DESARROLLO DE UNA APLICACIÓN MÓVIL PARA TRANSMISIÓN Y RECEPCIÓN DE PARÁMETROS FISIOLÓGICOS BÁSICOS DETECTADOS CON UN MÓDULO PORTÁTIL.

REALIZADO POR: Quintana Barillas Vanessa Alexandra

TUTOR: Prof. Iván Escalona

FECHA: Caracas, 27 de Febrero de 2013

UNIVERSIDAD CATÓLICA ANDRÉS BELLO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE
TELECOMUNICACIONES

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA TRANSMISIÓN Y RECEPCIÓN DE PARÁMETROS FISIOLÓGICOS BÁSICOS DETECTADOS CON UN MÓDULO PORTÁTIL.

REALIZADO POR: Quintana Barillas Vanessa Alexandra

TUTOR: Prof. Iván Escalona

FECHA: Caracas, 27 de Febrero de 2013

Dedicatoria

Quiero dedicar este trabajo especial de grado a toda mi familia.

A mi padre, Omar, que me ha dado todo su cariño y amor, y que sin su atención, preocupación, dedicación y apoyo emocional constante no hubiese podido realizar mis estudios en la Universidad Católica.

A mi madre, Ladys, quien en el transcurso de mi carrera y de mi vida, me ha dado todo su amor y la considero no solo mi madre, sino mi mejor amiga y compañera, siempre a mi lado para apoyarme, guiarme, y que en todos aquellos momentos que creí desfallecer y no poder seguir adelante, me alentó para continuar.

También se lo dedico a mi hermano Jesús, quien a pesar de su fuerte temperamento y carácter, siempre ha estado pendiente de mi salud y de mi carrera, y ha compartido conmigo muchos momentos, algunos pequeños o sencillos, y otros que me marcaron y fueron de gran importancia en mi vida.

Por último, le dedico este trabajo a mi hermano menor José Manuel. Si no fuera por su bondad, amor incondicional y el regalo de vida que me dio, yo no hubiese sido capaz de continuar con mis estudios, ni me encontraría disfrutando de esta nueva experiencia.

Para todos ustedes, las palabras de agradecimiento jamás serán suficientes. Los amo con todo mi corazón.

Agradecimientos

Le doy gracias a la vida por darme constantemente nuevas experiencias, muchas buenas, y algunas no tan buenas, que me han permitido madurar y crecer, y me han convertido en la persona que soy actualmente. Le doy gracias, por brindarme toda la salud que necesito, y por permitirme estar rodeada de personas que me aman y me apoyan, pase lo que pase.

Le agradezco a toda mi familia que me ha guiado, y con sus enseñanzas, experiencias y conocimientos, me ha convertido, día a día, en una persona mejor.

A mis amigos, que han estado siempre a mi lado, que me han permitido compartir con ellos mis secretos y mis momentos de alegría, tristeza y diversión.

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA TRANSMISIÓN Y RECEPCIÓN DE PARÁMETROS FISIOLÓGICOS BÁSICOS DETECTADOS CON UN MÓDULO PORTÁTIL.

Vanessa Alexandra Quintana Barillas
vanequintana29@gmail.com
Escuela de Ingeniería de Telecomunicaciones, UCAB
Caracas-Venezuela

Resumen

La telemedicina permite a los profesionales de la salud realizar una monitorización a distancia, para la evaluación, consulta y diagnóstico de pacientes que se encuentran ubicados en sitios remotos haciendo uso de las telecomunicaciones, aplicaciones web y móviles, entre otros. Esta es una solución muy eficaz ante la falta de personal médico en el país y la cantidad de pacientes que necesitan de una opinión médica y no tienen como movilizarse y acceder a los hospitales. Ante esta situación se desarrolla una aplicación para teléfono móvil y Tableta basada en el sistema operativo Android, para monitorizar parámetros biomédicos básicos como temperatura, frecuencia cardíaca y frecuencia respiratoria que son recibidos desde el paciente vía Bluetooth haciendo uso de un dispositivo portátil llamado BioHarness BioModule y que posteriormente son enviados a un servidor Web vía WiFi. Para el desarrollo de este proyecto se trabajó con un sistema MVC (Model View Controller) que separa el área de trabajo en 3 capas: la capa de presentación, que se conoce como la interfaz grafica de la aplicación; la capa de negocios que está relacionada con el servidor que manejará todas las peticiones realizadas por el usuario; y la base de datos y la capa de datos que maneja dicha base de datos. Todas las capas son independientes, por lo que se puede trabajar cada una por separado. El módulo de vista está implementado en Android, haciendo uso del entorno de desarrollo Eclipse. La capa de negocios se trabajó con un servidor Web, basado en Apache y desarrollado en Node.js.

Palabras clave: dispositivo portátil, telemedicina, parámetros biomédicos, *Android*.

Índice General

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

Durante los últimos años la Telemedicina se ha venido desarrollado de forma rápida en Venezuela, permitiéndole a los médicos realizar diagnósticos, consultas y monitorización a distancia con pacientes ubicados en zonas rurales o que no tienen acceso a especialistas o personas entrenadas en el área médica. La telemedicina se ha convertido en un recurso de vital importancia para pacientes y médicos, debido también a que existe una gran fuga de talentos médicos que prefieren ejercer su carrera en otros países, como España, ya que estos les ofrecen seguridad, buenos salarios y muchos otros beneficios, mientras que pueden continuar, a distancia, el control de pacientes radicados en el país.

Por otra parte, hay que destacar que la tecnología de comunicaciones se ha venido desarrollando de forma acelerada, en especial todo lo relacionado a teléfonos móviles inteligentes (Smart Phones) y dispositivos portátiles de tercera o cuarta generación. En la actualidad la mayoría de los venezolanos tienen acceso a teléfonos celulares, PDAs, Tablets y otros dispositivos móviles con conexiones vía Bluetooth y Wi-Fi, lo que ha incrementado la necesidad de programadores e ingenieros capaces de desarrollar aplicaciones en el área de la telemedicina, que permitan monitorizar a los pacientes desde cualquier punto de acceso a la red.

De acuerdo a lo mencionado anteriormente se propuso desarrollar una aplicación para teléfono móvil y Tableta que permita la transmisión y recepción de parámetros fisiológicos básicos (frecuencia cardíaca, frecuencia respiratoria y temperatura de la piel), mediante el uso del dispositivo portátil BioHarness BT BioModule. Dichas señales básicas son captadas por el sensor y enviadas al teléfono móvil o Tableta, los cuales a su vez enviarán los datos vía Wi-Fi a un servidor que almacenará la información. De esta forma el paciente no tendrá que movilizarse a un centro médico, sino que el especialista podrá monitorizar sus datos y realizar la toma de decisiones en cuanto a un tratamiento o cualquier otra evaluación que deba llevarse a cabo.

La estructura del proyecto consta de cinco (5) capítulos principales. El primer capitulo describe el planteamiento del problema, así como los objetivos que se desean cumplir, las limitaciones y alcances, y la justificación del mismo. El segundo capitulo contiene el marco teórico que recoge toda la teoría sobre las herramientas de desarrollo utilizadas, sistema empleado, características de los parámetros que se van a enviar, entre otros. El tercer capitulo contiene la metodología y desarrollo del trabajo de grado. El análisis de los resultados obtenidos, como la aplicación móvil, el servidor y las pruebas de conexión, se ubican en el cuarto capitulo. En el último capitulo se ubican todas las conclusiones y recomendaciones obtenidas del desarrollo del proyecto. Al culminar todos los capítulos se encuentra todas las fuentes bibliográficas consultadas, y seguidamente se ubican los anexos que complementan la información sobre el uso de la aplicación móvil la aplicación y el dispositivo BioHarness.

CAPÍTULO I

PLANTEAMIENTO DEL PROYECTO

En este capítulo se plantea el problema que da origen al proyecto en general. También se establecen los objetivos, se justifica el proyecto y la importancia en el área de las telecomunicaciones, y finalmente se definen los alcances y limitaciones que delimitan de forma clara y precisa hasta dónde se va a desarrollar el proyecto.

I.1. Planteamiento del problema

Ante la realidad incierta que afronta el país en lo económico, político, social, muchos profesionales consideran que no hay futuro en el país, aprovechando la oferta que le brindan países desarrollados. Actualmente, aunque en Venezuela no hay cifras exactas sobre la fuga de talentos médicos, existen indicadores (como el bajo número de inscritos en estudios de postgrado y especialización) que evidencian que se trata de cantidades importantes las referidas al número de profesionales de la medicina que emigran anualmente. Muchos médicos dejan su país con la intención de estudiar y regresar pero con el tiempo encuentran empleo y se radican, prefiriendo ejercer en países como España, Canadá, Inglaterra, Estados Unidos entre otros ya que en estos se ofrecen mejores condiciones socioeconómicas y laborales que las que se brindan en Venezuela, dejando sin relevo a la generación actual. Por otro lado, este grupo de médicos que ya tenía tiempo ejerciendo la medicina en el país, atendía a una parte importante de la población venezolana de los cuales conocía su historial médico y que ahora se ve parcialmente desamparada sin el cuidado respectivo. Detener esta fuga de talentos es tarea difícil en un país en crisis, sin embargo el reto está en generar fortalezas, tanto en las empresas como en el país, y en lograr la optimización de los servicios de atención en salud, ahorrando tiempo, desplazamientos innecesarios y facilitando atención de especialistas en zonas distantes. Esto se suma a un cuadro ya

complicado por las dificultades bien conocidas del sistema público de salud y los costos del sistema privado.

En este sentido la tecnología de la telecomunicaciones, que se ha desarrollado en las últimas décadas con un ritmo de cambio nunca antes conocido, ofrece campos de actividad innumerables con múltiples aplicaciones de comunicación a distancia: sería un paliativo ante la ausencia de un médico, a través del uso de la TELEMEDICINA como Medicina practicada a distancia, que incluye tanto diagnóstico y tratamiento: igualmente, facilitaría la educación médica, haciendo uso de los elementos telemáticos como soportes informáticos de distinta índole (computadoras con conexión a Internet, teléfonos celulares con sistema GPRS, etc.)

Es importante destacar que en Venezuela durante los últimos años ha venido aumentando el uso de la telefonía celular y de los dispositivos móviles. Según la Comisión Nacional de Telecomunicaciones (Conatel) a finales del 2010 un 97% de una población de aproximadamente 28 millones de habitantes poseía un teléfono celular y a inicios del primer trimestre del 2011 aproximadamente un 27% de dichos usuarios se conectaba a la red haciendo uso de estos dispositivo (Pasquier, 2010). La mayoría de los venezolanos posee teléfonos de 3ra y 4ta generación, también conocidos como Smart Phones, los cuales son óptimos para realizar conexión a Internet y descargar aplicaciones móviles.

Aprovechando esta situación este trabajo se orienta a desarrollar una aplicación para teléfono móvil y Tableta que permita la transmisión y recepción de ciertos parámetros fisiológicos básicos (frecuencia cardíaca, frecuencia respiratoria y temperatura de la piel), mediante el uso del dispositivo portátil BioHarness BT BioModule. Dichas señales básicas son captadas por el sensor y enviadas al teléfono móvil o la Tableta los cuales a su vez enviarán los datos vía Wi-Fi a un servidor, diseñado especialmente en el marco de este trabajo, para que permita la visualización de la información. De esta forma el paciente no tendrá que movilizarse a un centro médico, para hacer seguimiento de sus signos vitales básicos, sino que podrá monitorizarlos

por ejemplo desde su casa y enviar la información. De esta forma se podrán dejar libres mas plazas en los ambulatorios y hospitales, para que las personas que estén mas complicadas de salud puedan asistir a la consultas de forma presencial, permitiendo así que muchas personas puedan ser monitorizadas desde sus hogares.

I.2. Objetivos

I.2.1. Objetivo General

Desarrollar una aplicación móvil para la transmisión y recepción de parámetros fisiológicos básicos detectados con un módulo portátil.

I.2.2. Objetivos específicos

- ✓ Realizar un estudio detallado de las capacidades y funcionamiento del dispositivo portátil BioHarness BT BioModule existente en la Escuela de Ingeniería en Telecomunicaciones.
- ✓ Realizar un estudio de las diferentes formas de adquisición de la información por parte del teléfono móvil y la Tableta.
- ✓ Estudiar la forma de transmisión de los datos hacia el servidor, utilizando Wi-Fi.
- ✓ Desarrollar una aplicación móvil que permita la visualización gráfica de los valores recibidos por el teléfono móvil y la Tableta.
- ✓ Implementar un servidor que permita la recepción y visualización de las bioseñales básicas

I.3. Alcances y limitaciones

Este trabajo especial de grado incluye los siguientes aspectos:

- ✓ El desarrollo de una aplicación muy sencilla y básica que incluye la visualización de los datos enviados por el dispositivo portátil BioHarness BT BioModule tanto en un teléfono móvil como en una Tableta con sistema operativo *Android*.

- ✓ La transmisión, recepción y visualización de por lo menos tres de los parámetros que permite manejar el dispositivo portátil de BioHarness BT BioModule.

- ✓ La utilización exclusiva de los equipos que pertenecen a la Escuela de Telecomunicaciones de la Universidad Católica Andrés Bello.

- ✓ El envío y visualización de la información al servidor de la Universidad Católica Andrés Bello.

Este trabajo especial de grado no incluye los siguientes aspectos:

- ✓ El desarrollo extenso y complejo de programación del servidor.

- ✓ El desarrollo de extensiones de la aplicación para otros sistemas operativos distintos a *Android*.

I.4. Justificación

Este trabajo especial de grado tiene como objetivo principal aportar un tipo de solución a problemas causados por la escasez de médicos y especialistas, originada por la fuga de estos talentos, y que de cierta forma ha afectado a los pacientes a los que controlaban de forma constante, o bien en el caso de los pacientes ubicados en

zonas alejadas de los centros urbanos y con dificultades de acceso a centros hospitalarios y la atención médica adecuada.

Lo que se plantea es el desarrollo de una aplicación en *Android* para teléfonos móviles y tabletas que permita, haciendo uso de la telemedicina, que un médico pueda monitorizar los parámetros fisiológicos básicos de sus pacientes, sin necesidad de que éstos tengan que trasladarse al centro clínico, sino solo en caso de que el médico así lo indique. De esta forma, el paciente también podrá tomar éstos parámetros desde cualquier lugar que se encuentre haciendo uso de un módulo portátil especial y de su dispositivo móvil.

La aplicación de este proyecto no pretende abarcar casos que sean de emergencia, ya que a pesar de que se trabajará en tiempo real, el diagnóstico y evaluación del paciente dependerá de la rapidez con la que el especialista médico verifique la información y realice el contacto con el paciente para realizar la evaluación del caso y tomar las decisiones correspondientes.

Este proyecto permite aplicar las telecomunicaciones en el área de la medicina, ayudando de esta forma a desarrollar una aplicación que le permita al medico verificar los parámetros fisiológicos y datos de sus pacientes sin la necesidad de verlos en consulta y desahogando un poco los consultorios de pacientes que no ameritan un chequeo físico de emergencia.

CAPÍTULO II

MARCO TEÓRICO

En este capitulo se van a introducir todos los conceptos básicos de los parámetros fisiológicos (temperatura, frecuencia cardiaca y frecuencia respiratoria) con los que se va a trabajar, al igual que el dispositivo y la tecnología con el que se recibirán y transmitirán dichos parámetros a la aplicación móvil y posteriormente al servidor.

También se explicará claramente el sistema MVC (Modelo Vista Controlador) de tres capas junto con sus características y las herramientas de desarrollo, funciones y librerías que se van a utilizar para la realización de la aplicación móvil y el servicio web.

En la Figura 1. se muestra el esquema del marco teórico que representa de forma mas clara todos los puntos a tratar en este capitulo.

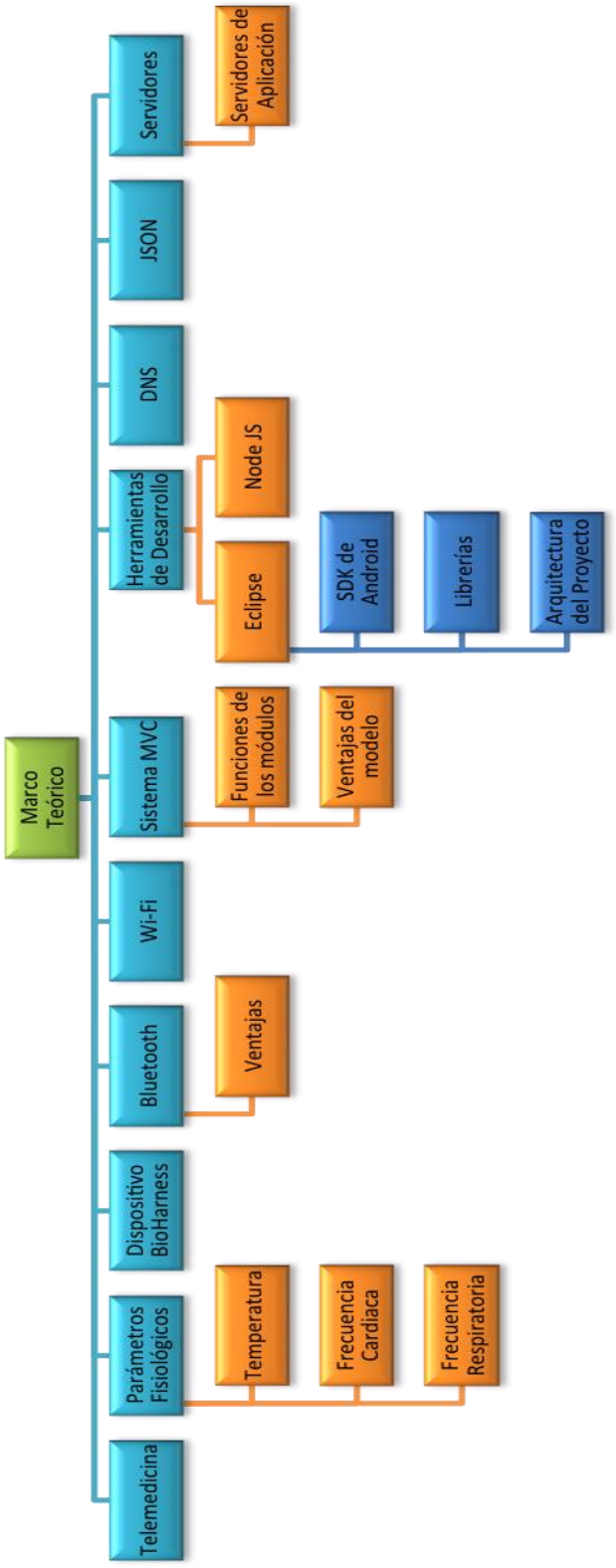


Figura 1. Esquema del Marco Teórico.

Fuente: Elaboración Propia.

II.1. Telemedicina

La telemedicina consiste en aplicar la medicina a distancia haciendo uso de las tecnologías de información y comunicación. Según la ATA (American Telemedicine Association) la telemedicina es usada para el intercambio de información médica entre un lugar y otro, haciendo uso de la comunicación electrónica. Asociado a este término se encuentra el de Telesalud, el cual se encarga también del monitoreo remoto de pacientes pero no siempre involucra servicios clínicos. (American Telemedicine Association, 2012)

La telemedicina tiene varias aplicaciones en el ámbito médico, de las cuales hay que destacar el telediagnóstico, la teleconsulta, la teleconferencia, y el almacenamiento digital de datos o fichas médicas. También ofrece servicios de videoconferencia, transmisión de imágenes fijas, portales para pacientes, monitoreo remoto de signos vitales, y educación tanto para médicos como enfermeras. (American Telemedicine Association, 2012)

II.2. Parámetros Fisiológicos

Para poder trabajar, entender y realizar una aplicación que muestre correctamente los parámetros fisiológicos de un paciente se debe primero conocer de forma clara el comportamiento de los valores temperatura, frecuencia cardíaca y frecuencia respiratoria, para luego compararlos con los valores normales dependiendo de la persona que se esté evaluando.

II.2.1. Temperatura

La temperatura es una magnitud física que expresa el nivel de calor que ostenta un cuerpo determinado, un objeto, un ambiente, entre otros, en tanto, la misma se encuentra estrechamente vinculada a las nociones de frío (menor temperatura) y de calor (mayor temperatura). (Florencia, 2009)

La temperatura de un individuo puede variar dependiendo de muchos factores como el sexo de la persona, momento del día, actividad que está realizando en ese momento, la alimentación e ingesta de líquidos (Vorvick, 2011). La medición de la temperatura puede realizarse en distintas zonas del cuerpo, como las axilas, el recto y la boca, entre otros. Dependiendo de la zona en que se tome el valor puede tener una variación entre ± 0,3 º C.

Según la Asociación Médica Americana, los valores promedios de temperatura de una persona normal oscilan entre 36,5º C - 37,2º C (grados Celsius) ó 97,8º F - 99º F (grados Fahrenheit): cualquier valor que sobrepase el valor del límite superior se puede considerar fiebre.

II.2.2. Frecuencia Cardíaca

La frecuencia cardíaca, también conocida como pulso cardíaco, es el número de veces que se contrae el corazón en un tiempo determinado y se mide en latidos por minuto o contracciones por minuto. Los valores de frecuencias pueden variar de forma significativa, dependiendo de las demandas que hace el cuerpo humano. Una persona que está durmiendo tendrá la frecuencia cardíaca mucho más baja que una que está realizando ejercicio (Nordqvist, 2011).

Los factores que pueden afectar la frecuencia cardíaca son la temperatura del aire, la posición del cuerpo, el tamaño corporal, peso, y medicación que use la persona (American Heart Association, 2012). Los rangos de valores adecuados o no adecuados para cada sexo y edad se muestran en la Tabla 1 que se muestra a continuación.

HOMBRES	Mala	Normal	Buena	Muy Buena
20-29	86 o más	70-84	62-68	60 o menos
30-39	86 o más	72-84	64-70	62 o menos
40-49	90 o más	74-88	66-72	64 o menos
50-59	90 o más	74-88	68-74	66 o menos
60 o más	94 o más	76-90	70-76	68 o menos

MUJERES	Mala	Normal	Buena	Muy Buena
20-29	96 o más	78-94	72-76	70 o menos
30-39	98 o más	80-96	72-78	70 o menos
40-49	100 o más	80-98	74-78	72 o menos
50-59	104 o más	84-102	76-82	74 o menos
60 o más	108 o más	88-106	78-88	78 o menos

Tabla 1. Valores Promedios de frecuencia cardiaca

Fuente: **(Hernández)**.

II.2.3. Frecuencia respiratoria

La frecuencia respiratoria es el número de veces que una persona respira por minuto. Se suele medir cuando la persona está en reposo, y consiste simplemente en contar el número de respiraciones durante un minuto contando las veces que se eleva su pecho. La frecuencia respiratoria es un signo vital que puede variar con la edad, el sexo, la tolerancia al ejercicio y la condición médica del individuo. La frecuencia respiratoria normal de un adulto que esté en reposo oscila entre 12 y 18 respiraciones por minuto (Dugdale, 2011).

II.3. Dispositivo BioHarness BioModule

Este dispositivo es una nueva tecnología de sensores inteligentes que trabaja con tecnología Bluetooth, permitiendo así la captura y transmisión de datos fisiológicos de un usuario a través de las redes de datos móviles y fijos. Es decir, este sensor permite el monitoreo y control remoto del rendimiento y la condición humana en el mundo real, evaluando respuestas impulsivas básicas e importantes como: la temperatura de la piel, frecuencia cardíaca y respiratoria, posición y aceleración. (Zephyr Technology Corporation, 2010)

La recepción de los datos obtenidos por el sensor puede realizarse mediante un radio o telefonía móvil (Bluetooth) para luego ser enviados a las personas interesadas en tomar decisiones críticas, que serán basadas en la fisiología del individuo. El Bluetooth utilizado para esta tecnología transmite en una frecuencia aproximada de 2,4 GHz y el receptor debe encontrarse posicionado en un rango de 10 mts del equipo que recibirá la señal. (Zephyr Technology Corporation, 2010). En la Figura 2 se puede observar una imagen del dispositivo BioHarness. Del lado izquierdo se observa el dispositivo principal que contiene todos los sensores encargados de recoger toda la información del paciente. Este maneja valores de tiempo, frecuencia cardíaca, frecuencia respiratoria, posición, temperatura y aceleración del individuo.

Figura 2. Dispositivo BioHarness BioModule.

Fuente: Elaboración Propia.

II.4. Bluetooth

Luego de estudiar las dos tecnologías con las que trabaja el dispositivo BioHarness (USB y Bluetooth), se consideró que para el desarrollo de una aplicación móvil la tecnología mas adecuada es Bluetooth ya que esta permitirá al usuario movilidad siempre dentro del rango de distancias que cubre este tipo de señal.

Esta tecnología fue diseñada principalmente para el soporte de redes inalámbricas simples de dispositivos de consumo personal como teléfonos celulares, PDAs, entre otros. Las señales inalámbricas transmitidas mediante Bluetooth cubren distancias cortas, generalmente como máximo los 10 mts. (Mitchell)

La tecnología Bluetooth se basa en el estándar de IEEE 802.15.1 y trabaja en el rango de frecuencias de 2,4GHz. El Bluetooth no es un reemplazo de Wi-Fi, porque en comparación las redes Bluetooth son mucho más lentas, son limitadas en rango y soportan muchos menos dispositivos. A pesar de esto la tecnología de Bluetooth incluye seguridad e interoperabilidad con otros estándares de redes (Mitchell). Las características de esta tecnología se pueden ver en la Tabla 2.

PARÁMETROS	VALORES
Frecuencia	2.4 GHz
Tecnología	Spread Spectrum
Potencia de transmisión:	Clase1: 1mW Cobertura de 1m. Clase 2: 2,5mW cobertura de 10m. Clase 3: 100mW cobertura de 100m (Alecrim, 2011)

Cobertura	10 metros
Alimentación	2.7 Voltios
Interferencia	Es mínima, se implementan saltos rápidos en frecuencia de 1600 veces / segundo.

Tabla 2. Características del Bluetooth.

Fuente: (Pérez & Joffre, 2000)**.**

II.4.1. Ventajas de la tecnología Bluetooth

La tecnología Bluetooth presenta una serie de ventajas importantes para el caso particular de este trabajo de grado. Según Gabriela L. Sparacino (2003) de la Universidad Rafael Belloso Chacín las ventajas que tiene esta tecnología son:

- ✓ Permite rápidas y seguras transmisiones tanto de voz como de datos.
- ✓ Bluetooth es una tecnología que nos permite conectar nuestros dispositivos digitales sin utilizar cable.
- ✓ Para la transmisión tanto de voz como de datos, Bluetooth emplea una combinación de conmutación de circuitos y de paquetes. Cada canal de voz soporta un enlace síncrono a 64 Kbps.
- ✓ A diferencia de otros sistemas de comunicaciones inalámbricos como los basados en infrarrojos, Bluetooth no requiere que haya línea de visión directa entre los dispositivos (Sparacino, 2003).

II.5. Wi-Fi

Wi-Fi es una tecnología para comunicaciones inalámbricas basada en ondas y es una de las más utilizadas hoy en día. Sus siglas significan Wireless Fidelity (Núñez, Peña, & Garzon, 2009). Cuando se habla de ésta tecnología se hace referencia a la Wi-Fi Alliance, que es una organización sin ánimos de lucro, conformada por un amplio grupo de fabricantes encargado de promocionar la tecnología inalámbrica en redes de área local, y mantener estándar IEEE 802.11 entre todos los fabricantes (Castro, 2005).

Originalmente se creó como una tecnología para que los dispositivos se conectaran a una red inalámbrica local (WLAN), pero actualmente es utilizada para que los dispositivos como laptops, PDAs, tabletas, celulares, entre otros, tengan acceso a Internet.

La tecnología Wi-Fi trabaja bajo los protocolos que engloba el estándar IEEE 802.11; dependiendo del protocolo con el que se trabaje variará la velocidad de transmisión, entre otras características. En la Tabla 3, se muestran los diferentes estándares de la IEEE 802.11 y la descripción de las características de cada estándar.

Estándar	Descripción
802.11	Estándar WLAN original. Soporta de 1 a 2 Mbps.
802.11a	Estándar WLAN de alta velocidad en la banda de los 5 GHz. Soporta hasta 54 Mbps.
802.11b	Estándar WLAN para la banda de 2.4 GHz. Soporta 11Mbps.
802.11e	Está dirigido a los requerimientos de calidad de servicio para todas las interfaces IEEE WLAN de radio

802.11f	Define la comunicación entre puntos de acceso para facilitar redes WLAN de diferentes proveedores.
802.11g	Establece una técnica de modulación adicional para la banda de los 2.4 GHz. Dirigido a proporcionar velocidades de hasta 54 Mbps.
802.11h	Define la administración del espectro de la banda de los 5 GHz para su uso en Europa y en Asia-Pacífico.
802.11i	Está dirigido a superar la vulnerabilidad actual en la seguridad para protocolos de autenticación y de codificación.

Tabla 3. Familia de protocolos IEEE 802.11.

Fuente: (Núñez, Peña, & Garzon, 2009).

II.6. Sistema MVC

El sistema MVC (Model-View-Controller), fue el sistema elegido para la solución de la propuesta original. El sistema MVC es un patrón de diseño que consiste en separar el área de trabajo en 3 capas.

La capa de presentación también es conocida como la interfaz gráfica con la que se relaciona el usuario y ésta debe tener la característica de ser amigable y de fácil entendimiento para el usuario. Esta capa se comunica únicamente con la de negocios. Luego tenemos la capa de negocios que actúa como capa de comunicación entre la capa de presentación y la capa de datos. Su trabajo consiste en recibir las solicitudes de la capa de presentación y mostrar los resultados obtenidos, y de comunicarse con la capa de datos para almacenar o recuperar datos en caso de que sea necesario. Por último la capa de datos es donde residen todos los datos y se encarga de recibir las

solicitudes de almacenamiento o recuperación de datos que vienen de la capa de negocios. (Calle, 2008).

II.6.1. Funciones de los módulos de diseño

- ✓ Modelo: Es la información o los datos con los que trabaja la aplicación, también representa las reglas de negocio que rigen el acceso y la actualización de los datos. Esta parte debe estar soportada por una base de datos.

- ✓ Vista: Es la forma en la que interactúa el usuario con la aplicación (La interfaz gráfica). Muestra y accede al contenido de un modelo y es responsable de mantener la coherencia en caso de que exista un cambio en el modelo.

- ✓ Controlador: Es el comportamiento o lógica de las acciones que realiza la aplicación. Es decir traduce las interacciones del módulo de vista con las acciones que se deben tomar en el modelo. Estas interacciones pueden ser clics en los botones o interacciones de un menú. (Eckstein, 2007).

En la Figura 3 se puede observar un diagrama de la estructura del sistema MVC y se mencionan las características principales con respecto a la función de cada una.

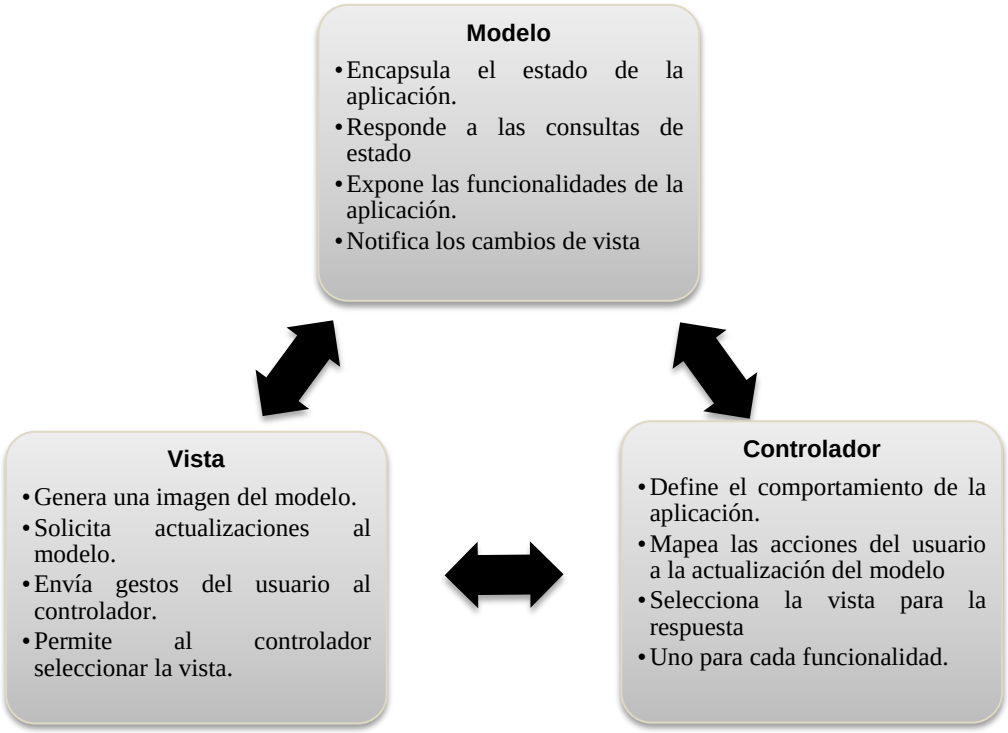


Figura 3. Diagrama de Estructura del MVC.

Fuente: (Eckstein, 2007)

II.6.2. Ventajas de la utilización del Modelo MVC

La separación por capas permite un mejor manejo al tratar sistemas cliente-servidor ya que proporciona características de escalabilidad, rentabilidad, fácil y flexible estructuración del código. Los modelos mencionados anteriormente pueden ser separados como tres bloques diferentes, lo que permite que sean desarrollados y manejados de forma individual y que la conexión entre cada uno de ellos sea claramente localizable.

Las ventajas de usar este modelo son:

- ✓ Cualquiera de esas tres capas puede ser modificada en cualquier momento sin tener ningún impacto sobre el resto de ellas.
- ✓ Simplifica el mantenimiento y la mejora del sistema.
- ✓ Crea distintas representaciones de los mismos datos de manera sencilla. (Business Development Software).

Todas estas ventajas hay que tomarlas en cuenta durante el desarrollo de nuestro sistema y facilitarán el desarrollo del mismo.

II.7. Herramientas de Desarrollo

En este apartado se expondrán todas las herramientas que se van a utilizar tanto para el desarrollo de la aplicación móvil en *Android* como el desarrollo del servidor.

II.7.1. Eclipse

Eclipse fue creado originalmente por la empresa IBM y es una plataforma de desarrollo o entorno de desarrollo integrado (IDE, Integrated Development Environment) de código abierto, basado principalmente en el lenguaje de programación Java, a pesar de que permite otros lenguajes de programación como C, C++ y COBOL (Gallardo, 2007).

Esta aplicación facilita las tareas de compilación, ejecución y edición de programas durante su fase de desarrollo. Es un entorno que puede soportar varios lenguajes de programación, pero el que vamos a utilizar para el desarrollo de nuestra aplicación es el lenguaje Java. Para la compilación y ejecución de nuestra aplicación en la plataforma Eclipse, se necesita el Java Development Kit o JDK. (Martinez, 2007)

II.7.1.1. SDK de *Android*

La aplicación móvil que se va a desarrollar para la recepción de los parámetros biomédicos será implementada bajo el sistema operativo *Android*, por lo cual debemos instalar en Eclipse el kit de desarrollo de software para *Android* (SDK siglas en ingles para Software Development Kit). Este es un conjunto de herramientas de desarrollo que permite al programador desarrollar aplicaciones para un sistema concreto (Adrian, 2010). La aplicación se desarrollará ~~bajo el SDK de *Android* versión 2.2.~~bajo el API Level – 8 y con el target name Android 2.2.

II.7.1.2. Librerías de *Android*

Las librerías son aquellas que contienen una serie de funciones ya preestablecidas o creadas por otros usuarios facilitando de esta forma la creación y edición de las aplicaciones. (android scenebeta.com, 2011).

Para el desarrollo de la aplicación móvil utilizamos tres librerías importantes aparte de las comunes que trae el entorno de desarrollo. La primera es *BioHarnessBT.jar* la cual trae algunas funciones que nos permiten la comunicación con el dispositivo BioHarness y el manejo de los paquetes de temperatura, frecuencia cardíaca y frecuencia respiratoria que deseamos usar. Otra librería muy importante es la *achartenging1.0.0.jar*, la cual permitirá graficar en tiempo real todos los datos que vaya transmitiendo el dispositivo BioHarness a través de Bluetooth. La última librería es la *gson-2.1.jar*. que es una librería de Google que permite convertir un objeto a un String JSON, convirtiendo así los datos que se desean enviar a un formato simple y compatible con el servidor web (Google).

II.7.1.3. Arquitectura de un proyecto en *Android*

En Eclipse tenemos la opción de realizar proyectos para *Android*. Al crear un proyecto, la plataforma de desarrollo genera, por default, una estructura básica de carpetas y paquetes que son indispensables para el buen funcionamiento de la aplicación.

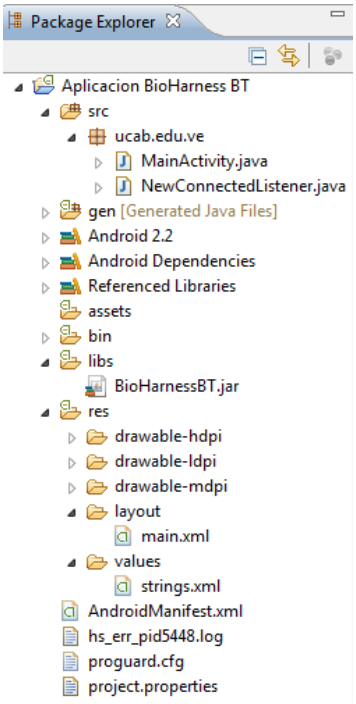


Figura 4. Estructura de un proyecto en *Android*.

Fuente: Elaboración Propia

De la estructura que se observa en la Figura 4 hay que destacar que existen 5 carpetas que son las más importantes y de las que se hará uso en el desarrollo de la aplicación móvil.

- ✓ src: contiene todo el código fuente de la aplicación, al igual que el código de la interfaz gráfica, clases auxiliares creadas, etc. Estos archivos siempre

tendrán la extensión *.java* y siempre existirá una clase principal o *main* desde donde se controlarán todas las demás clases.

- ✓ gen: aquí se ubican todos los archivos que Eclipse genera de forma automática. Estos archivos no pueden ser modificados ni sobrescritos por nadie, la plataforma se encargará de editarlo a medida que se desarrolla el proyecto.

- ✓ libs: en esta carpeta se ubican todas las librerías que serán necesarias en el proyecto, adicionales a las que ya provee la SDK.

- ✓ /res/layout: contiene los archivos *.xml* donde se diseña toda la interfaz gráfica del usuario, botones, imágenes, textos, haciendo uso de las vistas. Todas las vistas serán manejadas desde la clase principal, permitiendo así separar el código de la aplicación de la interfaz.

- ✓ /res/values: también contiene archivos *.xml* pero estos contendrán todas las cadenas de caracteres, arreglos y paletas de colores.

- ✓ *Android* 2.2: aquí se encuentran todas las librerías que tiene la SDK de la versión 2.2.

II.7.2. Node.js

Node.js fue creada en el 2010 y es una tecnología que permite utilizar JavaScript de lado del servidor, lo que no era muy común. Con él se logra realizar funciones que en años anteriores solo podían lograrse con lenguajes nativos de servidor, como PHP, Python, Ruby. Por otra parte, gracias a la implementación del motor V8, se logra trabajar las peticiones realizadas por cada socket (usuario conectado) a grandes velocidades (Realza, 2012). El motor V8 es usado por Google en su navegador de Chrome y se encarga de interpretar y ejecutar de una forma muy rápida el código en

JavaScript (Terkaly, 2012). Este servidor posee una serie de características muy importantes:

- ✓ Permite programación orientada a eventos.
- ✓ Es multiplataforma, es decir se puede implementar en Windows, Linux y Macintosh, aunque trabaja mucho mejor bajo sistemas de UNIX.
- ✓ Permite redes fácilmente escalables y presenta potentes capacidades ante eventos I/O (Entrada/Salida) haciéndola mas eficiente. Esto la convierte en la plataforma ideal para uso intensivo de aplicaciones con datos en tiempo real que se ejecutan a través de dispositivos distribuidos (Node .js)

De esta plataforma interesa una librería en particular, *Socket I/O*. Esta será la plataforma que se utilizará para el desarrollo del servidor web que recibirá los datos que serán enviados desde la aplicación móvil.

II.8. DNS

DNS es el acrónimo para *Domain Name System* y es un sistema que convierte los nombres de Host y de dominios en direcciones IP ya sea para Internet o redes locales que trabajen bajo el protocolo TCP/IP. Este tipo de sistemas permite que los usuarios no tengan que aprenderse una series de números cada vez que quieran acceder a una pagina web, sino que solamente deben aprender el nombre del dominio, que generalmente se organizan de forma jerárquica y terminan en ".com", ".net", ".ve", etc. Es decir, que aprenderse una dirección como www.google.com es mucho más sencilla que aprenderse la siguiente IP "173.194.69.147".

En la Figura 5 se muestra un uso sencillo de un DNS. El cliente desde su computadora o dispositivo móvil desea acceder a una página Web e ingresa el nombre del dominio *host-a.example.microsoft.com*. Inmediatamente se envía la

consulta del cliente a un servidor DNS. Ahí compara el nombre del dominio con su base de datos para ubicar la dirección IP, y le envía al cliente (Microsoft, 2005).

Figura 5. Funcionamiento básico de un sistema DNS.

Fuente: (Microsoft, 2005).

II.9. JSON

JSON es el acrónico de *JavaScript Object Notation* (Notación de objetos JavaScript). Según la pagina principal de JSON, éste se podría considerar como "un formato ligero de intercambio de datos" (JSON) que permite modelar y presentar estructuras de datos como arreglos y objetos como un "texto serializado" (Job, Malaikai, & Schenker, 2010) de una forma ordenada y de fácil acceso. Este formato se rige bajo el estándar ECMA-262 tercera edición.

JSON se ha ido incrementando en uso a nivel mundial, ya que se puede usar como una alternativa para el formato XML. Gracias a esto es reconocido por una amplia cantidad de lenguajes de programación como Java, PHP, C++, entre otros (Alexander, 2008).Este formato se rige por dos estructuras principales:

- ✓ Colección de Pares de nombre/valor, también conocidos como objetos.

- ✓ Listas ordenadas de valores, también llamados arreglos.

II.10. Servidores

Es un dispositivo o programa que realiza algunas tareas en beneficio de otras aplicaciones llamadas clientes, es decir, es un equipo que tiene acceso a Internet y puede acceder tanto a la información del cliente como suministrar esta información en caso de que sea solicitada. Algunos servicios habituales son de archivos, que permiten a los usuarios almacenar y acceder a los archivos de una computadora, y los servicios de aplicaciones, que realizan tareas en beneficio directo del usuario final. Es posible que una computadora cumpla simultáneamente las funciones de cliente y de servidor.

II.10.1. Servidores de Aplicaciones

Trabajan bajo el protocolo HTTP y manejan una serie de recursos, que no son solamente archivos estáticos sino también archivos dinámicos que contienen el código que se ejecutará en nombre de los clientes que realizan la petición. Cuando estos servidores reciben una petición del protocolo, éste analiza la petición para saber que recurso se está solicitando**.** Generalmente esta petición concierne al código ejecutable del servidor. Una vez ejecutado el código, la información obtenida es la que se envía al cliente (Groussard, 2010).

Es importante saber para el desarrollo de este proyecto lo que es un cliente http. Un cliente http es una aplicación basada en el protocolo http (*Hypertext Transfer Protocol*), que accede o solicita recursos o servicios a un servidor, haciendo uso de una serie de mensajes y métodos para realizar la petición.

El protocolo http es usado por World Wide Web desde 1990, y se basa en estándares URL para indicar la dirección del recurso al que se está referenciando. La sintaxis de

una URL se conforma por el "protocolo" con el que se está trabajando, que en este caso es el http, la "dirección IP" a la que se quiere solicitar el recurso, el "puerto" por el que se quiere acceder, y el "path" (Sara, 2012). Una dirección URL lucirá de la siguiente forma. ***"http" ://"dirección ip" "Puerto"/ "Path".***

II.10.1.1. Mensajes de http

http está conformado por dos mensaje principales: el de petición o *Request* y el de respuesta o *Response.* .

- ✓ Request: es un mensaje que se envía al servidor para realizar algún tipo de petición de recursos. También permite tener acceso a toda la información que se envíen entre el cliente y el servidor y debe tener un formato determinado, donde se menciona el método que se está realizando, URI que es el identificador de recursos uniforme, y por ultimo la versión. (Universidad de Valencia, 2012)

- ✓ Response: este mensaje se envía una vez que el servidor ha recibido y procesado la solicitud del cliente, y envía un mensaje de respuesta al mismo. Su formato consiste en una línea de estado, que es la primera línea del mensaje de respuesta donde se tiene la versión del protocolo, luego viene un código de estado junto con una frase que explica el número del código.

II.10.1.2. Métodos de http.

EL protocolo http implementa una serie de métodos que se encargan de realizar diferentes tipos de peticiones entre el cliente y el servidor. Los métodos son OPTIONS, GET, HEAD, POST, PUT, DELETE y TRACE. Solamente se mencionarán los mas importantes y los usados para el desarrollo del cliente http.

- ✓ GET: Método que contiene un mensaje de solicitud de datos del cliente, haciendo una solicitud o devolución del recurso que se encuentra identificado en la URL (Macias, 2007).

- ✓ POST: Este mensaje tiene varias funciones. Primero le indica al servidor que se prepare a recibir información del cliente, este acepta el contenido del servidor y ubica el recurso pedido a través del identificador URI. Aquí se incluye la información que se quiere enviar al servidor. (Macias, 2007).

- ✓ PUT: Este mensaje envía el recurso identificado por la URL desde el cliente al servidor y carga el contenido enviado. (Macias, 2007)

CAPÍTULO III

METODOLOGÍA Y DESARROLLO

El siguiente capitulo tiene como finalidad presentar y describir la metodología utilizada para el desarrollo del sistema de transmisión y recepción de parámetros fisiológicos haciendo uso de una aplicación móvil. El capítulo esta constituido por seis (6) etapas, en cada una de las cuales se especifican las actividades realizadas que permitieron cumplir con los objetivos planteados inicialmente.

El esquema de la Figura 6 que se muestra a continuación, ofrece una visión de la estructura de la metodología empleada.

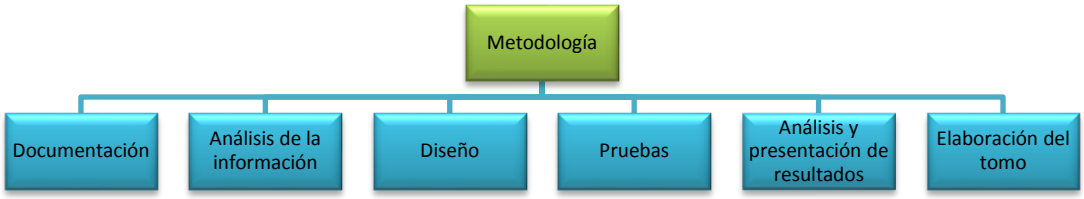


Figura 6. Estructura de la Metodología.

Fuente: Elaboración Propia.

III.1. Documentación.

En esta etapa se recopiló toda la documentación e información y se realizó la lectura de la bibliografía referente al dispositivo BioHarness BioModule, los parámetros fisiológicos que recibe y transmite el dispositivo, las mejores herramientas para el desarrollo de la aplicación móvil y el manejo del servidor. Toda la información

referente a esta fase fue recopilada principalmente de fuentes electrónicas, artículos electrónicos y libros digitalizados online.

La documentación correspondiente al dispositivo BioHarness, que es el encargado de tomar las bioseñales del paciente, se recopiló directamente del cd que viene con la compra del equipo y que contiene todos los instructivos, manuales y programas que ayudan al manejo, funcionamiento y configuración del dispositivo portátil. Otra fuente utilizada para esta parte fue la página Web de Zephyr Corporation que es la compañía que produce y distribuye el dispositivo.

III.2. Análisis de la información.

Paralelamente al desarrollo de la Fase I, se procedió al análisis de la documentación y bibliografía que se estaba investigando, permitiendo seleccionar un sistema adecuado para la transmisión y recepción de los datos, seleccionar la tecnología que es más adecuada para la transmisión de los datos del biomódulo al teléfono móvil o la Tableta, al igual que la tecnología para la transmisión de los datos del teléfono o tableta al Servidor. Adicionalmente se realizó un diagrama del sistema de trasmisión y recepción que se iba implementar y se dividió en etapas especificando para cada una las tecnologías seleccionadas.

Para la selección de las tecnologías a implementar, se dividió el sistema en dos etapas principales.

Etapa 1: está conformada por el dispositivo BioHarness y la tableta o teléfono móvil. La conexión entre estos dos equipos se realiza a través de Bluetooth, ya que ambos equipos los tendrá el usuario, y la distancia entre ellos no será muy larga. El uso de la tecnología Bluetooth permitirá la movilidad del usuario siempre y cuando cumpla con las limitaciones de ésta.

La tableta y el teléfono móvil trabajan bajo el sistema operativo *Android* y se les instalo la aplicación desarrollada para detectar todos los parámetros fisiológicos que envía el dispositivo BioHarness.

Etapa 2: conformada por el teléfono móvil o la tableta con la aplicación en *Android* y el servidor web. La tecnología que se seleccionó para la transmisión de los datos desde los dispositivos móviles hasta el servidor web, es Wi-Fi para ambos casos. El servidor que recibe los datos del usuario será implementado con Node.js, que permite trabajar el servidor a través de eventos.

En la Figura 7 a continuación se muestra el diagrama del sistema que se va a implementar en este proyecto junto con todas las tecnologías seleccionadas para cada etapa.

Figura 7. Diagrama del Sistema.

Fuente: Elaboración propia

III.3. Diseño

En esta fase se procedió con el desarrollo y diseño del sistema a ser implementado. Ésta se dividió en tres etapas principales, la primera orientada a la configuración del dispositivo BioHarness, en donde se delimitaron los parámetros que se desean transmitir, las velocidades de transmisión de los mismos y las claves de acceso del Bluetooth del dispositivo. Luego se procedió con el diseño de la aplicación móvil que es la encargada de establecer la conexión del dispositivo BioHarness BioModule con el dispositivo móvil (Teléfono o Tableta) y de realizar el envío de los datos al servidor web correspondiente. Y finalmente se desarrolló e implementó el diseño del servidor web que, para este proyecto, solamente recibirá los datos enviados por la aplicación móvil y estará elaborado en la plataforma Node.js.

III.3.1. Configuración del BioHarness BioModule

Al estudiar el dispositivo BioHarness BioModule se observó que los parámetros que éste toma del paciente son enviados en un período de tiempo constante. El tiempo de transmisión de los datos puede ser establecido o definido a través del software de configuración del dispositivo. Hay que destacar que los programas de instalación de las aplicaciones que trae el CD del dispositivo están desarrolladas para Windows, por lo que para esta etapa se utilizó una PC con Windows 7.

Para la configuración de los parámetros del dispositivo portátil se instaló en la computadora el *software* del BioHarness Configuration Tool que viene en el CD del dispositivo y se siguieron los pasos correspondientes para la instalación. Luego se conectó vía USB el dispositivo a la computadora y se abrió la herramienta de configuración. Automáticamente en el botón de selección de dispositivo se selecciona el equipo con el que estamos trabajando y en la pantalla de solo lectura se muestran todos los datos correspondientes al dispositivo, como:

- ✓ Numero Serial: ZBH990848

- ✓ MAC Address: 00:07:80:44:29:41
- ✓ Nombre de Bluetooth: BH ZBH990848

Para esta aplicación se configuró el tiempo en 1000 ms ya que de esta forma los datos son tomados de forma constante y posteriormente permite graficar en la aplicación móvil los datos que se están enviando.

Haciendo el uso del mismo software se configuró el dispositivo para que luego de 10000 ms que no se reciban datos del usuario, éste deje de transmitir información. En la Figura 8 se muestra la configuración realizada con el dispositivo donde se observan el período de transmisión, el tiempo de espera previo antes de que deje de transmitir el dispositivo y la data de sólo lectura que fue mencionada con anterioridad.

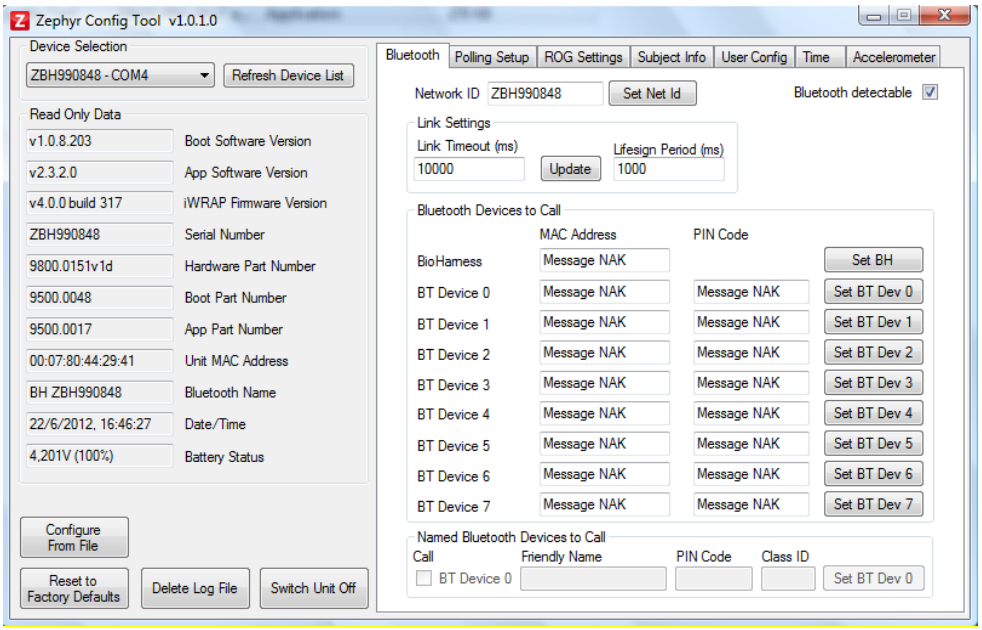


Figura 8. Zephyr Configuration Tool.

Fuente: Elaboración Propia.

Es muy importante conocer los datos de la MAC Address y del Bluetooth ya que ésta es la forma en la que la aplicación, instalada en el dispositivo móvil, se va a conectar vía Bluetooth con el dispositivo portátil BioHarness. El teléfono o la tableta a utilizar debe tener activa la conexión Bluetooth, para que cuando el BioHarness se encienda la aplicación pueda establecer la conexión por esta vía.

Para realizar la conexión con el Bluetooth del dispositivo es necesario corroborar el buen funcionamiento de éste, por lo cual se debe hacer la prueba de conexión y recepción de datos mediante el uso del "BioHarness Bluetooth Test Application V2". En la Figura 9 se puede observar la recepción de todos los parámetros que envía el dispositivo BioHarness, por lo que se verifica que el Bluetooth está funcionando de forma correcta.

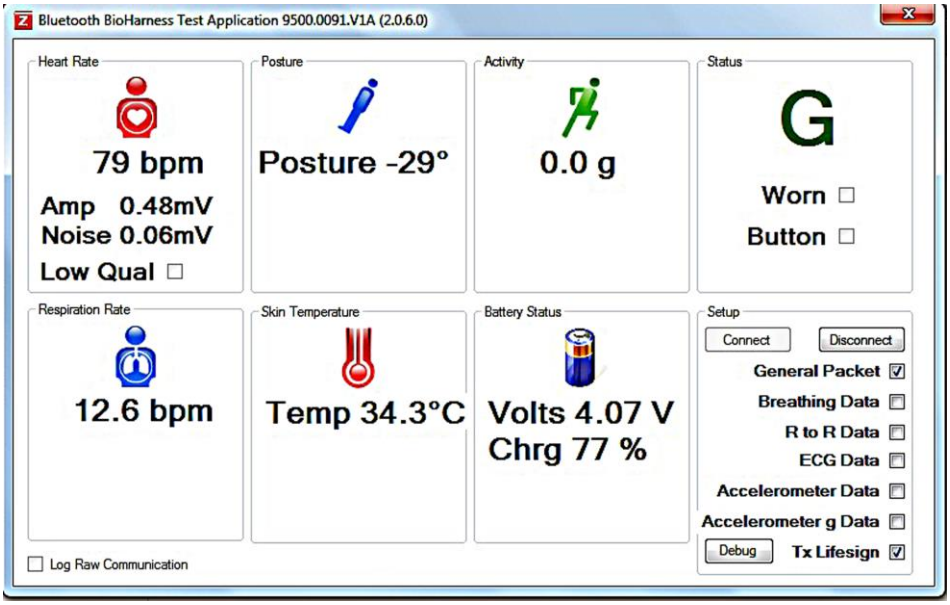


Figura 9. Aplicación de prueba de Bluetooth.

Fuente: Elaboración Propia.

III.3.2. Desarrollo de la aplicación

Para el desarrollo de la aplicación se instaló la plataforma de desarrollo Eclipse Juno versión 4.2.1 la cual fue descargada del link http://www.eclipse.org/downloads/ . Una vez finalizada la descarga se corrió el ejecutable y se siguieron las instrucciones para la instalación. Al correr el programa por primera vez se le debe dar una ruta por defecto donde se guardarán todos los proyectos que se vayan creando y luego se desplegará la consola principal como se muestra en la Figura 10.

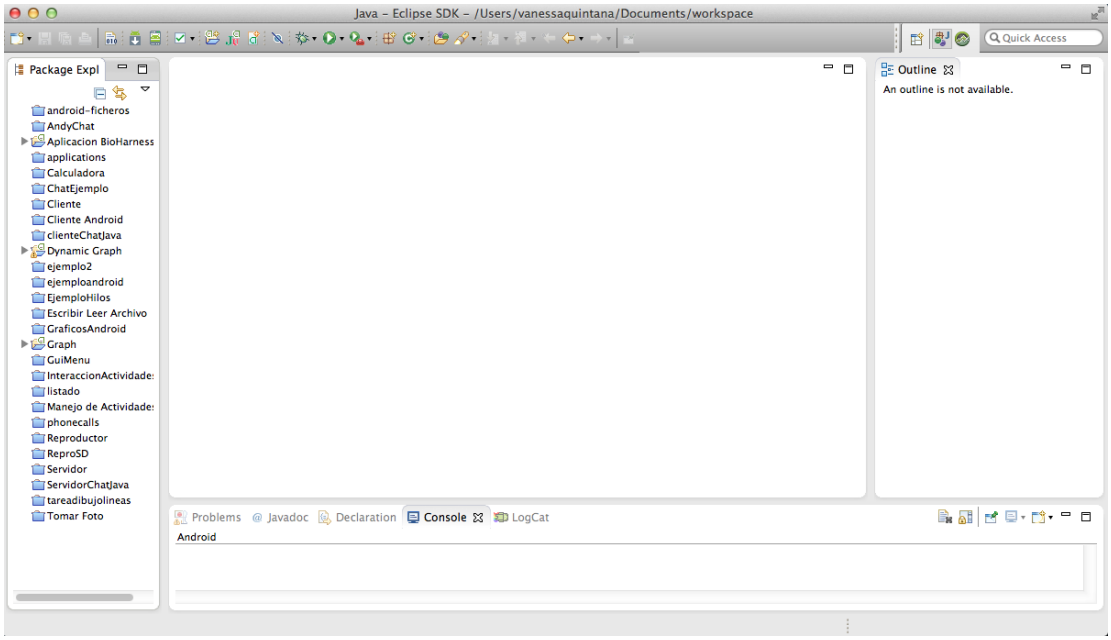


Figura 10. Consola principal de Eclipse.

Fuente: Elaboración Propia.

Para empezar con el desarrollo de la aplicación hay que seleccionar la pestaña de *"New File"*, luego el botón de otros y se desplegará una ventana como la de la Figura 11. Hay que destacar que la plataforma de Eclipse le permite a los desarrolladores trabajar con diferentes tipos de archivos; para este proyecto se debe seleccionar la opción de *"Android Application Project"*

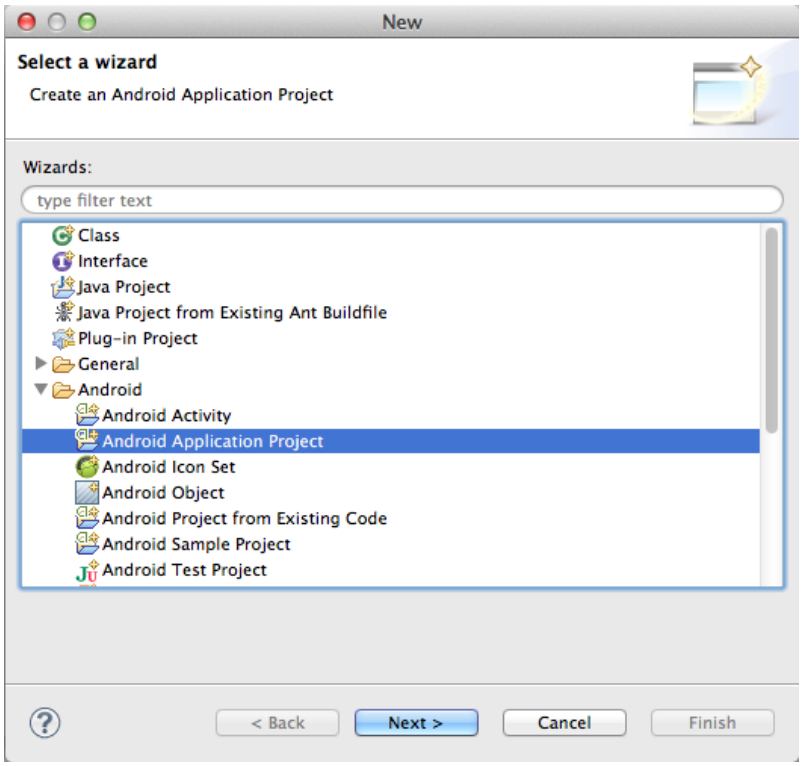


Figura 11. Nuevo proyecto de aplicación para *Android*

Fuente: Elaboración Propia.

Ésta genera una ventana en la que se deben introducir el nombre de la aplicación el nombre del proyecto, el nombre del paquete y la versión de *Android* con la que se desea trabajar. Para este proyecto la aplicación se nombró "*BioHarness Application"* y la versión de SDK utilizada para el desarrollo es *Android* 2.2 (API Level 8), tal y como son mostrados en la Figura 12.

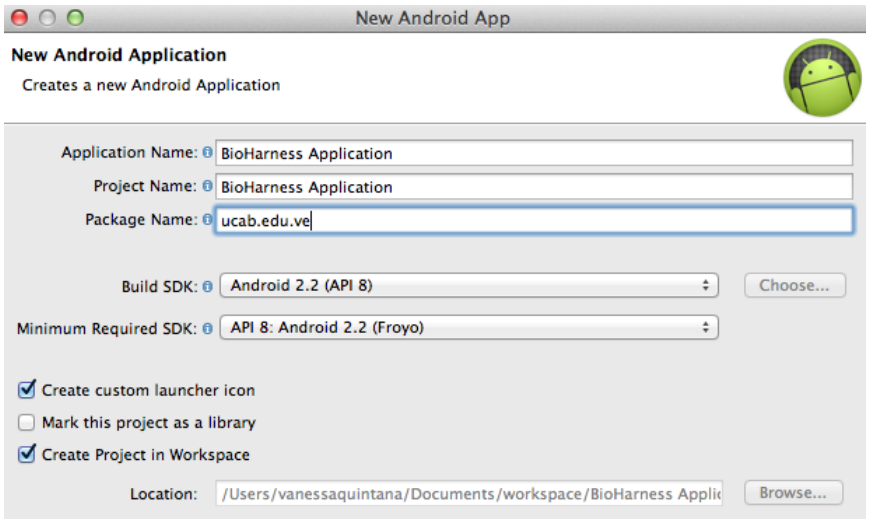


Figura 12. Nueva aplicación de *Android*.

Fuente: Elaboración Propia.

Al finalizar este proceso se desplegará la ventana principal en la que aparece el proyecto que se acaba de crear junto con todas las carpetas que la conforman. Para esta aplicación se va a trabajar principalmente con la carpeta de src que es donde se encuentra el código principal de la aplicación.

III.3.2.1. Clases implementadas

El desarrollo de esta aplicación se trabajó haciendo uso de las clases que se mencionan y explican a continuación.

- BioHarnessMainActivity.java: es la clase principal en donde se manejan y controlan todas las demás clases.

- NewConnectedListener.java: esta es la clase que se encarga de abrir la conexión con el dispositivo y de escuchar y recibir los paquetes de cada uno de los parámetros.

- Point.java: esta genera un punto a partir de dos valores X y Y de tipo doble.

- LineHeartRateGraph.java: aquí se generan los gráficos de frecuencia cardíaca, y se controlan todos los parámetros y características correspondientes a ellos.

- LineRespirationRateGraph.java: esta clase genera los gráficos de frecuencia respiratoria, controla los parámetros y características correspondientes a ellos

- LineTemperatureGraph.java: se genera el gráfico de temperatura de la piel y controla las características específicas del gráfico (color, puntos, ejes, etc.).

- Write_text.java: es la actividad que se llama cuando se selecciona el menú de ayuda y contiene el texto de ayuda al usuario.

También se desarrolló la clase HTTPClientHandler.java o cliente http. En ella se programaron todas las funciones necesarias para realizar la conexión y comunicación con el servidor. Esta clase es la que recibe la URL (*Uniform Resource Locator*) del servidor con el que se desea conectar. La URL no es más que un localizador de recursos uniforme, es decir que permite acceder a recursos compartidos que se encuentran en la red. Ésta debe seguir una estructura básica: protocolo://ip_del_servidor:puerto/ruta

El protocolo seleccionado es el http, la IP del servidor como va a depender de la red en la que se encuentre conectado se asignó mediante DNS y el puerto es el numero 3000.

Por último se utilizaron las clases StatsJson.java y StatusJason.java. La primera es la que se encarga de agrupar en un solo objeto los tres parámetros con todos los valores enviados en forma de arreglo. Posteriormente, en la aplicación principal son convertidos en un String de JSON haciendo uso de la librería Gson, y luego es enviado al servidor. La segunda clase tiene una función parecida a la anterior pero que se encarga de tomar los datos que envía el servidor a través del request.

III.3.2.2. Configuración del DNS.

EL DNS que se utilizó se creó de forma gratuita accediendo por Internet a la dirección http://freedns.afraid.org/. Primero se tuvo que realizar el registro de usuario llenar los datos que se solicitan en la pantalla y seguir las instrucciones para el envío del e-mail de confirmación,

Para crear el subdominio, se ingresó en la pestaña de "*Subdomain*" que envía a otra pantalla en la que hay que seleccionar "*Add*". Luego se mostrará una pantalla como la de la Figura 13 en donde hay que llenar los datos que se solicitan.

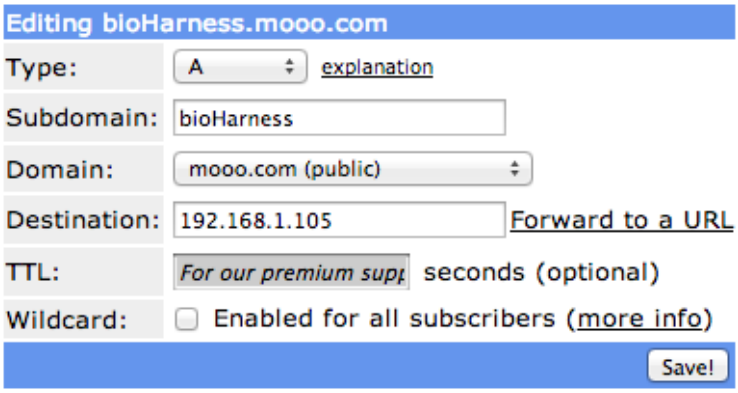


Figura 13. Crear un subdominio.

Fuente: Elaboración Propia

Type: se seleccionó el tipo A, que consiste en apuntar a un subdominio.dominio.com que tiene una IP determinada.

Subdomain: en este campo se colocó el nombre que seleccionamos para el subdominio. Hay que verificar que el nombre no esté utilizado, ya que la página trabaja con dominios y subdominios de tipo público.

Domain: se seleccionó cualquiera de los que están en la lista, dado que son dominios públicos ya existentes.

Destination: en este campo se debe colocar la IP del servidor al que se quiere apuntar. Para el caso de nuestra aplicación esta IP va a variar dependiendo de la red en la que se encuentre conectado el servidor.

Luego se debe salvar la información, y automáticamente se crea el subdominio. El Subdominio queda como en la Figura 14. Donde "*BioHarness.mooo.com*" es la dirección del subdominio, "A" es el tipo de subdominio seleccionado y "192.168.1.105" es la IP del servidor, que varía dependiendo de la red. Hay que destacar que los cambios que se realicen en el subdominio se actualizarán de forma automática en la Web.

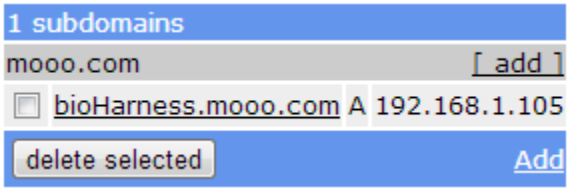


Figura 14. Subdominio.

Fuente: Elaboración propia.

III.3.3. Configuración del servidor

En el desarrollo del servidor se nos presentó un inconveniente. Originalmente se planteó instalar el servidor directamente en el servidor de la Escuela de Telecomunicaciones, pero por inconvenientes presentados con los códigos de acceso para la conexión remota de la VPN, otorgados por la Dirección de Tecnologías e Información, se tuvo que realizar un servidor local. Para este servidor se corresponde una IP privada que se asigna de forma automática dependiendo de la red en la que se encuentre conectado el servidor. La importancia de la misma la veremos en el desarrollo de la configuración del servidor.

Para desarrollar el servidor web se trabajó con la plataforma Node.js, que como se mencionó anteriormente permite la implementación de operaciones de entrada y salida y trabaja con eventos. El servidor se instaló en una computadora que trabaja con el sistema operativo MAC OS X el cual está basado en UNIX. Aquí hay que hacer la instalación de algunos programas previos que facilitan la instalación de Node.js. Primero se ingresa a la página http://mxcl.github.com/homebrew/ y se descarga *Homebrew*, que es el manejador de paquetes para MAC. Se abre el terminal de la computadora y se coloca el comando **ruby -e "$(curl -fsSkL raw.github.com/mxcl/homebrew/go)"** y se espera a que se realice la instalación. Luego se procede a instalar Node.js haciendo uso del comando **brew install nodejs**, y para finalizar se instaló *MongoDB*, con el comando **brew install mongodb** que permite la lectura de bases de datos. En la Figura 15 se muestran los comandos mencionados.

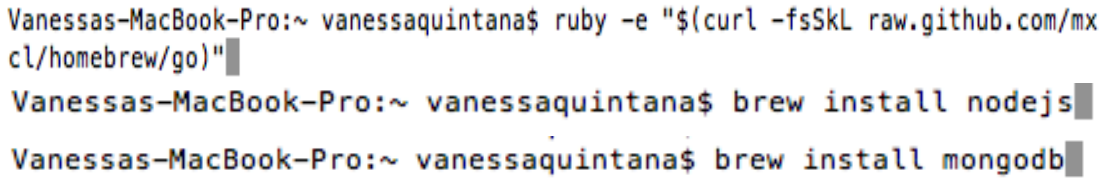


Figura 15. Comandos para la instalación del servidor.

Fuente: Elaboración propia.

Luego de finalizadas las instalaciones de todos los programas necesarios, se procedió a la programación del servidor. La programación básica del mismo se puede ver en la Figura 16. Hay que tomar en cuenta que este servidor permite la conexión con la aplicación móvil y que solamente permite visualizar en forma de arreglo, los valores de los parámetros recibidos y no realiza ninguna otra actividad.

```
app.post('/all',onlyApplicationJson,function(req,res) {
    console.log(req.body);

    estadisticas = req.body;
    console.log(estadisticas);
    if(estadisticas == undefined || estadisticas.HeartRate == undefined || estadisticas.Temperature
== undefined || estadisticas.RespRate == undefined )
        res.send({
            msg:"Datos invalidos",
            code: "invalid_data_error"
        });


    retorno = {
        msg: "prueba",
        code: "OK"
    };
    var instance = new Stats();


    instance.heartRate = estadisticas.HeartRate;
    instance.temp = estadisticas.Temperature;

    instance.resp = estadisticas.RespRate;
    instance.date = Date.now();

    instance.save(function(err) {
        if(err instanceof Error) {
            console.log(error("Ocurrio un error."));
            res.send({
                msg:"Cannot save in database",
                code: "database_error"
            });
        } else {
            res.send(retorno);
        }
    });


});

http.createServer(app).listen(app.get('port'), function(){
    console.log(notice("Express server listening on port " + app.get('port')));
});
```

Figura 16. Programación del servidor.

Fuente: Elaboración propia.

Este servidor se configuró para que trabajara con el puerto 3000, pero se puede sustituir por cualquier puerto que no este ocupado en el momento de su uso, y su IP va a variar dependiendo de la red en la que se encuentre. Como se está trabajando con un servidor local las IPs serán de tipo privado y se deben cambiar constantemente en la aplicación para que exista conexión entre ellas. A fin de evitar esta instalación constante de la aplicación para realizar el cambio de la IP se trabajó con un DNS, en el cual se puede cambiar la IP cuando sea necesario.

Al finalizar la programación del servidor se guardó el archivo como app.js, donde la extensión .js es el formato para Node.js.

III.4. Pruebas

Esta etapa se realizó una vez finalizado el desarrollo de todo el sistema de transmisión y recepción de los datos. Aquí se realizaron todas las pruebas de comunicación y transmisión entre todos los equipos que intervienen en el sistema, como son el dispositivo BioHarness, el teléfono móvil, tableta y servidor de aplicaciones. Se verificó que los datos enviados y recibidos fueran los mismos y en los casos que se presentaban inconvenientes se realizaron las mejoras correspondientes.

III.5. Análisis y presentación de los resultados

En esta fase se presentan todos los resultados obtenidos luego de realizar todas las pruebas y mejoras correspondientes al sistema implementado. La conexión entre los dispositivos tecnológicos que intervienen en el sistema se expondrá mostrando las imágenes tomadas durante la aplicación de pruebas y simulaciones.

Los resultados obtenidos de las pruebas con las personas que hicieron uso del dispositivo BioHarness y la aplicación móvil se visualizarán en tablas y gráficos, y se analizarán y compararán los resultados obtenidos en las pruebas con los valores adecuados que debería tener un paciente para una correcta interpretación por parte del médico tratante.

III.6. Redacción y elaboración del tomo.

En esta fase se expone de forma escrita y en un lenguaje técnico adecuado, cada uno de los capítulos que conforman el proceso de investigación y experimentación del Trabajo Especial de Grado. Todo el contenido fue seleccionado, organizado y presentado de forma clara y precisa para la fácil comprensión del lector. Todas las referencias bibliográficas del documento se trabajaron bajo las normas APA.

CAPÍTULO IV

PRESENTACIÓN DE RESULTADOS

En este capítulo se exponen todos los resultados obtenidos durante el desarrollo de las fases anteriormente mencionadas y que dan respuesta a los objetivos planteados. Se verificó que las herramientas de desarrollo, modelos y tecnologías de transmisión seleccionadas durante la fase de documentación y análisis de la información fueron acertadas. También se muestra tanto la aplicación móvil como el servidor en funcionamiento y los resultados de las pruebas realizadas en diferentes pacientes.

En la Figura 17 se muestra un esquema de los resultados obtenidos para cada una de las fases de documentación, diseño, y elaboración de pruebas.

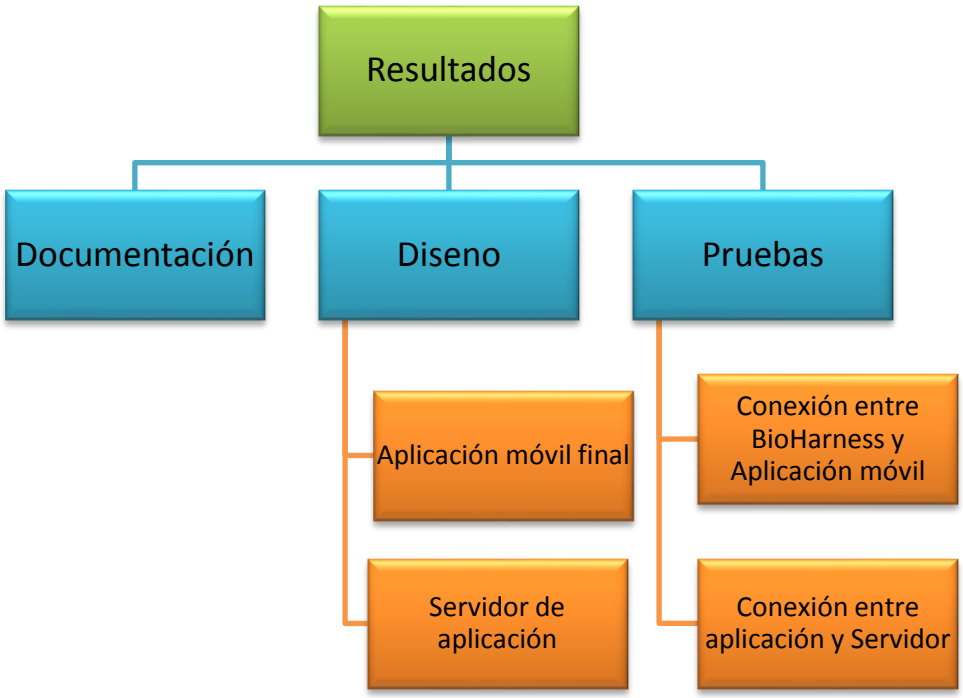


Figura 17. Esquema de la presentación de resultados.

Fuente: Elaboración Propia.

IV.1. Documentación.

En primer lugar se desarrollaron los conocimientos sobre el funcionamiento, las características principales y la configuración del dispositivo BioHarness, lo cual nos permitió entender de forma más precisa como se realiza la toma y el envío de los parámetros fisiológicos del paciente. También se investigó y estudió la librería que tiene el dispositivo para *Android*, conocida como *BioHarnessBT.jar* que contiene todas las funciones necesarias para el manejo de los datos en la aplicación móvil.

Adicionalmente, se ampliaron los conocimientos sobre la herramienta de desarrollo Eclipse que nos permitió desarrollar la aplicación BioHarness App y trabajar con todos los elementos que ésta nos brinda.

Luego se evaluaron las formas de transmisión de los datos del dispositivo BioHarness. Estos se podían realizar mediante el cable USB o vía Bluetooth, y ya que se deseaba tener una aplicación móvil donde el usuario pudiera realizarse las pruebas en todo momento, se seleccionó la conexión vía Bluetooth para la transmisión de información entre ellos. Esta le permite al usuario la movilidad necesaria para la toma de las bioseñales siempre y cuando este mantenga la distancia mínima de alcance del Bluetooth

Por otra parte se seleccionaron las tecnologías apropiadas para el envío de los datos desde la aplicación hacia el servidor. Primero se realizó la transmisión de la información vía Wi-Fi cumpliendo con el objetivo planteado originalmente.

Finalmente se estudiaron los elementos que conforman el sistema de transmisión de datos, y se seleccionaron las tecnologías de transmisión y equipos adecuados para el buen funcionamiento del mismo.

IV.2. Diseño.

De la fase de diseño se obtuvieron dos elementos resultantes: la aplicación que es la encargada de establecer la conexión con el dispositivo BioHarness y de enviar los datos hacia el servidor, y el servidor web que se encarga de recibir y mostrar en pantalla los datos enviados por la BioHarness App.

Ambos elementos se lograron implementar de forma correcta haciendo uso de las herramientas de desarrollo seleccionadas para el proyecto. A continuación se observa la interfaz gráfica de la aplicación BioHarness App desarrollada junto con las actividades que esta realiza y luego se muestra el diseño del servidor web.

IV.2.1. Aplicación móvil final

La aplicación BioHarness App fue desarrollada tanto para teléfonos celulares como Tabletas con el sistema operativo *Android*. Esta aplicación permite la visualización grafica de los parámetros de frecuencia cardíaca, frecuencia respiratoria y temperatura en tiempo real.

La aplicación funciona de la siguiente manera:

Para que la aplicación móvil pueda establecer la comunicación con el dispositivo BioHarness, es necesario que el teléfono o tableta tenga encendido el Bluetooth. Automáticamente el equipo móvil ubica el dispositivo con el que se está trabajando y solicita una clave de acceso (1234) la cual queda guardada para futuras conexiones con el dispositivo. En la siguiente figura se puede observar gráficamente la ubicación del dispositivo por Bluetooth y la solicitud de la clave.

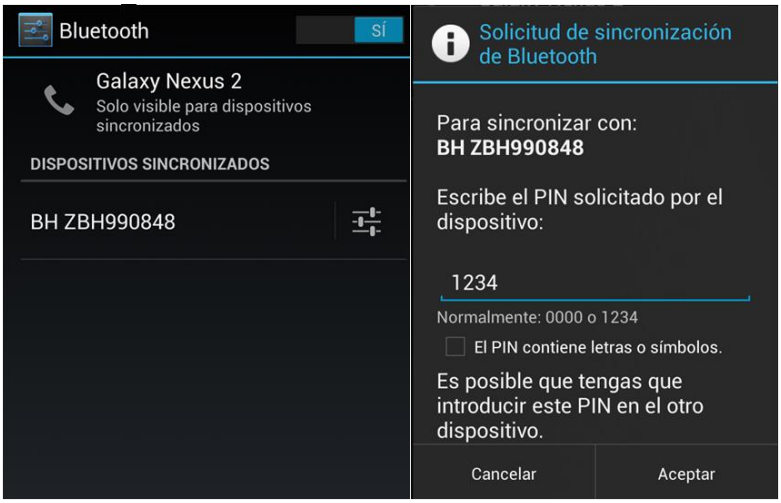


Figura 18. Conexión Bluetooth del teléfono o tableta.

Fuente: Elaboración Propia.

En la Figura 19 podemos observar la ventana principal de la aplicación, en la cual se encuentran 3 botones principales: "Conectar", "Desconectar" y "Enviar", un TextView que indica el status en el que está la aplicación con respecto al dispositivo, un ScrollView en el que se ubican las tres gráficas correspondientes a cada uno de los parámetros recibidos y por último un menú de ayuda para el usuario. Al iniciar la aplicación el estatus aparece como "No Conectado al dispositivo BioHarness".

El botón "Conectar" es el encargado de establecer la conexión con el dispositivo BioHarness. Si al pulsar el botón, el Bluetooth del equipo móvil se encuentra apagado, el status cambiara "Imposible Conectar", en caso contrario mostrará "Conectado al BioHarness: BH ZBH990848 ". El botón "Desconectar" se encarga de cortar la conexión con el dispositivo, es decir que la aplicación no seguirá recibiendo datos del dispositivo BioHarness. Una vez desconectado el status cambiará a "Desconectado del BioHarness". El botón de "Enviar" permite que la aplicación se conecte con el servidor y se realice el envío de los datos al mismo.

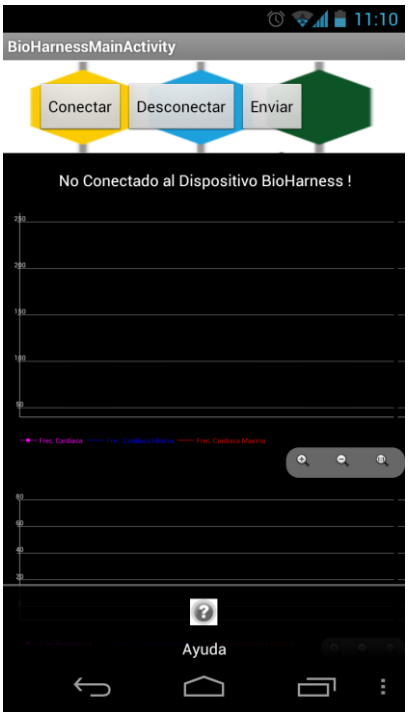


Figura 19. Ventana principal de la BioHarness App.

Fuente: Elaboración Propia.

IV.2.2. Servidor de aplicaciones

El servidor web fue desarrollado en Node.js y guardado con el nombre app.js. En la Figura 20 se observan los comandos y pasos a seguir para arrancar el servidor desde el terminal de la MAC. Para compilar el programa del servidor en Node.js, hay que estar ubicado en el directorio y el archivo en el que este se encuentra. Luego escribiendo el comando **node app.js** en el terminal de la computadora se arranca el servidor. Se debe esperar a que aparezca el mensaje de "Express server listening on port 3000" para estar seguro de que el servidor está corriendo, en caso contrario se mostrará que hay un error con la conexión.

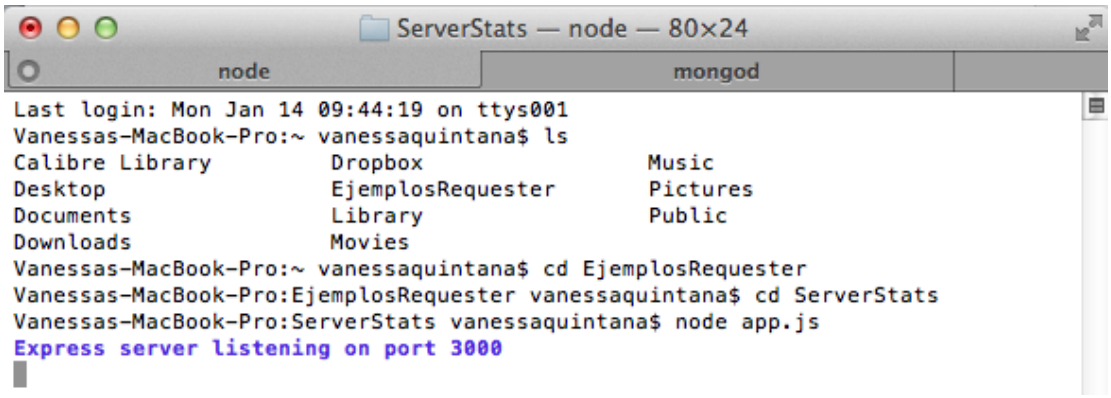


Figura 20. Comando de arranque del servidor.

Fuente: Elaboración propia.

IV.3. Pruebas.

A continuación se pueden ver los resultados obtenidos luego de realizar las pruebas de la aplicación y servidor. En ella podemos ver la correcta recepción de los datos para ambos casos.

IV.3.1. Conexión entre BioHarness y aplicación móvil

Para la realización de pruebas entre el dispositivo BioHarness y la aplicación BioHarness App se encendió el dispositivo, y se siguieron los pasos correspondientes (mostrados en el menú de ayuda) para realizar la conexión. Luego se entró en la aplicación y se hizo clic en el botón de conectar. Inmediatamente se empezaron a graficar cada uno de los valores en tiempo real. En la Figura 21 se muestran los gráficos para cada uno de los parámetros. En estos gráficos se puede observar que los valores obtenidos del dispositivo se encuentran en el rango normal. También podemos visualizar en los gráficos cada uno de los valores de forma numérica que arroja el dispositivo.

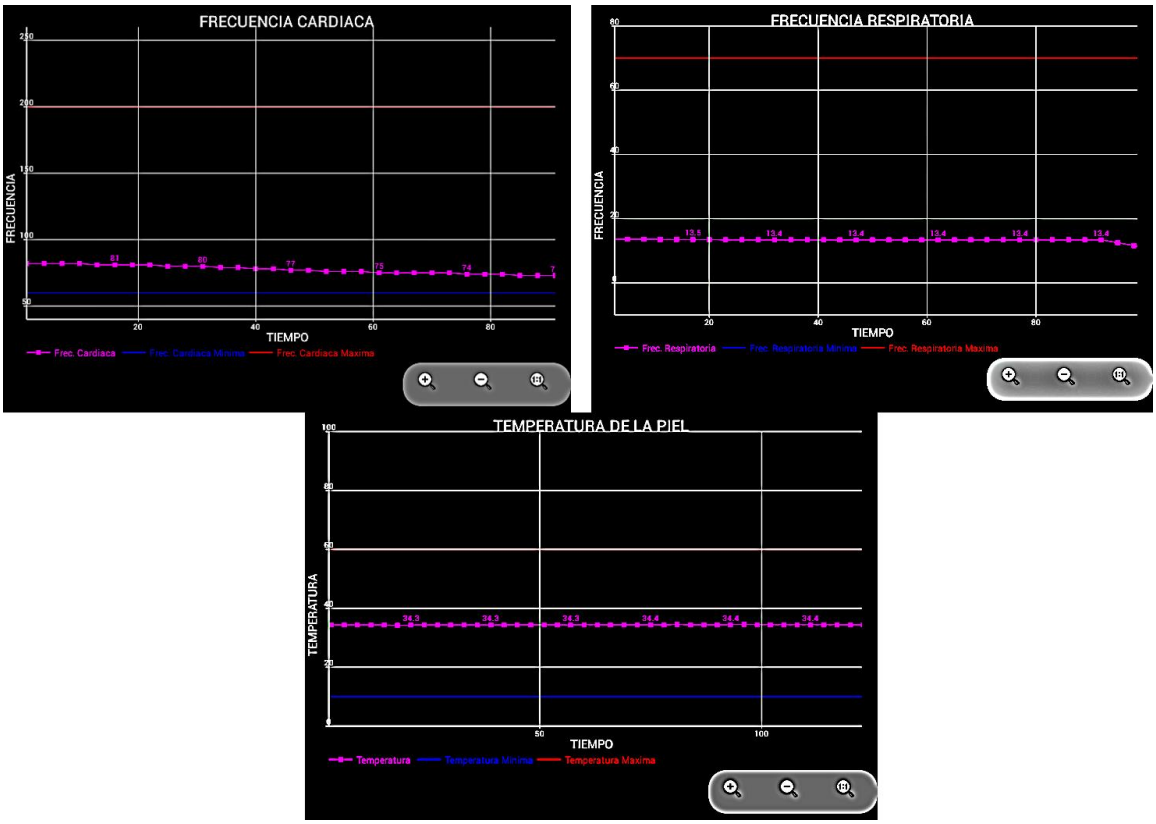


Figura 21. Gráficos de los parámetros en tiempo real.

Fuente: Elaboración propia.

Se realizaron varias pruebas para verificar que los dos dispositivos se conectaran de forma correcta, observándose que al colocar el dispositivo al paciente y encender el BioHarness, era necesario esperar entre 3 y 4 minutos antes de conectarlo a la aplicación, para que el dispositivo se estabilizara y empezara a enviar datos que correspondieran a los reales..

También se observó, en algunos casos, que al encender la aplicación no se logra visualizar el primer gráfico a pesar de que se continuaba viendo el envío de los datos correspondientes a los otros gráficos. En estos casos la solución es salir de la aplicación y volver a conectarse al dispositivo.

IV.3.2. Conexión entre la aplicación móvil y el servidor

Se arrancó el servidor con los comandos que se explicaron anteriormente, se inició la aplicación y se activó la conexión Wi-Fi del teléfono celular o tableta. Luego se procedió a conectar la aplicación al dispositivo para que se iniciara el envío de los datos y los gráficos de cada uno de los valores.

El servidor que se implementó, a diferencia de la aplicación, no trabaja en tiempo real. Así que se desconectó del dispositivo portátil, luego de unos segundos, y se pulsó la tecla "Enviar". En ese momento aparece la barra de carga, Figura 22, lo que indica que se está realizando el envío de la información.

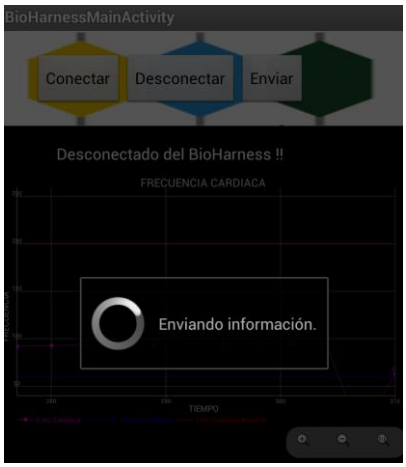


Figura 22. Dialogo que indica el envío de información al servidor.

Fuente: Elaboración propia.

El envío de los datos se realiza aproximadamente en tiempos entre 12 ms y 80 ms, Figura 23, este tiempo de envío va a depender de la velocidad de la red en la que se esté conectada.

```
POST /all 200 12ms - 37
```

Figura 23. Tiempo de envío de los datos.

Fuente: Elaboración propia.

Al presionar el botón de envío, se genera automáticamente el JSON, Figura 24 . Aquí se agrupan como arreglo los tres parámetros que se van a enviar desde la aplicación. Este arreglo solamente se visualiza desde la cónsola de la herramienta de desarrollo *Eclipse*. Para esto de debe compilar la aplicación y usarla cundo el teléfono esté conectado a la computadora por el puerto USB.

EL JSON {"HeartRate":["68","69","69","70","71","71","71","71"],"RespRate":[⏎
"35.7","35.4","35.3","35.3","35.4","35.3","35.3","35.3"],"Temperatu ⏎
re":["10.3","10.3","10.2","10.2","10.2","10.2","10.1","10.1"]}

Figura 24. Arreglo en Eclipse enviado por la aplicación.

Fuente: Elaboración propia.

En la Figura 25 se observan los datos que recibió el servidor. Los mismos se presentan en forma de arreglo y corresponden exactamente a los enviados desde la aplicación.

{ HeartRate: ['68', '69', '69', '70', '71', '71', '71', '71'],
  RespRate: ['35.7', '35.4', '35.3', '35.3', '35.4', '35.3', '35.3', '35.3'],
  Temperature: ['10.3', '10.3', '10.2', '10.2', '10.2', '10.2', '10.1', '10.1'
] }
POST /all 200 12ms - 37

Figura 25. Arreglo recibido en el servidor.

Fuente: Elaboración propia.

En ninguna de las pruebas se observaron errores al comparar los datos enviados por la aplicación con los recibidos en el servidor, lo que es de suma importancia en este tipo de aplicaciones, ya que se si llega algún dato errado, la lectura que puede realizar un médico será errada. Igualmente en futuros trabajos se deberían realizar pruebas más extensas para verificar que no existan errores en los datos y calcular el margen de error.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan las conclusiones obtenidas del desarrollo y análisis de los resultados expuestos en este Trabajo Especial de Grado. También se hacen las recomendaciones necesarias para mejorar a largo plazo la aplicación móvil y el sistema de transmisión implementado.

V.1. Conclusiones

A lo largo de todo el proyecto se presenta el desarrollo e implementación de una aplicación móvil que permite la toma de señales básicas médicas de un paciente. De esta forma se logró aplicar la telemedicina para que el paciente pueda ser monitorizado sin necesidad de trasladarse de su hogar o del lugar donde se encuentre recibiendo atención primaria. La implementación de este proyecto se logró gracias al uso de tecnologías de envío de datos como Wi-Fi, las herramientas de desarrollo, y por supuesto Internet.

El desarrollo de este proyecto permitió adquirir conocimientos más extensos en el área de la telemedicina, específicamente en el área de telemetría o monitorización remota de pacientes, logrando experimentar la importancia y necesidad que existe en el mundo actual de crear aplicaciones que permitan el control y monitorización de parámetros biomédicos como los evaluados en este trabajo de grado y otros de gran importancia.

También se logró aplicar y ampliar los conocimientos adquiridos a lo largo de la carrera, principalmente en el área de programación y transmisión de datos, así como de nuevas tecnologías. Esto aportó un gran nivel de aprendizaje sobre fundamentos teóricos y nuevas tecnologías de la información y de la comunicación.

Es importante destacar que la aplicación desarrollada en este proyecto, se encarga de recibir los valores que toma el dispositivo BioHarness, por lo cual va a depender del dispositivo y del buen funcionamiento del mismo, el que los valores representados permitan una clara y real interpretación de los resultados.

Por último se puede concluir que el servidor implementado para este proyecto funciona de forma correcta con la conexión de Wi-Fi. Conviene señalar que no fue posible realizar las pruebas por otras vías de transmisión de datos (3G, EDGE, etc.), debido a dificultades de acceso al servidor de la Escuela de Telecomunicaciones. No obstante, el programa desarrollado para el servidor, una vez instalado en un servidor con IP pública, podrá permitir el envío de los datos por estas vías.

V.2. Recomendaciones

Para futuros grupos que deseen tomar este proyecto como punto de partida para el desarrollo de otras aplicaciones similares, se recomienda:

- Realizar pruebas más extensas con pacientes, para que se puedan obtener resultados concluyentes con respecto al uso extenso de la aplicación, ya que para esta aplicación las pruebas realizadas fueron entre 2 y 3 min.

- Diseñar, para una segunda etapa de este proyecto, el sistema que permita la visualización de los datos por el médico, para lo cual se debe desarrollar otra aplicación móvil bajo el ambiente *Android* que permita la selección y monitorización de diferentes pacientes.

- Desarrollar una base de datos que permita el almacenamiento de la información de todos los pacientes que hagan uso del dispositivo. Esta puede ser implementada con MongoDB que es un sistema de base de datos de nueva generación NoSQL que está orientado a documentos, ya que es compatible con los programas que se están trabajando.

- Implementar el mismo programa que fue usado para el servidor de este proyecto en el servidor de la Escuela de Telecomunicaciones, lo que permitirá que la aplicación se pueda conectar al servidor a través de una IP pública y que se puedan enviar los datos no solo por Wi-Fi sino por la red celular haciendo uso de tecnologías como 3G, EDGE, entre otros. Así mismo se recomienda agregar el código necesario para realizar el envío de los datos del servidor con el móvil del médico que vaya a evaluar dichos datos.

REFERENCIAS BIBLIOGRÁFICAS

Zephyr Technology Corporation. (2010). Recuperado el 20 de Febrero de 2012, de http://www.zephyr-technology.com/store/faqs.html

Zephyr Technology Corporation. (2010). Recuperado el 20 de Febrero de 2012, de http://www.zephyr-technology.com/bioharness-bt

android scenebeta.com. (6 de Abril de 2011). Recuperado el 19 de Mayo de 2012, de http://android.scenebeta.com/tutorial/librerias

Universidad de Valencia. (13 de Septiembre de 2012). Recuperado el 24 de Noviembre de 2012, de http://www.uv.es/jac/guia/asp/asp9.htm

Adrian. (11 de Septiembre de 2010). *El androide libre.* Recuperado el 18 de Mayo de 2012, de http://www.elandroidelibre.com/2010/09/que-significa-el-diccionario-android.html

Alecrim, E. (16 de Mayo de 2011). *Info Wester*. Recuperado el 28 de Junio de 2012, de http://www.infowester.com/bluetooth.php

Alexander, K. (2008). RDF/JSON: A Specification for serialising RDF in JSON. United Kindom.

American Heart Association. (04 de Abril de 2012). Recuperado el 16 de Abril de 2012, de http://www.heart.org/HEARTORG/Conditions/More/MyHeartandStrokeNews/Heart-Rate---Pulse_UCM_438850_Article.jsp

American Telemedicine Association. (2012). Recuperado el 22 de Mayo de 2012, de http://www.americantelemed.org/i4a/pages/index.cfm?pageID=3333

Business Development Software. (s.f.). Recuperado el 18 de Mayo de 2012, de http://www.proyectosbds.com/software-y-programacion-a-medida/programacion-php-lamp-zf/mas-sobre-el-patron-mvc/120/

Calle, F. (24 de Junio de 2008). *Arquitectura de 3 Capas.* Obtenido de http://www.slideshare.net/Decimo/arquitectura-3-capas

Castro, R. (Sptiembre de 2005). Avanzando en la seguridad de las redes WIFI. *Boletin de RedIRIS*(73), 10.

Dugdale, D. C. (20 de Febrero de 2011). *University of Maryland Medical Center*. Recuperado el 18 de Abril de 2012, de http://www.umm.edu/ency/article/002341.htm

Eckstein, R. (Marzo de 2007). *Java SE Application Design With MVC.* Recuperado el 18 de Mayo de 2012, de http://www.oracle.com/technetwork/articles/javase/index-142890.html

Gallardo, D. (Julio de 2007). *Getting started with the Eclipse Platform*. Recuperado el 19 de Abril de 2012, de http://www.ibm.com/developerworks/library/os-ecov/

Google. (s.f.). *google_gson*. Recuperado el 24 de Noviembre de 2012, de http://code.google.com/p/google-gson/

Groussard, T. (2010). *Java Enterprise Edition- Desarrollo de aplicaciones Web con JEE6.* Barcelona: Ediciones ENI.

Hernández, J. A. (s.f.). *Mundo Atletismo.* Recuperado el 10 de Abril de 2012, de http://www.mundoatletismo.com/Site/atletismopopular/01d67c944b0dec402.html

Job, N., Malaikai, S., & Schenker, M. (03 de Mayo de 2010). *Developers Work.* Recuperado el 24 de Noviembre de 2012, de IBM: http://www.ibm.com/developerworks/ssa/xml/library/x-db2JSONpt1/

JSON. (s.f.). *JSON Organization*. Recuperado el 24 de Noviembre de 2012, de http://www.json.org/json-es.html

Macias, J. L. (2007). *Universidad de Huelva.* Recuperado el 24 de Noviembre de 2012, de http://www.uhu.es/josel_alvarez/NvasTecnProg/recursos/ProtocoloHTTP.pdf

Martinez, J. C. (26 de Septiembre de 2007). *Introducción al entorno de desarrollo Eclipse.* Recuperado el 19 de Mayo de 2012, de http://www-gris.det.uvigo.es/wiki/pub/Main/MiscResources/Manual_Eclipse.pdf

Microsoft. (21 de Enero de 2005). *Windows Server*. Recuperado el 24 de Noviembre de 2012, de http://technet.microsoft.com/en-us/library/cc787920(v=ws.10).aspx#feedback

Mitchell, B. (s.f.). *About.com*. Recuperado el 22 de Febrero de 2012, de http://compnetworking.about.com/cs/bluetooth/g/bldef_bluetooth.htm

Node .js. (s.f.). Recuperado el 22 de Mayo de 2012, de http://nodejs.org/

Nordqvist, C. (08 de Octubre de 2011). *Medical News Today*. Recuperado el 16 de Mayo de 2012, de http://www.medicalnewstoday.com/articles/235710.php

Núñez, C. V., Peña, J. C., & Garzon, C. L. (2009). Análisis comparativo de tecnologías. En *Ingenieria y Desarrollo* (págs. 200-217). Barranquilla, Colombia.

Pasquier, D. R. (13 de Agosto de 2010). *El efisgomanómetro o "tensiómetro"*. Recuperado el 23 de febrero de 2012, de Medicina Preventiva: http://www.medicinapreventiva.com.ve/auxilio/tensiometro.htm

Pérez, N., & Joffre, E. (Diciembre de 2000). *Universidad Central de Venezuela*. Recuperado el 19 de Abril de 2012, de Facultad de Ingenieria.: http://neutron.ing.ucv.ve/revista-e/No8/Joffre%20Perez/Bluetooth.1.htm

Realza, J. A. (27 de Abril de 2012). *Introducción a Node.js*. Recuperado el 22 de Mayo de 2012, de http://www.skillcorp.com.ve/articulo/Introduccion-a-NodeJS

Sara, A. (19 de Septiembre de 2012). *desarrollo Web*. Recuperado el 24 de Noviembre de 2012, de http://www.desarrolloweb.com/articulos/protocolo-http-ftp.html

Sparacino, G. L. (2003). TECNOLOGÍA INALÁMBRICA BLUETOOTH SOBRE LOS SERVICIOS DE COMUNICACIONES EN LOS ÁMBITOS SOCIAL Y EMPRESARIAL. *Revista electronica de estudios telematicos, 2*, 14.

Telemedicina Mexico. (s.f.). Recuperado el 22 de Abril de 2012, de http://www.telemedicina.org.mx/

Terkaly, B. (Marzo de 2012). Write Scalable, Server-side JavaScript Aplications with Node.js. *MSDN Magazine*.

Vorvick, L. J. (17 de Febrero de 2011). *University of Maryland Medical Center*. Recuperado el 08 de Abril de 2012, de http://www.umm.edu/esp_ency/article/001982.htm

ANEXOS

Anexo A

Glosario de términos

ACRÓNIMO	TÉRMINO EN INGLES	TÉRMINO EN ESPAÑOL
APA	American Psychological Association	Asociación americana de psicología
API	Application Programming Interface	Interfaz de programación de aplicaciones
ATA	American Telemedicine Association	Asociación de Telemedicina Americana
DNS	Domain Name System	Sistema de nombres de dominio
EDGE	Enhanced Data Rates for GSM Evolution	Tasas de Datos Mejoradas para la evolución de GSM
GPRS	General packet radio service	Servicio general de paquetes radio
HTTP	Hypertext Transfer Protocol	Protocolo de transferencia de Hipertexto
IBM	International Business Machines	Maquinas de negocio internacional
IDE	Integrated Development Environment	Ambiente de desarrollo integrado
IEEE	Institute of Electrical and Electronics Engineers	Instituto de ingeniería eléctrica y electrónica
IP	Internet Protocol	Protocolo de internet
JDK	Java Development Kit	Kit de desarrollo de java
JSON	JavaScript Object Notation	Notación de objetos de JavaScript

MVC	Model View Controller	Modelo Vista Controlador
PDA	Personal digital assistant	Asistente digital personal
PHP	Hypertext Pre Processor	Pre procesador de hipertexto
SDK	Software Development Kit	Kit de desarrollo de Software
SQL	Structured Query Language	Lenguaje de consulta estructurado
TCP	Transfer Control Protocol	Protocolo de control de transmisión
UNIX	Uniplexed Information and Computing Service	Información Uniplexed y Servicio de informática
URI	Uniform Resource Identifier	Identificador uniforme de recursos
URL	Uniform Resource Locator	Localizador uniforme de recursos
USB	Universal Serial Bus	Bus universal en series
VPN	Virtual Private Network	Red Privada Virtual
WIFI	Wireless Fidelity	Fidelidad inalámbrica
WLAN	Wireless Local Area Network	Red de área local inalámbrica
XML	eXtensible Mark-up Language	Lenguaje de Marcado Extensible

Anexo B

Características y descripción del BioHarness BioModule

Zephyr BioHarness ™ BT

DESCRIPCIÓN DEL PRODUCTO

El BioHarness ™ BT es un módulo electrónico compacto. Está conectado a una correa de tela ligera inteligente que incorpora sensores de ECG y respiración de detección.

El módulo BioHarness ™ puede transmitir datos fisiológicos a través de Bluetooth o grabarlo en la memoria interna.

Equipo	**Precio en $**
BioHarness BioModule	650 $
Base o cargador	25 $
Banda Ajustable	55 $

Especificaciones del equipo.

Parámetros	Notas	Min	Tipo	Max	Unidades
General					
Registro de Capacidad	1		570		Horas
Voltaje de alimentación	USB		5	5.5	V
Duración de batería-transmisión de radio	2		24	21	Horas
Registro de duración de batería	3		3		Horas
Tiempo de carga	Entre cargas		6		Horas
Almacenamiento	4		300		Ciclos
Ciclos de carga			10		Bits
Impedancia de entrada DC		20			MΩ
Frecuencia Cardiaca					
Rango		25		240	BPM
	A 60 bpm		7		s
Tiempo sin respuesta	60 a 0 bpm		7		s
Intervalo de muestreo del sensor ECG			4		ms
Rango de entrada dinámica		0.1		10	mVpp
Frecuencia respiratoria					
Rango de valores		3		70	BPM
Intervalo de muestreo del sensor de respiración			10		ms
Temperatura de la piel					
Rango	5	10		60	C
Precisión	30 a 40C		0.2		C
Tiempo de respuesta			5		s
intervalo de muestreo	6		60		s
Actividad					
VMU(Unidades de vector magnitud)	1.			16	g
intervalo de muestreo			8		ms
Ancho de banda		0.06		9	Hz
Rango dinámico		-16		16	g
Sensibilidad			10		mg
Ruido			7.2		mg
Postura					
Rango dinámico	2.	-180		+180	Grados
intervalo de muestreo			8		ms
Sensibilidad		8		1	Grados

Características RF.

Conformidad del Bluetooth	versión 2.0 + EDR
Perfil de soporte	Puerto Serial
Frecuencia de operación	2.4 a 2835 GHz
Potencia de salida	2mW
Rango de operación	10 m radio típico en interior (línea de vista)
Tipo de antena	Interna

Estatus de indicación Rojo/ Naranja/ Verde.

Este es un valor que se calcula en el dispositivo y depende de cuatro umbrales:

- Frecuencia cardíaca mínima
- Frecuencia cardíaca máxima
- Frecuencia respiratoria mínima
- Frecuencia respiratoria máxima

Los valores actuales y previos de frecuencia cardíaca y frecuencia respiratoria se utilizan junto con el nivel de actividad para establecer el estado de un sujeto, haciendo uso de algoritmos de Zephyr.

Los niveles umbral se almacenan en el dispositivo y se pueden configurar por USB.

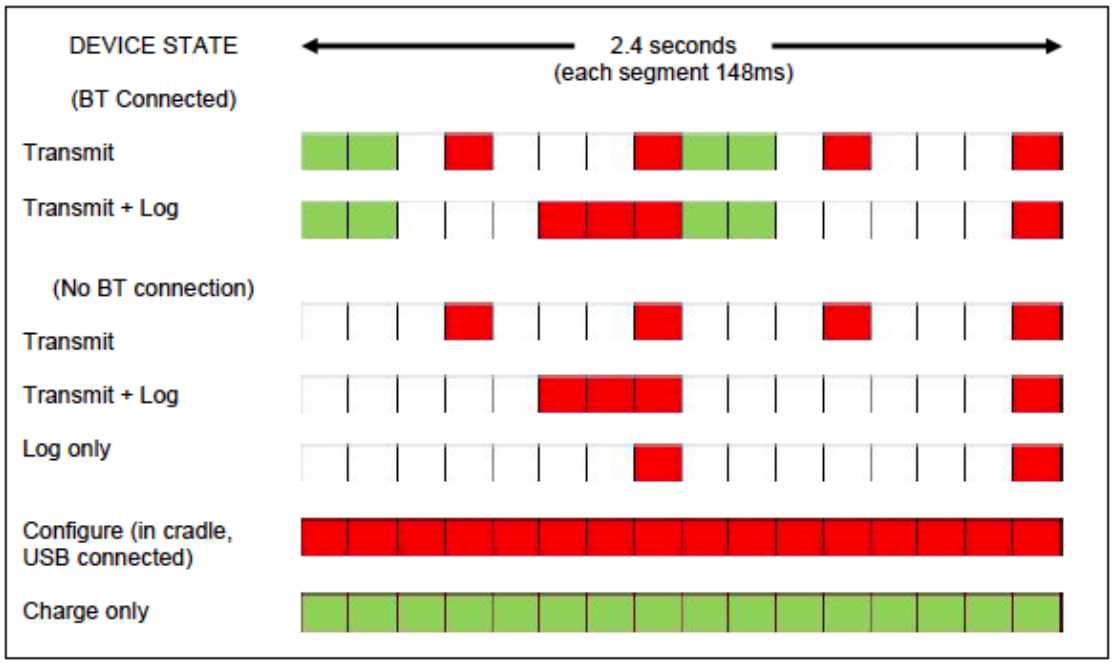
Comportamiento de los LED.

El modulo BioHarness puede operar en tres modos:

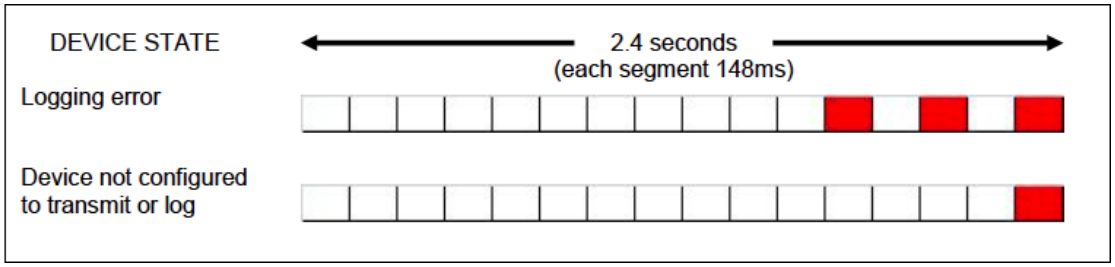
- Transmisión de datos vía Bluetooth.
- Registro de datos en la memoria interna (sin transmisión)
- Transmitir y registrar los mismos datos al mismo tiempo

El dispositivo se puede configurar para estos modos usando el BT Config Tool. Un LED rojo parpadeante indicará en cuál de los modos anteriores se encuentra el dispositivo.

A BioHarness puede transmitir datos, pero debe de haber establecido una conexión Bluetooth con un dispositivo receptor. Un dispositivo puede transmitir datos e indicarlo mediante un parpadeo del LED rojo, incluso a pesar de que no está conectado a un dispositivo receptor. Un parpadeo del LED verde indica que la conexión está presente.



Indicaciones de error con el LED



Anexo C

Guía de usuario para la aplicación en Android.

El documento que se muestra a continuación describe la funcionalidad del API BioHarness App. Aquí se presenta un resumen de todas las clases, métodos y variables que fueron implementadas en el API para establecer comunicación con el dispositivo BioHarness.

El documento va dirigido al público de programadores en Android que necesiten una rápida introducción a la aplicación desarrollada en el trabajo de investigación.

A continuación, se muestran las clases utilizadas en el desarrollo de la BioHarness App, y las variables y métodos implementados en cada una de ellas.

La clase LineHeartRateGraph.java.

Esta clase es la encargada de manejar todo el contenido para el desarrollo de los gráficos. Genera los conjuntos de datos múltiples que corresponden a cada una de las series que se van a visualizar en el gráfico de frecuencia cardíaca, personaliza todas las características de la misma y recibe los diferentes valores de los datos dinámicos.

Nombre Variable	Tipo de clase Variable	Propósito
view	GraphicalView	Guarda la vista gráfica
maxHeartRate	TimeSeries	Almacena un valor mínimo constante correspondiente a la frecuencia cardíaca
minHeartRate	TimeSeries	Almacena un valor máximo constante correspondiente a la frecuencia cardíaca
HeartRate	TimeSeries	Almacena el valor dinámicos de la frecuencia cardíaca

mDataset	XYMultipleSeriesRenderer	Guarda el conjunto de los valores máximos, mínimos y dinámicos.
maxRenderer	XYSeriesRenderer	Guarda la personalización especifica de la serie de frecuencia máxima.
minRenderer	XYSeriesRenderer	Almacena la personalización especifica de la serie de frecuencia mínima
heartRateRenderer	XYSeriesRenderer	Almacena la personalización del grafico en si.
mRenderer	XYMultipleSeriesRenderer	Guarda la personalización de cada uno de los gráficos en un conjunto

Los métodos para esta clase son:

Nombre del método	Propósito del método
LineHeartRateGraph()	Constructor donde se generan los conjuntos de los datos y la personalización de las líneas.
getView(Context)	Genera la vista gráfica de todas las series, con sus características.
addNewPoints(Point)	Regresa un punto p, conformado por un valor del eje X y otro del eje Y
minConstantNewPoints(Point)	Crea el conjunto de datos para el valor mínimos constante de la frecuencia cardíaca.
maxConstantNewPoints(Point)	Crea el conjunto de datos para el valor máximos constante de la frecuencia cardíaca.

La clase LineTemperatureGraph.java.

Esta clase funciona de forma parecida a la clase de *LineHeartRateGraph*, pero esta contiene todas las variables y métodos correspondientes a la data de Temperatura de la piel.

Nombre Variable	Tipo de clase Variable	Propósito
view	GraphicalView	Guarda la vista gráfica

maxTemperature	TimeSeries	Almacena un valor mínimo constante correspondiente a la temperatura
minTemperature	TimeSeries	Almacena un valor máximo constante correspondiente a la temperatura
Temperature	TimeSeries	Almacena el valor dinámicos de la temperatura
mDataset	XYMultipleSeriesRenderer	Guarda el conjunto de los valores máximos, mínimos y dinámicos.
maxRenderer	XYSeriesRenderer	Guarda la personalización especifica de la serie de temperatura máxima.
minRenderer	XYSeriesRenderer	Almacena la personalización especifica de la serie de temperatura mínima
temperatureRenderer	XYSeriesRenderer	Almacena la personalización del grafico en si.
mRenderer	XYMultipleSeriesRenderer	Guarda la personalización de cada uno de los gráficos en un conjunto

Los métodos correspondientes a esta clase son:

Nombre del método	Propósito del método
LineTemperatureGraph()	Constructor donde se generan los conjuntos de los datos y la personalización de las líneas.
getView(Context)	Genera la vista gráfica de todas las series, con sus características.
addNewPoints(Point)	Regresa un punto p, conformado por un valor del eje X y otro del eje Y
minConstantNewPoints(Point)	Crea el conjunto de datos para el valor mínimos constante de la temperatura.
maxConstantNewPoints(Point)	Crea el conjunto de datos para el valor máximos constante de la temperatura.

La clase LineRespirationGraph.java.

Al igual que las clases anteriores, esta clase se encarga de recibir y graficar los valores correspondientes a la frecuencia respiratoria.

Nombre Variable	Tipo de clase Variable	Propósito
view	GraphicalView	Guarda la vista gráfica
maxRespirationRate	TimeSeries	Almacena un valor mínimo constante correspondiente a la frecuencia respiratoria
minRespirationRate	TimeSeries	Almacena un valor máximo constante correspondiente a la frecuencia respiratoria
RespirationRate	TimeSeries	Almacena el valor dinámicos de la frecuencia respiratoria
mDataset	XYMultipleSeries Renderer	Guarda el conjunto de los valores máximos, mínimos y dinámicos.
maxRenderer	XYSeriesRenderer	Guarda la personalización especifica de la serie de frecuencia respiratoria máxima.
minRenderer	XYSeriesRenderer	Almacena la personalización especifica de la serie de frecuencia respiratoria mínima
RespirationRateRenderer	XYSeriesRenderer	Almacena la personalización del grafico en si.
mRenderer	XYMultipleSeries Renderer	Guarda la personalización de cada uno de los gráficos en un conjunto

Los métodos para esta clase son:

Nombre del método	Propósito del método
LineRespirationRateGraph()	Constructor donde se generan los conjuntos de los datos y la personalización de las líneas.
getView(Context)	Genera la vista gráfica de todas las series, con sus características.
addNewPoints(Point)	Regresa un punto p, conformado por un valor del eje X y otro del eje Y
minConstantNewPoints(Point)	Crea el conjunto de datos para el valor mínimos constante de la frecuencia respiratoria.
maxConstantNewPoints(Point)	Crea el conjunto de datos para el valor máximos constante de la frecuencia respiratoria.

La clase HTTPClientHandler.java.

Esta clase se encarga de crear el cliente http, abrir la conexión con el servidor, manejar todos los mensajes de request, response, get, put, etc, que se realizan entre el cliente y el servidor, y realizar el envío de los valores de cada uno de los parámetros Biomédicos en forma de arreglo.

Nombre Variable	**Tipo de clase Variable**	**Propósito**
baseURL	String	Contiene el String de la dirección URL en el que se encuentra ubicado el Servidor
httpClient	DefaultHttpClient	Contiene el cliente http con el que se va a trabajar.
errorType	String	Almacena el tipo de error ocurrido
responseCode	int	Guarda el código de respuesta del servidor

Los métodos para esta clase son:

Nombre del método	**Propósito del método**
getBaseURL()	Método que regresa el String de la dirección URL
createHttpClient()	Crea el cliente Http por default con todas sus características.
pruebaRequest()	Recibe un String JSON que contiene todos los valores de los parámetros que van a ser enviados al servidor,
getHttpClient()	Regresa el cliente Http
conexionDisponible()	Verifica si la conexión con el servidor esta disponible
obtenerDatosServidor(URI,HttpEntity)	Método que realiza un Request POST
obtenerDatosServidorGet(URI)	Método que realiza un Request GET
obtenerDatosServidorPut(URI,HttpEntity)	Método que realiza un Request PUT
getErrorType()	Regresa el error que ocurrió.

setErrorType(String)	Recibe el tipo de error que ha ocurrido.

La clase NewConnectedListener.java.

Se encarga de manejar el procesamiento de los paquetes de entrada desde el BioHarness y es responsable de la creación de objetos que analizan la secuencia de entrada de datos y otros objetos y métodos que muestran los datos en la pantalla del teléfono.

Nombre Variable	**Tipo de clase Variable**	**Propósito**
HEART_RATE	int	Contiene el valor de la frecuencia cardíaca
RESPIRATION_RATE	int	Contiene el valor de la frecuencia respiratoria
SKIN_TEMPERATURE	int	Contiene el valor de la temperatura
GP_MSG_ID	int	Guarda el numero de identificación de mensaje del paquete general
BREATHING_MSG_ID	int	Almacena la identificación del mensaje de cada paquete de frecuencia respiratoria del dispositivo BioHarness
ECG_MSG_ID	int	Almacena la identificación del mensaje de cada paquete de frecuencia cardíaca del dispositivo BioHarness
ACCEL_100mg_MSG_ID	int	Almacena la identificación del mensaje de cada paquete de aceleración del dispositivo BioHarness
SUMMARY_MSG_ID	int	Contiene el mensaje de identificación del Summary Packet
GP_HANDLER_ID	int	Es el número de identificación del manejador del paquete general.
GPInfo	GeneralPacketInfo	Guarda la información general de todos los paquetes de datos que vienen del BioHarness
ECGInfoPacket	ECGPacketInfo	Almacena la información del paquete de datos de frecuencia cardiaca.

BreathingInfoPacket	BreathingPacketInfo	Almacena la información del paquete de datos de frecuencia respiratoria.
RqInfoPacket	PacketTypeRequest	Almacena un objeto de la clase *PaquetTypeRequest* que permite habilitar o deshabilitar los diferentes tipos de paquetes que soporta el API.

Los métodos para esta clase son:

Nombre del método	**Propósito del método**
NewConnectedListener(Handler,Handler)	Crea la conexión con el dispositivo Bluetooth, y permite la conexión con los datos

La clase Point.java.

Se encarga de tomar los valores del eje X, y del eje Y y los convierte en un punto. Este maneja el punto dinámico para cada una de las graficas.

Nombre Variable	**Tipo de clase Variable**	**Propósito**
x	int	Guarda los valores del eje de coordenadas X.
y	double	Guarda los valores del eje de coordenadas Y.

Los métodos que corresponden a la clase son los siguientes:

Nombre del método	**Propósito del método**
Point(int, double)	Genera un punto en el plano de coordenada con los puntos X y Y.
getX( )	Obtiene el valor de X
getY( )	Muestra el valor de Y

La clase StatusJson.java.

Nombre Variable	Tipo de clase Variable	Propósito
code	String	Almacena el String del código de respuesta del servidor.
msg	String	Guarda el String del mensaje de respuesta del servidor.

Los métodos que corresponden a la clase son los siguientes:

Nombre del método	Propósito del método
StatusJson(String,String)	Genera un objeto de tipo String que será usado para generar el JSON.
getCode( )	Obtiene el código de tipo String correspondiente.
setCode(String)	Método que establece el código String.
getMsg( )	Obtiene la línea del mensaje tipo String correspondiente.
setMsg(String)	Método que establece la línea del mensaje que viene del servidor.

La clase StatsJson.java.

Nombre Variable	Tipo de clase Variable	Propósito
HeartRate	String[]	Almacena un arreglo de String de datos de frecuencia cardíaca.
Temperature	String[]	Guarda un arreglo de String de datos de Temperatura.
RespRate	String[]	Almacena un arreglo de String de datos de frecuencia respiratoria.

Los métodos que corresponden a la clase son los siguientes:

Nombre del método	Propósito del método
StatsJson (String[], String[]. String[])	Método constructor que agrupa todos los arreglos de cada parámetro.

getHeartRate()	Obtiene el arreglo de String de la frecuencia cardíaca.
setHeartRate(String[])	Método que establece el arreglo de String con todos los valores de la frecuencia cardíaca
getTemperature()	Obtiene el arreglo de String de la temperatura.
setTemperature(String[])	Método que establece el arreglo de String con todos los valores de la Temperatura
getRespirationRate()	Obtiene el arreglo de String de la frecuencia respiratoria
setRespirationRate(String[])	Método que establece el arreglo de String con todos los valores de la frecuencia respiratoria.