Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería

Reporte Final del Proyecto de Integración

Licenciatura en Ingeniería en Computación

Modalidad: Proyecto de Integración

Prototipo para agendar citas médicas

Alumno:
Mario Antonio Serrano Licea
Matrícula: 210304123

Asesor:
María Lizbeth Gallardo López

Co asesor:
Beatriz Adriana González Beltrán

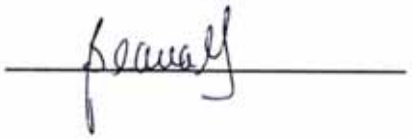
Trimestre 2015-Inverno
Fecha de entrega: 23 de abril de 2015

Yo, María Lizbeth Gallardo López, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Asesor:

Yo, Beatriz Adriana González Beltrán, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

CoAsesor:



Yo, Mario Antonio Serrano Licea, 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.

Alumno:

Resumen

El presente reporte describe el desarrollo de un prototipo para agendar citas médicas. La aplicación que resultó de este proyecto se llama *Medical. MediCal* es una agenda que tiene la característica de ser compartida y opera a través de dispositivos móviles con sistema operativo Android.

Medical fue pensada para pacientes con alguna enfermedad crónica, quienes con frecuencia acuden al médico y generalmente los acompaña un familiar. Esta agenda provee un medio para que el paciente gestione sus citas médicas, y el acompañante conozca en todo momento los movimientos de la agenda médica del paciente.

Medical consta de dos módulos principales: *Medical Server* el cual se encarga de realizar la sincronización de la agenda del paciente con la agenda del acompañante; este módulo opera desde una máquina servidor, y *Medical* el cual se encarga de gestionar las citas médicas localmente, tanto en el dispositivo del paciente como en el dispositivo del acompañante. La gestión de la agenda incluye las operaciones: registrar, modificar, eliminar, consultar y alertar.

Índice

Índice de figuras

Índice de tablas

1. Introducción

En la actualidad los pacientes con enfermedades crónicas, se someten a tratamientos largos que los obliga a tener un control estricto de citas médicas con sus distintos médicos; por ejemplo médico especialista, médico del dolor, médico de nutrición, entre otros. Esto provoca la omisión u olvido de alguna cita necesaria para dicho tratamiento. Este tipo de pacientes siempre debe ir acompañado por algún familiar responsable que pueda aportar información relevante y realizar algunos trámites sencillos.

El objetivo principal del *Prototipo para agendar citas médicas* fue proporcionar una agenda compartida que permita registrar las reuniones con el médico, a través de un dispositivo móvil que cuente con el sistema operativo Android. Esta agenda compartida permite gestionar al paciente sus citas médicas y notificar sobre los cambios en la agenda a las personas con quienes la comparte. La gestión involucra: registrar fecha, hora y lugar específicos para la cita médica, así como una breve descripción del evento (análisis clínicos, pruebas físicas, especialista, estudios médicos); notificar sobre empalmes en las citas, modificar los datos de una cita; eliminar una cita; así como programar una alarma que recuerde el evento. Dado que la agenda es compartida por al menos dos personas, los propietarios de la agenda pueden realizar modificaciones o eliminaciones, la cuales se notificarán a todos los que comparten la agenda.

Como resultado del desarrollo de este proyecto de integración, se tiene una aplicación llamada *Medical*, la cual consta de dos módulos principales: *Medical Server* el cual se encarga de realizar la sincronización de las citas médicas programadas por un paciente en la agenda de su dispositivo móvil, con la agenda de su acompañante en su dispositivo móvil, y *Medical* el cual se encarga de gestionar las citas médicas de manera local en cada dispositivo.

El presente documento está formado por 10 secciones. La sección 2 presenta los antecedentes relacionados al proyecto, tanto referencias externas como internas. La sección 3 muestra la justificación del proyecto. La sección 4 describe los objetivos, general y particulares que se plantearon en el proyecto. La sección 5 muestra el marco teórico. La sección 6 presenta el desarrollo del proyecto, como lo es la metodología, diseño y uso del sistema. La sección 7 presenta los resultados del proyecto. La sección 8 muestra el análisis y discusión de resultados. La sección 9 presenta las conclusiones, y la sección 10 muestra las perspectivas del proyecto.

2. Antecedentes

2.1. Referencias externas

Sistema automático de control y administración de contenido de información [1]**.**

Es una aplicación encargada de dar seguimiento a la información de los investigadores del Instituto de Ecología de la UNAM; reportes, curiculums, etc. Para el manejo de la información se hizo uso de la base de datos Oracle 9ias (*Internet Aplication Server*) como herramienta de la base de datos. La similitud con la propuesta es que ayuda a tener registrada información de las personas que es relevante para ellos. La diferencia es que en la propuesta la información necesaria es reflejada en que los pacientes necesitan saber cuáles han sido las últimas citas que han tenido en un periodo determinado de tiempo.

Smart phone **basado en un calendario de medicamentos, recordatorio y monitoreo** [2]**.**

Esta aplicación sirve para el oportuno control de la toma de los medicamentos correspondientes a una hora determinada previamente; además, registra el número de tomas de un medicamento. La aplicación trabaja para los calendarios disponibles en los teléfonos inteligentes. La semejanza con la propuesta es que de igual forma se trabajara con calendarios. La diferencia es que en esta propuesta se trabajará con el calendario pero para agendar citas médicas, no establece nada sobre los medicamentos.

2.2 Referencias Internas.

1) Aplicación Colaborativa para Dispositivos Móviles con Sistema Operativo Android [3]**.**

Es una aplicación compartida que facilita la colaboración entre miembros de un grupo de trabajo para la edición de figuras geométricas y texto a través de una red punto-a-punto. El parecido de esta aplicación con la propuesta es que funciona para dispositivos con S.O. Android. La diferencia consiste en que la colaboración es para poder editar trabajos, mientras que en la agenda compartida una vez registrada la cita, la agenda no deberá ser cambiada.

2) Sistema Colaborativo para médicos especialistas en medicina alternativa [4]**.**

Es un portal web colaborativo enfocado en médicos titulados, el cual proporciona la información necesaria para la medicina alternativa; la información está basada en casos de pacientes que los médicos han tratado en el pasado. Emplean el lenguaje de programación JAVA[1] y *Web Builder*[2] para el contenido semántico. El parecido con la propuesta es que

[1]Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases.

[2]*WebBuilder* es una suite de productos que sirven como plataforma para el desarrollo de aplicaciones y portales semánticos.

también es un sistema colaborativo, pero la diferencia es que este sistema es visto desde el punto de vista médico, mientras que esta propuesta es vista desde el punto de vista del paciente.

3) Aplicación web de administración de horarios para estudiantes [5]**.**
La aplicación permite a un estudiante organizar sus horarios, tanto de cursos como de actividades extraescolares dentro de un calendario. El usuario puede establecer distintos periodos o ciclos escolares en los que está inscrito, sus materias y las actividades de cada una de ellas. Todos los eventos que gestiona este calendario son individuales y no hay eventos grupales, lo cual se tiene planeado en esta propuesta.

3. Justificación

El paciente con una enfermedad crónica debe poder organizar sus citas médicas de tal manera que no haya empalmes; sin embargo, frecuentemente las citas se anotan en papel y en ocasiones cuando deben registrar otra cita, tanto el paciente como el familiar se olvidan de que ya tienen una cita para ese día y esa hora. El no acudir en la fecha y hora indicadas por el médico puede resultar en una complicación grave en la enfermedad del paciente; por ejemplo, el médico proporciona medicamentos para un determinado periodo de tiempo, y si el paciente no acude a la próxima cita, no podrá contar con los medicamentos apropiados para su tratamiento.

Algunas agendas que existen actualmente en el mercado como son: *Cal*, *EventFlow* o *DigiCal* poseen toda la gestión para organizar nuestro tiempo, pero ninguna de las antes mencionadas posee la capacidad de ser compartida y poder notificar de la reunión a otras personas, característica que se propone en este proyecto. La importancia del *Prototipo para agendar citas médicas* radica en que se buscará facilitar la gestión compartida de las citas, y se propondrán medios visuales que faciliten la observación de las citas programadas.

4. Objetivos

4.1. Objetivo general

Desarrollar una agenda compartida para gestionar citas médicas desde un dispositivo móvil que cuente con sistema operativo Android.

4.2. Objetivos particulares

- Diseñar e implementar un módulo de una agenda médica que incluya las operaciones de: registrar, modificar, eliminar, consultar y alertar sobre las citas médicas.
- Diseñar e implementar un módulo que permita la comunicación y sincronización entre los dispositivos que compartirán la agenda.
- Diseñar e implementar un módulo para la persistencia de los datos que se manejarán en la agenda médica.
- Diseñar e implementar una interfaz gráfica para dispositivos móviles que facilite al usuario la interacción con la agenda médica.

5. Marco Teórico

5.1 Sistema Operativo *Android*

Android es un sistema operativo basado en Linux, fue diseñado para dispositivos móviles tales como Smartphone y tablets. *Android* fue desarrollado por *Android Inc*., empresa que

en 2005 pasó a ser propiedad de Google Inc. La popularidad de este sistema operativo se incrementó a partir del año 2008, año en que Google se une al proyecto de *Open Handset Alliance*, un consorcio formado por 48 empresas de desarrollo hardware, software y telecomunicaciones, a favor de la promoción del software libre. Siendo Google quien ha aportado la mayor parte del código fuente del sistema operativo. A esta causa también se le sumó la ayuda de Apache, fundación que da soporte a proyectos de código abierto.

Android permite programar en un entorno de trabajo (*framework*) de Java. Las aplicaciones son lanzadas 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 (ver *Figura 1*):

- *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*.
- *AndroidRuntime*: 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.

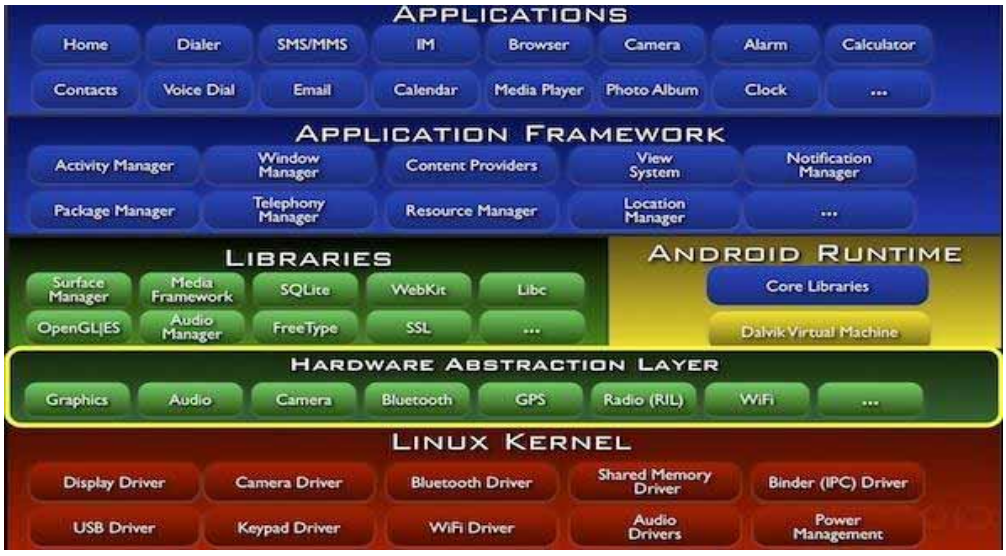


Figura 1. Android architecture.

5.2 Leguaje de programación Java

Java es **un lenguaje de programación orientado a objetos** que se popularizó a partir del lanzamiento de su primera versión comercial, la JDK[3] 1.0 en 1996.Las principales características del lenguaje son: orientado a objetos: encapsulación, herencia, polimorfismo; facilita la creación de aplicaciones distribuidas; es portable, porque es posible ejecutarlo sin importar la arquitectura; multihilo, porque permite la sincronización de múltiples hilos en ejecución.

5.3 IDE *Android* Studio.

Android Studio es un nuevo entorno de desarrollo basado en *IntelliJ IDEA* [6]. Contiene nuevas características y mejoras sobre *Eclipse Android Development Tools* [7]. *Android* Studio es oficialmente el Entorno de Desarrollo Integrado, IDE por sus siglas en inglés: *IntegratedDevelopmentEnvironment* para *Android*. Las funcionalidades de *Android Studio* son:

- *Android Virtual Device (Emulador de dispositivos Android)*: Es una herramienta que permite emular distintos dispositivos *Android.*
- *User Interface (Interfaz de desarrollo visual)*: Facilita el desarrollo porque permite agregar componentes visuales de forma directa en la interfaz de la aplicación.

5.4SQLite y MySQL.

Android integra en su propio API[4]un manejador de bases de datos llamado *SQLite* [8]; en los dispositivos de prueba se cuenta con la versión 3.7.11. *SQLite* es un motor de bases de datos que maneja archivos de tamaño pequeño, no necesita ejecutarse en un servidor, cumple con el estándar *SQL-92* [9] y es de dominio público. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos) son guardados como un único archivo en la máquina host. Por lo tanto, *SQLite* maneja toda la base de datos como único archivo, el cual bloquea al iniciar cada transacción. Esto es, al mismo tiempo su gran virtud y su mayor inconveniente, ya que debido a que bloquea totalmente el archivo de la base de datos en cada transacción dispone de latencias muy bajas, pero también impide múltiples accesos a la base de datos.

MySQL [10] es un sistema de administración relacional de bases de datos. Una base de datos relacional almacena datos en tablas separadas en vez de colocar todos los datos en un

[3]Java Development Kit (JDK), es un software que provee herramientas de desarrollo para la creación de programas en Java.

[4]La interfaz de programación de aplicaciones (API) es el conjunto de subrutinas, funciones y que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

gran archivo. Esto impacta en la velocidad de acceso y proporciona flexibilidad para la modificación. Las tablas mantienen relaciones que hacen posible combinar datos de las diferentes tablas.

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 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 [11].

Para el desarrollo de *MediCal,* 2 semanas constituyen una iteración. En la fase de inicio se realizaron dos iteraciones (I1, I2), 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: Análisis dela lógica del negocio, y definición del alcance del proyecto. Planificación de actividades de acuerdo al calendario escolar. Bosquejo de las interfaces de *MediCal*. Instalación y acondicionamiento del entorno de desarrollo tanto para *MediCal* como para *MediCal Server*.

Elaboración: Definición de la arquitectura basada en dos capas. Identificación de los requisitos, tales como: Registro de las citas médicas en tiempo y forma, eliminación y modificación de las citas registradas, poder compartir las citas con un acompañante, así como 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.

Construcción: Diseño e implementación definitiva de la base de datos, diseño e implementación iterativa de los módulos para los requisitos de *Medical*: `Registro`, `Eliminacion` y `Modificacion` de las citas agendadas por el paciente. Se continuó y finalizó el diseño e implementación de los módulos para *Medical Server:* Registro de usuarios de *MediCal*, Registro de las citas agendadas por los usuarios de *MediCal* `Sincronizacion` de las citas agendadas por el paciente; eliminación, modificación y registro de las citas y la suscripción del acompañante a la aplicación MediCal del paciente.

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 dominio, y modelo de robustez, los cuales sirvieron de base para el desarrollo de *Medical.*

6.2.1 Diagramas de casos de uso

La aplicación *Medical* comprende dos casos de uso principales: `Gestionar Agenda` y `Sincronizar mi agenda con la de un paciente` (ver Figura 2).

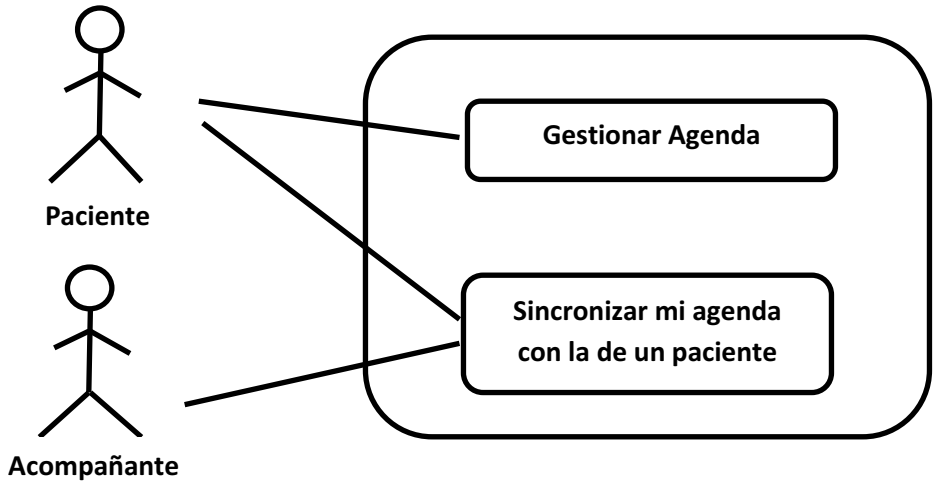


Figura 2. Diagrama de Casos de Uso General

`Gestionar Agenda`: Se encarga de administrar todas las operaciones de la agenda, a saber: Agendar Cita, Buscar Citas, Consultar Cita, Eliminar Cita y Modificar Cita. El paciente podrá gestionar sus citas médicas a través del calendario de la aplicación.

`Sincronizar mi agenda con la de un paciente`: Se encarga de manejar los requerimientos para sincronizar las agendas, la visualización se logra a través de un calendario. El acompañante podrá mantener, en su propio calendario, las citas agendadas por el paciente.

A continuación se explican los casos de uso correspondientes al caso uso Gestionar agenda (ver Figura 3).

Agendar Cita: Programa una cita en la agenda, en el dispositivo móvil, y la registra en *MediCal* y *Medical Server*; además, realiza la sincronización de la agenda del acompañante con la del paciente.

Buscar Citas: Busca citas agendadas, a partir de un intervalo de fechas definido por el paciente o por el acompañante.

Consultar Cita**:** Consulta el detalle de una cita, a partir de la búsqueda realizada previamente, visualizando los datos completos: Fecha de la cita, hora de la cita, tipo de cita, lugar de la cita y el recordatorio de la cita

Eliminar Cita**:** Elimina una cita, a partir de la búsqueda realizada previamente, mostrando un cuadro de dialogo para confirmar la acción.

Modificar Cita**:** Modifica los datos de una cita, a partir de la búsqueda realizada previamente.

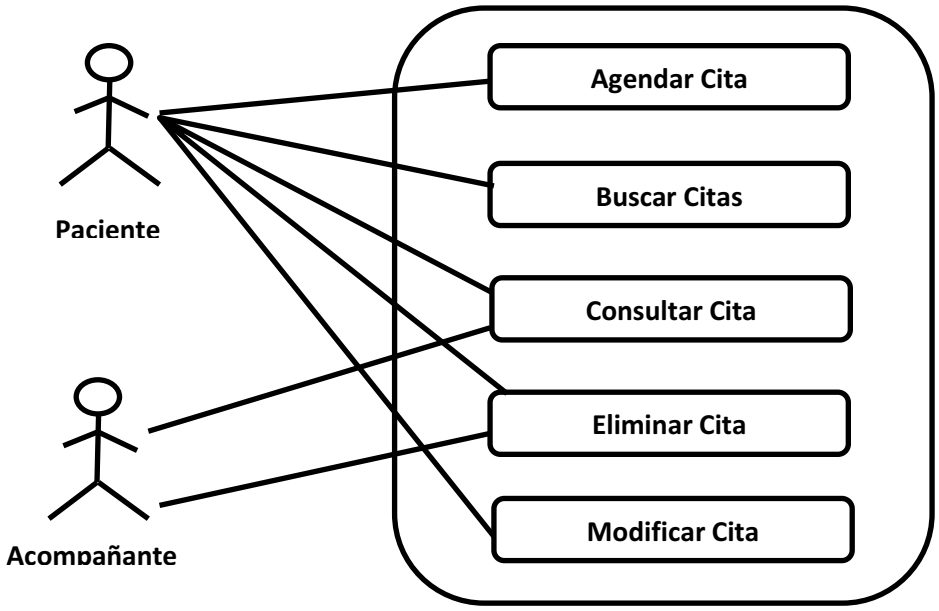


Figura 3. Casos de uso particulares para `Gestionar Calendario`

A continuación se explican los casos de uso correspondientes al caso uso Sincronizar mi agenda con la de un paciente (Figura 4), a saber: Registrarse en *MediCal Server* y Suscribirse a una agenda.

Registrarse en *Medical Server***:** Registra a los actores paciente y acompañante en el servicio de *Google Cloud Messaging*. El paciente deberá registrarse en *MediCal Server* con el fin de que sus citas sean registradas en el servicio de Google. El acompañante deberá registrar separa posteriormente suscribirse a la agenda del paciente.

Suscribirse a una agenda**:** Permite que un acompañante se registre en la agenda de un paciente para recibir notificaciones sobre los movimientos que el paciente realice sobre suscitas médicas. Además de recibir las notificaciones, la agenda del acompañante se actualizará con las modificaciones que el paciente realice. Cabe mencionar que una persona concreta puede ser un acompañante y, a su vez, ser un paciente.

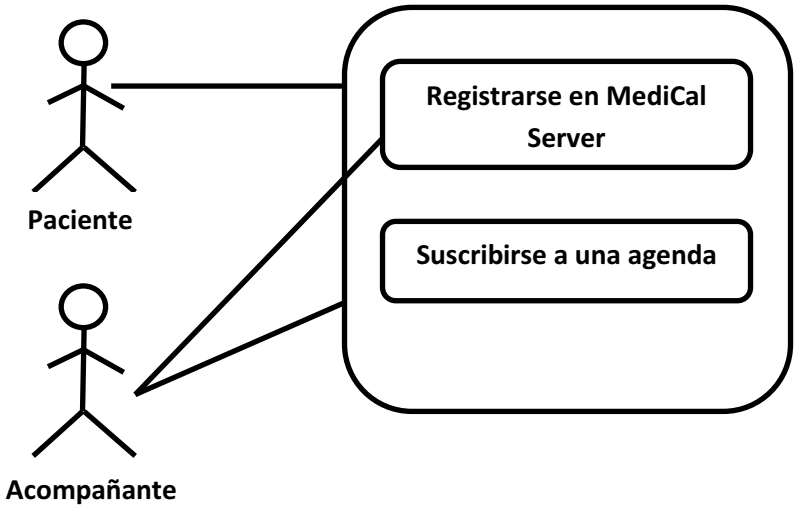


Figura 4. Casos de uso particulares de sincronizar mi agenda con la de un paciente

6.2.2 Casos de uso de texto

Tomaremos los casos de uso `Registrarse en Medical Server`, `Suscribirse a una agenda` y `Agendar cita` para ejemplificar, a lo largo del documento, las diferentes etapas del diseño. Los detalles del resto de los casos de uso de las Figuras 2 y 3 se encuentran en la memoria de diseño del sistema.

CU01: Registrarse en MediCal Server

1.- CU01: Registrarse en *MediCal Server*

2.- Personal involucrado e intereses:

- Paciente: Desea registrarse en *MediCal Server* para que sus citas puedan ser notificadas al acompañante.
- Acompañante: Desea registrarse en *MediCal Server* para recibir notificaciones de la agenda de un paciente específico.

3.- Precondiciones:

La aplicación *MediCal* debe estar instalada en el teléfono.
Contar con una cuenta de google instalada en el teléfono.
Tener instalado Play Store en el teléfono.
Contar con internet al momento de realizar el registro.

4.- Garantías de éxito (Pos condiciones):

El Paciente o el Acompañante fue registrado satisfactoriamente en el servidor.

5.- Flujo(s) Básico(s):

1. El paciente selecciona una cuenta de correo electrónico Gmail, aquella que esté configurada en el teléfono.
2. El paciente elige la opción Registrarse.
3. La aplicación *MediCal* despliega un mensaje indicando que los datos se están enviando a la aplicación *MediCal Server*.
4. La aplicación *MediCal Server* registra la cuenta de Gmail seleccionada, asignándole un único código de identificación.
5. La aplicación *MediCal* recibe una notificación indicando que el registro fue correcto.
6. La aplicación *MediCal* despliega un mensaje indicando que se ha registrado satisfactoriamente.
7. El paciente elige aceptar
8. La aplicación *MediCal* despliega el calendario, el cual constituye la pantalla principal de la aplicación.

6.- Extensiones (o Flujos Alternativos):

a* Si en el momento del registro, el teléfono no cuenta con internet se mostrará un mensaje avisando al usuario que debe contar con internet para realizar la operación. No se podrá realizar la operación hasta que se cuente con internet en el dispositivo y la información no será guardada localmente en *MediCal*.

3a. El paciente elige cancelar la operación registrarse.

1. La aplicación *MediCal* mostrará un mensaje indicando que la operación fue cancelada. Los datos del registro no serán mandados a *MediCal Server* y la información de registro no será guardada localmente.

CU02: Suscribirse a una agenda

1.- CU02:suscribirse a una agenda.

2.- Personal involucrado e intereses:

- Acompañante: Poder suscribirse a la agenda de su paciente para sincronizar con la suya las citas que realice el paciente.

3.- Precondiciones:

Haber cumplido con éxito el caso de uso Registrarse en *Medical Server*.
Contar con internet al momento de la suscripción.

4.- Garantías de éxito (Pos condiciones):

El Acompañante fue suscrito satisfactoriamente al calendario de otro usuario.

5.- Flujo(s) Básico(s):

1. El acompañante ingresa al menú desde la pantalla principal o en su defecto a través de los tres puntos situados en la esquina superior derecha de la aplicación *MediCal*.
2. El acompañante elige la opción Suscribirse a un calendario.
3. La aplicación *MediCal* hace una solicitud a la aplicación *Medical* Server para obtener la lista de pacientes registrados en *MediCal.*
4. La aplicación MediCal Server envía la lista de pacientes registrados.
5. La aplicación *MediCal* despliega la lista de pacientes registrados.
6. El acompañante escoge un paciente.
7. La aplicación *Medical* envía una solicitud de suscripción a la agenda del paciente seleccionado a través del botón `Enviar solicitud`.
8. La aplicación *Medical Server* recibe la solicitud, y envía una notificación sobre la solicitud a la aplicación *MediCal* del paciente.
9. El paciente acepta la solicitud y la aplicación *Medical* envía la respuesta a *Medical Server*.
10. *Medical Server* renvía la aceptación a la aplicación *Medical* del Acompañante,
11. El Acompañante recibe la notificación de aceptación.
12. La aplicación *MediCal Server* envía la lista de citas registradas en la agenda del paciente.

6.- Extensiones (o Flujos Alternativos):

a* Si en el momento de la suscripción el teléfono no cuenta con internet se mostrará un mensaje indicando que debe contar con internet para realizar la operación. No se podrá realizar la operación hasta que se cuente con internet en el dispositivo y la información no será guardada localmente en *MediCal*.

7a. La operación fue cancelada, al momento de mandar la solicitud:

1. Si el acompañante se encuentra dentro de la aplicación mostrará un mensaje indicando que la operación fue cancelada sin mandar la solicitud de suscripción. Si se sale de la aplicación la solicitud no será enviada. Se mostrará un mensaje al acompañante indicando que la operación fue cancelada y por lo tanto la solicitud no será enviada.

9a. El paciente rechaza la solicitud enviada por el acompañante

1. El acompañante recibirá una notificación indicando que su solicitud fue rechazada.

CU03: Agendar Cita

1.- CU03: Agendar Cita

2.- Personal involucrado e intereses:

- Paciente: Quiere agendar una cita médica, y que ésta quede almacenada tanto en el teléfono como en el teléfono de su acompañante.
- Acompañante: Quiere recibir la cita agendada por el paciente en la agenda de su teléfono.

3.- Precondiciones:

El Paciente elige la opción `Agendar` desde la ventana principal de la aplicación o mantiene una pulsación largar sobre algún día en el calendario y elige la opción Agendar Cita.

4.- Garantías de éxito (Pos condiciones):

Se agenda la cita médica, esta queda registrada en el teléfono a través de la aplicación *MediCal*, y en el servidor a través de la aplicación *MediCal Server*.

5.- Flujo(s) Básico(s):

1. El Paciente establece la fecha de la cita médica a través de la opción `Fecha de la cita`.
2. El Paciente establece la hora de la cita médica a través de la opción `Hora de la cita`.
3. El Paciente establece el tipo de cita a través de la opción `Tipo de cita`.
4. El Paciente establece el lugar de la cita médica a través de la opción `Lugar de la cita`.
5. El Paciente establece el tiempo que desea para que se le recuerde la cita médica a través de la opción `Recordatorio`.
6. El Paciente agenda la cita haciendo uso de la opción `Hecho`.
7. La aplicación *Medical* registra la cita en el teléfono
8. La aplicación *Medical* envía la cita a la aplicación *MediCal Server.*
9. La aplicación *MediCal Server* registra la cita en el servidor remoto.
10. La aplicación *Medical* coloca un icono en la vista del calendario, en la fecha de la cita.
11. La aplicación *MediCal Server* envía la cita a la aplicación *MediCal* del acompañante.
12. La aplicación *Medical* del acompañante despliega una notificación del envío.
13. El Acompañante elige aceptar.
14. La aplicación *Medical* del acompañante registra la cita localmente.
15. La aplicación *Medical* del acompañante marca en el calendario la cita recibida.

6.- Extensiones (o Flujos Alternativos):

a* En cualquier momento el Paciente puede presionar el botón “Atrás” del teléfono para abandonar el registro de una cita médica.

1a. `Fecha de la cita` no válida:

1. La aplicación encuentra que la fecha de la cita es una fecha ya pasada y no permite que se realice el registro de la cita médica hasta que se proporcione una fecha a partir del día en que intenta el registro de la cita.

2a. `Hora de la cita` no válida:

1. La aplicación encuentra que la hora de la cita es una hora que ya pasó y no permite que se realice el registro de la cita médica hasta que se proporcione una hora a partir de la hora en que se intenta el registro de la cita.

4a. La opción `Lugar de la cita` está vacío:

1. La aplicación encuentra que el lugar de la cita está vacío y no permite que se realice el registro de la cita médica hasta que se proporcione un lugar de la cita médica.

6a. El teléfono no cuenta con internet al momento de `Agendar` la cita.

1. Se muestra un aviso diciendo que necesita conexión a internet para poder realizar la operación. Los datos no se guardaran localmente ni serán enviados a *Medical Server*.

6.2.3 Modelo del dominio

El modelo de dominio para *Medical* se muestra en la `Figura 5`, donde las principales entidades son:

Un `Paciente` puede agendar `Citas Medicas` y, a su vez, puede compartir su `Calendario` con uno o más `Acompañantes`, este a su vez, se puede suscribir al calendario (la cual representa la agenda) que el Paciente comparte. Ambos, tanto el paciente como el acompañante, pueden consultar la `Descripcion de las citas` agendadas.

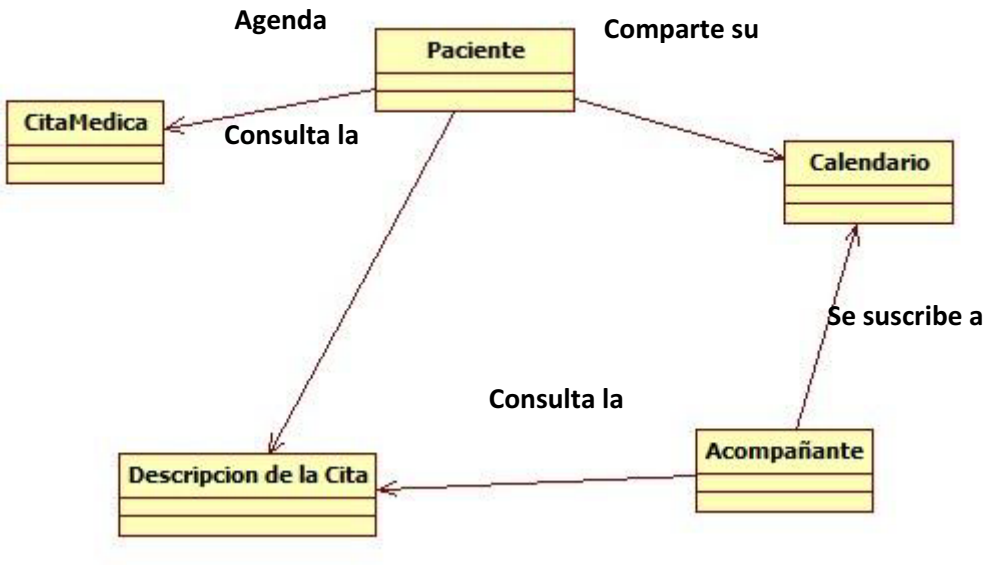


Figura 5. Modelo de dominio de la aplicación *MediCal*

6.2.4 Diagramas de paquetes y diagrama de clases.

6.2.4.1 Descripción del diagrama de paquetes para la aplicación *Medical*

La aplicación *Medical* está compuesta por 5 paquetes (ver Figura 6). El paquete `com.example.too.myapplication` contiene las clases controlador que están a la espera de cualquier evento que ocurra cuando el usuario interactúe con el dispositivo. El paquete `BaseDeDatos` contiene las clases dedicadas a actualizar y consultar la base de datos de la aplicación. El paquete `clases` contiene las clases del modelo para gestionar los datos de las citas médicas. El paquete `Sincronizacion` contiene las clases para realizar la sincronización de la aplicación *Medical* tanto del paciente como del acompañante, y el paquete `adaptadores` contiene las clases encargadas de visualizar los datos de las citas que se consultan.

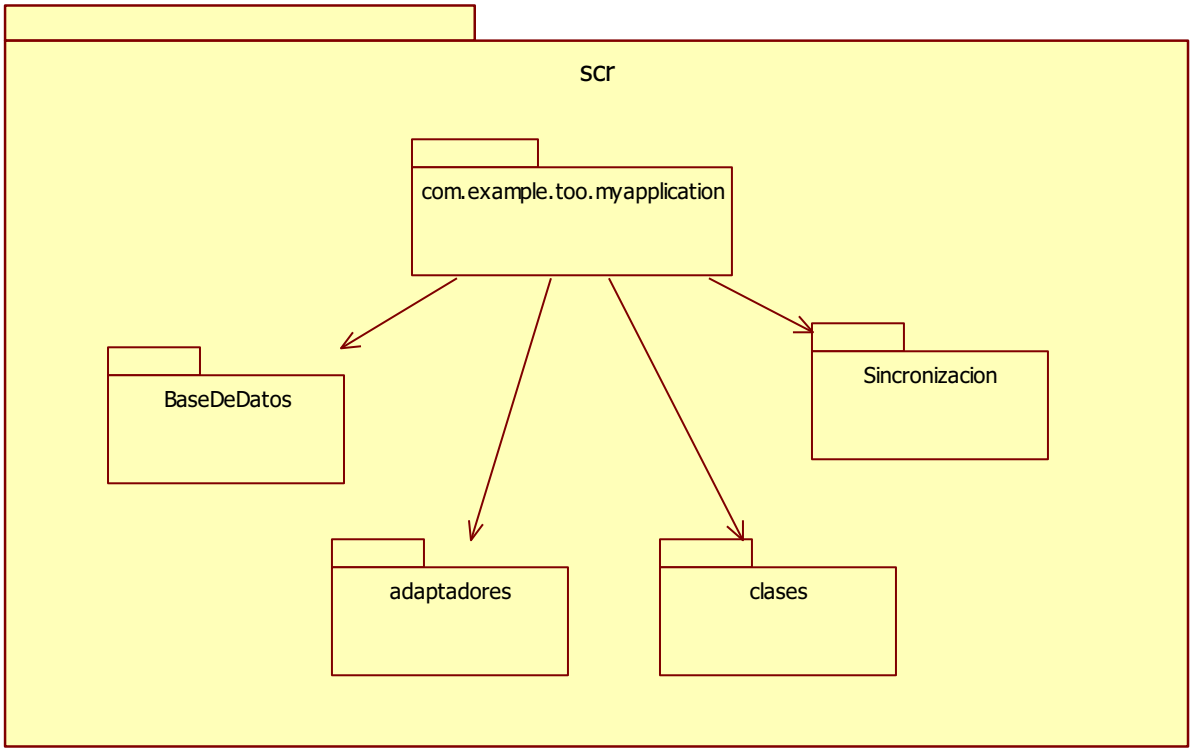


Figura 6.Diagrama de paquetes

Diagrama de clases del paquete:com.example.too.myapplication

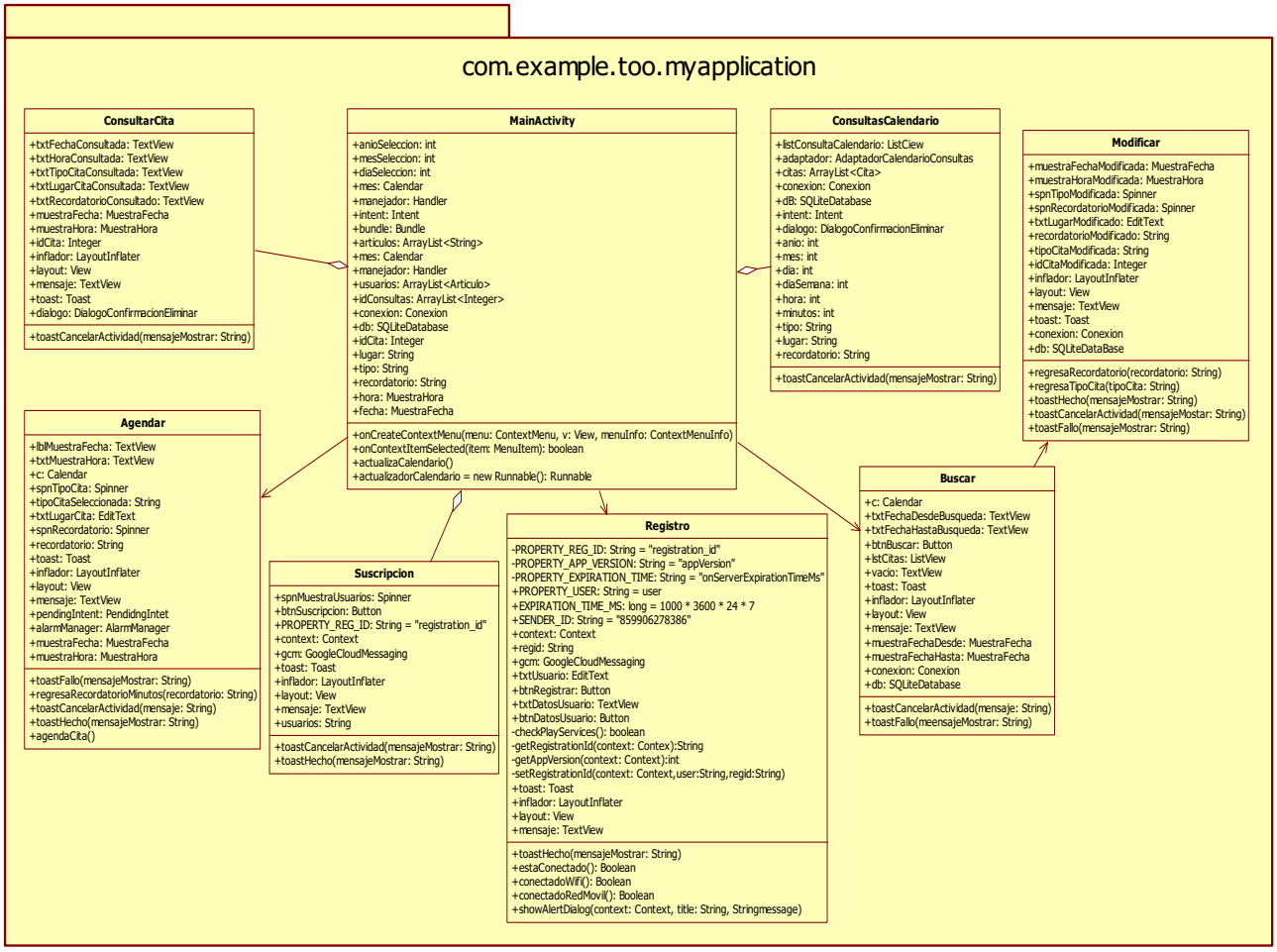


Figura 7. Diagrama de clases del paquete com.example.too.myapplication

La Figura 7muestra el diagrama de las clases controlador. El punto de entrada para la aplicación *Medical* es la *Activity* MainActivity.

La explicación de sus entidades y su relación se hace a continuación:

La clase `Agendar` hereda de ActionBarActivity, se encarga de agendar una cita de manera local, así como de manera remota en Medical Server. También, lleva a cabo la sincronización con los teléfonos que comparten la agenda. Contiene un objeto de la clase `Conexion` para hacer la persistencia de las citas agendadas de manera local. Además contiene un objeto de la clase `OperacionesServidor` para agendar la cita en Medical

Servidor y de esta manera poder realizar la sincronización con los teléfonos que compartan la agenda.

La clase `ConsultarCita` herede de ActionBarActivity, se encarga de realizar las consultas de la citas agendadas, esto de manera local.

La clase `MainActivity` hereda de ActionBarActivity, es la ventana principal que muestra un calendario, en ella se selecciona la fecha en el que se quiere agendar o consultar una cita agendada, a través de una pulsación larga sobre la fecha específica.

La clase `Modificar` hereda de ActionBarActivity, se encarga de modificar citas agendadas en el teléfono, esto de manera local.

La clase `Buscar` hereda de ActionBarActivity, se encarga de buscar citas agendadas en el teléfono a partir de un intervalo de fechas, esto de manera local. Contiene uno o más objetos de la clase `Cita` para almacenar cada cita encontrada en un objeto de este tipo. Contiene un objeto de tipo `AdaptadorCitasBuscadas` para visualizar correctamente las citas encontradas.

La clase `Registro` hereda de ActionBarActivity, se encarga de realizar el registro en el servidor de Google Cloud Messaging, para recibir notificaciones sobre las citas agendadas. Contiene un objeto de tipo `OperacionesServidor` para que tanto el paciente como el acompañante puedan registrarse en el *Medical Server* y. Con estos registros *Medical Server* podrá realizar la sincronización de la agenda del paciente con la agenda del acompañante.

La clase `Suscripcion` hereda de ActionBarActivity, se encarga de realizar la suscripción de un acompañante a la aplicación *MediCal* de un paciente. Contiene un objeto de tipo `OperacionesServidor` para registrar la cuenta de correo electrónico a Medical Server, el cual administra la sincronización.

La clase `ConsultasCalendario` se encarga de visualizar las consultas de las citas médicas a través de una pulsación larga en alguna fecha del calendario en la `MainActivity`, esto de manera local.

Diagrama de clases del paquete: BasesDeDatos.

La Figura 8muestra el diagrama de la clase encargada de realizar la conexión con la base de datos de la aplicación *MediCal.*

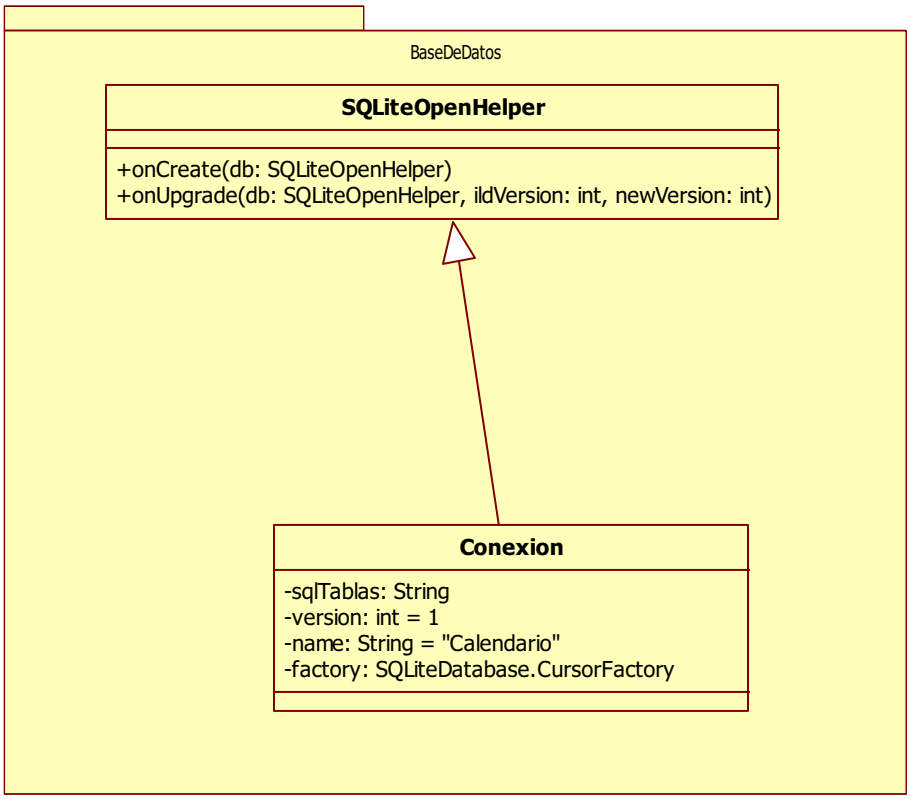


Figura 8. Diagrama de clases del paquete BasesDeDatos

La explicación de las entidades y sus relaciones se hace a continuación:

La clase `Conexion` se encarga de administrar las operaciones CRUD, acrónimo de *Create, Read, Update and Delete*(en español crear, recuperar, actualizar y borrar) de la base de datos local. Hereda de la clase `SQLiteOpenHelper` para conectarse con la base de datos del teléfono.

La clase `SQLiteOpenHelper` se encarga de realizar la conexión con la base de datos local.

Diagrama de clases del paquete: adaptadores.

La Figura 9muestra el diagrama de las clases encargadas de visualizar las citas consultadas.

La explicación de las entidades y su relación entre si se hace a continuación:

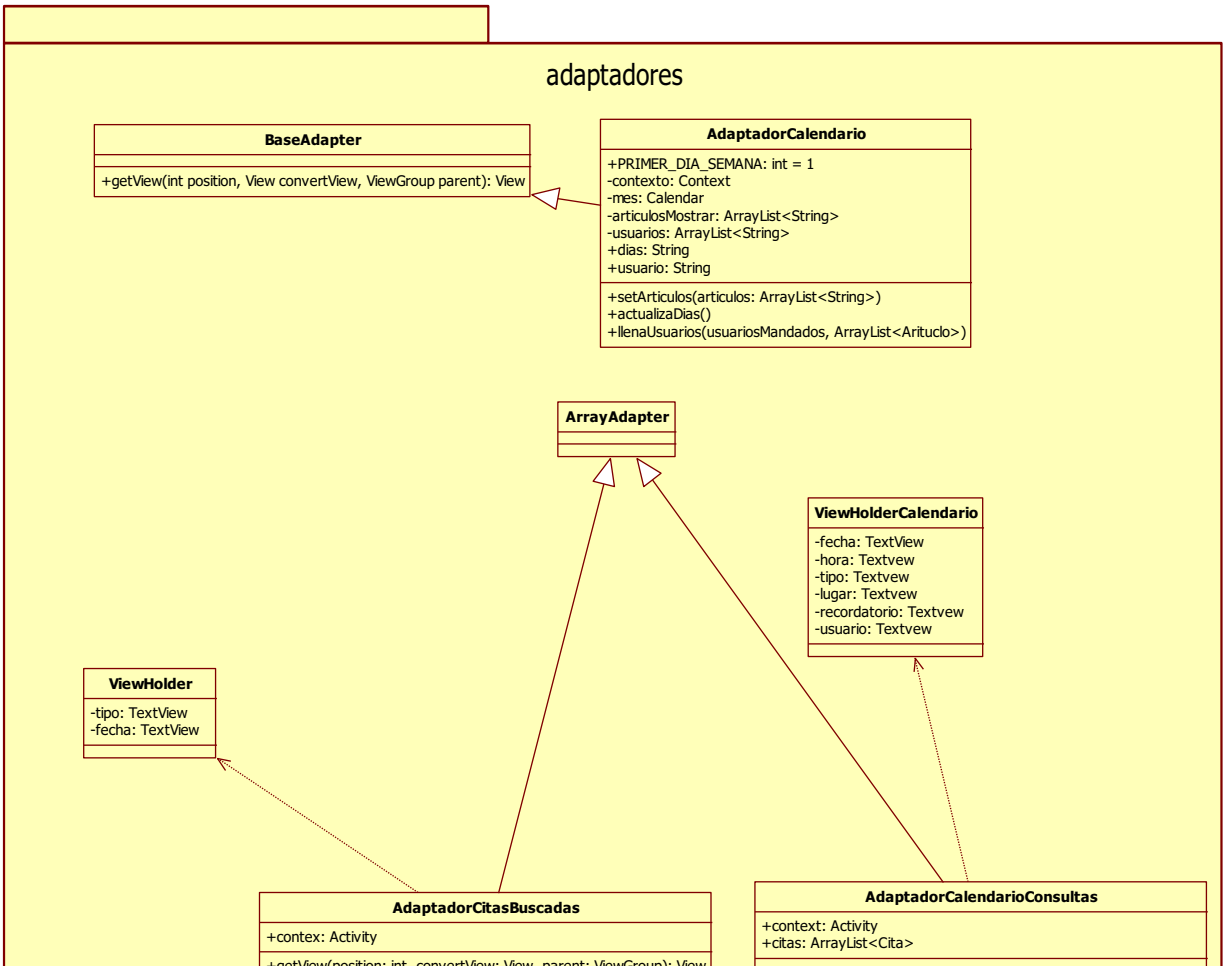


Figura 9. Diagrama de clases del paquete adaptadores.

La clase `AdaptadorCitasBuscadas` se encarga de crear un adaptador que contenga diferentes citas encontradas, para que éstas puedan ser visualizadas adecuadamente. Hereda de la clase `ArrayAdapter` y es dependiente de la clase `ViewHolder` para mostrar los datos de las citas.

La clase `ArrayAdapter` se encarga de crear un adaptador para visualizar los datos de manera correcta en el teléfono.

La clase `AdaptadorCalendarioConsultas` se encarga de crear un adaptador para visualizar los datos de una cita de manera correcta luego de una pulsación larga en alguna fecha de la vista del calendario

La clase `ViewHolderCalendario` se encarga de almacenar los datos que serán mostrados a través de la clase `AdaptadorCalendarioConsultas`.

La clase `ViewHolder` administra los datos de una cita para que sean mostrados en la clase `AdaptadorCitasBuscadas`.

La clase `AdaptadorCalendario` se encarga de administrar las operaciones disponibles en la vista del calendario, haciendo uso de un objeto de tipo `BaseAdapter`.

La clase `BaseAdapter` se encarga de administrar la lista de los días visibles en la vista del calendario.

Diagrama de clases del paquete: clases.

La Figura 10muestra el diagrama de las clases encargadas del modelo puesto que administran los datos de las citas agendadas.

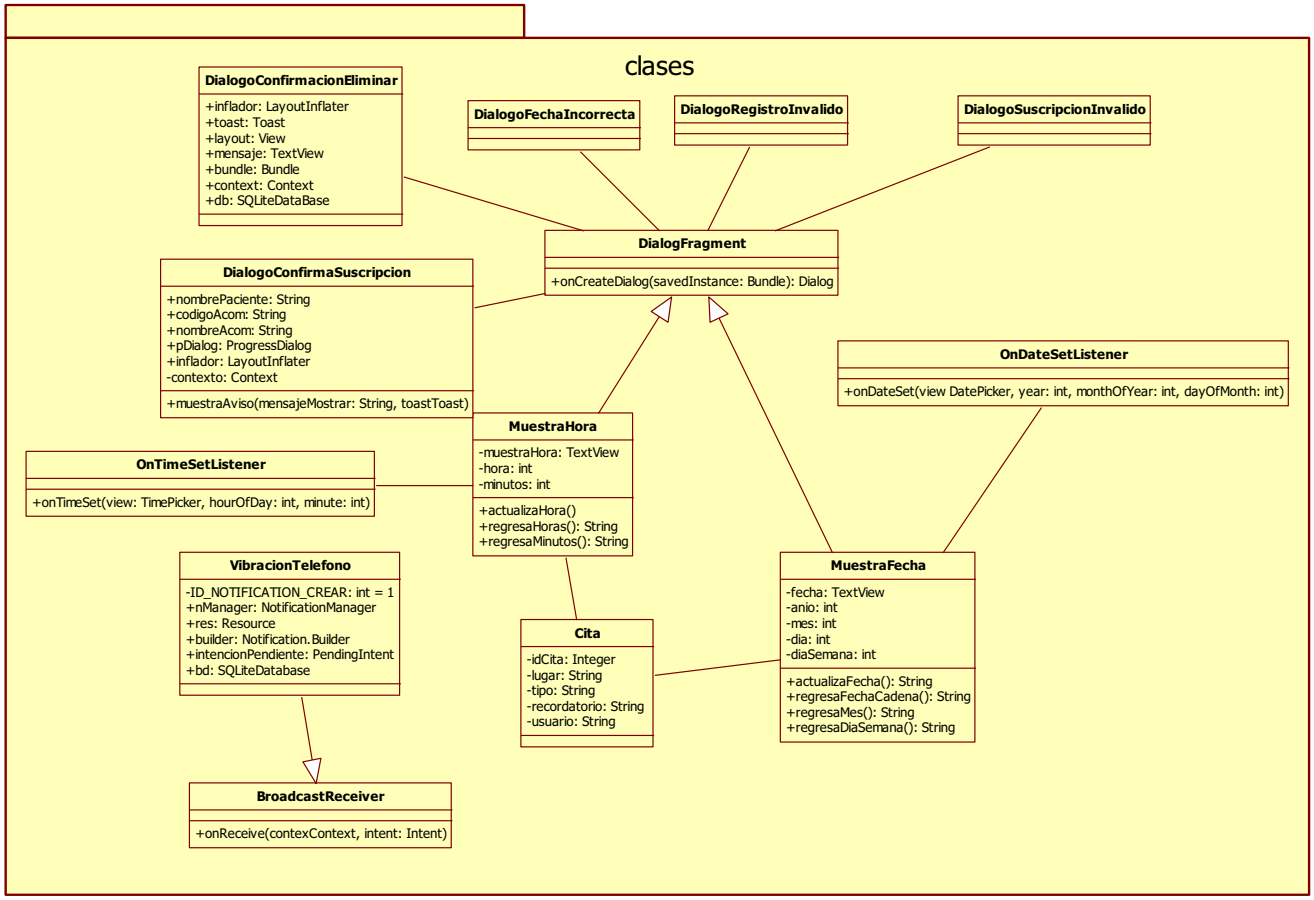


Figura 10. Diagrama de clases de paquete clases

La explicación de las entidades y su relación entre si se hace a continuación:

La clase `Cita` es una clase contenedora, almacena los datos de las citas agendadas. Contiene un objeto `MuestraFecha` para seleccionar y modificarla fechas de cada cita, así como un objeto de tipo `MuestraHora` para seleccionar y modificar la hora de la cita.

La clase `MuestraFecha` es una clase contenedora, almacena los datos de las fechas agendadas en las citas, año, mes y día. Implementa la interfaz `OnDateSetListener` para poder recuperar los datos del año, mes y día de la fecha seleccionada, a través de un

cuadro de dialogo; la creación del cuadro de dialogo se realiza gracias a que hereda de la clase `DialogFragment`.

La clase `OnDateSetListener` se utiliza para recuperar los datos de la fecha elegida en un cuadro de dialogo.

La clase `MuestraHora` es una clase contenedora, almacena los datos de la hora para las citas agendadas, hora y minutos. Implementa la interfaz `OnTimeSetListener` para recuperar los datos de la hora y minutos seleccionados, a través de un cuadro de dialogo, la creación del cuadro de dialogo se realiza gracias a que hereda de la clase `DialogFragment`.

La clase `OnTimeSetListener` se utiliza para recuperar los datos de la hora elegida en un cuadro de dialogo.

La clase `DialogFragment` se utiliza para crear y mostrar los cuadros de dialogo donde se eligen tanto la fecha como la hora de la cita.

La clase `DialogoConfirmacionEliminar` se encarga de mostrar un cuadro de dialogo que pide la confirmación de la eliminación de una cita. Contiene un objeto de tipo `DialogFragment` para crear el cuadro de dialogo, así como un objeto de tipo `Conexion` para borrar los datos de la cita dentro de la base de datos local.

La clase `DialogoConfirmarSuscripcion` se encarga de mostrar un cuadro de dialogo que pide la confirmación o rechazo de una suscripción por parte de un acompañante. Contiene un objeto de tipo `DialogFragment` para crear el cuadro de dialogo.

La clase `DialogoFechaIncorrecta` se encarga de mostrar un cuadro de dialogo informando que no se pueden agendar citas en esa fecha en caso de que se quiera agendar a través de una pulsación larga en la vista del calendario. Contiene un objeto de tipo `DialogFragment` para poder crear el cuadro de dialogo.

La clase `DialogoRegistroInvalido` se encarga de mostrar un cuadro informando que no se puede registrar si ya se ha registrado anteriormente. Contiene un objeto de tipo `DialogFragment` para poder crear el cuadro de dialogo.

La clase `DialogoSuscripcionInvalido` se encarga de mostrar un cuadro de dialogo informando que no se puede suscribir a un calendario si no se ha registrado previamente. Contiene un objeto de tipo `DialogFragment` para poder crear el cuadro de dialogo.

La clase `VibracionTelefono` se encarga de administrar las notificaciones para los recordatorios de las citas agendadas. Hereda de la clase `BroadcastReceiver` ya que es

a través de esta que el teléfono podrá recibir las notificaciones de las alarmas. Contiene un objeto de tipo `Conexion` para recuperar los datos que se necesitan para poder visualizar la cita al momento de que la alarma suene.

La clase `BroadcastReceiverse` encarga de administrar los recursos del teléfono para que pueda recibir la notificación del recordatorio.

La Figura 11 presenta el diagrama de clases completo de la aplicación, así como las relaciones entre las entidades.

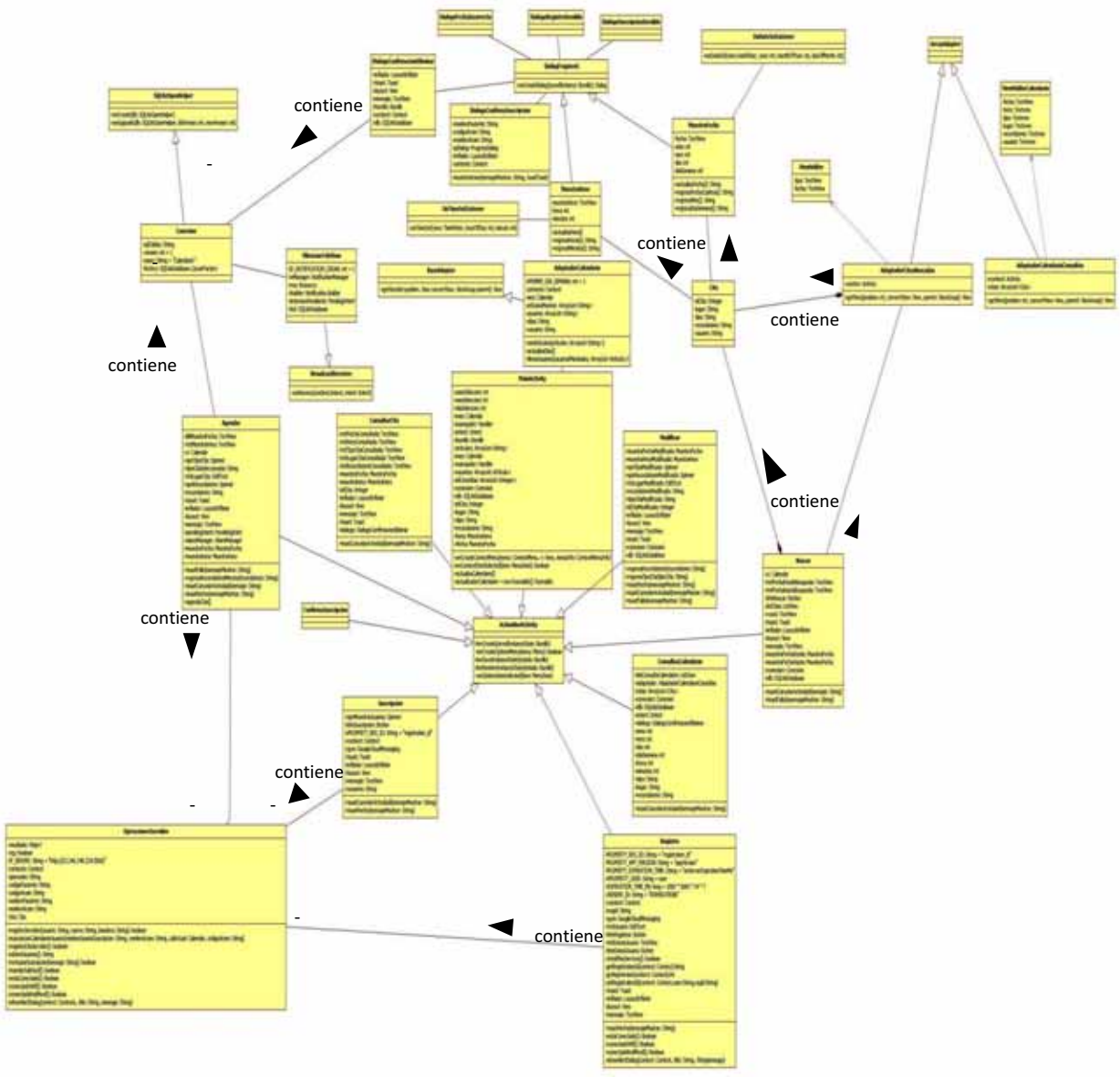


Figura 11. Diagrama de clases de la aplicación *MediCal*

Diagrama de clases del paquete: Sincronizacion.

La Figura 12muestra el diagrama de la clase encargada de la sincronización entre aplicaciones *MediCal*.

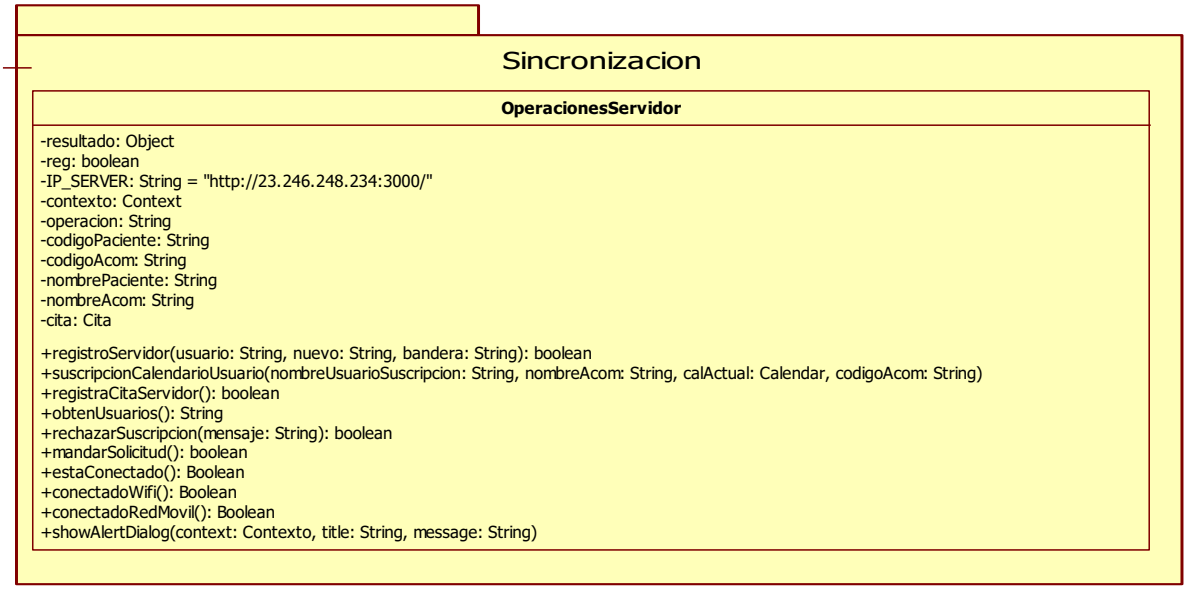


Figura 12. Diagrama de clases del paquete Sincronizacion

Las entidades y sus relaciones se explican a continuación:

La clase `OperacionesServidor` se encarga de administrar todas las operaciones pertenecientes a la sincronización de los dispositivos móviles a partir de los cuales se desea compartir una agenda, tales como:`registroServidor()`, el cual registra a un paciente o acompañante a la aplicación *MediCal Server* o `registraCitaServidor()`, el cual se encarga de registrar una cita agendada por el paciente en el servidor *MediCal Server*.

6.2.4.2 Descripción del diagrama de paquetes para la aplicación *Medical Server*

El diagrama de clases paquetes de la aplicación *Medical Server*s e muestra en la Figura 13.

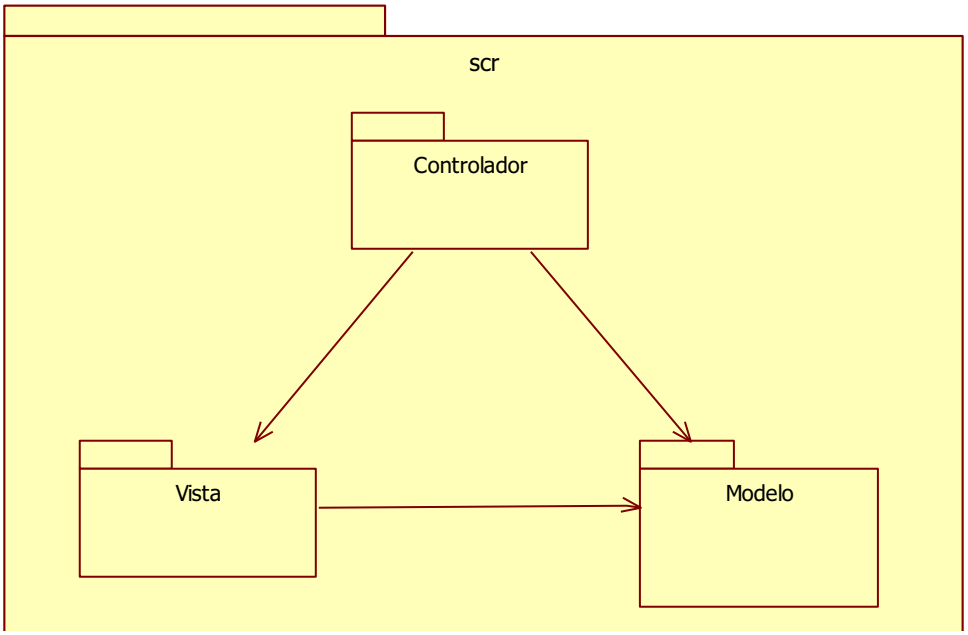


Figura 13. Diagrama de paquetes de MediCal Server

Diagrama de clases del paquete: Controlador

La Figura 14 muestra el diagrama de clases del paquete Controlador, la descripción de sus entidades se describe a continuación:

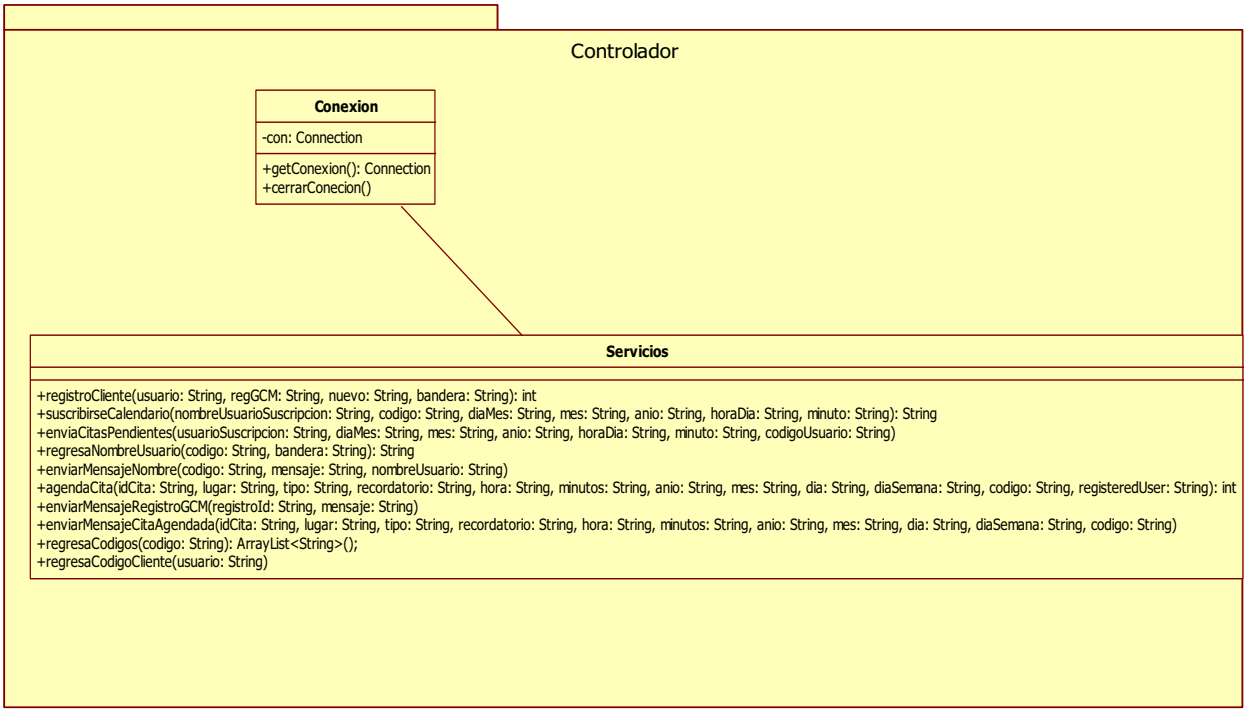


Figura 14. Diagrama de clases del paquete Controlador

La clase `Conexion` se encarga de crear una conexión a la Base de Datos (BD) del servidor, a través de ella es que la clase Servicios puede acceder a la BD.

La clase *`Servicios`* se cuenta con las operaciones: `registroCliente`(), `suscribirseCalendario`(), `agendaCita`().

Diagrama de clases del paquete: Vista

La Figura 15muestra el diagrama de clases del paquete Vista.

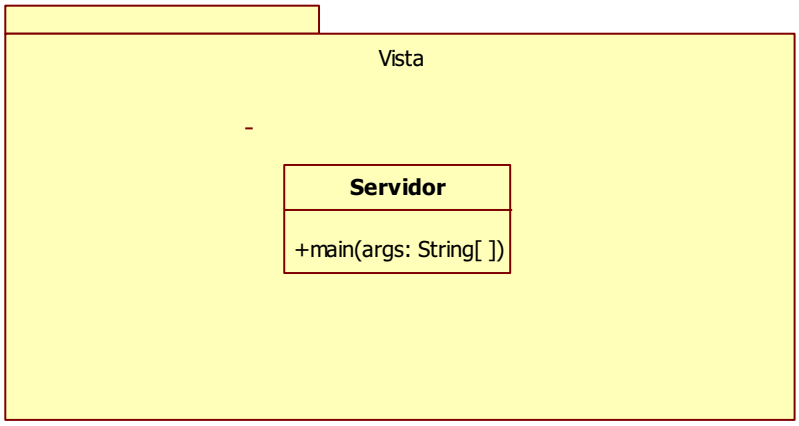


Figura 15. Diagrama de clases del paquete Vista.

La clase Servidor se encarga de realizar el llamado a una UI (*User Interface*), a través de la cual se llama a la clase controladora *Servicios*.

Diagrama de clases del paquete: Modelo

La Figura 16muestra el diagrama de clases del paquete Modelo.

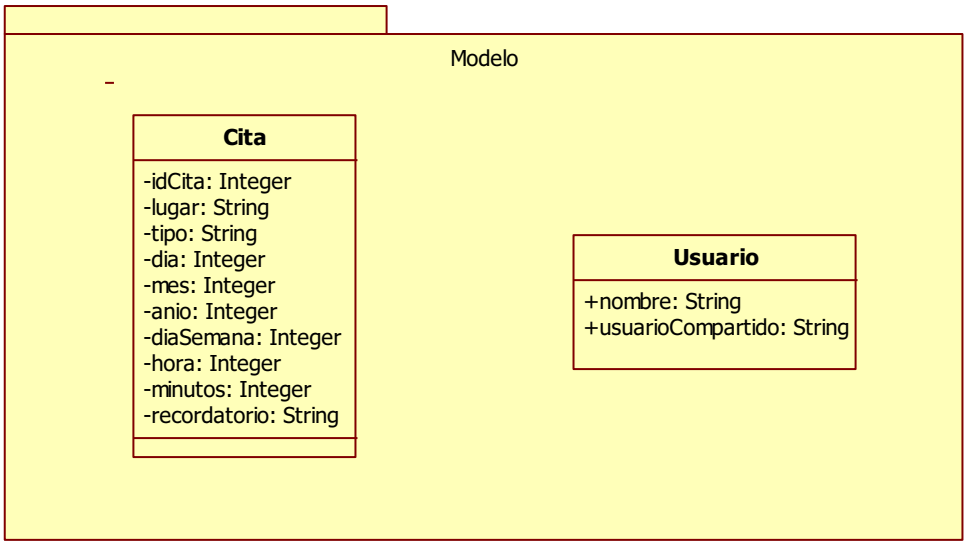


Figura 16. Diagrama de clases del paquete Modelo.

`Cita` es una clase contenedora que es utilizada para almacenar los datos de las citas agendadas desde la aplicación *MediCal.*

`Usuario` es una clase contenedora usada para recuperar los datos de la tabla usuario en la BD.

El diagrama de clases de la aplicación *MediCal Server* se muestra en la Figura 17.

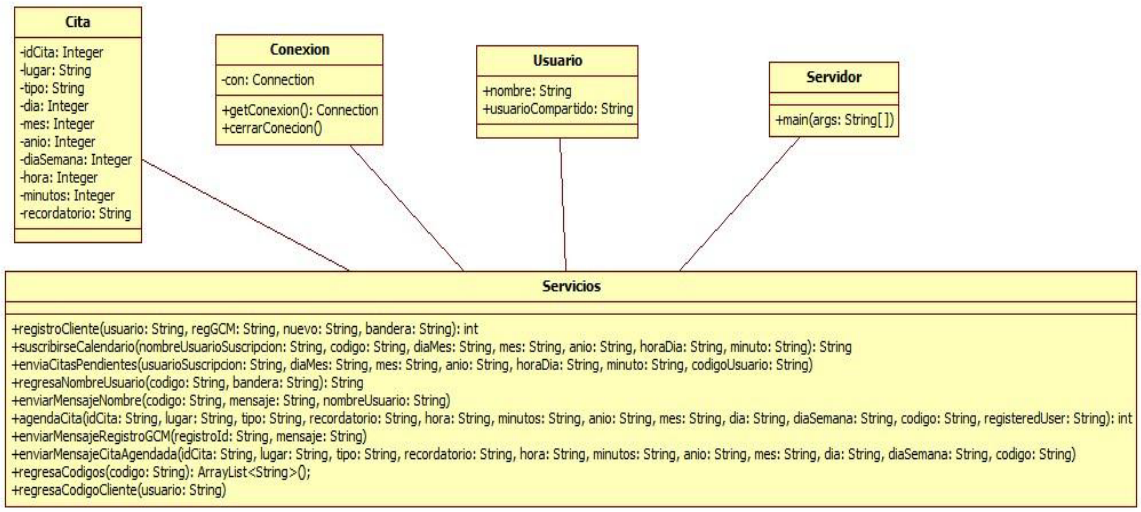


Figura 17. Diagrama de clases de la aplicación MediCal Server.

6.2.5 Arquitectura del sistema.

El patrón de arquitectura de software de la aplicación *MediCal* es el Modelo-Vista-Controlador (MVC). En la Figura 18se observa el árbol de directorios de la aplicación *Medical* con las distintas partes de la arquitectura. En la Figura 19 se observa el árbol de directorios de la aplicación Medical Server.

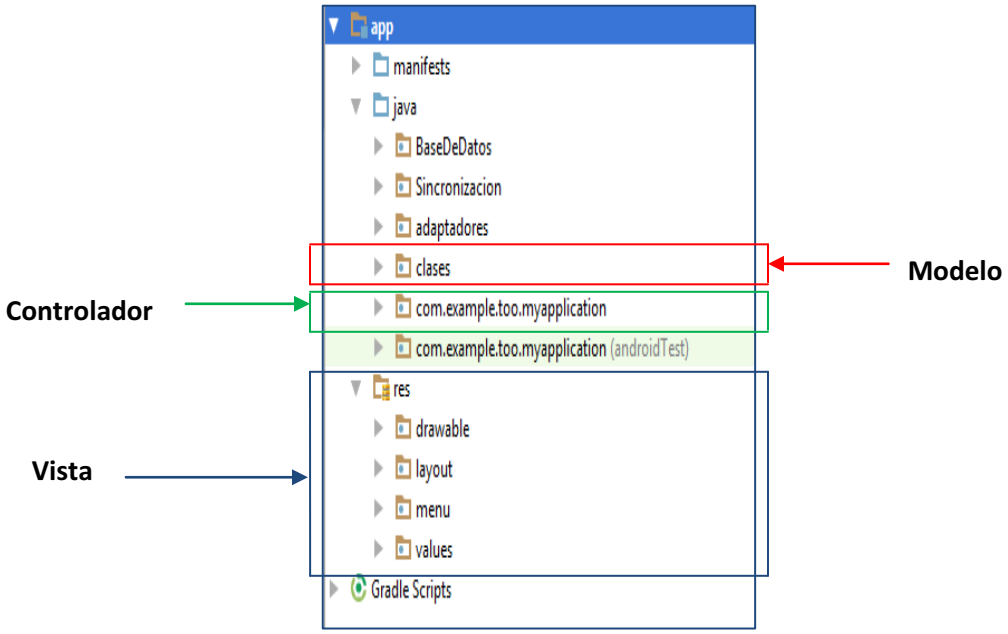


Figura 18. Árbol de directorios de la aplicación MediCal.

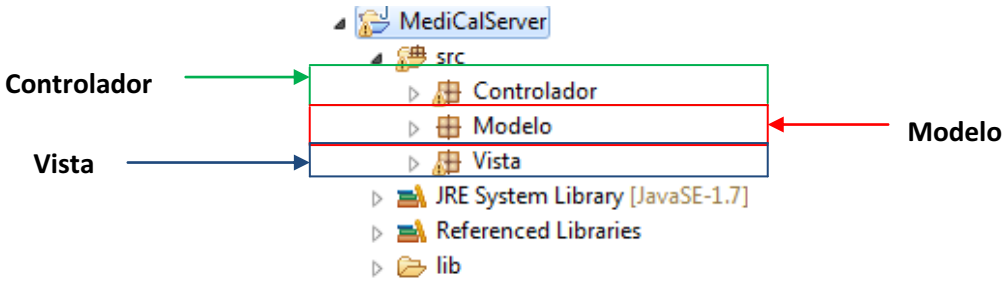


Figura 19. Árbol de directorios de la aplicación MediCal Server.

6.2.5.1 Aplicación *Medical* para el dispositivo móvil.

El modelo está representado por las clases contenidas en el paquete clases. Estas clases crean, consultan y actualizan la base de datos. En este conjunto de clases se encapsula la lógica de negocio de *Medical*.

La vista está formada por la carpeta \res y sus subcarpetas: drawable, layout, menu y values contienen todo el diseño de la vista de la aplicación *MediCal,* saber: íconos, vistas del dispositivo, menús generados por el dispositivo y los mensajes escritos en las vitas del dispositivo, respectivamente.

El controlador está formado por las clases del paquete com.example.too.myapplication. Se define la respuesta del sistema a determinados eventos generados por el usuario, a través del dispositivo móvil.

6.2.5.2 Aplicación *Medical Server*.

El modelo está representado por las clases contenidas en el paquete Modelo. Crean los objetos necesarios para poder manipular las operaciones CRUD

El paquete Vista contiene la clase `Servidor` que pone en marcha a *Medical Server,* y a través de la UI (desde línea de comandos) se operan las funcionalidades de *Medical Server*.

El controlador está formado por las clases del paquete `Controlador`. Se define las operaciones que serán llamadas a través de la aplicación *Medical*, siendo las principales: `registroCliente`(), `suscribirseCalendario`(), `agendaCita`().

6.2.5.3 Modelo arquitectónico de dos capas.

Se estableció una arquitectura de dos capas ver Figura 20 para comunicar el cliente (dispositivo móvil) con el servidor lo que nos permite encapsular o separar elementos de nuestra aplicación en partes claramente definidas.

La primera capa está compuesta por la aplicación *Medical*, la cual se ejecuta en el dispositivo móvil. La segunda capa está compuesta por la aplicación *Medical Server,* la cual aloja además de lo descrito en el punto 6.2.5.2, la base de datos `calendario`. La comunicación entre estas dos capas se realiza a través del protocolo HTTP. El servidor Web *Apache Tonca* es el encargado de ejecutar la aplicación Web *Medical Server*.

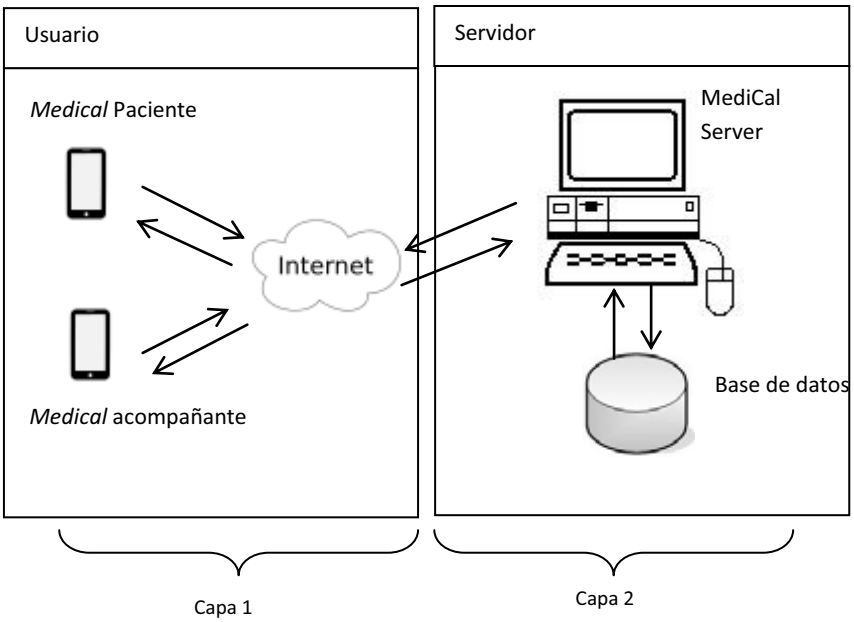


Figura 20. Arquitectura de dos capas, inspirado de [16]

6.2.6. Estructura de la base de datos

6.2.6.1 Base de datos para aplicación *Medical*

La aplicación *Medical* utiliza una base de datos local para el dispositivo móvil, la cual consiste de una tabla llamada `Cita` (ver Figura 21),para las citas agendadas por el paciente. La tabla 1 describe cada uno de los campos que conforman la tabla Cita.

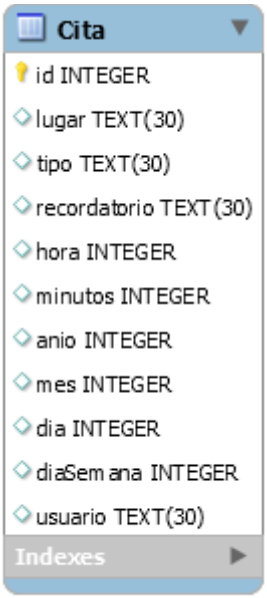


Figura 21. Tabla Cita.

Cita								
	Tipo	Unique	Null	Notnull	Default	Autoincrement	Llave	Descripción
Id	INTEGER			X		X	PK	Identificador de la cita
Lugar	TEXT							Lugar de la cita
Tipo	TEXT							Tipo de la cita
recordatorio	TEXT							Recordatorio de la cita
Hora	INTEGER							Hora de la cita
Minutos	INTEGER							Minutos de la cita
Anio	INTEGER							Año de la cita
Mes	INTEGER							Mes de la cita
Dia	INTEGER							Día de la cita
diaSemana	INTEGER							Día de la semana de la cita
Usuario	INTEGER							Usuario que agendo la cita.

Tabla 1. Descripción de la tabla Cita en la base de datos del teléfono

6.2.6.2 Base de datos para aplicación Medical Server.

Para la aplicación *Medical Server* se creó la base de datos `calendario` la cual contiene tres tablas. La tabla `usuario` guarda el nombre de usuario que se registra en *MediCal Server* y el código asignado por el servidor de Google Cloud Messaging (GCM), la tabla `Cita` que almacena los datos agendados por el paciente en su aplicación *MediCal* siendo la `fecha, hora, tipo de cita, lugar de la cita, el recordatorio de la cita`, así como el usuario que la programó. La tabla `Lista` que almacena los usuarios que se encuentran suscritos a distintos calendarios *MediCal*. La Figura 22 muestra el diagrama. La tabla `Cita` no tiene relación con ninguna tabla. La tabla *usuario* tiene una relación de uno a muchos con la tabla Lista ya que un usuario puede tener uno o más acompañantes.

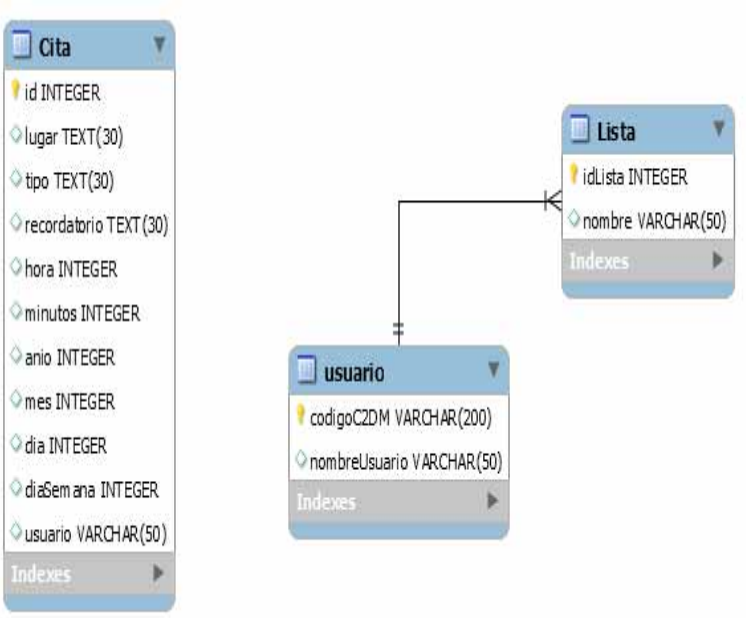


Figura 22. Diagrama de BD del servidor.

La tabla 2 muestra la descripción de la tabla Cita, la tabla 3 la descripción de la tabla usuario y la tabla 4 la descripción de la tabla Lista.

Cita								
	Tipo	Unique	Null	Notnull	Default	Autoincrement	Llave	Descripción
id	INTEGER			X		X	PK	Identificador de la cita
lugar	TEXT							Lugar de la cita
tipo	TEXT							Tipo de la cita
recordatorio	TEXT							Recordatorio de la cita
hora	INTEGER							Hora de la cita
minutos	INTEGER							Minutos de la cita
anio	INTEGER							Año de la cita
mes	INTEGER							Mes de la cita
día	INTEGER							Día de la cita
diaSemana	INTEGER							Día de la semana de la cita
usuario	INTEGER							Usuario que agendo la cita.

Tabla 2. Descripción de la tabla cita en la base de datos de MediCal Server.

Usuario								
	Tipo	Unique	Null	Notnull	Default	Autoincrement	Llave	Descripción
nombreUsuario	Varchar(50)							Nombre del usuario que se registra
CodigoC2DM	Varchar(200)						PK	Código de registro

Tabla 3. Descripción de la tabla usuario en MediCal Server

Lista								
	Tipo	Unique	Null	Notnull	Default	Autoincrement	Llave	Descripción
idLista	INTEGER					X	PK	Identificador
CodigoCompartido	Varchar(200)						FK	Llave foránea a la tabla usuario columna(CodigoC2DM)
nombre	Varchar(50)							Nombre del acompañante

Tabla 4. Descripción de la tabla Lista en la aplicación Medical Server

6.3 Uso del sistema

Pantalla Registro

Al abrir por primera vez la aplicación, se encontrará con la ventana de registro (mostrada en la Figura 23). Para realizar el registro, el dispositivo móvil debe contar con conexión a la red de internet. Este registro se hace con el fin de que el paciente comparta su agenda con el acompañante. Para realizar el registro a nuestra aplicación *MediCal Server*, siga los siguientes pasos:

1. Seleccione una cuenta de Gmail (ver Figura 23), aquella que esté configurada en su dispositivo móvil:

Figura 23. Ventana de registro

2. Una vez seleccionada la cuenta de correo (ver Figura 24), deberá presionar el botón `Registrar` y los datos serán enviados a la aplicación *MediCal Server*.

Figura 24. Envío delos datos de registro

3. Posteriormente la aplicación *MediCal* de su dispositivo lo llevará a la vista del calendario (ver Figura 25),y al mismo tiempo recibirá una notificación indicando que el registro se llevó a cabo correctamente, observe la Figura 26. La vista del calendario es el punto de acceso a las distintas funcionalidades que ofrece *MediCal*.

   A partir del registro del paciente, y siempre que se cuente con conexión a la red de internet, toda cita se guardará en *MediCal Server.*

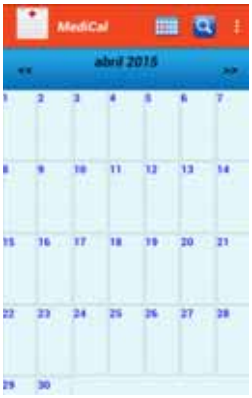


Figura 25. Vista del calendario

Figura 26. Notificación: registro correcto

Suscripción a una agenda

Un acompañante se suscribe a la agenda de un paciente para apoyarlo en recordar las citas médicas. Una vez suscrito a la agenda, el acompañante recibirá notificaciones de las citas registradas por el paciente; también recibirá la información de las citas en su propia agenda *MediCal* y se activará el recordatorio. Es decir, las agendas se sincronizan para que tanto el paciente como el acompañante cuenten con la misma información. Los siguientes pasos muestran como llevar a cabo la suscripción a una agenda.

1. El acompañante, estando en la vista del calendario de *MediCal,* deberá ir al menú suscribirse a una agenda, tal como se muestra en la Figura 27. Este menú se encuentra en la esquina superior derecha de la pantalla, a veces está representado por tres puntos verticales o, en su defecto, presionando el botón para mostrar opciones adicionales del dispositivo móvil.

Figura 27. Menú para suscribirse a una agenda

2. Dentro de la ventana para la `Suscripción` deberá seleccionar el nombre del paciente, a partir de la lista de pacientes registrados en *MediCal Server*, una vez seleccionado el paciente, deberá presionar el botón `Enviar Solicitud`. La Figura 28 muestra un ejemplo de esto.

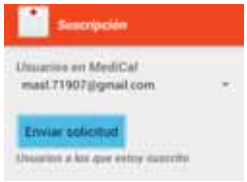


Figura 28.Enviar solicitud de suscripción

3. Una vez presionado el botón `enviar solicitud`, el correo electrónico y la solicitud serán enviados a *MediCal Server*; el cual, a su vez, notifica al paciente que un usuario (identificado por su correo electrónico) (ver Figura 29 quiere suscribirse a su agenda. Luego, la aplicación *MediCal* pregunta al paciente si desea aceptar o rechazar la suscripción (ver Figura 30). Al responder, la aplicación *MediCal Server* notifica al acompañante sobre la respuesta del paciente. La Figura 31muestra la notificación a un acompañante con el mensaje "suscrito correctamente".

Figura 29. Notificación que llega al paciente

Figura 30. Confirmación por parte del paciente

Figura 31. Notificación que recibe el acompañante

Agendar Cita

Al `Agendar` una cita en el calendario de *MediCal*, esta quedará almacenada en el dispositivo móvil. Si la agenda es compartida con al menos un acompañante, la cita será notificada al acompañante, y luego registrada en su propia agenda *MediCal*. Para llevar a cabo esta operación, se tiene dos formas diferentes, a saber:

1. Desde la vista del calendario deberá seleccionar el menú `agendar`, ubicado en la parte superior derecha, ver la Figura 32.
2. Desde la vista del calendario deberá mantener una pulsación larga en el día correspondiente a la fecha en que desea agendar una cita, y luego seleccionar, del menú emergente, la opción `Agendar Cita`, ver Figura 33.

Figura 32. Menú Agendar

Figura 33. Menú Agendar

3. Dentro de la ventana `agendar` deberá establecer los campos necesarios para la cita médica. Una vez completados los campos, deberá presionar el botón `Hecho`, ubicado en la parte superior derecha. La Figura 34 muestra un ejemplo.

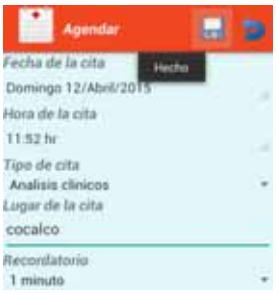


Figura 34. Ventana para Agendar una cita

4. Una vez presionado el botón `Hecho`, se enviarán los datos a la aplicación *MediCal Server*, se mostrará un mensaje "Cita registrada con éxito", y también, se mostrará un ícono en la fecha de la cita, ver la Figura 35.Si el calendario es compartido con un acompañante, la aplicación *MediCal Server* enviará dos notificaciones; i) al acompañante informando que el paciente ha agendado una nueva cita, ver la Figura 36 ii) al paciente indicando la "Cita Notificada Correctamente" al acompañante, ver Figura 37.

Figura 35. Cita agendada con éxito

Figura 36. Notificación que recibe el acompañante

Figura 37. Notificación que recibe el paciente

6.4. Hardware y software necesario

Para el desarrollo de la aplicación *MediCal* se utilizó un equipo de cómputo con las características siguientes:

6.4.1 Hardware.

- Memoria RAM 2GB.
- Sistema Operativo Windows 7 de 64 bits.
- Procesador Intel Celeron 560 a 2.13GHz.
- Disco duro de 148GB.

6.4.2 Software.

El software utilizado para el desarrollo de la aplicación, así como sus características se describe a continuación:

1. Sistema Operativo *Android*, versión 4.1.1 [12].
2. Lenguaje de programación JavaJ2ME.
3. *IDE Android Studio*, versión 1.1 [13].
4. Base de datos: En la aplicación móvil *MediCal* se empleó *SQLite*, versión 3.8.9 [14]. En la aplicación *MediCal* Server se empleó *MySQL* [15].

7. Resultados

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

- Se logró la implementación de los módulos que administran la agenda, a saber: registrar, buscar, consultar, modificar, alertar y eliminar.
- Se logró la implementación del módulo para la sincronización de las citas, este módulo corresponde a la aplicación *Medical Server*, la cual opera desde una máquina servidor.
- Se logró la implementación de dos bases de datos: una que opera localmente en el dispositivo móvil y otra que opera en el servidor. A través de estas bases de datos se logra la consistencia de los datos en las dos agendas.
- Se logró la implementación de las interfaces gráficas de la aplicación *Medical*, las interfaces gráficas son sencillas, desde las cuales se pueden operar las funcionalidades de la aplicación, además de ser intuitivas para el usuario.

8. Análisis y discusión de resultados.

Las pruebas realizadas a la aplicación *Medical* se hicieron tanto en la aplicación en el móvil como en la aplicación en el *Medical Server*, para verificar la persistencia de los datos en los dispositivos móviles y en el servidor; además, de validar la sincronización entre dispositivos. Las pruebas realizadas fueron las siguientes.

- Una vez agendada una cita en la aplicación *MediCal,* en el dispositivo móvil, se revisó que sonara una alarma en tiempo y forma establecida en el recordatorio.
- Una vez agendada una cita en la aplicación *MediCal*, se revisó que en la vista principal (calendario) fuera visible el ícono de la cita agendada en la fecha correspondiente.
- Una vez agendada la cita, se revisó que los datos de ella fueran correctamente ingresados tanto en la base de datos de la aplicación *MediCal* como en la base de datos de la aplicación *MediCal Server*.
- Al `Agendar` una cita, se comprobó que la cita fuera notificada solo al o los acompañantes que estaban suscritos a la agenda del paciente.
- Al buscar las citas agendadas en un periodo de tiempo, se verificó que las citas mostradas estuvieran dentro del rango de fechas indicadas.
- Al `Eliminar` una cita, se verificó que ésta fuera correctamente notificada a la aplicación *MediCal* del acompañante; además, se verificó que la cita eliminada por el paciente en su dispositivo, también fuera eliminada en el dispositivo del acompañante.

- Al `consultar` una cita, se revisó que los datos de ésta correspondieran a los ingresados por el paciente cuando agendó la cita.
- Al `Modificar` una cita, se verificó que la cita fuera correctamente notificada a la aplicación *MediCal* del acompañante; además se verificó que la cita modificada por el paciente en su dispositivo, también fuera modificada en el dispositivo del acompañante.
- Al momento de registrarse un paciente o un acompañante en la aplicación *MediCal Server,* se comprobó que el registro permanece en su base de datos.
- Al suscribirse a una agenda en la aplicación *MediCal Server*, se verificó que el acompañante quedara registrado en su base de datos.

A partir de las pruebas realizadas encontramos algunas limitaciones de la aplicación *Medical,* las cuales se explican a continuación.

1. Los recordatorios de las citas agendadas no pudieron ser almacenados, de tal forma que solo se recuerda la última cita agendada.
2. Al realizar la sincronización, una cita agendada por el paciente es notificada al acompañante, pero no es posible consultarla a partir de la notificación. Para consultar la cita, se debe ingresar desde la vista del calendario de *Medical* a través de la operación consultar.

9. Conclusiones

La aplicación que resultó de este proyecto se llama *MediCal. MediCal* es una agenda que tiene la característica de ser compartida y opera a través de dispositivos móviles con sistema operativo Android. Esta agenda provee un medio para que un paciente gestione sus citas médicas, y su acompañante (familiar) conozca en todo momento los movimientos de la agenda médica del paciente.

Medical consta de dos módulos principales: *Medical Server* el cual se encarga de realizar la sincronización de la agenda del paciente con la agenda del acompañante; este módulo opera desde una máquina servidor, y *Medical* el cual se encarga de gestionar las citas médicas localmente, tanto en el dispositivo del paciente como en el dispositivo del acompañante.

El cumplimiento de los objetivos presentados en la propuesta del proyecto de Integración se describe a continuación:

- Diseñar e implementar un módulo de una agenda médica que incluya las operaciones de: registrar, modificar, eliminar, consultar y alertar sobre las citas médicas. Este objetivo se cumplió de forma parcial puesto que la aplicación no alerta de todas las citas médicas, solo de la última agendada.

- Diseñar e implementar un módulo que permita la comunicación y sincronización entre los dispositivos que compartirán la agenda. Este objetivo se cumplió completamente, porque una vez registrada la agenda de un paciente en el servidor, todo registro o cambio en las citas agendadas serán sincronizadas en el dispositivo del acompañante.

- Diseñar e implementar un módulo para la persistencia de los datos que se manejarán en la agenda médica. Este objetivo se cumplió completamente, todos los datos de las citas agendadas persisten tanto de manera local, en el dispositivo, como de manera remota, en el servidor.

- Diseñar e implementar una interfaz gráfica para dispositivos móviles que facilite al usuario la interacción con la agenda médica. Este objetivo se cumplió completamente. Las interfaces gráficas de *Medical* son fácil de manejar e intuitivas.

10. Perspectivas del proyecto.

Existen mejoras que pueden realizarse a la aplicación *Medical*, las cuales se listan y describen a continuación:

1. Implementar un módulo que elimine automáticamente los datos obsoletos, tales como las citas que ya se realizaron.
2. Implementar un módulo en *Medical Server* que permita a un acompañante darse de baja de una agenda.
3. Al momento de recibir una notificación sobre una cita agendada por el paciente, permitir consultar los datos de la cita al seleccionar dicha notificación.
4. Mejorar la interfaz gráfica del dispositivo móvil con apoyo de un diseñador de interfaces o un diseñador gráfico.
5. Convertir a la aplicación *Medical Server* en un servicio web que cuente con una interfaz gráfica que facilite la gestión de la información en el servidor.

Bibliografía

[1] J. L. Razo, «Sistema automático de control y administración de contenido de información,» D.F., 2004.

[2] Zao, J.K., Mei-Ying Wang, Peihsuan Tsai, Liu, J.W.S., "Smart phone basado en un calendario de medicamentos, recordatorio y monitoreo" , Articulo, e-Health Networking Applications and Services (Healthcom), 12th Conferencia internacional de la IEEE, 2010.

[3] J. P. Alcántara Olivares, José Daniel López Jaimes , "Aplicación Colaborativa para Dispositivos Móviles con Sistema Operativo Android", Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2013.

[4] J. Arcos García, L. C. Garcia Morales, " Sistema Colaborativo para médicos especialistas en medicina alternativa", Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2010.

[5] A. M. Velázquez Canales **"**Aplicación web de administración de horarios para estudiantes", Proyecto Terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2009.

[6] «IntelliJ IDEA,» [En línea]. Available: https://vaadin.com/book/es/-/page/getting-started.idea.html. [Último acceso: 28 Marzo 2015].

[7] «Android Developers,» [En línea]. Available: http://developer.android.com/tools/sdk/eclipse-adt.html. [Último acceso: 28 Marzo 2015].

[8] «SQLite,» [En línea]. Available: http://www.sqlite.org/. [Último acceso: 28 Marzo 2015].

[9] C.J. y H. Darwen, «A guide to the SQL Standard,» de *A guide to the SQL Standard*, Massachusetts, Addison-Wesley, 1997.

[10] H. Darwen, « Introducción a los sistemas de bases de datos,» de *Introducción a los sistemas de bases de datos*, Massachusetts:, Prentice Hall, 2001.

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

[12] «Android,» [En línea]. Available: https://www.android.com/. [Último acceso: 21 Abril 2015].

[13] «Android Developers,» [En línea]. Available: http://developer.android.com/sdk/index.html.

[Último acceso: 28 Marzo 2015].

[14] «SQLite,» [En línea]. Available: http://www.sqlite.org/. [Último acceso: 17 Abril 2015].

[15] «MySQL,» [En línea]. Available: http://www.mysql.com/downloads/. [Último acceso: 28 Marzo 2015].

[16] A. J. M. Sierra, «Struts,» de Struts, España, RA-MA Editorial, 2007.