

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Facultad de Ciencias Químicas e Ingeniería

Maestría y Doctorado en Ciencias e Ingeniería



Desarrollo de un Framework para el Monitoreo Remoto de Signos Vitales Utilizando Dispositivos Móviles

TESIS

QUE PARA OBTENER EL TÍTULO DE

MAESTRO EN CIENCIAS

Presenta:

GELACIO LÁZARO MARTÍNEZ

Director de Tesis

DR. GUILLERMO LÍCEA SANDOVAL

TIJUANA, B.C., MÉXICO

FEBRERO DE 2014

Universidad Autónoma de Baja California
FACULTAD DE CIENCIAS QUÍMICAS E INGENIERÍA
COORDINACIÓN DE POSGRADO E INVESTIGACIÓN

FOLIO No. 116

Tijuana, B. C., a 26 de Febrero de 2014

C. Gelacio Lazaro Martinez
Pasante de: Maestro en Ciencias
Presente

El tema de trabajo y/o tesis para su examen profesional, en la
Opción TESIS

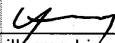
Es propuesto, por el C. Dr. Guillermo Licea Sandoval

Quien será el responsable de la calidad de trabajo que usted presente, referido al
tema "DESARROLLO DE UN FRAMEWORK PARA EL MONITOREO REMOTO DE
SIGNOS VITALES UTILIZANDO DISPOSITIVOS MOVILES"

el cual deberá usted desarrollar, de acuerdo con el siguiente orden:

- I.- INTRODUCCION
- II.- MARCO TEORICO
- III.- ANALISIS Y DISEÑO
- IV.- PRUEBAS Y RESULTADOS
- V.- CONCLUSIONES


Q. Noemí Hernández Hernández
Sub-Director


Dr. Guillermo Licea Sandoval
Director de Tesis


Dr. Luis Enrique Palafox Maestre
Director

A Conejo.

A la memoria de Beto.

Agradecimientos

Este trabajo fué posible gracias al apoyo de mucha gente.

Deseo agradecer al Dr. Guillermo Licea por apoyarme y guiarme durante el desarrollo de este trabajo y el transcurso de la maestría.

A la familia Rodriguez Ortiz.

A mis compañer@s de la maestría que hicieron más agradable el trayecto de la maestría.

A mis maestr@s Puga, Juan Ramón y Olivia por sus observaciones y recomendaciones.

A mi papá y mamá(Solomón y Lupita).

Familia y amigos

Resumen

La comunicación móvil ofrece un medio eficaz de acercar servicios de salud a los ciudadanos de países en vía de desarrollo. Con teléfonos móviles de bajo costo y la penetración global de redes de telefonía móvil, millones de personas utilizan dispositivos móviles como herramientas diarias para la comunicación y transferencia de datos. Esta explosión del uso del teléfono móvil tiene el potencial de mejorar la prestación de servicios de salud en una escala masiva. Este trabajo se presenta el desarrollo de un framework en Java para el monitoreo remoto de signos vitales en pacientes. Con el propósito de auxiliar a los doctores en la detección a tiempo de enfermedades mediante las aplicaciones que se desarrollen utilizando el framework.

Abstract

The mobile communication provides an effective means of bringing health services to citizens of countries in development. With low-cost mobile phones and the global penetration mobile phone networks, millions of people use mobile devices as everyday tools for communication and data transfer. This blast from the use of mobile phone has the potential to improve the provision of health services on a scale mass. This paper presents the development of a Java framework for monitoring remote patient vital signs. With the purpose of helping the physicians in time of disease detection by applications that are developed using the framework.

Índice General

1. Introducción	2
1.1. Antecedentes	2
1.1.1. Trabajo previo	4
1.2. Protocolo	4
1.2.1. Planteamiento del problema	4
1.2.2. Justificación	6
1.2.3. Hipótesis	8
1.2.4. Objetivos	8
1.2.5. Metas	9
1.2.6. Metodología	10
1.3. Como está organizado este documento	11
2. Marco Teórico	13
2.1. Frameworks de Software	14
2.2. mHealth	16

2.2.1. Educación y Concientización.	17
2.2.2. Colección Remota de Información.	18
2.2.3. Monitoreo Remoto.	18
2.2.4. Comunicación y Capacitación a trabajadores de la Salud.	19
2.2.5. Seguimiento de enfermedades y brotes epidémicos.	19
2.2.6. Apoyo en el Diagnóstico y Tratamiento.	19
2.3. Signos Vitales	20
2.4. Android	20
3. Análisis y Diseño	23
3.1. Requerimientos del Framework	23
3.1.1. Requerimientos funcionales	23
3.1.2. Tecnologías Utilizadas.	25
3.2. Intefaz Gráfica del Usuario	26
3.2.1. HeartRateGraphView	28
3.2.2. TemperatureGraphView	29
3.2.3. PieChartView	29
3.3. Conección a Servicios Web	29
3.4. Servicios Web	32
3.4.1. RESTful	32
3.4.2. RESTful APIs	35

4. Pruebas y resultados	41
4.1. Utilización del Framework	41
4.2. Desarrollo de aplicación sin el framework.	49
5. Conclusiones	50
5.1. Trabajo futuro	52

Lista de Figuras

1.1. Arquitectura de un sistema de monitoreo[2]	5
1.2. Modelo de espiral	10
3.1. Organización de componentes de interfaz de usuario.	27
3.2. Diagrama de Interfaz de Pacientes.	29
3.3. Interacción de clases para conexión a un servicio web	30
3.4. Modelo de espiral	35
3.5. Diagrama de actividades del API createPatient.	37
3.6. Diagrama de actividades del API getPatients.	38
4.1. Ejemplo 1 de la utilización del framework	43
4.2. Ejemplo 3 de la utilización del framework	44
4.3. Ejemplo 2 de la utilización del framework	45

Listados

- 4.1. Inflando el layout para utilizarlo en la actividad. 44
- 4.2. Requisición de datos de temperatura al servicio web. 46
- 4.3. Recepción de datos temperatura del servicio web. 48
- 4.4. Notificación de error en el proceso de requisición de datos en el servicio web. . 49

Capítulo 1

Introducción

1.1. Antecedentes

En la Facultad de Ciencias Químicas e Ingeniería se ha trabajado en un proyecto para el monitoreo remoto de signos vitales de pacientes utilizando una red de sensores inalámbricos experimental. La idea primordial es atacar las enfermedades cardiovasculares que representan la causa más frecuente de mortalidad en el ámbito mundial. Previniendo o detectando oportunamente los padecimientos cardiovasculares para aminorar futuras condiciones críticas que llevan a costosos tratamientos y altos riesgos de salud. Se está en la etapa de desarrollo de un sistema no intrusivo utilizado un dispositivo sensor de parámetros fisiológicos por medio de una pulsera compacta que se trae en la muñeca del paciente. El dispositivo soporta una red inalámbrica de sensores para la comunicación a corta distancia y un modem GSM para comunicación a largo alcance. Es un sistema de monitoreo para el hogar que incluye toma de muestra y envío de señales fisiológicas de temperatura, frecuencia cardíaca y movilidad. Habilita el sistema para monitoreo de manera remota y continua sin que este interfiera en las actividades normales del paciente de movilidad [6]. Las mediciones de temperatura,

frecuencia cardíaca y movilidad son enviados a un servidor de base de datos.

El costo de la salud y el envejecimiento de la población mundial se está incrementando, esto ha requerido monitorear el estado de salud los pacientes mientras se encuentran fuera de los hospitales y se encuentran en su ambiente personal. Para satisfacer esta demanda, una gran variedad de sistemas de prototipo y productos comerciales se han construido en el transcurso de los años recientes. Estos pretenden proveer información de la condición de salud en tiempo real a los pacientes mismos, a los centros médicos o a los supervisores de los médicos así como alertar sobre las condiciones críticas de los pacientes.

Los sistemas de monitoreo de salud puede comprender varios tipos de mini-sensores , alámbrico/inalámbrico o implantable incluso. Estos biosensores tienen la capacidad de medir importantes parámetros fisiológicas como el ritmo cardíaco, presión sanguínea, temperatura corporal, saturación de oxígeno, frecuencia respiratoria, etc. Las mediciones obtenidas son comunicados a un central de datos mediante dispositivos alámbricos o inalámbricos. Esto ejemplifica de que los sistemas médicos puede abarcar sensores, fuentes de poder, módulos de comunicación inalámbricas y enlaces, unidades de control y procesamiento, interfaz gráfica para el usuario; software y algoritmos para la extracción de datos y toma de decisiones.

La arquitectura de la Figura 1.1 se muestra una arquitectura de acuerdo a la funcionalidad y componentes del sistema descrito. Sin embargo esto no se debe tomar como un estándar de diseño de los sistemas de monitoreo de pacientes ya que algunos sistemas pueden adoptar una variación [2].

La primera fase el proyecto consiste en el desarrollo del dispositivo en forma de pulsera que se encarga de medir o tomar las muestras del paciente y transmitirlos a un servidor de base de datos MySQL. En esta pulsera se encuentran colocados los biosensores, una red inalámbrica de sensores de comunicación de corto alcance, un módem GSM para la comunicación de largo alcance. Un sistema para el monitoreo en el hogar que abarca la toma y envío de

señales fisiológicas como temperatura, frecuencia cardíaca y movilidad, de manera remota, continua y sin afectar las actividades y movilidad del paciente. De la arquitectura de la Figura 1.1 se puede decir que están incluidos las partes de los biosensores, transmisión inalámbrica de datos, comunicación de sensores, control de señales y la base de datos de señales.

La segunda fase del proyecto se pretende desarrollar un framework en Java para facilitar y acelerar el tiempo de desarrollo de aplicaciones en el área de monitoreo de signos vitales. En esta fase se pretende abarcar la parte de Interfaz Gráfica de Usuario, implementar algunos APIs en un servidor para extraer los datos de la base de datos e implementar un diseño para que las aplicaciones en que se utilice el framework puedan fácilmente interactuar con el servidor de bases de datos para la obtención de las base de datos a través de la Internet. Completando de esta manera la arquitectura que se muestra en la Figura 1.1 aunque como lo menciona Pantelopoulos[2], la arquitectura que se propone no es un estándar y puede tener variaciones y en algunos sistemas incluso no todos los módulos de la arquitectura se incluyen o abarcan.

1.1.1. Trabajo previo

1.2. Protocolo

1.2.1. Planteamiento del problema

El sistema de monitoreo remoto de signos vitales se encuentra en su primera fase. Por el momento sensa los signos vitales del paciente tomando la frecuencia cardíaca, la temperatura y la movilidad. Transmite la información para ser almacenados a un servidor de base de datos. En la segunda fase del sistema se requiere que la información pueda estar disponible a los interesados como los doctores, enfermeras, etc.

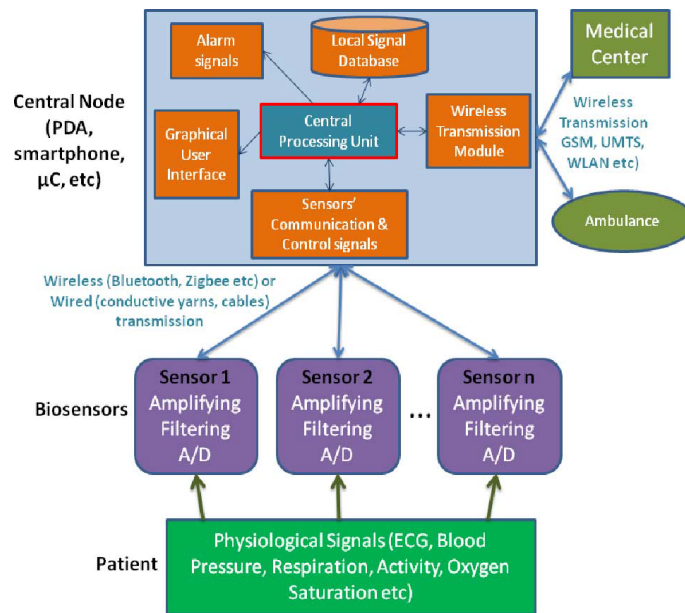


Figura 1.1: Arquitectura de un sistema de monitoreo[2]

Se plantea el desarrollo y construcción de un framework para disponer de la información de signos vitales de pacientes almacenados en un servidor de base de datos MySQL. Se requiere que la información este disponible y este al alcance fácilmente de los doctores mediante aplicaciones de fácil desarrollo con componentes de software de arrastrar y soltar. Se propone el desarrollo de clases o componentes para que las conexiones al servidor de bases de datos sea fácil de usar y prácticamente transparente al usuario. Se proyecta el desarrollo de Web Services o APIs despachar las requisiciones de signos vitales de las aplicaciones cliente al servidor de bases de datos MySQL.

1. Clases o componentes de Interfaz de Usuario para desplegar los signos vitales. Aunque en la primera fase del proyecto abarca hasta la medición de temperatura corporal, frecuencia respiratoria y movilidad, en esta segunda etapa se propone cubrir los 4 signos vitales y tener todas las interfaces preparadas para cuando las presión arterial y el pulso cardíaco se lleven a cabo:

- a)* Presión arterial
 - b)* Respiración
 - c)* Pulso
 - d)* Temperatura
- 2. Clases o componentes para realizar transacciones de requisición de signos vitales con el con el servidor de base de datos MySQL.
- 3. Clases o componentes de Interfaz de Usuario para informar al usuario de transacciones en progreso con el servidor de base de datos o los datos de signos vitales se encuentran en procesamiento.
- 4. Diseñar e implementar APIs o Servicios Web necesarios para a través de estos las aplicaciones cliente interactúen con el servidor de base de datos MySQL y de esta manera se tenga acceso a los signos vitales almacenados. Con el desarrollo del framework que se plantea en este trabajo se complementa el sistema o se cubren la mayor parte de los módulos o partes de la arquitectura de la Figura 1.1

1.2.2. Justificación

La primera década del siglo XXI se ha cerrado, gobernantes de muchos países en desarrollo pueden estar orgullosos de sus enormes esfuerzos para mejorar la vida de sus ciudadanos. En muchas partes del mundo, los ciudadanos de las economías emergentes han saboreado el aumento de sus ingresos y mayor acceso a herramientas que prometen la mejora de la calidad de vida. No obstante, sigue habiendo barreras muy fuertes. Los problemas de salud de hoy podría decirse que es la barrera más importante para el desarrollo mundial sustentable. Las enfermedades y la falta de una adecuada atención médica preventiva tienen fuerte impacto en los países en desarrollo. A pesar de los fuertes avances económicos de la década, el informe

de la ONU sobre los avances realizados para alcanzar los Objetivos de Desarrollo del Milenio (ODM) indica que sigue habiendo condiciones deplorables en la áreas cruciales de la salud pública. Por ejemplo, un niño nacido en un país en vías de desarrollo tiene 33 veces más de probabilidad de morir dentro de los primeros 5 años de vida que un niño nacido en un país industrializado, aunque las principales causas de muerte (neumonía, la diarrea, el paludismo y el sarampión) son prevenibles mediante servicios básicos y vacunas. Cada minuto, al menos una mujer muere a causa de complicaciones relacionadas con el embarazo o parto. Y para cada mujer que muere en el parto, otras 20 sufren de lesiones, infecciones, cerca de 10 millones cada año. Se estima que 2.5 millones de personas fueron infectados por HIV en el 2007. Enfermedades transmisibles y perfectamente evitables, como la tuberculosis (TB) y la malaria siguen cobrando vidas debido a factores prevenibles como la falta de acceso a medicamentos y adecuado tratamiento médico. Según estimaciones actuales cumplir con la meta del ODM de reducir a la mitad la tasa de prevalencia de tuberculosis en el 2015 es poco probable.

La capacidad de los países en desarrollo para enfrentar estos problemas tiene varios obstáculos, entre ellos la escasez global de los trabajadores de salud. Según la Organización Mundial de Salud, entre 57 países, principalmente los que están en vías de desarrollo, existe un grave déficit de trabajadores de la salud, el cual representa un déficit total de 2.4 millones de trabajadores de la salud mundial [7].

Esta limitación de recursos humanos intensifica la presión cada vez mayor de desarrollo de sistemas de salud mundial. No solo tiene que hacer frente conteniendo la propagación de enfermedades asociadas a la pobreza extrema, sino que tiene que hacer frente también a la creciente incidencia de enfermedades crónicas, como la diabetes, las enfermedades del corazón. Los gobiernos, empresas, organizaciones no gubernamentales, fundaciones y organización reconocen la importancia de aprovechar las nuevas tecnologías y soluciones para hacer frente a estas distintas pero interrelacionadas desafíos en materia de salud.

Las comunicaciones móviles ofrecen un medio eficaz para llevar el servicio de salud a los ciudadanos de los países en desarrollo. Con el bajo costo de los teléfonos móviles y la penetración de la redes de telefonía móvil a escala mundial, decenas de millones de ciudadanos que nunca antes habían tenido accesos a un teléfono fijo o equipo, ahora utilizan teléfonos móviles, como herramienta para la comunicación diaria y transferencia de información. El 64 % de los usuarios de teléfonos móviles se encuentran en los países en desarrollo [8]. Por otra parte las estimaciones indican que para el año 2012, la mitad de los individuos en áreas remotas tendrán un teléfono móvil. Esta creciente ubicuidad de los teléfonos móviles es un elemento central en la promesa de las tecnologías móviles para la salud.

1.2.3. Hipótesis

Es posible reducir el tiempo, esfuerzo y costo en el desarrollo de aplicaciones de monitoreo de signos vitales a través de un framework que presente interfaces médicas estandarizadas y métodos de comunicación con un servidor de datos.

1.2.4. Objetivos

Objetivo general

El objetivo principal de la investigación es el desarrollo de un framework para facilitar el desarrollo de aplicaciones móviles en plataforma Android en el área de monitoreo del Signos Vitales utilizando interfaces médicas estandarizadas. Con el propósito de apoyar a los médicos en la toma de decisiones con las aplicaciones que se puedan desarrollar con el framework.

Objetivos específicos

1. Construir librerías necesarias para desarrollo de aplicaciones cliente de monitoreo de signos vitales en dispositivos móviles.
2. Desarrollar APIs or Servicios Web para que mediante estas funciones las aplicaciones cliente se comuniquen con el servidor de base de datos.
3. Desarrollar una aplicación (framework) por el lado del servidor en MySQL para la interacción cliente-servidor
4. Investigar los requerimientos necesarios para las aplicaciones de monitoreo de signos vitales.
5. Investigar e implementar las Interfaces de Usuario estandarizados para presentar datos de signos vitales.

1.2.5. Metas

1. Contar con un framework para el desarrollo de aplicaciones de Monitoreo de Signos Vitales en plataforma Android.
2. Contar con las Interfaces gráficas o componentes estándares necesarias para mostrar signos vitales en las aplicaciones móviles.
3. Contar con la documentación del framework necesaria para la implementación o desarrollo. Documentación técnica y documentación del usuario.
4. Probar la facilidad de uso del framework en estudiantes de Ingeniería en Computación.
5. Probar la facilidad del uso del framework en desarrolladores experimentados de Software.



Figura 1.2: Modelo de espiral

6. Desarrollar 3 aplicaciones para probar la facilidad de uso del framework.

1.2.6. Metodología

Dado que el presente trabajo trata de desarrollo de Software se utiliza el modelo de espiral o incremental de desarrollo de software.

El modelo del espiral consiste en hacer ciclos o iteraciones sobre las actividades de desarrollo bien definidos que se repiten en forma de espiral comenzando desde el centro. En cada ciclo del espiral se va generando un prototipo ejecutable que se puede probar acercándose cada vez a los objetivos finales del producto, en esta caso un framework, como se muestra en la Figura 1.2.

El framework se desarrollará utilizando el modelo incremental de desarrollo de software para el cual se realizarán las siguientes actividades de manera iterativa:

1. Estudio del estado actual de las aplicaciones de Monitoreo del Signos Vitales
2. Análisis de requerimientos para el desarrollo de los frameworks.
3. Analizar los patrones de diseño que se pueden usar para la implementación de framework.

4. Diseñar la implementación del framework
5. Implementación del framework.
6. Documentación del framework. Escribir el documento técnico y el documento del usuario.
7. Desarrollo de aplicaciones para diferentes plataformas y tipos de usuario utilizando el framework
8. Escribir/Actualizar un plan de pruebas.
9. Correr el plan de pruebas.
10. Corrección de errores del framework.
11. Desarrollar tres aplicaciones para probar el framework.

1.3. Como está organizado este documento

El presente trabajo esta organizado en 6 capítulos

- **Capítulo 1:** Se hace una descripción del origen del proyecto, del trabajo previo realizado al presente. La arquitectura no estándar de los sistemas de monitoreo remoto de salud. Se expone en que consiste el presente proyecto. La justificación de porque se está desarrollando el presente trabajo. Así como los objetivos y metas a alcanzar; la metodología de desarrollo que se planea seguir.
- **Capítulo 2:** Es esta parte se explica un poco que son los frameworks. Se cita las definiciones de mHealth y los campos en que se subdivide para ubicar al lector en en área se encuentra el presente trabajo. También se cita la definción de los signos vitales

para introducir al lector en este campo ya que el proyecto trata de mediciones de signos vitales de manera remota.

- **Capítulo 3:** En ésta sección se explica los requerimientos con los cuales debe cumplir o satisfacer el framework de software que se desarrolla en este trabajo. Los requerimientos funcionales, las limitaciones así como las tecnologías o herramientas de software que se utilizan para el desarrollo del framework.
- **Capítulo 4:** En este apartado se expone como se desarrolla la interfaz gráfica del usuario así como los diferentes componentes que se desarrollaron y la arquitectura de las conexiones al servidor como parte del framework.
- **Capítulo 5:** En esta sección se exponen 3 aplicaciones que se desarrollaron haciendo uso del framework. Se describe las bondades o ventajas del software, la manera en que se utilizan los componentes de interfaz de usuario y la facilidad de uso de las funciones o métodos que interactúan con el servidor de la base de datos.
- **Capítulo 6:** Se exponen las conclusiones mencionando las ventajas del uso del framework así como el trabajo pendiente y mejoras que se le pueden realizar al framework.

Capítulo 2

Marco Teórico

mHealth es la abreviación de mobile Health y es un término utilizado en medicina y salud pública para referirse a la práctica médica que se apoya en dispositivos móviles. mHealth ha emergido en años recientes como un segmento de eHealth(Electronic Health). Aunque no existe un acuerdo amplio sobre definición de estas áreas, la comunidad pública de salud se ha unido en torno a estas definiciones:

- eHealth: Es la utilización de la tecnologías de información y las comunicaciones, como computadoras, teléfonos móviles y comunicación satelital, en el cuidado de la salud.
- mHealth: Es la utilización de las tecnologías de de información y las comunicaciones, como PDAs, tabletas electronicas, teléfonos móviles , para servicios de información y el cuidado de la salud.

mHealth y eHealth están estrechamente ligados, ambos se utilizan para mejorar la salud y sus tecnologás trabajan en conjunto para este fin.

El presente trabajo pertenece al área conocida como mHealth y consiste en un framework de software para el monitoreo de Signos Vitales. En la siguiente sección se describen que es un

framework de software, los sub campos de mHealth y debido a que el trabajo está relacionado con los signos vitales es preciso ofrecer una definición de estos.

2.1. Frameworks de Software

Los frameworks son un área que ya sido ampliamente discutido y definido por varios autores.

Los frameworks son una técnica de reutilización orientado a objetos. Existe una gran confusión de si los framework son un patrón a gran escala o si son solo otro tipo de componente. Las definiciones de framework varía entre las cuales encontramos las siguientes:

Gamma, Helm, Johson y Vlissides definen un framework como un conjunto de clases en colaboración que conforman un diseño reutilizable para una clase especifica de software. Un framework proporciona un arquitectura dividiendo el diseño en clases abstractas y la definición de sus responsabilidades y colaboración. Un desarrollador personaliza el framework a una aplicación en particular creando subclases y composición de instancias de las clases del framework [10].

Los frameworks se definen como un conjunto de clases que incorpora un diseño abstracto para solucionar a una familia de problemas relacionados o un conjunto de objetos en colaboración para realizar o llevar a cabo un conjunto de responsabilidades de un dominio de aplicación de un subsistema [15] [16].

Reberts y Johnson define a los frameworks como diseños reutilizables en todo o una parte de un sistema de software descrito como un conjunto de clases abstractas y la manera en que las instancias de estas clases colaboran [17].

El framework tambien se ha considerado como un conjunto de clases en colaboración, ambos abstractos y concretos, que hacen un diseño reusable para un tipo especifico de soft-

ware [1].

Un framework es una aplicación reusable, semi-completa que puede estar especializado para producir aplicaciones personalizadas. Los frameworks promueven la reusabilidad explotando el dominio del conocimiento y el esfuerzo de desarrolladores experimentados para evitar recrear las soluciones comunes a los requerimientos de aplicaciones recurrentes [13].

Una de las ideas clave de un framework es la de la clase abstracta. Una clase abstracta es una clase que no tiene instancias y solo se usa como una superclase.

Un framework es un diseño de gran escala que describe como un programa es descompuesto dentro de un conjunto de interacción de objetos.

El primer framework ampliamente utilizado se desarrollo a finales de los 70's. Este fue el Smaltalk-80, un framework de interface de usuario llamado Model/View/Controller(MVC). MVC mostró que la programación orientada a objetos fué bien adaptado para la implementación de interfaces de usuario gráficas. Este divide una interface de usuario en 3 tipos de componentes: modelos, vistas y controladores. Estos objetos trabajan en tríos consistiendo de una vista y un controlador interactuando con un modelo. Un modelo es un objeto de la aplicación y se supone que es independiente de la interface de usuario. Una vista administra una región del display y lo mantiene consistente con el estado del modelo. Un controlador convierte eventos de usuario como los movimientos del mouse y las presiones de teclado en operaciones en sus modelos y vistas.

Los frameworks no están limitados a las interfaces de usuario, también puede ser aplicados cualquier área de diseño de software.

Un framework reutiliza código porque facilita la construcción de una aplicación partiendo de una librería de componentes existentes. Estos componentes pueden ser fácilmente usados unos con otros porque todos ellos utilizan las interfaces del framework. Un framework también reutiliza código porque un nuevo componente puede heredar la mayoría de la implementación

de una super clase abstracta. Pero el reuso es mejor cuando no se tiene que entender los componentes que se están reusando, y la herencia requiere un profundo entendimiento de una clase que lo esta usando como un componente, entonces es mejor reusar componentes existentes que hacer uno nuevo.

Por supuesto, la razón principal que un framework es hacer posible el reuso de código porque es un diseño reusable. Este proporciona algoritmos abstractos reusables y un alto nivel de diseño que descompone un sistema grande en pequeños componentes y describe la interfaces internas entre los componentes. Estos interfaces estándares hacen posible mezclar y match componentes, y construir una amplia variedad de sistemas de un pequeño número de componentes existentes. Nuevos componentes que cumplen con estas interfaces encajaran dentro del framework, entonces los diseñadores de componentes también reúsan el diseño de un framework.

Los frameworks es solo una de muchas técnicas de reuso. EL uso ideal de tecnología provee componentes que pueden ser fácilmente conectados para hacer un nuevo sistema. Los desarrolladores no tienen que saber como los componentes fueron implementados, y la especificación del componente es fácil de entender. Como resultado el sistema será eficiente, fácil de mantener, y confiable.

2.2. mHealth

Mobile Health (mHealth) es un término utilizado en el campo de la medicina y se basa en el uso de dispositivos móviles. El campo mHealth ha surgido como una rama de eHealth, que se basa en el uso de Tecnología de Información y Comunicaciones(ICT), como computadores, teléfonos móviles, comunicaciones satélites, monitores de pacientes, etc., para proveer servicios e información para el cuidado de la Salud [9].

Las aplicaciones de mHealth se basa en el uso dispositivos móviles para la recolección de datos clínicos, suministro de información relacionado al cuidado de la salud a practicantes, investigadores y pacientes, monitoreo de signos vitales en tiempo real y la prestación de servicios médicos a través de estos dispositivos conocidos como telemedicina.

La Organización Mundial de la Salud lo define como la práctica médica y de salud pública con el apoyo de dispositivos móviles, tales como teléfonos móviles, dispositivos de monitoreo a pacientes, asistentes digitales personales(PDAs) y otros dispositivos inalámbricos. mHealth implica el uso de voz y el servicio de mensajes cortos(SMS) como utilidad principal, así como funcionalidades y aplicaciones mas complejas, incluyendo GPRS, telecomunicaciones móviles de tercera y cuarta generación, el sistema de posicionamiento global(GPS) y la tecnología Bluetooth[14].

Las aplicaciones de mHealth están organizados por área de aplicación, desde el menos especializado al mas especializado. Aunque algunas aplicaciones abarcan mas de un área, estas se clasifican por su función mas especializada. A continuación se dan descripción de las diferentes áreas:

2.2.1. Educación y Concientización.

En aplicaciones de educación y concientización, mensajes SMS son enviados a los usuarios de teléfonos para ofrecer información de pruebas y tratamientos disponibles en servicios de salud y combate a las enfermedades. Estudios formales y anécdotas demuestran que los mensajes de alerta SMS han tenido mayor influencia incluso mayor que la radio y televisión. Los mensajes SMS tienen la ventaja de de ser relativamente discretos, ofreciendo confidencialidad en zonas donde las enfermedades especialmente HIV/AIDS es a veces un tabú. En los países en desarrollo, los mensajes de alerta SMS tiene el alcance de penetrar áreas rurales donde la ausencia de clínicas, falta de doctores y el escaso acceso a la información de salud a veces no

permite a la personas tomar buenas decisiones para cuidar su salud.

2.2.2. Colección Remota de Información.

Colección de datos es un componente crucial en los programas de salud pública. Los reguladores y centros de salud a nivel nacional, estatal y municipal requieren de información exacta para evaluar las políticas y programas existentes y para formar nuevos programas. En lo países en desarrollo, el área de recolección de información es particularmente importante desde debido a que en muchos partes de la población es raramente posible visitar un hospital. Colectar información en el lugar donde los pacientes viven es vital, la información debería idealmente ser actualizado y accesible en tiempo real. El proceso de colección de datos es más eficiente usando teléfonos móviles y PDAs que rellenando formatos en papel para después teclearse manualmente en la base de datos.

La recopilación de datos por medio de teléfonos móviles ha sido utilizado en muchos países en desarrollo principalmente en proyectos piloto. Estas iniciativas están cerrando la falta de información que existe actualmente de los pacientes en países en desarrollo.

2.2.3. Monitoreo Remoto.

Una de las áreas mas únicas que crece junto con la tendencia de la tecnología móvil es el monitoreo remoto de pacientes. El monitoreo remoto abre nuevas posibilidades para el tratamiento de pacientes, una capacidad crucial en países en desarrollo donde el acceso a los hospitales y clínicas esta limitado. Este grupo de aplicaciones la comunicación puede ser unidireccional o bidireccional para monitorear las condiciones de salud, mantener las citas, o asegurar las medicaciones se sigan al pie de la letra en los pacientes. Algunas aplicaciones incluyen sensores para monitorear condiciones múltiples.

2.2.4. Comunicación y Capacitación a trabajadores de la Salud.

El escasez de médicos es un gran desafío para los países en desarrollo. La formación de nuevos profesionales de la salud y el empoderamiento de los trabajadores actuales con el fin de incrementar la satisfacción en el trabajo y reducir el desgaste son esenciales para satisfacer las necesidades del capital humano. Conectando médicos con fuentes de información por medio de teléfonos móviles es una base solida para el empoderamiento porque provee el soporte que ellos requieren para cumplir con su trabajo más efectivamente y autosuficiente.

Existe una fuerte presión para mejorar la comunicación entre los diferentes unidades de salud para facilitar más eficiente el cuidado de los pacientes. Debido a la escasez de teléfonos fijos y computadoras con acceso a la internet, es muy común por ejemplo, para un paciente que se ha enviado al hospital regional solo para descubrir que no hay una cama disponible. Los teléfonos móviles ayudan a cerrar estas brechas de comunicación que en el contexto de la salud puede significar la diferencia entre la vida y la pérdida de vidas humanas.

2.2.5. Seguimiento de enfermedades y brotes epidémicos.

Los brotes de enfermedades propagables a menudo comienzan en una población y cuando no se detecta a tiempo se puede convertir en una epidemia. Con el uso de dispositivos móviles, con la habilidad de rápidamente capturar y transmitir información sobre incidencia de enfermedades, puede ser decisivo en la prevención y la contención brotes.

2.2.6. Apoyo en el Diagnóstico y Tratamiento.

El diagnostico y tratamiento son de vital importancia en el cuidado de la salud. Un mal diagnostico o la incapacidad para diagnosticar podría tener consecuencias fatales. Las aplicaciones de este grupo están diseñadas para proporcionar un diagnostico y recomendaciones

sobre el tratamiento a los médicos a distancia a través del acceso inalámbrico de bases de datos de información médica o personal médico. Con diagnosticos mHealth los pacientes pueden recibir tratamientos en sus poblados y hogares, evitando las costosas visitas al hospital que están fueran del alcance de muchos pacientes.

2.3. Signos Vitales

Los signos vitales son mediciones de las funciones mas básicas del cuerpo. Los cuatro principales que los médicos y los profesionales de la salud examinan en forma frecuente son los siguientes. Los signos vitales comprenden el ritmo cardíaco, la frecuencia respiratoria, la temperatura y la presión arterial. Los signos vitales normales cambian con la edad, el sexo, el peso, la tolerancia al ejercicio y la enfermedad[11].

Los rangos normales de los signos vitales para un adulto sano promedio mientras esta en reposo con:

Presión arterial: 90/60 mm/Hg hasta 120/80 mm/Hg.

Respiración: 12 a 18 respiraciones por minuto.

Pulso: 60 a 100 latidos por minuto.

Temperatura: 36.5-37.2° C (97.8-99.1° F)/promedio de 37° C (98.6° F).

2.4. Android

Android proporciona un conjunto completo de software para dispositivos móviles: un sistema operativo, middleware y aplicaciones móviles clave.

Android es un sistema abierto. Android fué diseñado desde cero para permitir a los desarrolladores crear aplicaciones atractivas y sacar el máximo provecho de lo que un teléfono tiene que ofrecer. Construido para ser un sistema verdaderamente abierto. Cualquier aplicación puede tener la capacidad de llamar cualesquiera de las funcionalidades del teléfono, como hacer una llamada, enviar un mensaje SMS por citar algunas. Android se construyó sobre el Kernel de Linux. Además de eso, utiliza una máquina virtual personalizado diseñado para optimizar la memoria y los recursos del hardware en un ambiente móvil. Android es código abierto, puede ser libremente extendido para incorporar nuevas tecnologías conformen vayan emergiendo.

Android no diferencia entre las aplicaciones básicas del teléfono y las aplicaciones de terceros. Todas pueden ser construidas para tener igualdad de acceso a las capacidades de un teléfono que proporciona al usuario, una amplia gama de aplicaciones y servicios. Con los dispositivos basados en la plataforma Android, los usuarios son capaces de adaptar totalmente el teléfono a sus intereses.

Android rompe las barreras a la creación de aplicaciones nuevas e innovadoras. Por mencionar uno, un desarrollador puede combinar la información de la web con los datos en el teléfono móvil de un individuo -como los contactos del usuario, la agenda o la ubicación geográfica- para proporcionar una experiencia mas relevante al usuario. Con Android, un desarrollador puede crear una aplicación que permite a los usuarios ver la ubicación de sus amigos y ser alertados cuando se encuentren en las proximidades, dándoles un oportunidad de conectarse.

Android proporciona una amplia gama de bibliotecas y herramientas útiles que se pueden utilizar para construir aplicaciones. Por ejemplo, Android permite a los desarrolladores obtener la ubicación del dispositivo, y permite que los dispositivos se comuniquen entre si enriqueciendo de esta modo las aplicaciones de punto a punto.

Además Android incluye un conjunto completo de herramientas que se han construido desde cero junto a la plataforma para ofrecer a los desarrolladores una alta productividad y un profundo conocimiento de sus aplicaciones [3].

Capítulo 3

Análisis y Diseño

3.1. Requerimientos del Framework

Los requerimientos de un sistema son declaraciones abstractas de alto nivel de un servicio que debe proporcionar el sistema o una restricción de este. En el otro extremo, es una definición detallada y formal de una función del sistema[19]. Entre los tipos de requerimientos encontramos los funcionales y los no funcionales. Los funcionales son requerimientos de los servicios que debe proporcionar el sistema mientras que los requerimientos no funcionales son restricciones de las funciones o servicios ofrecidos por el sistema. A continuación se presentan los requerimientos funcionales y no funcionales del framework.

3.1.1. Requerimientos funcionales

1. El framework deberá contar con interfaces gráficas en forma de líneas, de barras y de pastel para cada uno de los siguientes signos vitales.
 - Frecuencia cardíaca

- Temperatura
- Pulso
- Presión arterial

Se requiere que los componentes de interfaz de usuario cuenten con las funciones necesarias para fácilmente cambiar sus propiedades y adaptables a las necesidades del desarrollador de aplicaciones.

2. Clases para hacer las requisiciones de información al servidor de base de datos. Las clases o métodos deberán contar con temporizadores para evitar que la conexión al servidor se bloquee y la aplicación cliente se quede indefinidamente esperando se colmine la conexión al servidor. Las clases deberán contar con una manera de notificar del éxito o fallo de la requisición de signos vitales al servidor. Las clases requieren revisar la disponibilidad de conexión a la Internet antes de realizar una requisición al servidor de datos.
3. El Servidor de datos deberá contar con un conjunto de funciones o métodos en el servidor de datos para atender las peticiones de las aplicaciones clientes.
4. Se requiere que los componentes gráficos de interfaz de usuario sea de fácil uso y de inclusión en las aplicaciones en desarrollo.

Limitaciones del Framework.

1. El framework funciona a partir de signos vitales almacenados en una base de datos MySQL.
2. El framework no funciona para monitoreo de signos vitales en tiempo real.

3. El propósito del framework es auxiliar a los médicos con las aplicaciones que de los desarrolladores implementen.
4. El framework no tiene la capacidad para automáticamente diagnosticar ningún tipo de enfermedad a partir de los signos vitales.
5. El framework no funciona para dispositivos Blackberry o dispositivos con sistema operativo iOS.

3.1.2. Tecnologías Utilizadas.

Para el desarrollo del framework se utilizarán las siguientes tecnologías o herramientas de software.

1. Apache Tomcat.
2. MySQL Community Server 5.6.13
3. JSON.
4. Eclipse IDE.
5. Jersey RESTful Web Services framework
6. Android SDK Tools
7. Android Development Tools (ADT).
8. Android Support Library package.
9. MySQL Connector/JDBC.

3.2. Intefaz Gráfica del Usuario

El interfaz de usuario es uno de los más importantes del cualquier sistema computacional, pues funciona como el vínculo entre el humano y la máquina. La interfaz de usuario es un conjunto de protocolos y técnicas para el intercambio de información entre una aplicación computacional y el usuario [12].

El interfaz de usuario en las aplicaciones Android se representa por una Actividad(Activity). Una Actividad(Activity) es el componente de una aplicación que proporciona una pantalla con el cual el usuario puede interactuar para hacer algo como marcar el teléfono, enviar un e-mail o visualizar un mapa [4]. Cada actividad se le asigna una ventana en donde se puede dibujar el interfaz de usuario. El interfaz de usuario de una actividad se maneja por medio de una herencia de vistas derivadas de la clase View. Cada vista controla un espacio de rectángulo dentro de una ventana de la actividad y responde a las interacciones con el usuario. Los layouts son vistas derivadas de ViewGroup que proporciona un modelo de estructura para las vistas que contiene.

También se puede derivar de la clase View o ViewGroup o subclases existentes para crear un componente de UI personalizado o adaptado y después ser utilizados en una actividad.

Android proporciona componentes de interfaz de usuario listos para ser utilizados en su SDK y diferentes tipos de layouts para organizar la estructura visual del interfaz de usuario. Pero estos componentes son los comunes que muchos otros SDK de intefaz de usuario proporcionan. En particular Android provee Button, TextView, EditText, ListView, CheckBox, RadioButton, Gallery, Spinner, AutoCompleteTextView, ImageSwitcher y TextSwitcher.

Cuando ninguno de los componentes predefinidos llena las expectativas del desarrollador de aplicaciones o los requerimientos de lo que es necesario desplegar, se puede crear uno propio extendiendo la clase View. Si solo se requiere hacer pequeños ajustes a los componentes ya

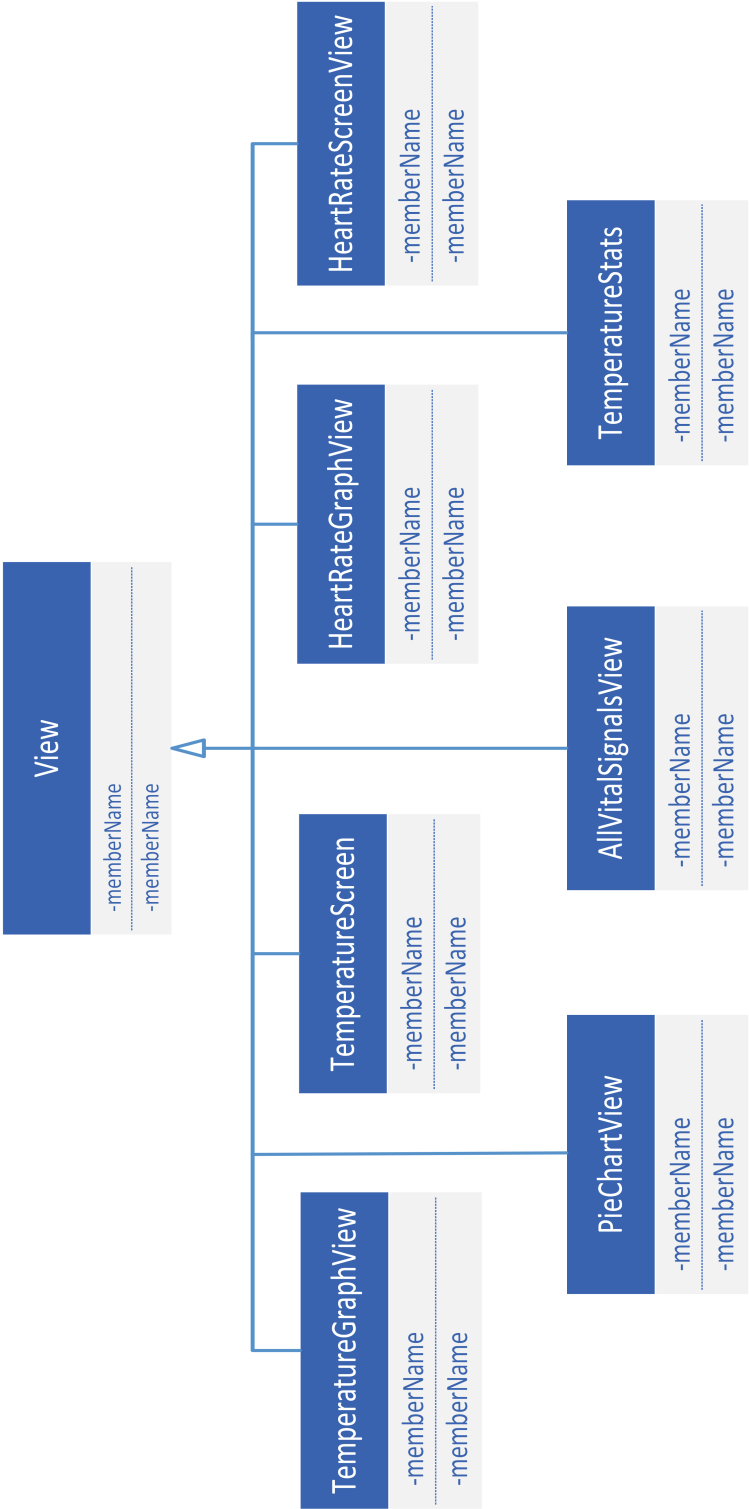


Figura 3.1: Organización de componentes de interfaz de usuario.

existentes, se puede extender la subclase ya existente y sobrecargar sus métodos.

Android no proporciona componentes básicos listos para visualizar aplicaciones de salud o de signos vitales. Por lo cual se han creado algunos componentes de interfaz de usuario. En esta sección se explica cada uno de ellos.

3.2.1. HeartRateGraphView

HeartRateGraphView es un componente personalizado que se hereda de la clase View. Con este componente se puede desplegar el ritmo cardíaco en forma de líneas y de barras. Se puede visualizar y analizar los cambios en el ritmo cardíaco de los pacientes en función del tiempo.

Este componente se puede incluir en una actividad por medio de un archivo plantilla(layout) o bien crear una instancia en forma de programación durante el inicio de la actividad.

Una vez incluido el componente en una actividad se le puede asignar los datos para su despliegue pasándole al componente el arreglo que contiene los signos vitales. Esto se hace mediante una invocación a un API web service mediante la clase ConcreteDataRequest. Las actividades donde se utiliza el HeartRateGraphView se requiere implementar la interface ConnectionListener, ya que por este medio la actividad es notificada por el éxito o fallo de la adquisición de la información en el servidor por medio de los métodos HandleInput() y HandleError().

La información de los signos vitales solicitada al API web es regresada en formato JSON y convertido a un arreglo de datos tipo double para su asignación al componente HeartRateGraphView que cuenta con los métodos setTemperatureValues() y setHorlabels() para este propósito.

3.2.2. TemperatureGraphView

Este componente de android se puede visualizar la temperatura corporal de un paciente en forma de linear y barras.

3.2.3. PieChartView

Con este componente se puede utilizar para representar proporciones y porcentajes estadísticos de los signos vitales. Con PieChartView el desarrollador puede fácilmente visualizar los signos vitales a los doctores para su examinación e interpretación. El médico podrá visualizar la cantidad de un determinado signo vital o signos vitales que están dentro de la normalidad, abajo de la normalidad y por arriba de lo normal esperado en los seres humanos. Por medio de las variables miembro mínimo y máximo se puede calcular el número de signos vitales que caen dentro de lo normal, abajo de lo normal y arriba de los normal.

3.3. Conexión a Servicios Web

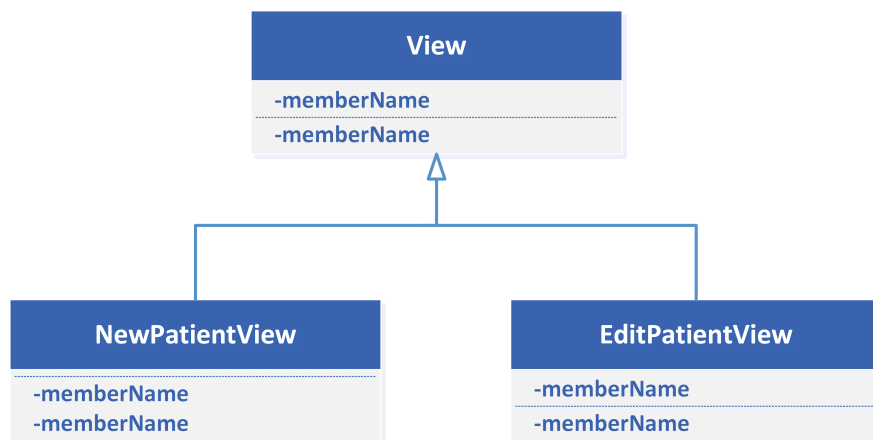


Figura 3.2: Diagrama de Interfaz de Pacientes.

La arquitectura u organización de las clases para hacer el requerimiento de los signos

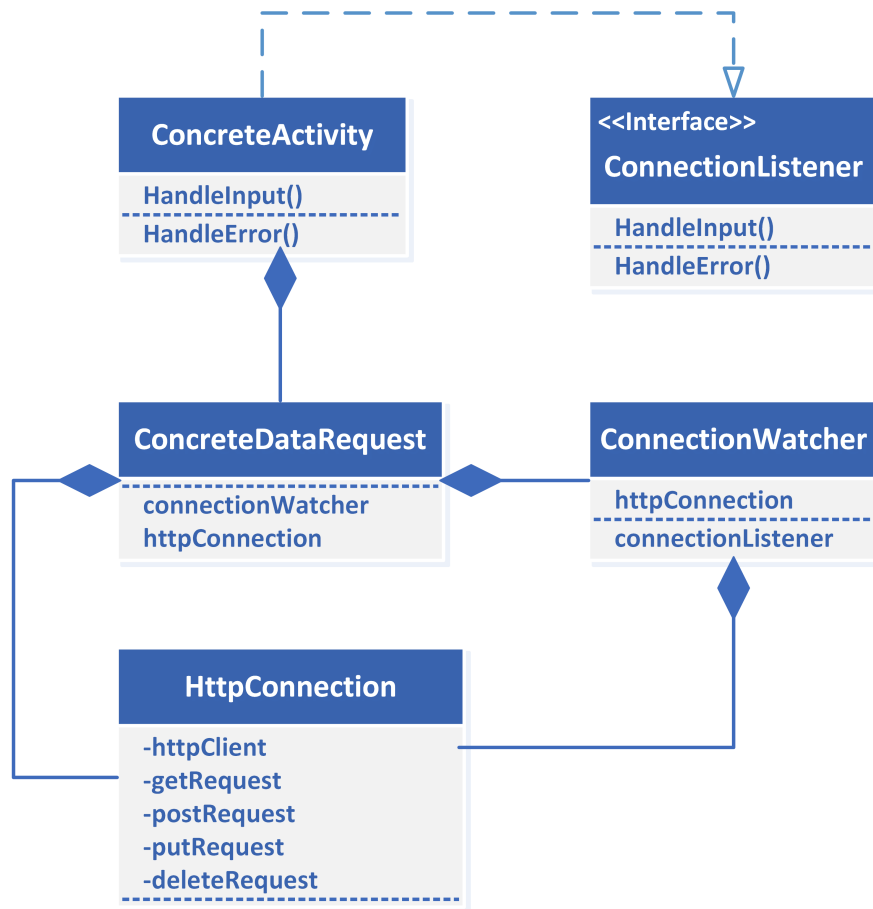


Figura 3.3: Interacción de clases para conexión a un servicio web

vitales de un determinado paciente se puede observar en la Figura 3.3.

Partiendo de una actividad de Android en concreto en donde se puede incluir cualesquiera de los componentes gráficos de signos vitales. Para llenar de datos el componente incluido, la actividad requiere implementar la interface de **ConnectionListener** que cuenta con los métodos `HandleInput()` y `HandleError()`. A través de los cuales la actividad es notificada del éxito de la solicitud o algún error se ha ocurrido durante la sesión correspondientemente.

Para requerir determinados signos vitales al servidor, se hace uso de un tipo concreto de pedido representado por la clase **ConcreteDataRequest**. La clase **ConcreteDataRequest** esta compuesto de objetos de tipo **HttpConnection**, **ConnectionWatcher** y **ConnectionListener**. El objeto **ConnectionListener** corresponde a la actividad de la cual se han requerido los datos

de signos vitales; el objeto de `HttpConnection` es el que se encarga de la transacción del recurso de los signos vitales utilizando una URI de acuerdo al tipo de transacción que esta en proceso. `ConnectionWatcher` se encarga de estar monitoreando la situación de la conexión http con el servidor cada determinado tiempo.

Dentro del objeto `ConcreteDataRequest` se inicializan 2 procesos al momento de hacer el requerimiento de datos solicitado al servidor de bases de datos. Estos procesos corresponden a los objetos `ConnectionWatcher` y `HttpConnection`. Al crear la instancia de estas clases se les pasa como argumento la actividad que esta requiriendo los datos como tipo `ConnectionListeners`.

`HttpConnection` es la clases que se encarga de la transacción con el servidor de bases de datos mientras que la `ConnectionWatcher` se encarga de estar supervisándolo cada determinado tiempo. Revisando si la transacción se ha concretado, o si algún error ha ocurrido o el tiempo limite de espera de transacción ha expirado. Asegurando de esta manera que la conexión con el servidor no permanezca por mucho tiempo y este cause que la aplicación se quede suspendida indefinidamente y afecte el funcionamiento del dispositivo Android.

`ConnectonWatcher` se encarga de notificar a la actividad desde la cual se hizo la requisición de signos vitales si la conexión con el servidor ha tenido éxito o no. Si la transacción con el servidor ha tenido éxito este lo comunica a la actividad por medio del objeto `ConnectionListener` utilizando el método `HandleInput()` a través del cual también le envía los datos obtenidos del servidor. En caso contrario notifica a la actividad a través el método `HandleError()` junto con el tipo de error que se ha ocurrido para que el desarrollador de aplicaciones pueda desplegar el error correspondiente para notifiaciñ al usuario.

3.4. Servicios Web

Un servicio web es un sistema de software diseñado para permitir la interoperabilidad máquina a máquina en una red. Cuenta con un interface descrito en un formato maquina-procesable(WSDL). Otros sistemas interactúan con el sistema web en la manera descrita usando mensajes SOAP transmitida mediante HTTP con una serialización XML en conjunto con otros estándares relacionadas con la Web [5].

Un servicio web es una noción abstracta que requiere ser implementado por un agente concreto. Un agente es una pieza concreto de software o hardware que envía y recibe mensajes, mientras que el servicio es un recurso caracterizado por un conjunto de funciones que provee.

El propósito de un servicio web es proveer alguna funcionalidad en favor de una organización, como un negocio o individuo. La entidad proveedor es la persona o organización que provee un agente apropiado para implementar un servicio particular.

La entidad solicitante es una persona o organización que desea hacer uso del servicio web de la entidad proveedor. Este utiliza un agente solicitante para intercambiar mensajes con el agente proveedor de la entidad proveedor.

Para los propósitos de desarrollo de este trabajo se utiliza Servicios Web RESTful.

3.4.1. RESTful

En Ingeniería de Software el término estilo de arquitectura de software normalmente se refiere a un conjunto de reglas de diseño que identifica los tipos de componentes y conectores que se pueden usar para componer un sistema o subsistema [18].

En el campo de los Servicios Web, REpresentational State Tranfer(REST) es un lenguaje de diseño clave que abarca una arquitectura cliente-servidor sin estado en el cual los servicios

web son vistos como recursos y se indentifican por sus URIs.

Los clientes de los Web Services que desean utilizar estos recursos acceden a una representación en particular al transferir el contenido de una aplicación usando un conjunto de métodos remotos globalmente definidos que describe la acción a realizar sobre el recurso.

REST es un tipo de diseño para acceder a las paginas web. REST estructuralmente representa URI con el método de transferencia del Solicitud preguntando el resultado desde el servidor e intercambia la información basado en el protocolo HTTP.

Utiliza una arquitectura client-servidor. REST no limita la comunicación client-servidor a un solo protocolo en especifico, pero el mas comun que utiliza es el HTTP. Los servicios web RESTful se puede describir utilizando el lenguaje de descripción de aplicaciones web. Un archivo WADL describe las peticiones que legitimante se pueden abordar en un servicio, incluso los URIs del servicio y los datos que el servicio espera y sirve.

REST se base en cuatro principios de diseño y cuatro propiedades.

1. Recurso

- Un recurso es cualquier cosa lo suficientemente importante como para ser referenciado. Normalmente un recurso es algo que puede ser almacenado en una computadora y representado como flujo de bits como un documento, un renglón de una base de datos, o el resultando de ejecutar un algoritmo.

2. URIs

- Un recurso debe contar con un identificador uniforme de recursos(URI). El URI es el nombre y dirección de un recurso. Si una pieza de información no cuenta con un URI no puede considerarse como un recurso y no esta realmente en la Web, pu un como un bit de dato describiendo algún otro recurso.

3. Direccionabilidad

- Una aplicación es direccionable si expone interesantes aspectos de sus datos como recursos. Los recursos son expuestos por medio de sus URIs, una aplicación direccionable expone un URI por cada pieza de información que sirve. Esto conduce en un infinito número de URIs.

4. Sin Estado

- Cada request HTTP del client al servidor se realiza en completo aislamiento. Cuando el cliente realiza un request HTTP al servidor, este incluye toda la información necesaria para que el servidor lleve a cabo la requisición. El servidor en ningún momento depende de información de requisiciones previas. El servidor nunca se basa en información de solicitudes anteriores para responder a una nueva.

5. Representaciones

- Cuando una aplicación se divide en diferentes recursos, se incrementa su área de superficie. Los usuarios pueden construir un URI apropiado y acceder a la aplicación en donde se requiere. Sin embargo, los recursos no son los datos, sino que son simplemente la idea del diseñador de servicios de la forma que divide los datos. Un servidor no puede enviar una idea, tiene que enviar una serie de bytes en un formato de archivo específico. Esta es la representación del recurso.

Un recurso de una fuente de representaciones, y una representación es solo datos acerca del estado actual de un recurso. El servidor puede presentar una lista de bugs abiertos como un documento XML, formato JSON, página web o texto separados por comas.

6. Interface Uniforme

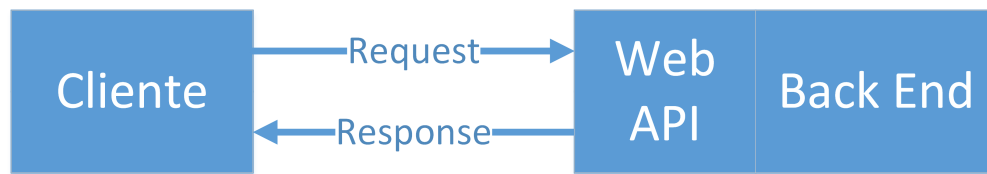


Figura 3.4: Modelo de espiral

- La interacción entre los componentes Web depende de la uniformidad de sus interfaces. Si cualquiera de los componentes no sigue los estándares establecidos la comunicación se viene abajo. Los componentes web operan dentro de cuatro restricciones identificadas por Fielding. Identificación de recursos Manipulación de los recursos por medio de representaciones. Mensajes Autodescriptivos. Hipermedia como el motor del estado de la aplicación.

Recursos REST basados en HTTP se puede acceder a través del estándar HTTP y una interfaz uniforme. En la Web, existe un conjunto de operaciones que se puede hacer sobre un recurso. HTTP proporciona cuatro métodos básicos para las operaciones más comunes. Crear, Leer, Actualizar y Eliminar (CRUD)

- Obtener la representación de un recurso (HTTP GET).
- Crear un nuevo recurso (HTTP POST).
- Actualizar un recurso existente (HTTP PUT)
- Borrar un recurso existente (HTTP DELETE).

3.4.2. RESTful APIs

Los programas del lado del cliente utilizan interfaz de programación de aplicaciones (API) para la comunicación con los servicios web. Un API expone un conjunto de datos y aplicaciones para facilitar las interacciones entre los programas de computación y así hacer posible el

intercambio de información 3.4. Para fines del presente trabajo, se requieren APIs para intercambiar información de signos vitales entre las aplicaciones móviles Android y el servidor de base de datos. Los APIs con los que se cuenta y que se han implementado para interactuar con el servidor de base de datos se explican a continuación. Se han clasificado en 2 partes y estos consisten en funciones para manipular o manejar el registro de los pacientes y funciones para el manejo de los registros de los signos vitales.

APIs Para Administración de Registros de Pacientes

1. createPatient

Este API se utiliza para insertar un nuevo registro en la tabla patients de la base de datos vitalsignsserver. Este función recibe como datos de entrada la descripción de datos generales del paciente y los agrega al tabla antes mencionado utilizando la clausula INSERT de MySQL. Se encarga de notificar a la aplicació cliente del éxito o fallo de la operación.

2. getPatients

- Este API se utiliza para obtener un lista de pacientes que corresponde a un cierto medico. Como dato de entrada recibe el número de indentificación del medico. Devuelve una lista de pacientes o el código de error en caso de fallo.

3. updatePatient

- Se puede presentar el caso en que un medico requiere cambiar o modificar determinado dato de un paciente. El API updatePatient se utiliza para modificar los datos de un paciente. Como dato de entrada recibe los nuevos datos de paciente y utilizando la clausula UPDATE de MySQL actualiza los datos del paciente. Regresa un código de error a la aplicación cliente o de fallo según sea el caso.

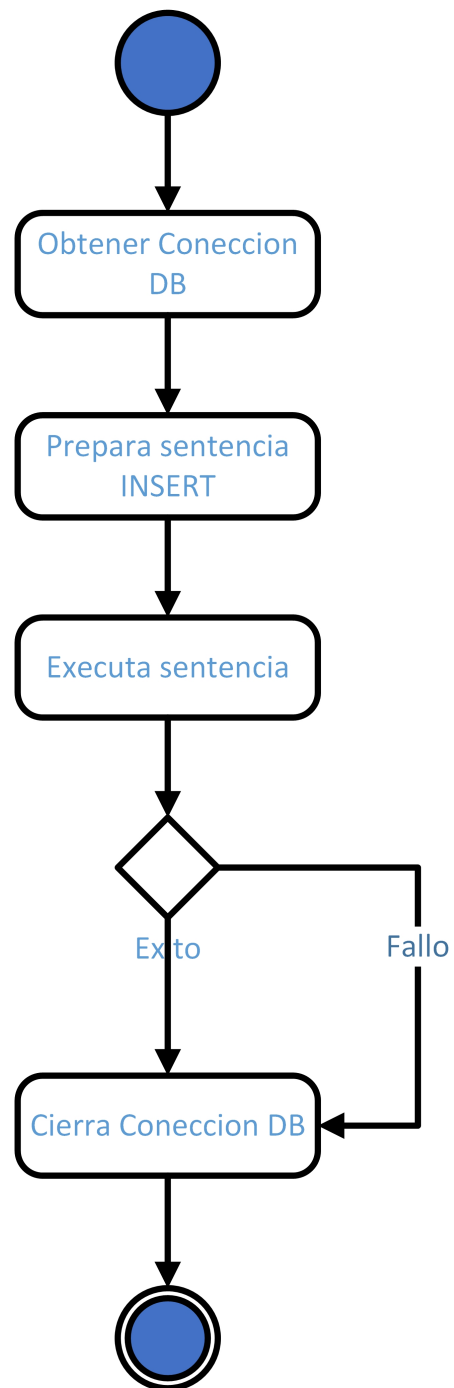


Figura 3.5: Diagrama de actividades del API createPatient.

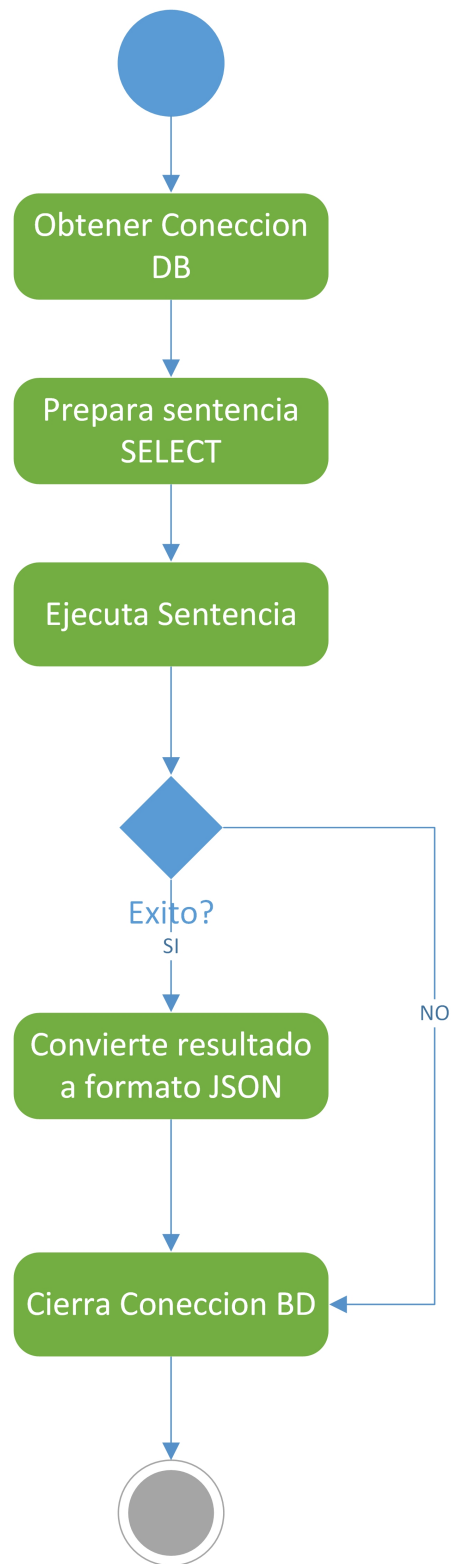


Figura 3.6: Diagrama de actividades del API getPatients.

4. removePatient

- Este API se utiliza para borrar el registro de un paciente.

5. getPatient

- Este API para obtener los datos de un determinado paciente. Como dato se entrada se requiere el numero de indentificacion del paciente. Se regresa en registro del paciente o código de error en caso de que no se haya hallado.

APIs Para Administración de Registros de Signos Vitales

1. getVitalSignsBySingleDate

- Este API se utiliza para obtener los signos vitales de un determinado fecha. Parametros: patient_id, measure_type y date.

2. getVitalSignsByRangeDate

- Este API se utiliza para obtener los signos vitales de un determinado rango de fechas y los devuelve en un arreglo JSON. Parametros: patient_id, measure_type, start_date y end_date.

3. getVitalSignsStatsBySingleDate

- Este API se utiliza para obtener los signos vitales de un determinado fecha. Obtiene la ultima muestra tomada, el maximo y el minimo muestra tomada durante la fecha indicada. Parametros: patient_id, measure_type y date.

4. getVitalSignsStatsByRangeDate

- Este API se utiliza para obtener los signos vitales de un determinado rango de fechas. Obtiene la ultima muestra tomada, el maximo y el minimo muestra tomada durante el rango de fecha indicado. Parametros: `patient_id`, `measure_type`, `start_date` y `end_date`.

Capítulo 4

Pruebas y resultados

4.1. Utilización del Framework

Se presentan 3 ejemplos como demostración de la utilización del framework. En el primer ejemplo se explican los componentes que se utilizan en la aplicación y la manera que estos presentan los signos vitales que se obtienen de la base de datos utilizando servicios web. La forma en que se obtiene los datos en los otros 2 ejemplos es muy similar o parecido a como se hace en el primer ejemplo. Por lo cual se ha decidido solo explicar el primero.

En la figura 4.1 se puede apreciar una demostración de la presentación de temperatura de un paciente. Es una actividad que contiene varios componentes del framework. Para el ejemplo de la utilización del framework se utiliza un layout en donde se han colocado los componentes de interfaz de usuario que se explican en la siguiente lista:

1. `DateRangeView`

- Este componente se puede utilizar para desplegar la fecha de inicio y de fin que corresponden a las mediciones de se muestran en un otro componente. En este caso se

utiliza para indicar el rango de fechas que corresponde la temperatura del paciente que se muestra en los componentes de `TemperatureGraphView`. `DateRangeView` contiene las propiedades de fecha de inicio y fecha de fin el cual una vez que se han asignado valores este se encarga automáticamente de desplegar los valores sin que el desarrollador de aplicaciones tenga que preocuparse por conversiones de tipo de fecha a los formatos de fecha.

2. `TemperatureStats`

- Este componente es una especie de tabla que muestra la cantidad de signos vitales de manera resumida para un análisis rápido de los signos vitales que están dentro de lo normal esperado de un paciente promedio normal, arriba de lo normal y abajo de lo normal; Así como el porcentaje que representa cada uno de las categorías y la descripción de cada renglón.

3. `PieCharView`

- Este componente muestra una gráfica en forma circular que se puede utilizar para visualizar porcentajes y proporciones de un determinado signo vital. En este ejemplo se utiliza para mostrar los signos vitales que están dentro de lo normal esperado, los signos vitales que estan arriba y abajo de lo normal.El componente solo se requiere pasarle el arreglo de valores de los signos vitales y este se encargara automáticamente de realizar los cálculos para gráficar las proporciones.

4. `TemperatureGraphView`

- Este componente despliega un determinado signo vital en forma de gráfica de líneas con capacidad de funciones de acercamiento y alejamiento. Muestra como un signo vital cambia con respecto al tiempo. Se puede visualizar si los signos

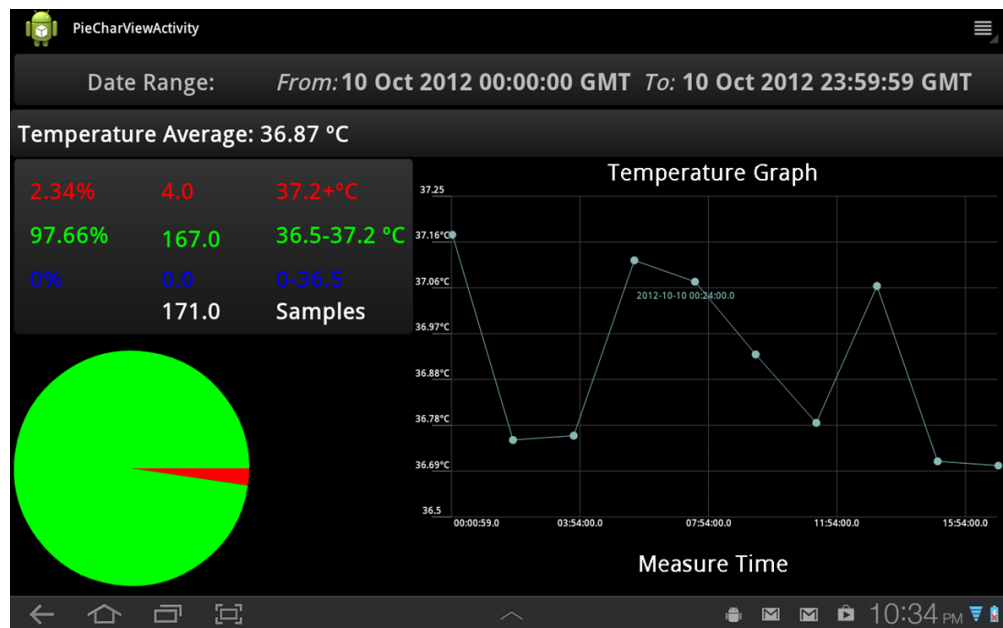


Figura 4.1: Ejemplo 1 de la utilización del framework

vitales están mejorando o empeorando. Muy útil para visualizar la tendencia de los signos vitales de un paciente dentro de un determinado periodo de tiempo.

En el ejemplo 2 de la utilización del framework el desarrollador de aplicaciones puede usar el componente en forma de lista para desplegar la temperatura del paciente. Cuando la temperatura corporal del paciente se encuentra fuera del rango de los normal, la información de la temperatura en el región utilizan un icono de termómetro de color rojo mientras para los casos que se encuentran dentro de la normalidad de utiliza el icono termómetro de color verde.

En el ejemplo 3 de la aplicación es un componente que muestra la máxima, mínima y el más reciente temperatura de un paciente de una determinada fecha. Con la aplicación el usuario tiene la habilidad de navegar a través de las fechas mediante los botones que se muestra en la parte inferior del componente.

El layout descrito en la figura anterior se utiliza en la Actividad y normalmente en Android



Figura 4.2: Ejemplo 3 de la utilización del framework

cuando se utiliza un layout en una actividad este se indica en el método `onCreate` haciendo uso del método `setContentView` para utilizar el layout como interfaz UI en la actividad. Esto se puede apreciar en el Listado 4.1.

```

1  @Override
2  public void onCreate(Bundle savedInstanceState)
3  {
4      super.onCreate(savedInstanceState);
5      setContentView(R.layout.activity_temperature_measures);
6      graphView = (TemperatureGraphView) findViewById(R.id.
7          temperatureGraphView1);
8      pieChartView = (PieChartView) findViewById(R.id.pieChartView1);
9      tempStats = (TemperatureStats) findViewById(R.id.temperatureStats1);
10     average = (TextView) findViewById(R.id.average);
11     dateRangeView = (DateRangeView) findViewById(R.id.dateRangeView1);
12     handler = new Handler();
13     registerForContextMenu(graphView);
14     getTemperatureDate();
15 }

```

Listado 4.1: Inflando el layout para utilizarlo en la actividad.

En el método `getTemperatureDate()` manda a llamar los datos de signos vitales. Para que esto sea posible, la actividad requiere implementar la interface `ConnectionListener` el cual

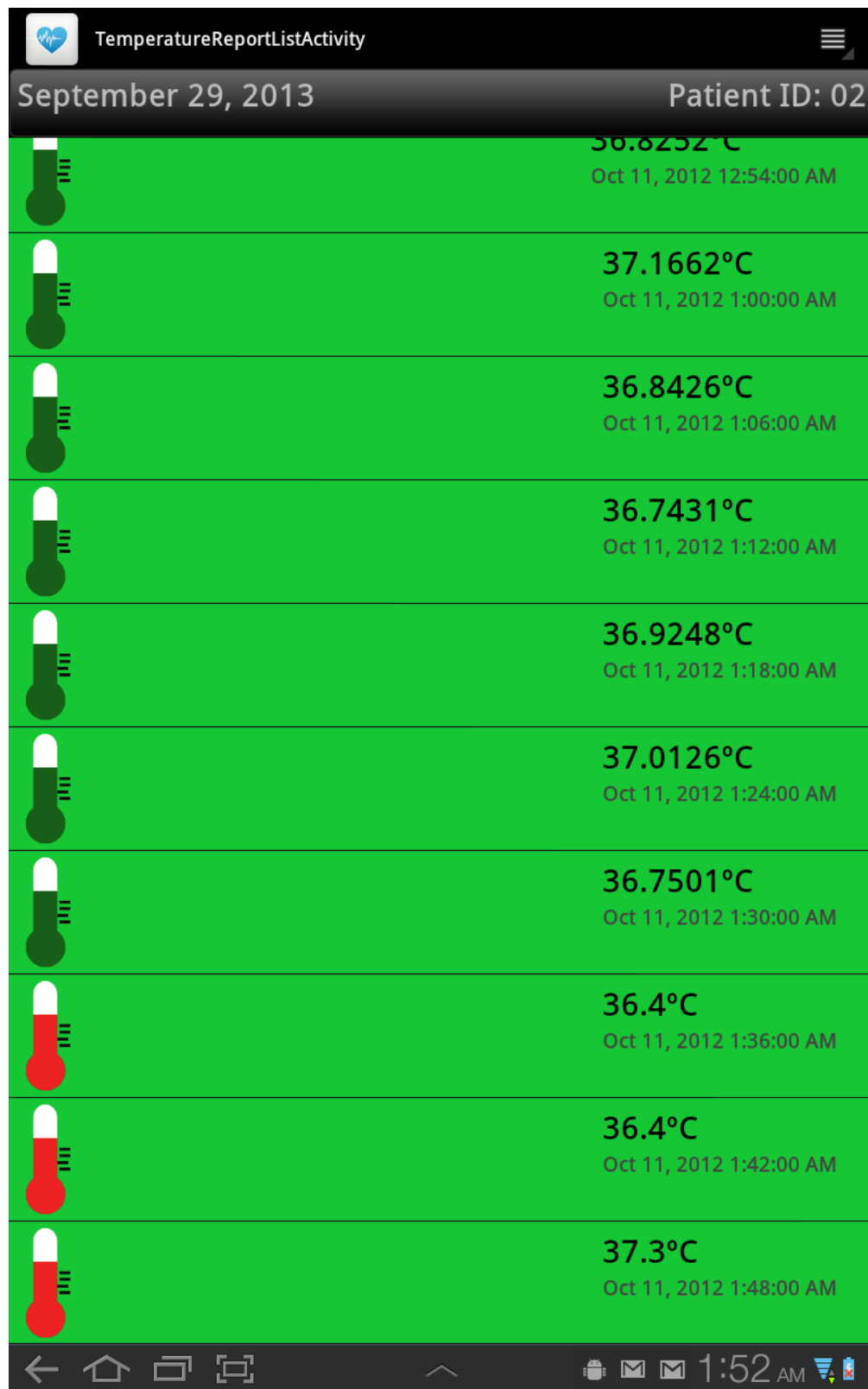


Figura 4.3: Ejemplo 2 de la utilización del framework

contiene los métodos abstractos *handleInput* y *handleError* y requieren ser implementados en la actividad. Listado 4.2.

```
1  private void getTemperatureDate ()
2  {
3      dialogLoading = ProgressDialog.show(PieChartViewActivity.this ,
4                                          "", Html.fromHtml("<big>Connecting
5                                          ...</big>"), true);
6
7      Calendar calendar = Calendar.getInstance();
8      calendar.set(Calendar.YEAR, 2012);
9      calendar.set(Calendar.MONTH, 9);
10     calendar.set(Calendar.DAY_OF_MONTH, 11);
11     calendar.set(Calendar.HOUR, 0);
12     calendar.set(Calendar.MINUTE, 0);
13     calendar.set(Calendar.SECOND, 0);
14     Date fromDate = calendar.getTime();
15
16     calendar.set(Calendar.HOUR, 23);
17     calendar.set(Calendar.MINUTE, 59);
18     calendar.set(Calendar.SECOND, 59);
19
20     Date toDate = calendar.getTime();
21     new Methods(this).getVitalSignsByRange(2, 1, fromDate, toDate);
22 }
```

Listado 4.2: Requisición de datos de temperatura al servicio web.

Del código se puede notar que la conexión al servidor para hacer la requisición de los datos es muy sencillo. Se activa la ventana de dialogo de “Connecting” para inmediatamente llamar el método *getVitalSignsByRange* que se encargara de obtener los datos del servicio web.

Después de haber llamado que se ha hecho la requisición de los datos solo queda esperar la notificación del fallo o éxito de la requisición de los datos. La actividad será notificada por medio de los métodos *handleInput* si la requisición tuvo éxito o por medio del método *handleError* en caso de fallo y desplegar el mensaje error ocurrido.

Para la recepción de los signos vitales de temperatura de igual manera es sencillo asignar los datos a los componentes en donde se requieren.

El método *handleInput* recibe los datos de signos vitales mediante un argumento de tipo

de cadena en formato JSON. El desarrollador de aplicaciones fácilmente puede convertir esta cadena a formato JSON y extraer los valores de temperatura con su correspondiente tiempo en que se tomo la muestra. Como se puede ver en Listing 4.3 esto se realiza de una manera muy sencilla. Los arreglos de tipo JSON se convierten a tipo ArrayList son asignados a los diferentes componentes de interfaz de usuario para su despliegue. Finalmente se desactiva el diálogo de **Connecting**.

En caso de alguna falla en la requisición de los signos vitales el método *handleError* es llamado. Este recibe como parametros el código de error y el mensaje de error de los cuales el desarrollador de aplicaciones se puede guiar para desplegar el mensaje al usuario después de haber desactivado la ventana de dialogo de **Connecting**.

```
1 public void handleInput(String input)
2 {
3     JSONObject obj;
4     try
5     {
6         obj = new JSONObject(input);
7         JSONArray measure_array = obj.getJSONArray("measures");
8         JSONArray measure_time = obj.getJSONArray("measures_time");
9         values = Utils.JSONArrayToArrayList(measure_array);
10        hor_desc = Utils.JSONArrayToArrayListString(measure_time);
11
12        handler.post(new Runnable()
13        {
14            public void run()
15            {
16                //Set data to the graph of lines
17                graphView.setTemperatureValues(values);
18                graphView.setHorlabels(hor_desc);
19                //Set data to the statistic graph
20                tempStats.setMaxValue(graphView.getMaxValue());
21                tempStats.setMinValue(graphView.getMinValue());
22                tempStats.setValues(values);
23                //Set data to the PieChartView
24                ArrayList<Integer> mSlides = new ArrayList<Integer>(1);
25                mSlides.clear();
26                mSlides.add(graphView.getHighPoints());
27                mSlides.add(graphView.getMediumPoints());
28                mSlides.add(graphView.getLowPoints());
29                pieChartView.setSlides(mSlides);
30                //Seth averaga temperature to the TextView
31                DecimalFormat df = new DecimalFormat("###.###");
32                average.setText("Temperature Average: " + df.format(
33                    graphView.getTemperatureAverage()) + " C");
34
35                dialogLoading.dismiss();
36            }
37        });
38    } catch (JSONException e)
39    {
40        e.printStackTrace();
41    }
42 }
```

Listado 4.3: Recepción de datos temperatura del servicio web.

```
1 public void handleError(int errorCode, String errorMsg)
2 {
3     dialogLoading.dismiss();
4     DisplayErrorMessage(errorMsg);
5 }
```

Listado 4.4: Notificación de error en el proceso de requisición de datos en el servicio web.

4.2. Desarrollo de aplicación sin el framework.

El desarrollo de la aplicación de la Figura 4.1 llevaría mas líneas de código y tiempo de desarrollo sin el framework. Solo el componente *TemperatureGraphView* consume mas de 1000 líneas de de código de desarrollo. Aunado a los componentes *DateRangeView*, *TemperatureStats*, *PieCharView* serían bastantes líneas de código comparado con las 200 lineas de código aproximadamnte que se requiere con la utilización el framework. La utilización del framework proporciona enormes ventajas como se puede observar mediante la reutilizacion de los componentes. Una ventaja mas es la requisición de datos utilizando APIs del servicio web. Solo se requiere crear una instancia de clase Methods y llamar el API que se requiere de acuerdo al componente que se quiere utilizar.

Capítulo 5

Conclusiones

En este trabajo se ha presentado el desarrollo de un framework en Java orientado al desarrollo de aplicaciones móviles de monitoreo de signos vitales.

El framework consiste es un conjunto de componentes de interfaz de usuario, clases para la requisición del datos cliente-servidor y en el lado del servidor un conjunto de APIs para atender la demanda de las aplicaciones cliente.

Los componentes de interfaz de usuario facilitan la presentación de la temperatura corporal, ritmo cardiaco, frecuencia respiratoria y la presión arterial de un paciente en las aplicaciones móviles.

Las clases para la requisición de datos al servidor hacen fácil la obtención de signos vitales en el servidor para ser utilizados en las aplicaciones móviles.

En el lado del servidor se desarrollaron un conjunto de APIs para atender las peticiones de datos de las aplicaciones cliente al servidor utilizando servicios web RESTful.

Se presentaron 3 ejemplos de aplicaciones utilizando el framework desarrollado con el propósito de mostrar la facilidad de uso del framework. En las aplicaciones se mostró lo

amigable que es el framework para el desarrollo de aplicaciones móviles orientado a monitoreo de signos vitales.

La propósito general de este trabajo es el desarrollo de un framework para hacer fácil del desarrollo de aplicaciones móviles. Con las 3 aplicaciones que se presentaron se puede concluir que el objetivo principal se ha alcanzado. Los componentes de interfaz de usuario se pueden reutilizar en diferentes aplicaciones en forma de librería. El conjunto de clases para la obtención de datos se datos al servidor se puede utilizar independientemente y utilizar los componentes de interfaz de usuario para desplegar los datos obtenidos o en los componentes propios introducidos por el desarrollador.

El conjunto de APIs de Servicios Web también son reutilizables. El desarrollador solo requerirá instalar las librerías de API en un servidor e indicar en la dirección IP del servidor MySQL y el servidor estará listo para atender las peticiones de las aplicaciones cliente requiriendo los signos vitales.

Con los componentes de interfaz de usuario, las clases de obtencion de datos y los APIs en el servidor se puede concluir que se han alcanzado con las metas. A partir del framework se pueden desarrollar aplicaciones sencillamente y en corto tiempo comparado que si se desarrollará la misma aplicación sin el empleo del framework.

Se han utilizado herramientas de software libre entre las cuales podemos mencionar Apache Tomcat, MySQL, Jersey Json y Jersey Server. Todas estas herramientas aunado al framework desarrollado en este trabajo hacen que las aplicaciones sean de bajo costo y el tiempo de desarrollo sea relativamente corto.

5.1. Trabajo futuro

El framework presentado en este trabajo hace uso de los signos vitales previamente almacenados en un servidor de base datos. Una posibilidad de trabajo a futuro es hacer monitoreo de los signos vitales en tiempo real. Se puede ampliar el framework agregando más componentes de interfaz de usuario. También en un futuro se podría hacer minería de datos con los signos vitales.

Bibliografía

- [1] Reusing and object oriented frameworks. 1995.
- [2] Nikolaos G. Bourbakis Alexandros Pantelopoulos. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE*, 36(1):1, 2010.
- [3] Open Handset Alliance. Android, December 2013.
- [4] Android. Api guides, February 2012.
- [5] W3C Consortium. Web services architecture, February 2004.
- [6] Lorena Guereña Constantino. *Monitoreo continuo y no intrusivo de signos vitales basado en una red inalámbrica de sensores experimental*. Universidad Autónoma de Baja California, Tijuana,B.C., 2012.
- [7] Vital Wave Consulting. mhealth for development: The opportunity of mobile technology for healthcare in the developing world. *Technology*, pages 6–7, 2009.
- [8] Vital Wave Consulting. mhealth for development: The opportunity of mobile technology for healthcare in the developing world. *Technology*, pages 6–7, 2009.
- [9] Vital Wave Consulting. mhealth for development: The opportunity of mobile technology for healthcare in the developing world. *Technology*, 2009.

-
- [10] Ralph Johnson John Vlissides Erich Gamma, Richard Helm. *Gang of Four - Design Patterns, Elements of Reusable Object Oriented Software*. Addison-Wesley, Baarn, Holland, 1995.
 - [11] Medline Plus Trusted Information for You. Vital signs, January 2013.
 - [12] J. Larson. *Interactive software*. Yourdon Press., Englewood Cliffs, New Jersey, 1992.
 - [13] Douglas C. Schmidt Mohamed Fayad. Object-oriented application frameworks, October 1997.
 - [14] World Health Organization. mhealth: New horizons for health through mobile technologies: Based on the findings of the second global survey on ehealth. *Global Observatory for eHealth Series*, 3, 2011.
 - [15] Brian Foote Ralph E. Johnson. Designing reusable classes. June 1998.
 - [16] Vincent F. Russo Ralph E. Johnson. Reusing object oriented designs. 1991.
 - [17] Johnson R. Roberts D. Evolving frameworks: A pattern language for developing object oriented frameworks, November 2013.
 - [18] M. Shaw and M. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996.
 - [19] Ian Sommerville. *Ingeniería del software*. Pearson Educación,S.A., Reading, Massachusetts, 2005.