Universidad Autónoma Metropolitana Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería

Reporte Final de Proyecto de Integración

Licenciatura en Ingeniería en Computación

Modalidad de Proyecto Tecnológico

Empleo de comandos de voz para el registro de citas médicas mediante una aplicación para dispositivos móviles con sistema operativo Android.

Alumno: Gerardo Hernández Vázquez
Matrícula: 210301361

Asesora: María Lizbeth Gallardo López

Co-asesora: Beatriz Adriana González Beltrán

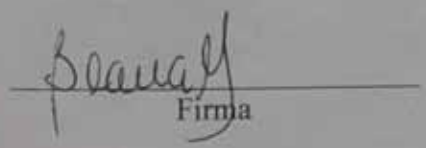
Trimestre 2015 Invierno

Fecha de entrega: 23 de abril de 2015

Nosotras,María Lizbeth Gallardo López y Beatriz Adriana González Beltrán, declaramos que aprobamos el contenido del presente Reporte de Proyecto de Integración y damos nuestra autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Firma

Asesora:
María Lizbeth Gallardo López
Profesor-investigador
Departamento de Sistemas
glizbeth@correo.azc.uam.mx



Firma

Co-asesora:
Beatriz Adriana González Beltrán
Profesor-investigador
Departamento de Sistemas
bgonzalez@correo.azc.uam.mx

Yo, Gerardo Hernández Vázquez, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco

Firma

Alumno:
Gerardo Hernández Vázquez
Matrícula: 210301361
al210301361@alumnos.azc.uam.mx

Resumen

El presente reporte describe el desarrollo del Proyecto de Integración *Empleo de comandos de voz para el registro de citas médicas mediante una aplicación para dispositivos móviles con sistema operativo Android*. La aplicación resultante de este proyecto se llama *Agenda por Voz Offline AVO*; se trata de una aplicación para dispositivos móviles con sistema operativo Android, cuyo propósito es gestionarlas citas médicas de pacientes; las operaciones que realiza la aplicación son: registrar, consultar, modificar, eliminar y recordar cita médica.

Una característica de *Agenda por Voz Offline AVO* es el empleo de comandos de voz para realizar las operaciones antes mencionadas, esto se logró con el empleo del API *PocketSphinx*. Otra característica de esta aplicación es el empleo de un enfoque diferente al de las agendas convencionales, en cuanto a la interacción con el usuario. Las agendas convencionales ponen especial interés en la fecha y hora de los eventos de un usuario; por lo tanto, se presenta inicialmente un calendario, donde se elige la fecha, después la hora y finalmente las características del evento. Mientras que la *Agenda por Voz Offline AVO,* pone énfasis en el evento y sus características; por lo tanto, la pantalla inicial no presenta un calendario, sino una pantalla donde es posible definir el tipo de evento antes de definir su fecha y hora.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. Introducción

En la vida diaria de cualquier persona se presentan eventos que deben ser planificados y luego recordados para cumplir con ellos, por ejemplo: pago de servicios, reuniones de trabajo y citas con el médico. A lo largo del tiempo se han propuesto diversas agendas (manuales y digitales) que permiten organizar y recordar los eventos importantes. Y desde la aparición de los teléfonos inteligentes, se han propuesto aplicaciones para planificar eventos.

Dentro de la variedad de aplicaciones para planificar eventos podemos encontrar las de diseño cercano a una agenda de cuadernillo, otras emplean etiquetas parecidas a una nota adherible, y recientemente se está proponiendo el empleo de voz para simular el dictado a un asistente personal. Se destacan también las aplicaciones que emplean la nube para planificar eventos permitiendo la sincronización simultánea desde cualquier dispositivo de cómputo.

En particular, las personas que tienen un padecimiento y se encuentran bajo algún tratamiento deben tener control sobre sus citas médicas: registrar, consultar, modificar, eliminar y que le sean recordadas. Algunos pacientes recurren a agendas de cuadernillo y otras recurren a la aplicación disponible en su teléfono inteligente. Quienes recurren a las aplicaciones, con frecuencia les resulta difícil registrar la cita médica, principalmente porque los pasos para lograrlo, incluyen: ubicar el mes en el calendario; luego, escoger la hora; posteriormente, escribir el texto que describa la cita; finalmente, deben confirmar para que la aplicación registre la información. Estos pasos suelen percibirse como excesivos, sobre todo cuando se trata de un paciente que requiere registrar no una sino varias citas médicas en un momento dado.

En este proyecto de integración se desarrolló una aplicación que permite a un paciente planificar sus citas médicas a través de comandos de voz; de tal manera que el paciente se comunique con la aplicación como si proporcionara indicaciones a un asistente personal. Con ello se ha logrado que la interacción se realice en un menor número de pasos que el requerido en una aplicación de agenda convencional. Además proponemos la opción de planificar las citas médicas a través del teclado del teléfono inteligente.

El reporte final cuenta con 11 secciones. La sección 1 presenta la introducción del proyecto; la sección 2 presenta los antecedentes del proyecto; la sección 3 presenta la justificación del proyecto; la sección 4 presenta los objetivos del proyecto; la sección 5 presenta el marco teórico del proyecto; la sección 6 presenta el desarrollo del proyecto; la sección 7 presenta los resultados obtenidos durante el desarrollo del proyecto; la sección 8 presenta el análisis y discusión de los resultados del proyecto; la sección 9 presenta las conclusiones del proyecto; la sección 10 presenta la perspectiva a futuro del proyecto y la sección 11 presenta la bibliografía completa del proyecto.

2. Antecedentes

A continuación se presentan varios proyectos relacionados con el proyecto *Empleo de comandos de voz para el registro de citas médicas mediante una aplicación para dispositivos móviles con sistema operativo Android*. Los proyectos relacionados están clasificados en: proyectos de integración, proyectos reportados en artículos, proyectos de tesis y software actualmente disponible.

2.1. Proyectos de integración.

Aplicación Colaborativa para Dispositivos Móviles con Sistema Operativo Android [1]. En este proyecto se desarrolló una aplicación colaborativa llamada SHAREDRAW que busca facilitar la interacción entre los miembros de un grupo de trabajo en la edición de figuras geométricas, así como en la edición texto. La relación con la propuesta presentada en este documento únicamente es que ambas son aplicaciones para dispositivos móviles con S.O. Android.

Sistema de Apoyo Clínico para el Tratamiento del Paciente Diabético [2]. Este proyecto presenta un sistema que reside en un servidor en Internet y cuenta con una base de datos relacional que almacena la información, gestiona el expediente clínico de los consultorios médicos y contiene un módulo especial para la atención de los pacientes diabéticos. La relación con la propuesta presentada en este documento es que ambos tienen un enfoque hacia pacientes con una enfermedad crónica.

SRCV: Sistema computacional interactivo basado en reconocimiento de comandos de voz para aplicaciones educativas [3]. El proyecto presenta un sistema SRCV que acepta comandos de voz (palabras simples dentro del "vocabulario del sistema"), como respuesta a preguntas sencillas, apoyadas con imágenes, las cuales hacen comprensible las respuestas por parte de los usuarios (niños en edad preescolar). El SRCV analiza las respuestas apoyándose en un Banco de Reactivos. La relación con la propuesta presentada en este documento es que el método de entrada del sistema SRCV son comandos de voz; sin embargo, nosotros emplearemos instrucciones específicas para manipular una agenda médica.

2.2. Artículos

Designing Android Applications Using Voice Controlled Commands: For Hands Free Interaction with Common Household Devices [4]. El artículo habla sobre el aprovechamiento de assistive tecnology para ayudar a personas con discapacidades. De forma más específica dentro del artículo se plantea la posibilidad de utilizar aplicaciones Android que por medio de comandos de voz le permitan al usuario controlar dispositivos

dentro de su hogar. El desarrollo del trabajo mencionado en el artículo se lleva a cabo en un Smartphone Samsung Galaxy S, y las pruebas consintieron en crear una interfaz en la que el Smartphone enviara señales vía Bluetooth a un control remoto de televisión, el cual convierte estas señales en instrucciones para la televisión. La relación con la propuesta presentada en este documento es que se hará uso de la biblioteca PocketSphinx, misma que emplean en el proyecto de este artículo.

2.3. Tesis

Proceso de desarrollo independiente de una aplicación móvil Android [5]. El autor hace un análisis sobre el futuro de las aplicaciones móviles en México y en el mundo; la evolución que tanto las aplicaciones y el S.O. Android han tenido, la forma de aprovechar el mercado y así obtener un beneficio económico de los proyectos desarrollados, la construcción de un ambiente de desarrollo adecuado, así como una metodología para el desarrollo de una aplicación móvil y la descripción de la aplicación misma. Con estos elementos, el autor logra que su tesis sea una referencia para aquellos desarrolladores que tengan en mente el crear una aplicación móvil.

2.4. Software

Asistente Agenda [6]. Aplicación Android que permite realizar el registro de eventos próximos en el calendario a través de comandos de voz utilizando el reconocedor de voz Google Voice perteneciente a Google Inc. Este reconocedor de voz requiere que el dispositivo del usuario esté conectado a una red de internet. La innovación de la aplicación a desarrollar como proyecto de integración es que a través de ella se podrá agendar citas por comandos de voz sin la necesidad de una conexión a la red de internet y de esta manera permitir el uso de la aplicación en cualquier momento y en cualquier lugar.

Seg-Social Cita Previa [7]. Aplicación Android que permite solicitar una cita previa. Esta aplicación busca ofrecer la opción de solicitar de forma ágil y rápida una cita en una de las oficinas de Seguridad Social, para evitar que un paciente espere inútilmente en el consultorio médico. Esta aplicación está enfocada al control de citas por parte de la seguridad social; mientras que la aplicación de nuestro proyecto está enfocada al control de las citas médicas de un paciente.

3. Justificación

Si bien existe una cantidad considerable de aplicaciones para planificar eventos, muchas de ellas frustran el interés de un paciente por emplearlas; con frecuencia les resulta difícil

registrar la cita médica, principalmente porque los pasos para lograrlo, incluyen: ubicar el mes en el calendario; luego, escoger la hora; posteriormente, escribir el texto que describa la cita; finalmente, deben confirmar para que la aplicación registre la información. Estos pasos suelen percibirse como excesivos, sobre todo cuando se trata de un paciente que requiere registrar no una sino varias citas médicas en un momento dado.

En este proyecto de integración proponemos desarrollar una aplicación que permita a un paciente planificar sus citas médicas a través de comandos de voz; de tal manera que el paciente se comunique con la aplicación como si proporcionara indicaciones a un asistente personal. Con ello buscamos que la interacción se logre en un menor número de pasos que el requerido en una aplicación de agenda convencional.

La aplicación que proponemos desarrollar representa una innovación para personas que padecen una enfermedad, ya que con su desarrollo buscamos aprovechar la capacidad que ofrecen los intérpretes de comandos de voz disponibles para teléfonos inteligentes. Cabe mencionar que los comandos de voz podrán ser procesados sin necesidad de una conexión de red (offline). Además proponemos la opción de planificar las citas médicas a través del teclado del teléfono inteligente.

4. Objetivos

4.1. Objetivo General

Desarrollar una aplicación Android para el registro de citas médicas mediante el empleo de comandos de voz.

4.2. Objetivos Específicos

1.-Diseñar e implementar el módulo para el reconocimiento de voz.

2.- Diseñar e implementar la interfaz de la aplicación.

3.- Diseñar e implementar el módulo para registrar una cita.

4.- Diseñar e implementar el módulo para consultar una cita.

5.- Diseñar e implementar el módulo para modificar una cita.

6.- Diseñar e implementar el módulo para eliminar una cita.

7.- Integrar los módulos para administrar una cita.

8.- Realizar pruebas de la aplicación.

5. Marco Teórico

A continuación se presenta el marco teórico del proyecto *Empleo de comandos de voz para el registro de citas médicas mediante una aplicación para dispositivos móviles con sistema operativo Android*, en este apartado se aborda la definición del sistema operativo Android que es el sistema operativo para el cual se desarrolla el proyecto; también se presenta la API PocketSphinx la cual se empleó para el procesamiento de comandos de voz.

5.1. Sistema operativo Android como plataforma de desarrollo de una aplicación

Androides un sistema operativo basado en Linux [8], fue diseñado para dispositivos móviles tales como *smartphones* y *tablets*. Android permite programar en un entorno de trabajo (framework) de Java. Las aplicaciones son ejecutadas sobre una máquina virtual llamada Dalvik. Dalvik es una variación de la máquina virtual de Java con compilación en tiempo real. Dado que Android está basado en el Kernel de Linux, tiene control sobre recursos como: controladores de pantalla, cámara, memoria flash, entre otros más. Android está formado por las siguientes capas [9]:

Applications: Está formada por la aplicaciones instaladas en el dispositivo.

Application Framework: Su objetivo es simplificar la reutilización de componentes, es decir, las aplicaciones pueden publicar sus capacidades y otras hacer uso de ellas.

Libraries: Incluye un conjunto de bibliotecas C/C++ usadas por los diferentes componentes de Android.

Android Runtime: Es la capa donde corren las aplicaciones, las cuales utilizan la máquina *Dalvik. Dalvik* es una variante de la máquina virtual de Java, pero diseñada para Android.

Linux Kernel: Proporciona servicios como seguridad, manejo de memoria, multiproceso, pila de protocolos y soporte para los drivers.

Durante el desarrollo de este proyecto, se utilizó la versión 4.1 y 4.3 de *Android* conocida como *Android JELLY BEAN* [10], la cual cuenta con las siguientes características:

- ✓ El *Kernel* del sistema operativo tiene un mejor rendimiento y mayor velocidad comparado con el que ofrecían las versiones anteriores.
- ✓ Se mejoró el consumo de energía durante la transmisión y recepción de datos a través de Bluetooth.
- ✓ Se incluye una opción de autocompletado más eficiente; al digitar números o letras en el teclado, el sistema sugerirá números o nombres completos de los contactos disponibles en el dispositivo.
- ✓ Se mejoró la compatibilidad de idiomas; se agregó un mayor soporte para los idiomas hebreo y árabe.

5.2 PocketSphinx para empleo de comandos de voz

PocketSphinx [11] es una versión del sistema de habla continua y reconocimiento de habla, el cual emplea el modelo oculto de *Márkov* o HMM, por sus siglas en inglés: *Hidden Márkov Model,* y un lenguaje de modelado estadístico de n-gramas. Sphinx se utiliza también en sistemas embebidos; por ejemplo, un procesador con arquitectura *Advanced RISC Machine ARM*[1] [12]. *PocketSphinx* está siendo evaluado para desarrollar e incorporar características como la aritmética de coma fija y algoritmos eficientes para el cálculo en modelos mezclados; un modelo mezclado es aquel en el que durante la captura de audio hay otros elementos presentes como pueden ser música, ruido y voces. PocketSphinx puede ser empleado en equipos portátiles y también en teléfonos móviles.

6. Desarrollo del Proyecto

6.1. Metodología empleada en el desarrollo del proyecto

El Proceso Unificado (UP por sus siglas en inglés de Unified Process) se ha convertido en un proceso de desarrollo de software de gran éxito para la construcción de sistemas orientados a objetos. El UP promueve un desarrollo iterativo e incremental; además, se organiza en iteraciones de duración fija. Cada iteración aborda nuevos requisitos y amplía el sistema incrementalmente [13].

Para el desarrollo de *Agenda por Voz Offline AVO* se definió una iteración con 2 semanas. En la fase de inicio realizaron dos iteraciones (I1, I2, I3), en la fase de elaboración tres iteraciones (E1, E2, E3), en la fase de construcción seis iteraciones (C1-C6) y en la fase final transición tres iteraciones (T1, T2, T3).

Inicio: Visión aproximada e inicio del documento de visión, análisis de la lógica del negocio, y definición del alcance del proyecto. Planificación de actividades de acuerdo al calendario escolar, e inicio de la documentación. Bosquejo de las interfaces del sistema. Instalación y acondicionamiento del entorno de desarrollo.

Elaboración: Refinamiento del documento de visión, definición de la arquitectura basada en patrones de diseño de software. Identificación de otros requisitos, y asignación de prioridad a los requisitos. Análisis iterativo a través de la especificación de los casos de uso de texto para los requisitos. Diseño, implementación y pruebas de la base de datos preliminar, implementación del módulo del menú principal, y su documentación. Inicio del diseño del módulo de registro de nueva cita. Comienzo de la redacción del manual técnico y la memoria de diseño.

[1] ARM es una arquitectura RISC (Reduced Instruction Set Computer, en español: Ordenador con Conjunto Reducido de Instrucciones) de 32 bits desarrollada por ARM Holdings.

Construcción: Diseño e implementación definitiva de la base de datos, diseño e implementación iterativa de los módulos para el resto de los requisitos; a saber se continuó y finalizó el diseño e implementación del módulo de registro de nueva cita y se inició y terminó el diseño e implementación de los módulos para: eliminar, modificar y consultar una cita, y la preparación para el despliegue. Finalización del manual técnico, manual de usuario, memoria de diseño, y reporte final del proyecto.

Transición: Pruebas unitarias, integración de los módulos, y pruebas de sistema. Evaluación y retroalimentación de la documentación, análisis de resultados en base a los objetivos propuestos. Identificación de limitantes del sistema y propuestas de posibles soluciones.

6.2. Diseño del sistema

A continuación se presentan los artefactos de diseño: diagrama de casos de uso, casos de uso de texto, y diagrama de clases, los cuales sirvieron de base para el desarrollo de *Agenda por voz Offline AVO*.

6.2.1. Diagrama de casos de uso de la aplicación *Agenda por Voz Offline AVO*

La aplicación *Agenda por Voz Offline AVO* comprende cinco casos de uso: Configurar aplicación, Registrar cita, Eliminar cita, Modificar cita, Consultar cita (Véase Figura 1).

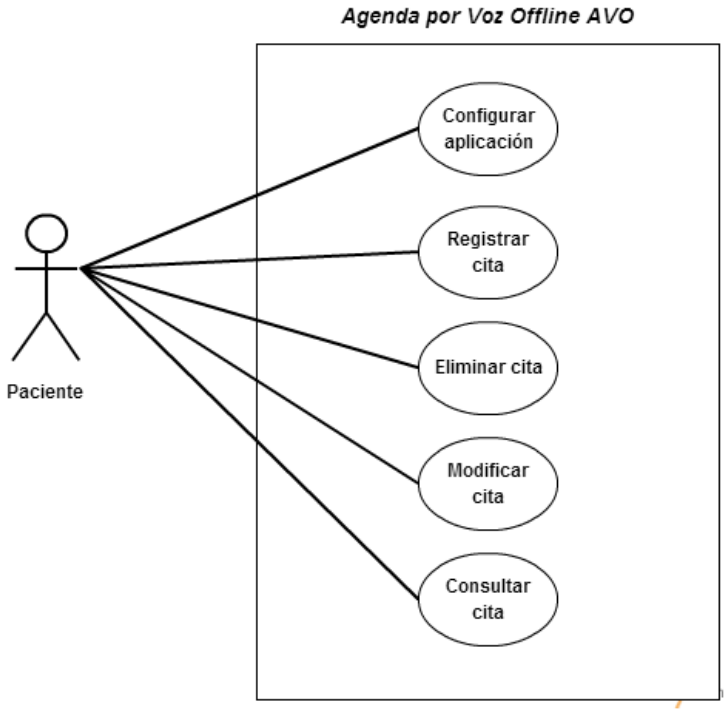


Figura 1. Casos de uso de la aplicación Agenda por Voz Offline AVO

6.2.2. Casos de uso de texto de la aplicación *Agenda por Voz Offline AVO*

Tomaremos los casos de uso `Establecer configuración` y `Registrar cita` para ejemplificar, a lo largo del documento, las diferentes etapas del diseño. Los detalles del resto de los casos de uso de la figura 1 se encuentran en la memoria de diseño del sistema.

6.2.2.1. CASO DE USO DE TEXTO CONFIGURAR APLICACIÓN

Identificador: 00EstablecerConfiguración
Nombre: Establecer Configuración
Descripción: El paciente establecerá la configuración que tendrá por defecto la aplicación *Agenda por Voz Offline AVO* desde el menú correspondiente.
Actor principal: Paciente
Pre-condiciones: • El paciente ha presionado el menú `Configuración` desde la *actionbar* en la pantalla principal.
Flujo principal: 1. El sistema despliega el formulario correspondiente para establecer la configuración. 2. El paciente presiona el botón `Frecuencia de citas`. 3. El paciente establece la frecuencia que tendrán sus citas médicas. 4. El paciente presiona el botón `Fijar Hora`. 5. El paciente establece la hora en que comúnmente serán sus citas. 6. El paciente presiona el botón `Fijar Especialista`. 7. El paciente establece el especialista con quien comúnmente serán sus citas. 8. El paciente presiona el botón `Recordatorio`. 9. El paciente establece con que anticipación se le recordará de sus citas. 10. El paciente presiona el botón `Guardar`. 11. El sistema indica que la configuración se guardó con éxito. 12. El sistema cierra el dialogo y vuelve a la pantalla principal.
Flujo alternativo: *El paciente en cualquier momento cancela la operación presionando el botón `Cancelar`, los cambios no se guardarán.
Post-condiciones: El paciente regresa a la pantalla principal y puede elegir la misma o alguna otra opción disponible; o bien, salir de la aplicación.
Requerimientos no funcionales: - La interfaz de la aplicación será clara y fácil de utilizar para el paciente. - La aplicación se ejecutará en la mayoría de los dispositivos que operan con Android a partir de la versión 4.0, pues a partir de esta versión, el sistema operativo ofrece un mejor rendimiento; esto es posible gracias a que su *kernel* aprovecha mejor el hardware del dispositivo y cuenta con un nivel 14 del API[2], lo cual permite el correcto despliegue de los *layouts*.

[2]API es un conjunto de subrutinas, funciones y procedimientos que ofrece una biblioteca determinada. Un API es utilizado por otro software como una capa de abstracción.

6.2.2.2. CASO DE USO DE TEXTO REGISTRAR CITA

Identificador: 01RegistrarCita
Nombre: Registrar cita
Descripción: El paciente registrará una nueva cita médica.
Actor principal: Paciente
Pre-condiciones: • El paciente cumplió con el caso de uso de texto Establecer Configuración. • El paciente ha elegido algún modo de operación en la pantalla principal. • El paciente ha presionado el botón `Registrar Cita` en el menú de opciones.
Flujo principal: 1. Aparece el formulario de registro de nueva cita con valores predeterminados. 2. El paciente elige guardar la cita con los valores por defecto. 3. El sistema despliega un dialogo de confirmación para guardar la cita. 4. El paciente confirma que desea guardar la cita. 5. El sistema realiza el registro de la cita en la base de datos. 6. El sistema realiza el registro de la cita en el calendario del dispositivo. 7. El sistema despliega una notificación indicando al paciente que la cita se guardó. 8. La aplicación regresa a la pantalla principal.
Flujo alternativo: 1.1 El paciente elige el control manual y no existen valores por defecto. 1.1.1 El paciente presiona el botón `Fijar fecha`. 1.1.2 El paciente establece la fecha de su cita. 1.1.3 El paciente presiona el botón `Fijar hora` 1.1.4 El paciente establece la hora de su cita. 1.1.5 El paciente presiona el botón `Fijar Especialista` 1.1.6 El paciente establece el especialista con quién será su cita. 1.1.7 El paciente presiona el botón `Recordatorio` 1.1.8 El paciente fija el recordatorio de su cita. 1.1.9 El paciente presiona el botón `Guardar` 1.1.10 El sistema despliega un dialogo de confirmación para guardar la cita. 1.1.11 El paciente confirma que desea guardar la cita. 1.1.12 El sistema realiza el registro de la cita en la base de datos. 1.1.13 El sistema realiza el registro de la cita en el calendario del dispositivo. 1.1.14 El sistema despliega una notificación indicando al paciente que la cita se guardó correctamente. 1.2 El paciente elige el control por voz de la aplicación. `1.2.1` El paciente modifica el campo `Fecha` haciendo uso de uno de los comandos: `<<DosSemanas>>, <<UnMes>>, <<MedioAño>>.` 1.2.2 El paciente modifica el campo `Hora` haciendo uso de uno de los comandos: `<<Siete>>, <<Nueve>>, <<MedioDía>>.` 1.2.3 El paciente modifica el campo `Especialista` haciendo uso de uno de los comandos: `<<Dentista>>, <<Pediatra>>, <<Urólogo>>.`

1.2.4 El paciente modifica el campo `Recordatorio` haciendo uso de uno de los comandos: `<<QuinceAntes>>, <<TreintaAntes>>`.

1.2.5 El paciente emplea el comando de voz `<<Guardar>>`

1.2.6 El sistema le pide al paciente la confirmación para guardar la cita.

1.2.7 El paciente confirma la acción, seleccionando manualmente la opción `Guardar`.

1.2.8 El sistema realiza el registro de la cita en la base de datos.

1.2.9 El sistema realiza el registro de la cita en el calendario del dispositivo.

1.2.10 El sistema despliega una notificación indicando al paciente que la cita se guardó correctamente.

1-a El paciente elige cancelar la operación y regresa al menú de opciones, presionando el botón `Regresar` del dispositivo.

1-b El paciente activa o desactiva el control por voz en cualquier momento.

Post-condiciones:

* El paciente regresa a la pantalla principal y puede elegir la misma o alguna otra opción disponible; o bien, salir de la aplicación.

Requerimientos no funcionales:

- La interfaz de la aplicación será clara y fácil de utilizar para el paciente.
- La aplicación se ejecutará en la mayoría de los dispositivos que operan con Android a partir de la versión 4.0, pues a partir de esta versión, el sistema operativo ofrece un mejor rendimiento; esto es posible gracias a que su *kernel* aprovecha mejor el hardware del dispositivo y cuenta con un nivel 14 del API, lo cual permite el correcto despliegue de los *layouts*.

6.2.3. Diagrama de clases del sistema y arquitectura del sistema

De manera general el proyecto de la aplicación *Agenda por Voz Offline AVO* está contenido en la carpeta `AVO`; la cual contiene jerárquicamente los módulos, carpetas y archivos de configuración y de construcción `Gradle`[3]. (Véase figura 2).

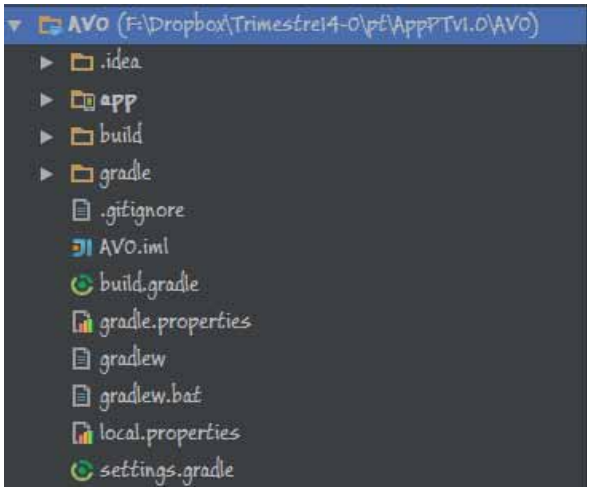


Figura 2. Panel "Project" de la aplicación Agenda por Voz Offline AVO

Carpeta `app`: Es la carpeta que representa el módulo de le aplicación *Agenda por Voz Offline AVO*. Al desplegarse se pueden ver tres carpetas: `build`, `libs` y `src` (Véase Figura 3).

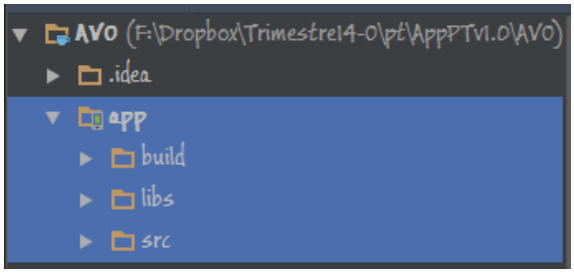


Figura 3. Contenido de la carpeta app.

[3] Gradle es una herramienta de Android Studio que permite realizar la construcción de un proyecto Android.

Carpeta `libs`: Contiene la biblioteca `pocketsphinx-android-0.8-nolib.jar` la cual contiene todos los métodos encargados del procesamiento de los comandos de voz. La versión empleada en este proyecto es la 0.8 (Véase Figura 4).

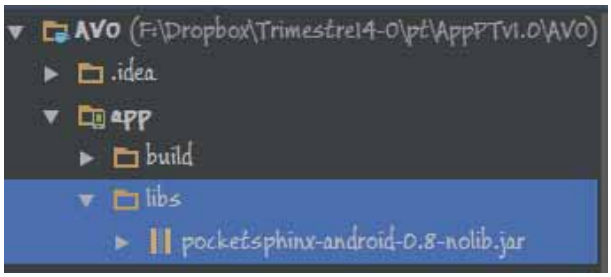


Figura 4. Contenido de la carpeta libs.

Carpeta `src`: Dentro de esta carpeta se encuentra la carpeta principal `main`, la cual contiene las carpetas: `assets`, `java`, `jniLibs` y `res` así como el archivo `AndroidManifiest.xml` (Véase Figura 5).

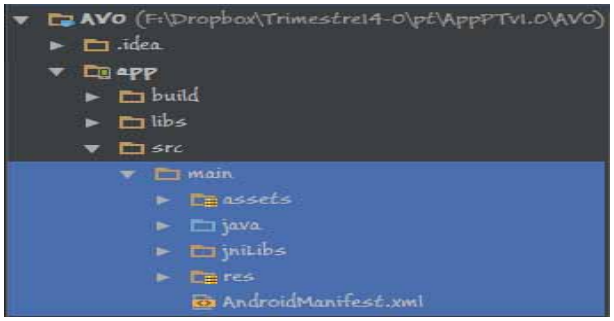


Figura 5. Estructura de la carpeta src.

Carpeta `java`: Contiene todos los archivos fuente Java de la aplicación, agrupados en su respectivo paquete (Véase la figura 6).

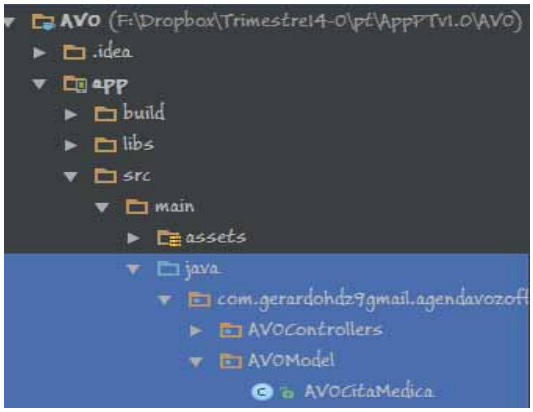


Figura 6. Estructura de la carpeta java.

Siguiendo la secuencia de las imágenes 2, 3, 4, 5 y 6 se accede al paquete `com.gerardohdz9gmail.agendavozoffline`, el cual contiene las clases java del proyecto agrupadas por paquetes (Véase Figura 7), los cuales serán explicados a continuación:

- `com.gerardohdz9gmail.agendavozoffline`: Representa el paquete que identifica la firma del desarrollador de la aplicación. Contiene los paquetes principales que organizan las clases java.
- `AVOActivityControllers`: Contiene las clases controladoras (`Activity`) de la aplicación *Agenda por Voz Offline.*
- `AVODBControllers`: Contiene las clases controladoras de la base de datos de la aplicación *Agenda por Voz Offline.*
- `AVOModel`: Contiene las clases que representan el modelo de la aplicación *Agenda por Voz Offline.*

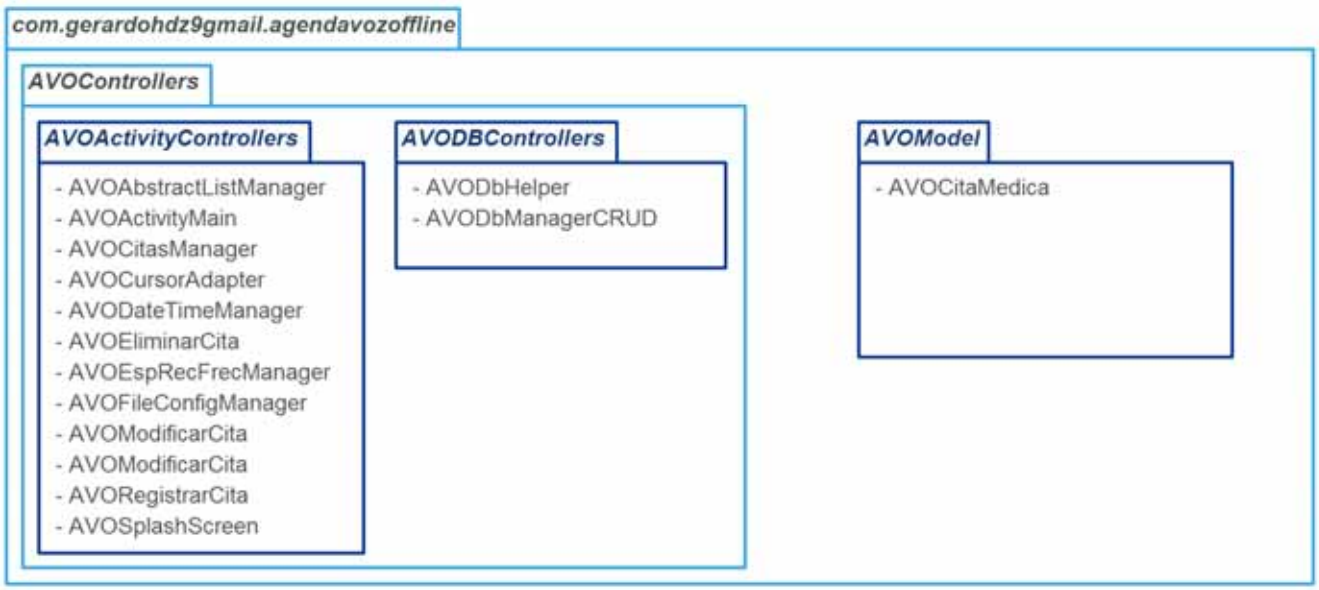


Figura 7. Diagrama de paquetes de la aplicación Agenda por Voz Offline

En el diagrama de paquetes de la aplicación *Agenda por Voz Offline AVO* se observa que el proyecto está compuesto por tres clases abstractas y 14 clases concretas. A continuación se describen[4] las clases de la aplicación *Agenda por Voz Offline AVO*:

[4]La descripción de las clases no incluye la descripción de sus métodos, para mayor información consultar la documentación del proyecto contenida en la carpeta AVOJavaDoc anexa en el CD del Proyecto de Integración.

Clases abstractas e interfaces:

- `Activity`: Es una clase abstracta, que constituye la base de toda aplicación Android; esta clase contiene métodos elementales encargados de crear, mostrar, pausar, reiniciar, ejecutar en segundo plano y parar cualquier `activity` de la aplicación.
- `RecognitionListener`: Es una `interface` que contiene los métodos utilizados para el reconocimiento y procesamiento de los comandos de voz que se dictan a la aplicación.
- `AVOAbstractListManager`: Es una clase abstracta que contiene los métodos responsables de recuperar, de la base de datos, la información de las citas, así como de desplegarlas en la `listview` presente en la pantalla de una `activity`; también contiene los métodos eliminar, modificar y consultar citas (Véase Figura 8)

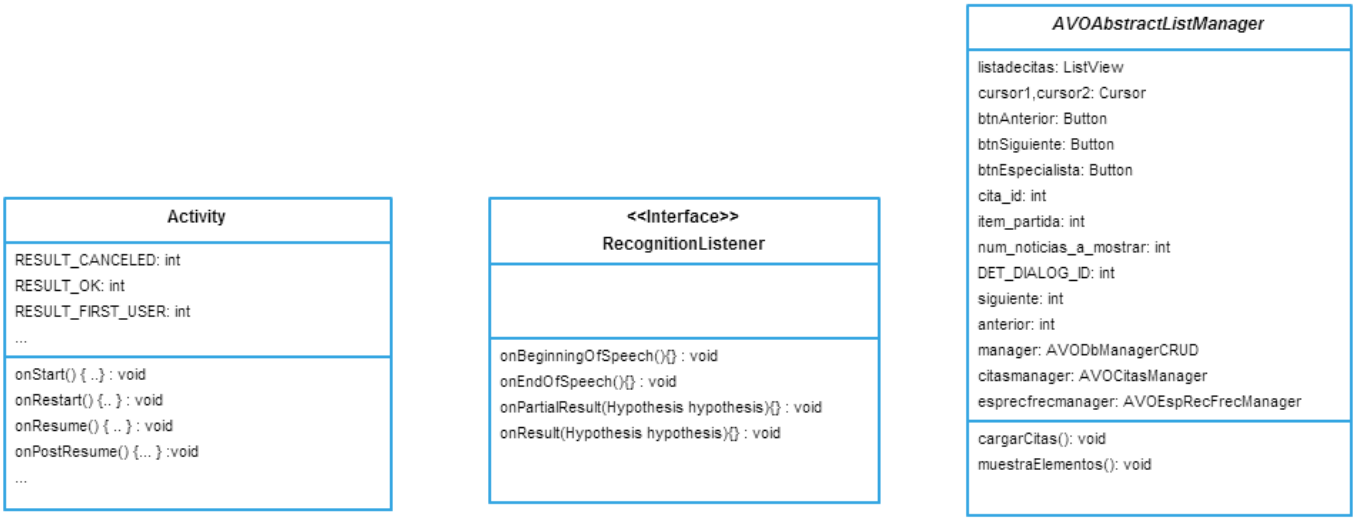


Figura 8. Clases abstractas e interfaces

Clases del paquete `AVOActivityControllers` (Parte 1/4):

- AVOActivityMain: Es la clase que controla la actividad principal de la aplicación que contiene los métodos lanzarRegistroCita(), lanzarEliminarCita(), lanzarModificarCita() y lanzarConsultarCita(), encargados de lanzar las actividades Registrar, Eliminar, Modificar y Consultar Cita respectivamente. Esta clase también se encarga de definir la lógica del control por voz de la pantalla principal(Véase Figura 9); esto se logró implementado los métodos proporcionados por la interface RecognitionListener; a saber onBeginningOfSpeech() permite definir alguna acción justo al iniciar el reconocimiento de comandos de voz, por ejemplo desplegar un mensaje que informe al paciente que el reconocimiento de voz ha iniciado; onEndOfSpeech() permite definir alguna acción justo después de terminar el reconocimiento de comandos de voz, por ejemplo reiniciar el reconocedor de voz para poder realizar el reconocimiento de otro comando; onPartialResult() permite definir acciones sobre un resultado parcial del procesamiento de voz, por ejemplo procesar el resultado parcial y definir si es conveniente detener el procesamiento del comando recibido con el objetivo de rápidamente descartar comandos erróneos sin necesidad de concluir su procesamiento completo; onResult() permite definir acciones sobre el resultado final del procesamiento de un comando de voz, por ejemplo validar de que comando se trata y con base en ello indicarle el comportamiento a seguir a la aplicación; setupRecognizer() permite inicializar el reconocedor de voz, por ejemplo permite indicarle con que diccionario y lista de comandos deberá trabajar el reconocedor de voz; switchSearch() permite dar inicio al reconocimiento de voz. En general estos métodos permiten iniciar el reconocimiento de voz y validar el uso de los comandos: <<nueva>>, <<eliminar>>, <<modificar>>, <<consultar>> y <<ayuda>>.

AVORegistrarCita: Es la clase que controla la actividad registrar una nueva cita, para lo cual emplea a las clases: AVODateTimeManager y AVOEspRecFrecManager. Las tres clases permiten obtener los atributos: fecha, hora, especialista y recordatorio de la cita. AVORegistrarCita guarda estos atributos en un objeto de la clase AVOCitaMedica, y lo envía a la clase AVOCitasManager con el propósito de registrar la cita en el calendario y en la base de datos del dispositivo. Esta clase también se encarga de definir la lógica del control por voz para registrar una cita (Véase Figura 9); esto se logró implementado los métodos proporcionados por la interface RecognitionListener, los cuales fueron descritos en la clase AVOActivityMain. Estos métodos permiten iniciar el reconocimiento de voz y validar el uso de los comandos: <<dos semanas>>, <<un mes>>, <<medio año>>para ajustar la fecha; <<siete>>, <<nueve>> y

<<medio día>> para ajustar la hora; <<dentista>>, <<pediatra>> y <<urólogo>> para ajustar el especialista; <<quince antes>> y <<treinta antes>> para ajustar el recordatorio, y por último <<ayuda>> que despliega una ventana con todos los comandos disponibles.

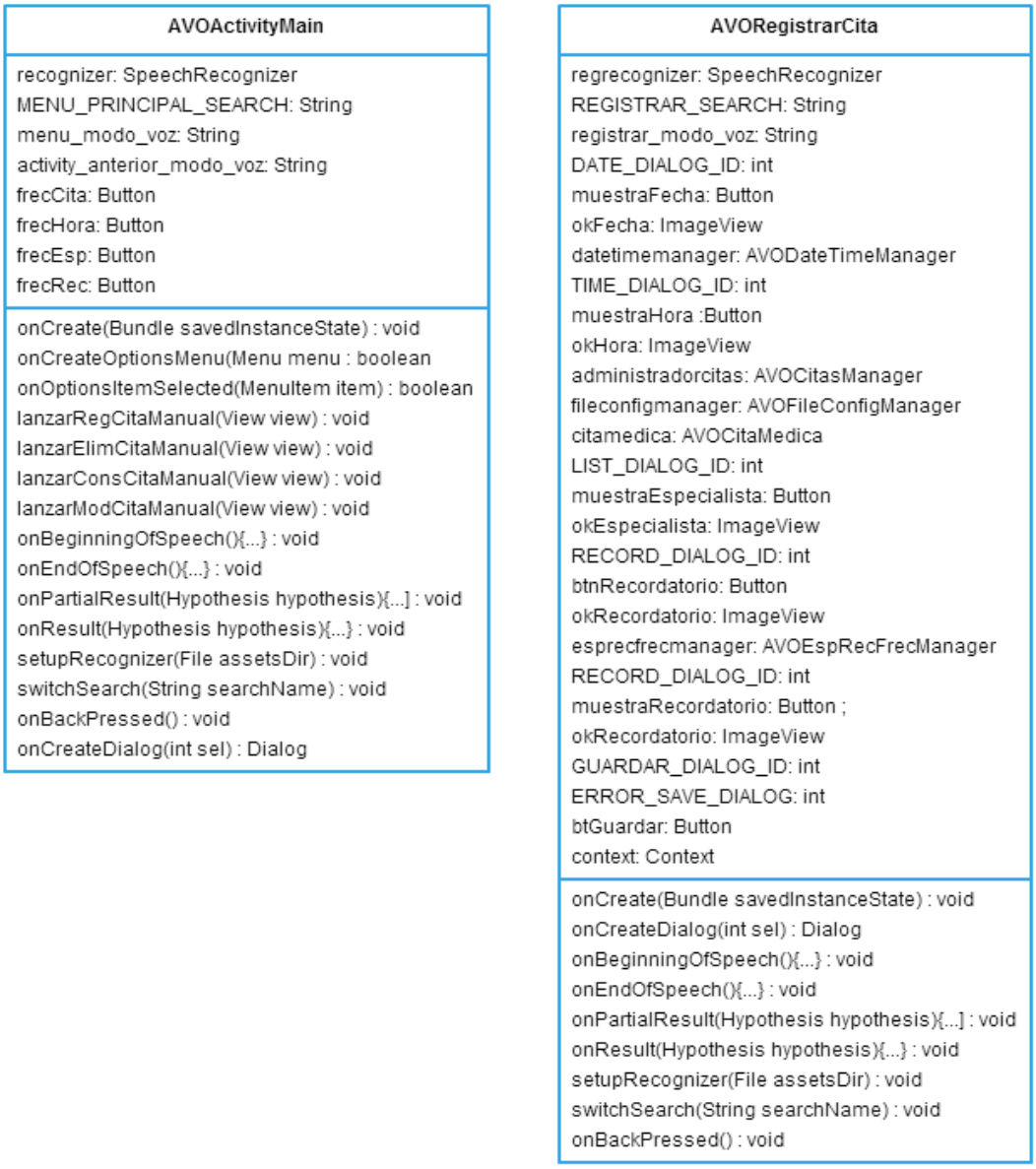


Figura 9. Clases del paquete AVOActivityControllers (parte 1/4)

Clases del paquete AVOActivityControllers (Parte 2/4):

- AVOEliminarCita: Esta clase controla la actividad eliminar una cita médica. Emplea la clase AVOAbstractListManager para desplegar las citas existentes en el dispositivo en una ListView. Emplea paginación en caso de existir más de seis citas registradas, valida y envía la petición de eliminación a la clase AVOCitasManager. Esta clase también se encarga de definir la lógica del control por voz para eliminar una cita (Véase Figura 10); esto se logró implementado los métodos proporcionados por la interface RecognitionListener, los cuales fueron descritos en la clase AVOActivityMain. Estos métodos permiten iniciar el reconocimiento de voz y validar el uso de los comandos: <<dentista>>, <<pediatra>>, <<urólogo>> y <<todos>>para realizar el filtrado de las citas por especialista, y <<siguiente>> y <<anterior>> para realizar el paginado.

- AVOModificarCita: Esta clase controla la actividad modificar una cita. Emplea la clase AVOAbstractListManager para desplegar las citas existentes en el dispositivo en una ListView. Emplea paginación en caso de existir más de seis citas registradas, valida y envía la petición de modificación a la clase AVOCitasManager. Esta clase también se encarga de definir la lógica del control por voz para modificar una cita (Véase Figura 10); esto se logró implementado los métodos proporcionados por la interface RecognitionListener, los cuales fueron descritos en la clase AVOActivityMain. Estos métodos permiten iniciar el reconocimiento de voz y validar el uso de los comandos: <<dentista>>, <<pediatra>>, <<urólogo>> y <<todos>>para realizar el filtrado de las citas por especialista, y <<siguiente>> y <<anterior>> para realizar la paginación.

- AVOConsultarCita: Es la clase que controla la actividad consultar una cita médica. Emplea la clase AVOAbstractListManager para desplegar las citas existentes en el dispositivo en una ListView. Emplea paginación en caso de existir más de seis citas registradas. Esta clase también se encarga de definir la lógica del control por voz para consultar una cita (Véase Figura 10); esto se logró implementado los métodos proporcionados por la interface RecognitionListener, los cuales fueron descritos en la clase AVOActivityMain. Estos métodos permiten iniciar el reconocimiento de voz y validar el uso de los comandos: <<dentista>>, <<pediatra>>, <<urólogo>> y <<todos>>para realizar el filtrado de las citas por especialista, y <<siguiente>> y <<anterior>> para realizar el paginado.

Figura 10. Clases del paquete AVOActivityControllers (parte 2/4)

Clases del paquete `AVOActivityControllers` (Parte 3/4):

- `AVOCitasManager`: Esta clase contiene los métodos encargados de validar la existencia del registro de una cita en una fecha determinada para evitar el registro de nuevas citas en la misma fecha. También contiene los métodos encargados de registrar, eliminar, modificar y consultar una cita en el calendario y en la base de datos del dispositivo (Véase Figura 11).

- `AVOEspRecFrecManager`: Esta clase contiene los métodos encargados de configurar parámetros como: especialista, recordatorio y frecuencia de las citas, en la actividad de registro de una cita (Véase Figura 11).

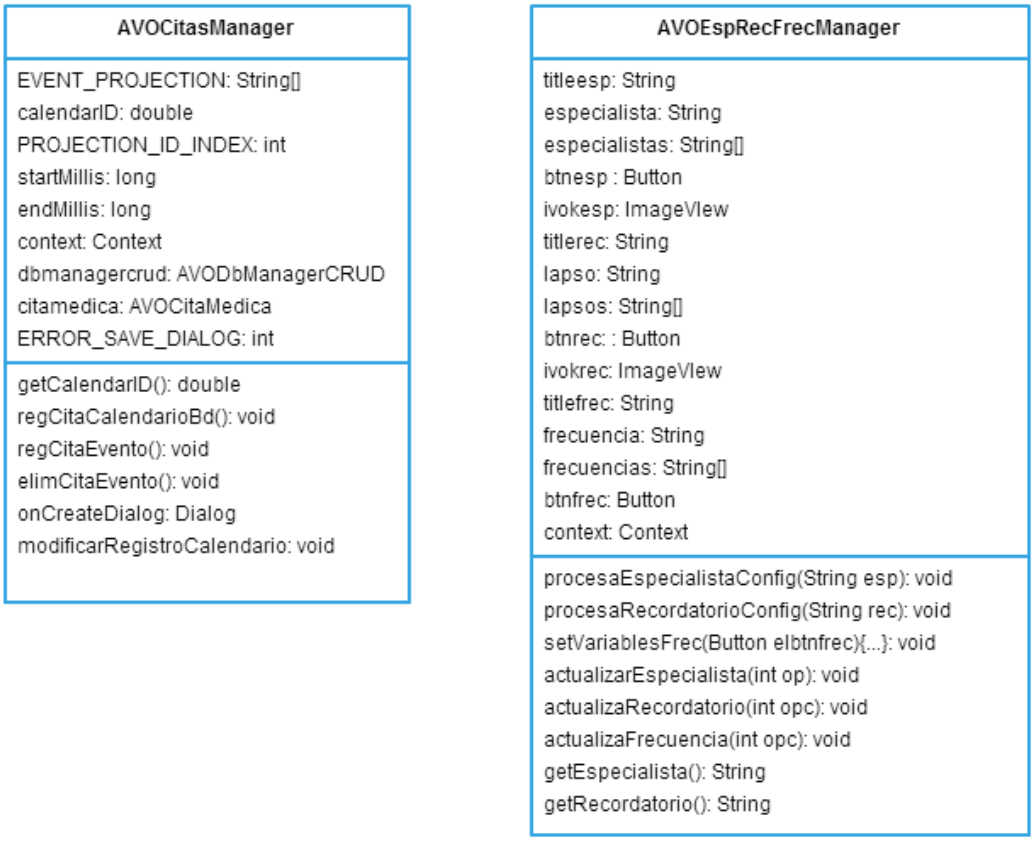


Figura 11. Clases del paquete AVOActivityControllers (parte 3/4)

Clases del paquete `AVOActivityControllers` (Parte 4/4):

- `AVOFileConfigManager`: Esta clase contiene los métodos encargados de guardar y recuperar los datos de configuración de la aplicación, elegidos por el paciente. Los datos de configuración se encuentran en el archivo `AVOSettings.txt`, ubicado en la carpeta de instalación `com.gerardohdz9gmail.agendavozofflinede` la aplicación (Véase Figura 12).

- `AVODateTimeManager`: Esta clase contiene los métodos encargados de procesar las opciones de fecha y hora en la actividad registrar una cita (Véase Figura 12).

- `AVOSplashScreen`: Esta clase controla la actividad de inicio; es decir, realiza el despliegue del `layout` cuyo fondo es la imagen que se muestra al iniciar la aplicación (Véase Figura 12).

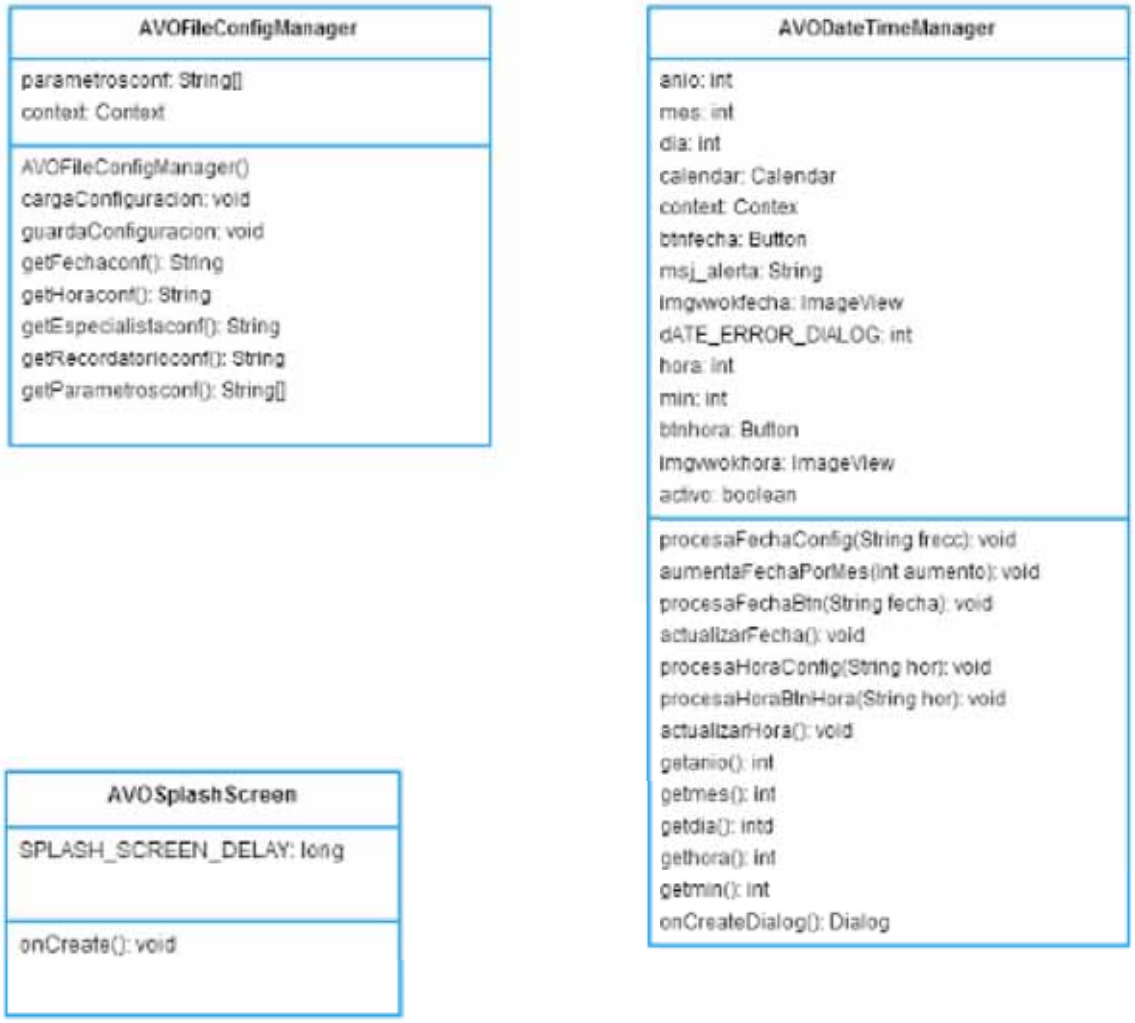


Figura 12. Clases del paquete AVOActivityControllers (parte 4/4)

Clases del paquete `AVOBDControllers`:

- `DBManagerCRUD`: Esta clase contiene los métodos encargados de insertar, eliminar, actualizar y consultar los registros de la base de datos (Véase Figura 13).

- `AVODBHelper`: Contiene la definición de la base de datos(Véase Figura 13)

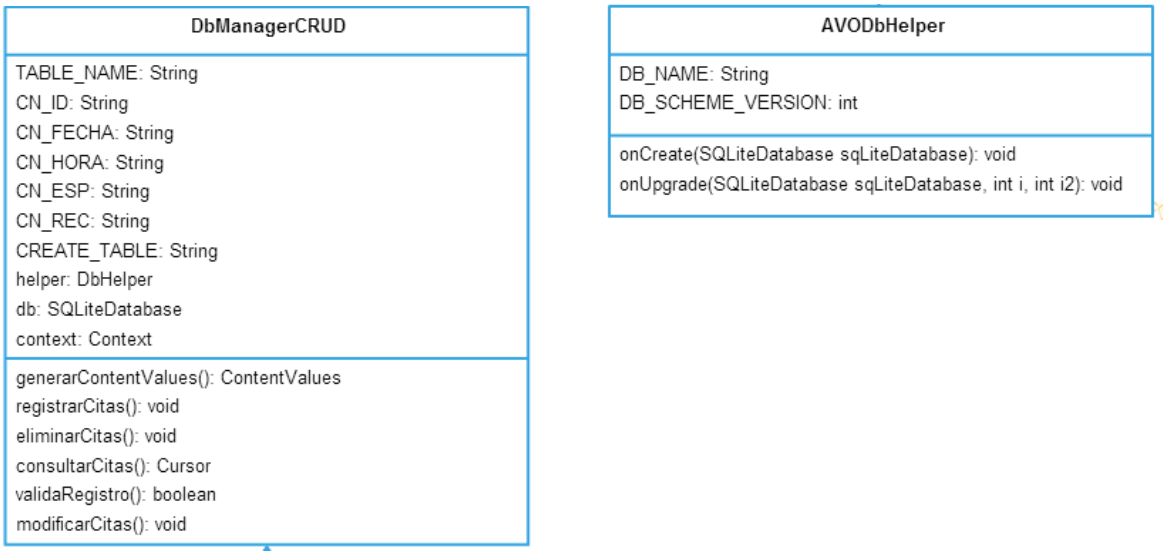


Figura 13. Clases del paquete AVOBDControllers

Clases del paquete `AVOBDControllers`:

- `AVOCitaMedica`: Esta clase representa el modelo del proyecto (Véase Figura 14).

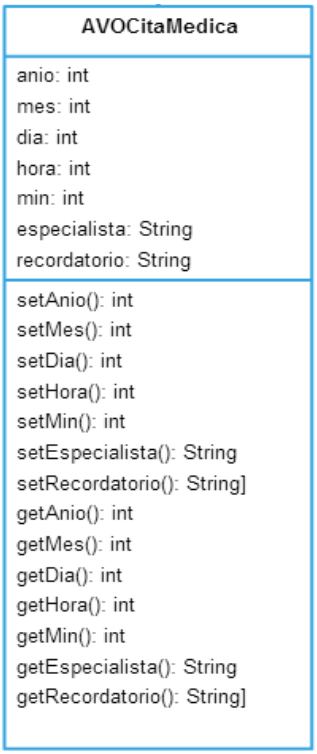


Figura 14. Clase del paquete AVOModel

Las relaciones que existen entre las clases antes descritas están representadas en el diagrama de clases de la aplicación *Agenda por Voz Offline* que se muestra a continuación:

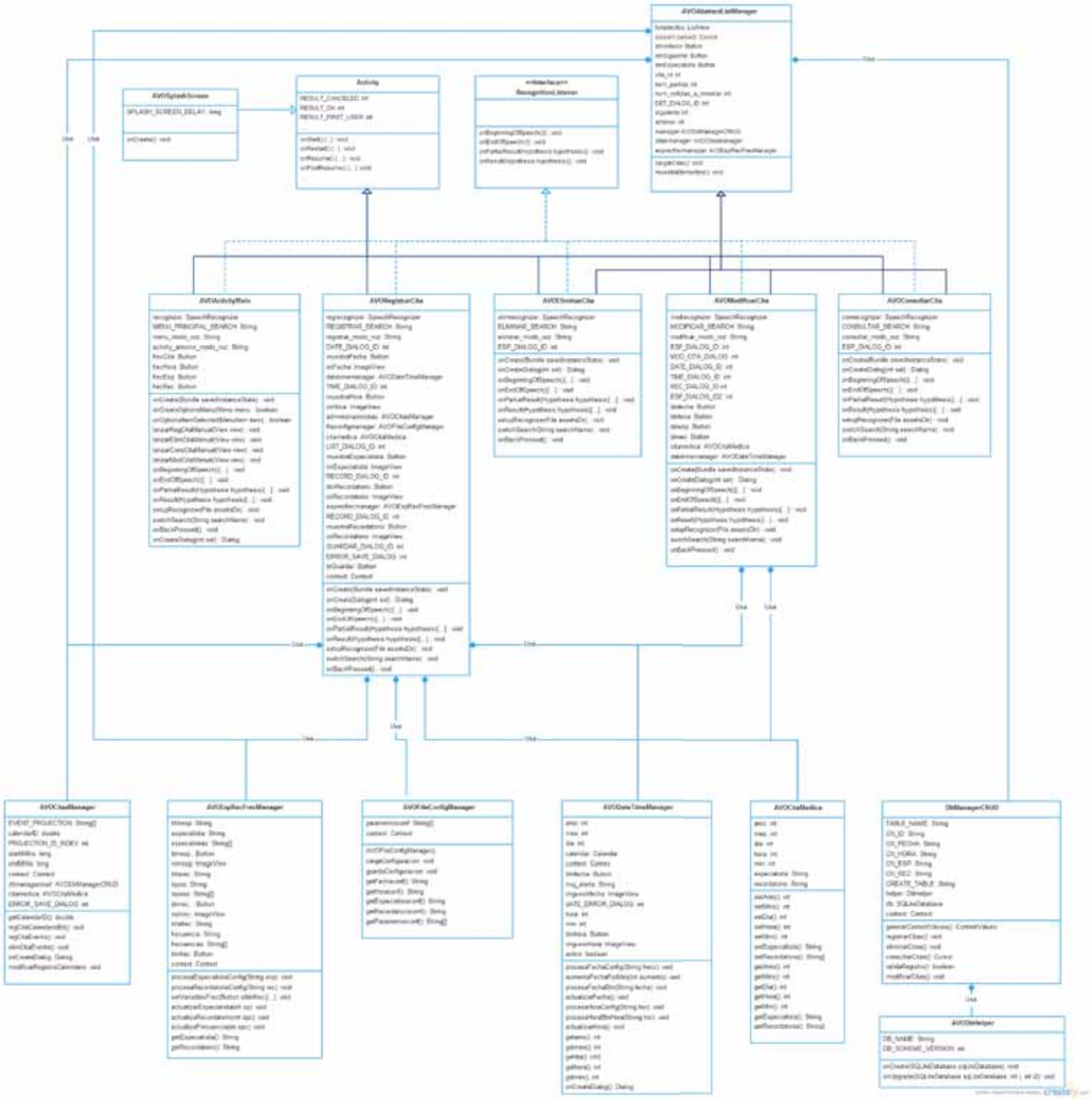


Figura 15. Diagrama de clases de la aplicación Agenda por Voz Offline AVO

6.2.4. Diagramas de secuencia

El registro de una nueva cita se realiza de la siguiente manera; el paciente inicia la aplicación, elige la opción registrar cita en el menú de la pantalla principal, fija la fecha, fija la hora, fija el especialista, fija el recordatorio y manda la petición de registro en la bases de datos y calendario del dispositivo, por último se le informa al paciente que su cita se ha registrado (Véase figura 16).

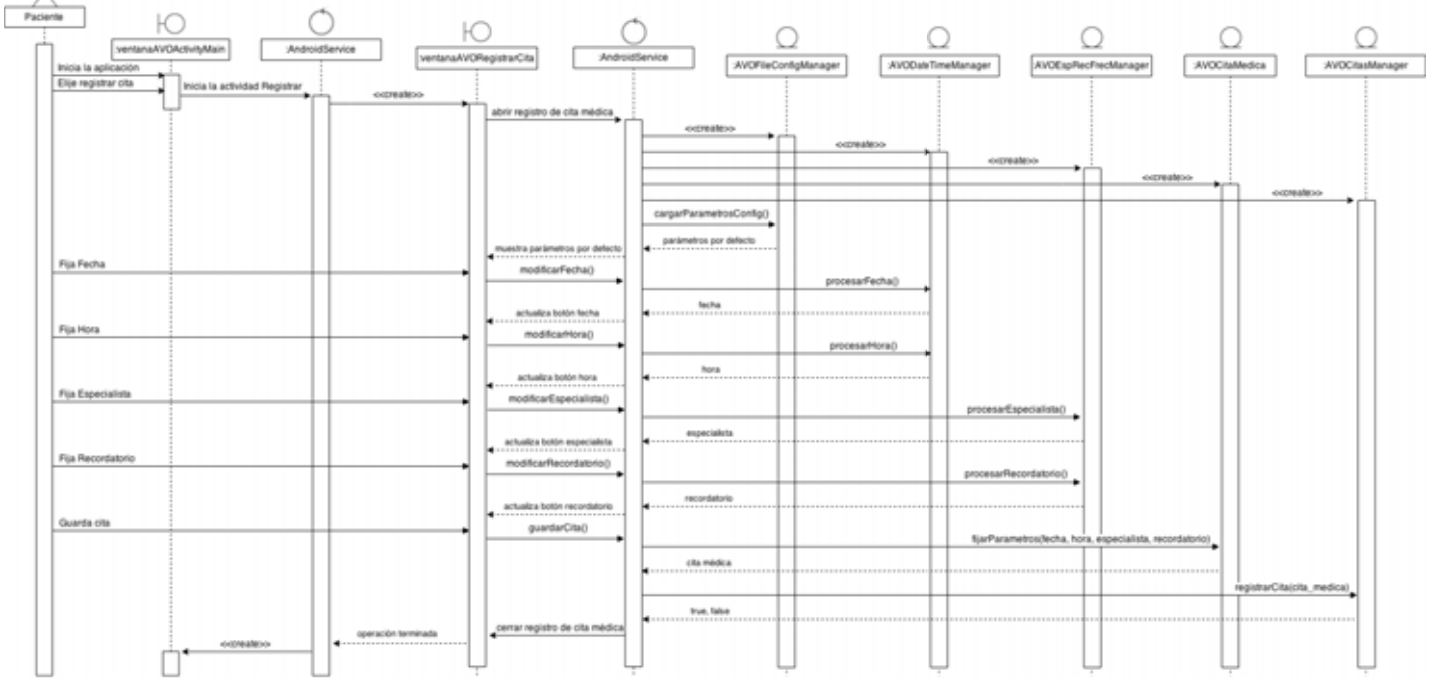


Figura 16. Diagrama de secuencia de la actividad de registro de una cita.

6.2.5 Estructura de la base de datos

La aplicación *Agenda por Voz Offline AVO* cuenta con una base de datos con el nombre de AVOBD, esta base de datos contiene una única tabla llamada CITA donde se almacenan los atributos de las citas del paciente (Véase Figura 17).

Figura 17. Tabla CITA de la base de datos AVOBD

6.3. Uso del sistema

Pantalla principal: Contiene el menú de opciones disponibles para el paciente(Véase Figura 18); a saber las opciones registrar, eliminar, modificar, consultar cita, configuración por default y activación del control por voz; los comandos de voz disponibles para esta sección son `<<nueva>>`, `<<eliminar>>`, `<<modificar>>`, `<<consultar>>` y `<<ayuda>>`(véase Figura22).

Figura 18. Pantalla principal de la aplicación Agenda por Voz Offline AVO

Menú: Es un menú básico disponible en toda aplicación `Android`, contiene las opciones de configuración y ayuda de la aplicación, las cuales están disponibles para el paciente. En este momento el menú no cuenta con comandos de voz, solo está disponible el control manual (véase Figura 19).

Figura 19. Menú de la aplicación Agenda por Voz Offline AVO

Diálogo de configuración: Permite al paciente definir las opciones por defecto de la aplicación, a saber: definir la frecuencia con que asiste a sus citas, definir la hora en que comúnmente son programadas sus cita, definir el especialista con el cual asiste con más frecuencia, y seleccionar con qué tiempo de anticipación quiere que se programe un alerta que le permita recordar la cita programada. En esta sección no hay comandos de voz disponibles, porque implica implementar el control por voz en el componente diálogo de Android, el cual está encapsulado y del cual desconocemos su implementación (véase Figura 20)

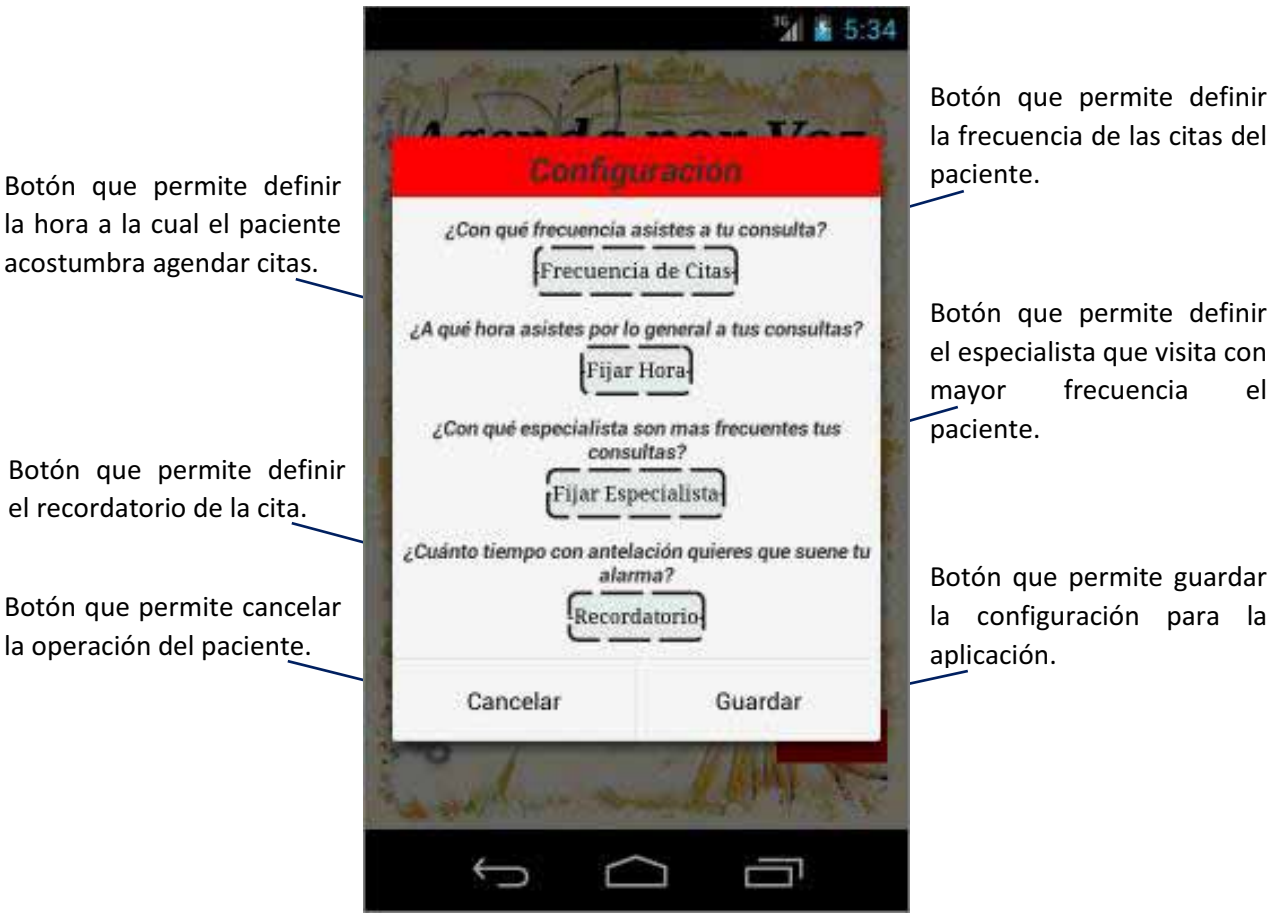


Figura 20. Diálogo de configuración de la aplicación Agenda por Voz Offline AVO

Pantalla para registrar una nueva cita: Esta pantalla muestra el formulario para el registro de una nueva cita (Véase Figura 21); el paciente fija la fecha; luego, fija la hora; después, fija el especialista, y por último, guarda la cita. También está disponible la opción activar el control por voz, la cual tiene como comandos disponibles: `<<dos semanas>>`, `<<un mes>>`, `<<medio año>>`para el ajuste en la fecha; `<<siete>>`, `<<nueve>>` y `<<medio día>>` para el ajuste en la hora; `<<dentista>>`, `<<pediatra>>` y `<<urólogo>>` para el ajuste del especialista; `<<quince antes>>` y `<<treinta antes>>` para ajustar el recordatorio, y por último `<<ayuda>>` que despliega una ventana con todos los comandos disponibles(véase Figura 23).

Figura 21. Pantalla de registro de nueva cita de la aplicación Agenda por Voz Offline AVO

Diálogo de comandos disponibles en la pantalla principal: Este diálogo muestra al paciente los comandos de voz que puede emplear para realizar una operación sobre su agenda. Es posible acceder a este diálogo desde el menú del dispositivo, a través de la opción `ayuda` o con el comando de voz `<<ayuda>>` (Véase Figura 22).

Figura 22. Diálogo de comandos disponibles en la pantalla principal.

Diálogo de comandos disponibles en la pantalla de registro de nueva cita: Este diálogo muestra al paciente los comandos de voz que puede emplear en el registro de nueva cita. Es posible acceder a este diálogo desde el menú del dispositivo, a través de la opción `ayuda` o con el comando de voz `<<ayuda>>` (Véase Figura 23).

Figura 23. Diálogo de comandos disponibles en la pantalla de registro de nueva cita.

6.4. Hardware y software necesario

6.4.1. Hardware para desarrollo de la aplicación

Para la implementación de la aplicación *Agenda por Voz Offline AVO* se utilizó el siguiente Hardware.

Equipo de desarrollo:

- Computadora portátil *Lenovo Ideapad Z570* Procesador *Intel Corei3 2310*, 8GB de memoria *RAM*, Sistema operativo. *Windows 8.1 Pro*.

Dispositivos para pruebas:

- Tablet *Ainol Novo 7 Venus,* Procesador *ATM Quad Core 1.2GHz*, 1GB de memoria *RAM*, Sistema Operativo *Android 4.1.1 JELLY BEAN.*
- Celular *Samsung Galaxy Fame GT-S6820M*, Procesador *ARM 1GHz*, 512MB de memoria RAM, Sistema Operativo *Android 4.1.2 JELLY BEAN.*

6.4.2. Software para desarrollo de la aplicación

- Sistema Operativo Android
- Android SDK para Windows
- PocketSphinx
- SQLite
- Android Studio 1.1.0

6.4.3. Requerimientos mínimos de hardware para el dispositivo móvil

Hardware:

- Procesador ARM mono núcleo de al menos 1 GHz.
- 512 MB de memoria RAM.
- 512 MB de espacio disponible en memoria de almacenamiento interno o externo.
- Pantalla touch.

6.4.4. Requerimientos de software

Software

Android 4.0, Ice Cream Sandwich

7. Resultados

A continuación se detallan los resultados obtenidos del desarrollo de este proyecto:

- Se logró implementar el reconocimiento de voz correctamente en la aplicación Agenda por Voz Offline AVO utilizando la biblioteca `pocketsphinx-android-0.8-nolib.jar` y un modelo de lenguaje basado en el idioma español.
- Se logró implementar una interfaz gráfica intuitiva y atractiva para el usuario, lo cual fue ratificado por siete de diez personas que probaron la aplicación, quienes participaron en una prueba sobre el uso de la aplicación.
- Se implementó el módulo que permite realizar el registro de una nueva cita, el cual tiene los siguientes comandos de voz disponibles:`<<dos semanas>>`, `<<un mes>>`, `<<medio año>>`para el ajuste en la fecha;`<<siete>>`, `<<nueve>>` y `<<medio día>>` para el ajuste en la hora;`<<dentista>>`, `<<pediatra>>` y `<<urólogo>>` para el ajuste del especialista;`<<quince antes>>` y `<<treinta antes>>` para ajustar el recordatorio, y por último `<<ayuda>>` que despliega una ventana con todos los comandos disponibles.
- Se implementó el módulo que permite eliminar una cita, cual tiene los siguientes comandos de voz disponibles:`<<dentista>>`, `<<pediatra>>`, `<<urólogo>>` y `<<todos>>`para realizar el filtrado de las citas por especialista;`<<siguiente>>` y `<<anterior>>` para realizar la paginación, y por último `<<ayuda>>` que despliega una ventana con todos los comandos disponibles.
- Se implementó el módulo que permite modificar una cita, los comandos disponibles para este módulo son:`<<dentista>>`, `<<pediatra>>`, `<<urólogo>>` y `<<todos>>`para realizar el filtrado de las citas por especialista;`<<siguiente>>` y `<<anterior>>` para realizar el paginado, y por último `<<ayuda>>` que despliega una ventana con todos los comandos disponibles.
- Se implementó el módulo que permite consultar una cita, los comandos disponibles para este módulo son:`<<dentista>>`, `<<pediatra>>`, `<<urólogo>>` y `<<todos>>`para realizar el filtrado de las citas por especialista;`<<siguiente>>` y `<<anterior>>` para realizar el paginado, y por último `<<ayuda>>` que despliega una ventana con todos los comandos disponibles.
- Se realizaron pruebas de uso sobre la aplicación las cuales se describen a detalle en el manual técnico del proyecto.

8. Análisis y Discusión de Resultados

Se realizaron pruebas de la aplicación *Agenda por Voz Offline AVO* en diez diferentes dispositivos móviles, de las cuales se pudo extraer información de gran valor para el futuro de la aplicación; dicha información se resume en los siguientes puntos:

- Debido a que la biblioteca `pocketsphinx-android-0.8-nolib.jar`se encuentra en fase de desarrollo, se observaron errores durante el reconocimiento de los comandos de voz, principalmente en ambientes con ruido. Cuando se empleó la aplicación en medio de voces o en medio de música, el reconocedor se confundió y reconoció múltiples comandos a la vez.

- Dado que el procesamiento de los comandos de voz depende de la capacidad de procesamiento del procesador del dispositivo donde está instalada la aplicación, en algunos dispositivos de gama baja, tomó un tiempo de aproximadamente tres segundos la inicialización del reconocedor de voz mientras que en los equipos de gama alta solo tomó entre 1 y 1.5 segundos.

- Algunos comandos de voz no pudieron ser incluidos en la aplicación debido a que la mayoría de los modelos de lenguaje están diseñados para el inglés; mientras que el modelo utilizado en la aplicación es el español; sin embargo, `pocketsphinx` no incluye la totalidad de palabras disponibles en el español.

- Siete de diez de los usuarios que participaron en las pruebas de la aplicación *Agenda por Voz Offline AVO* argumentaron que la interfaz es intuitiva y fácil de utilizar; por lo tanto, consideraron utilizar la aplicación, para programar sus citas médicas.

- Descartando los conflictos con el reconocedor de voz de la aplicación, la mayoría de los usuarios que participaron en las pruebas de la aplicación *Agenda por Voz Offline AVO* estuvieron de acuerdo en que el manual proporciona una introducción rápida a los comandos de voz de la aplicación, para emplearlos inmediatamente en la gestión de la agenda médica.

9. Conclusiones

Las conclusiones del proyecto de desarrollo de la aplicación *Agenda por Voz Offline AVO* tienen como fundamento el cumplimiento de los objetivos presentados en la propuesta del Proyecto de Integración (Véase Tabla 1):

Objetivo	Medida de cumplimiento
Desarrollar una aplicación Android para el registro de citas médicas mediante el empleo de comandos de voz.	Completo
Diseñar e implementar el módulo para el reconocimiento de voz.	Completo
Diseñar e implementar la interfaz de la aplicación.	Completo
Diseñar e implementar el módulo para registrar una cita.	Completo
Diseñar e implementar el módulo para consultar una cita.	Completo
Diseñar e implementar el módulo para modificar una cita.	Completo
Diseñar e implementar el módulo para eliminar una cita.	Completo
Integrar los módulos para administrar una cita.	Completo
Realizar pruebas de la aplicación.	Completo

Tabla 1. Objetivos y estatus de la aplicación

10. Perspectivas del proyecto

Existen mejoras que pueden realizarse a la aplicación *Agenda por Voz Offline AVO*, las cuales se listan y describen a continuación:

* Experimentar el reconocedor de voz Google Voice. Esta API requiere que los dispositivos cuenten con una conexión a la red de internet.
* De concluirse el desarrollo de la biblioteca PocketSphinx, se puede emplear la última versión de la biblioteca, revisando si la capacidad de procesar modelos de lenguaje combinados; es decir que soporte no solo el idioma inglés sino también el español.
* El diseño de la interfaz gráfica se puede mejorar solicitando el apoyo de un profesional del área de Ciencias y Artes para el Diseño de la Universidad. También se puede profundizar en el uso de patrones de diseño de interfaces en aplicaciones móviles.
* Corregir el error que aparece cuando la aplicación hace la validación para evitar que se registre una cita en fechas que ya pasaron; actualmente se despliega dos veces el diálogo de alerta que indica que no puede registrar citas para fechas pasadas; se hizo un análisis del problema pero no se ha logrado determinar si es un problema de lógica o del IDE de desarrollo.

* Desarrollar un módulo que permita identificar las citas que ya se realizaron y eliminarlas automáticamente de la base de datos, esto con la finalidad de reducir el consumo de recursos de almacenamiento del dispositivo.

11. Bibliografía

[1] J. P. Alcántara Olivares y J. D. López Jaimes, *Aplicación Colaborativa para Dispositivos Móviles con Sistema Operativo Android,* Mexico, 2013.

[2] B. A. Chávez Flores y V. Márquez Pérez, *Sistema de Apoyo Clínico para el Tratamiento del Paciente Diabético,* México, 2013.

[3] A. Díaz Mendoza, *SRCV: Sistema computacional interactivo basado en reconocimiento de comandos de voz para aplicaciones educativas,* México, 2012.

[4] N. Mulhern, «Designing Android Applications Using Voice Controlled Commands: For Hands Free Interaction with Common Household Devices,» de *Bioengineering Conference (NEBEC) 39th Annual Northeast*, NEBEC, 2013.

[5] G. A. Juárez Rodríguez, *Proceso de desarrollo independiente de una aplicación móvil Android,* México, 2013.

[6] Naukta, «Asistente Agenda,» 04 Marzo 2014. [En línea]. Available: https://play.google.com/store/apps/details?id=com.nautka.asistenteagenda&hl=es_419. [Último acceso: 12 Abril 2015].

[7] Gerencia Informática de la Seguridad Social, «Seg-Social Cita Previa,» 20 Mayo 2014. [En línea]. Available: https://play.google.com/store/apps/details?id=gov.es.segsocial.citaprevia&hl=es. [Último acceso: 12 Abril 2015].

[8] android.com, «Android, Sistema Operativo Android,» 2014. [En línea]. Available: http://www.android.com/. [Último acceso: marzo 2015].

[9] F. Briano, «Android, Android Architecture,» 15 Noviembre 2007. [En línea]. Available: http://picandocodigo.net/2007/android-la-nueva-plataforma-de-desarrollos-moviles/. [Último acceso: 22 Abril 2105].

[10] android.com, «Android, JELLY BEAN,» 2014. [En línea]. Available: http://www.android.com/versions/jelly-bean-4-3/. [Último acceso: marzo 2015].

[11] CMUSphinxAdministrator, «CMUSphinx, PocketsPhinx,» 2015. [En línea]. Available: http://cmusphinx.sourceforge.net/wiki/tutorialandroid. [Último acceso: marzo 2015].

[12] R. E. Marcelo, «Edudevices, Microcontroladores ARM,» [En línea]. Available: http://www.edudevices.com.ar/download/articulos/comentarios/Microcontroladores_de_

32_bits_ARM.pdf. [Último acceso: marzo 2015].

[13] L. Craig, UML y Patrones Una introducción al análisis y diseño orientado a objetos y al proceso unificado, Madrid: Pearson, 2003.

Apéndice A. Entregable: Listado del API del código fuente desarrollado

Apéndice B. Entregable: Manual de usuario

Apéndice C. Entregable: Memoria de diseño

Apéndice D. Entregable: Manual Técnico