Web Service para implementación del estándar SCP-ECG orientado al geoposicionamiento

Nelson Iván Fernández Suarez

Gabriel Enrique Suárez Colmenares

ESCUELA DE INGENIERIA DE SISTEMAS E INFORMATICA

FACULTAD DE INGENIERIAS FISICOMECANICAS

UNIVERSIDAD INDUSTRIAL DE SANTANDER

BUCARAMANGA

2011

Web Service para implementación del estándar SCP-ECG orientado al geoposicionamiento

Nelson Iván Fernández Suarez

Gabriel Enrique Suárez Colmenares

Proyecto de investigación para optar por el título de:

Ingeniero de sistemas

Directora:

MsC. Lola Xiomara Bautista Rozo

Codirectores:

PhD. Homero Ortega Boada

MsC(c). Celso Andrés Forero Flórez

ESCUELA DE INGENIERIA DE SISTEMAS E INFORMATICA

FACULTAD DE INGENIERIAS FISICOMECANICAS

UNIVERSIDAD INDUSTRIAL DE SANTANDER

BUCARAMANGA

2011

A mi familia por el apoyo incondicional durante toda mi vida, mis amigos por haberme acompañado en esta gran etapa de formación profesional, y a Natalia, la luz de mi vida y mi mejor amiga, sin ella nada de esto hubiera sido posible

- Gabriel Enrique Suárez Colmenares

Para mi padre, toda mi familia, la comunidad de la EISI, y Dios

- Nelson Fernández

AGRADECIMIENTOS:

Al Profesor Homero Ortega y a todos los integrantes del grupo RadioGIS por toda la gestión, el apoyo y la ayuda brindada a nosotros para lograr el desarrollo de este proyecto.

A la profesora Lola Bautista por su asesoría y apoyo desde que se hizo cargo como directora de este proyecto, su gran conocimiento en el campo fue importante para la culminación del trabajo de investigación.

Al gran concepto como evaluadora de la profesora Sonia Gamboa, quien con sus conocimientos en el campo de la investigación y desarrollo software contribuyo en la calificación del proyecto.

A mi compañero de proyecto, quien con su gran esfuerzo y conocimientos nunca se rindió a pesar de la adversidad.

Al ing. Omar Gallo, quien nos brindo apoyo vital para culminar con la evaluación de proyecto.

RESUMEN

TITULO*: WEB SERVICE PARA IMPLEMENTACION DEL ESTANDAR SCP-ECG ORIENTADO AL GEOPOSICIONAMIENTO.

AUTORES:** NELSON IVAN FERNÁNDEZ SUÁREZ
GABRIEL ENRIQUE SUÁREZ COLMENARES

Palabras Clave: Servicio Web, Telemedicina, Android, RadioGIS, GIIB.

DESCRIPCIÓN

Los avances tecnológicos de años recientes han permitido grandes desarrollos en materia de internet móvil, telecomunicaciones, teléfonos inteligentes y demás dispositivos avanzados que permiten mantener a los usuarios siempre conectados con la internet lo que ha incentivado el desarrollo de miles de aplicaciones prácticas y sencillas tanto para entretenimiento como interacción social. Sin embargo, no se ha explotado al máximo el gran potencial que tiene integrar tecnologías móviles con aplicativos ya existentes como lo son los servicios de telemedicina. Existen dispositivos para electrocardiografía portátiles actualmente en el mercado, que se encargan de monitorear la condición cardiaca de un paciente y algunos de estos son tan pequeños que pueden ser llevados a cualquier parte sin ninguna dificultad, pero carecen de conectividad con la red, lo que impide comunicar los datos tomados a entidades especializadas en el cuidado de la salud.

Utilizando tecnologías de última generación como lo son el sistema operativo Android, Java SE 6, servicios web REST y aplicaciones web con JavaServer Faces2.0, trabajando en conjunto con el servidor de aplicaciones GlassFish 3.0.1 y el motor de base de datos PostgreSQL 9.0 fue posible desarrollar un prototipo de servicio web orientado al monitoreo de una señal ECG enviada por un cliente móvil que utiliza el estándar SCP-ECG para el empaquetamiento de los datos. Además integrando Google Maps fue posible geo-posicionar el dispositivo que envía la señal, creando un servicio de telemedicina con un gran potencial de desarrollo para la industria de la medicina asistida por medios electrónicos y las telecomunicaciones.

---

* Trabajo de Grado – Modalidad: Investigación

**Facultad de Ingenierías Fisicomecánicas. Escuela de Ingeniería de Sistemas e Informática. Directora: MSc. Lola Xiomara Bautista Rozo. Codirectores: PhD. Homero Ortega Boada, Ing. Celso Andrés Forero Flórez

SUMMARY

TITLE*: WEB SERVICE FOR IMPLEMENTATION OF STANDARD SCP-ECG GEO-POSITIOTING-ORIENTED.

AUTHORS:** NELSON IVAN FERNÁNDEZ SUÁREZ
GABRIEL ENRIQUE SUÁREZ COLMENARES

Key Words: Web Service, Telemedicine, Android, RadioGIS, GIIB.

DESCRIPTION

Technological breakthrough of recent years, allowed mayor development in matter of mobile internet, telecommunications, Smartphones and some other advance devices that allow users to be always online which has encouraged the development of thousands of practical and simple mobile applications both for entertainment and social networking. However, it is not fully exploited the great potential of integrating mobile technologies with existing applications such as the telemedicine services. There are portable electrocardiograph devices available on the market, which are responsible for monitoring a patient's heart condition and some devices are so small that can be carried anywhere without difficulty, but lack of network connectivity, making it impossible to communicate the collected data to organizations specialized in health care

Using state of the art technologies like the Android Operating System, Java Standard Edition 6, REST web services and JavaServer Faces 2.0 web applications, working together with GlassFish 3.0.1 application server and the database engine PostgreSQL9.0 it was possible to develop a Web Service prototype oriented towards the monitoring of an ECG signal sent by a mobile client which uses the SCP-ECG standard for packaging the signal data. Also by integrating Google Maps into the JSF web application it was possible to geo-positionate the device that sends the signal, building a telemedicine service with huge potential of development for the electronically assisted medicine industry as well as telecommunications.

---

* Thesis – Research Work

**Physical Mechanical Engeneering Faculty. Systems Engineering and Informatics School. Director: MSc. Lola Xiomara Bautista Rozo. Co-Directors: PhD. Homero Ortega Boada, Eng. Celso Andrés Forero Flórez

TABLA DE CONTENIDO

LISTA DE FIGURAS

INTRODUCCIÓN

El gobierno Colombiano busca impulsar la masificación del uso de internet con su plan estratégico “Colombia Vive Digital” el cual plantea metas para aumentar la cantidad de conexiones existentes a banda ancha y lograr el uso masivo de las TIC (Tecnologías de la Información y Comunicación). Para ello ha visualizado lo que hoy se conoce como el ecosistema digital, con el fin de comprender en donde deben ser concentrados los esfuerzos del país. A partir de la lectura de este ecosistema es claro que el gobierno debe centrar importantes esfuerzos al desarrollo de aplicaciones de TIC.

En la actualidad, la calidad de vida guarda relación estrecha con el uso de las TIC. Esto se ve reflejado en la búsqueda del mejoramiento de múltiples servicios, como por ejemplo: la salud, puesto que la ciencia y la tecnología deben ser usadas con fines productivos y benéficos para la sociedad. La intervención de la ciencia para propósitos médicos permite el enfrentamiento a enfermedades que han tenido pocas posibilidades a una solución rápida y óptima. En este caso, las patologías cardiacas, que necesitan de una atención dinámica y pertinente. A partir de esto, nacen las aplicaciones de telemedicina – una práctica de la medicina con apoyo de las TIC.

Para la calidad de los servicios de la telemedicina se creó un estándar que formaliza la administración de los datos, para que estos tengan toda la información pertinente y sean más fáciles en su transmisión para los médicos y especialistas; este estándar es el SCP-ECG (*Standard Communications Protocol for Electrocardiography*) que está diseñado para la compresión, almacenamiento y comunicación específica de alta calidad de archivos ECG.

El desarrollo de telemedicina al que le apuesta este trabajo requiere de un dispositivo electrónico que permita obtener la información cardiaca del paciente durante las 24 horas del día. Además, incluye un mecanismo que permite la localización del enfermo por parte de los médicos y personas responsables del paciente, en caso de que ocurra una emergencia y sea necesario suministrar la mejor atención médica requerida. El dispositivo debe ser práctico, cómodo y fácil de llevar para el paciente para que pueda realizar sus actividades diarias y sacar el mejor provecho. Con los avances en hardware y la tecnología en Internet Móvil, teniendo en cuenta los alcances planteados en el programa “Vive Digital” del gobierno Colombiano, se puede garantizar que la información llegará al servidor donde será debidamente procesada y almacenada, junto al historial de datos del paciente.

1. PRESENTACIÓN DEL PROYECTO

1.1 OBJETIVO GENERAL

- Desarrollar un servicio basado en localización de telemedicina en la web que implemente el estándar SCP-ECG con geoposicionamiento para recibir y mostrar los signos vitales enviados por un electrocardiógrafo.

1.2 OBJETIVOS ESPECÍFICOS

- Desarrollar un prototipo software que envíe, interprete y muestre las señales de un electrocardiógrafo usando el estándar SCP-ECG.
- Implementar un módulo software de geoposicionamiento, que pueda ubicar la señal que se recibe del dispositivo y mostrarla en la aplicación.
- Construir un sitio web que muestre la información de la señal recibida de un electrocardiógrafo remoto relacionándola con su posición actual mediante el uso de una base de datos.

1.3 JUSTIFICACIÓN

Es de suma importancia brindar un seguimiento y un buen servicio de salud a los pacientes más delicados de cualquier región, razón por la cual se deben buscar nuevas formas de prestar el servicio médico a poblaciones de toda clase social, en cualquier lugar y en cualquier condición. Afortunadamente con el desarrollo de nuevas tecnologías en telecomunicaciones y con los grandes avances que se han realizado en los últimos años es posible aplicar este conocimiento y orientarlo hacia la atención médica con telemedicina. El desarrollo de una *web service* para la monitorización de la actividad cardiaca (ECG) de un paciente en todo momento y en cualquier lugar permitiría a los cuerpos de salud atender las emergencias que ocurran más eficientemente y a mantener un historial completo de las lecturas del corazón del paciente, lo que facilitaría el trabajo de los médicos a la hora de diagnosticar problemas o avances en la afección del paciente. Además, es un servicio construido con base a un estándar internacional como lo es SCP-ECG y que permitiría la interoperabilidad con diferentes dispositivos fabricados en distintas partes del mundo por lo cual se convertiría en una herramienta con disponibilidad de uso internacional.

Aprovechar las nuevas tecnologías en transporte de datos aplicadas a la medicina es un avance de gran importancia para cualquier sistema de salud de un país, esto permitiría llevar la medicina de alta especialización a cualquier parte de una región con un sencillo

acceso a internet y proveer atención de calidad a la población de esta locación. Con un desarrollo de este tipo se prevendría la muerte de muchos pacientes con enfermedades cardiacas, quienes pierden su vida por falta de recursos para recibir atención apropiada en una gran ciudad, a su vez los pacientes más delicados como aquellos de tercera edad o mujeres embarazadas podrían tener un seguimiento constante de su actividad cardiaca y además, aplicando georeferenciación al aprovechar la tecnología GPS y el servicio de google maps, se conoce la ubicación del paciente en todo momento, para atender las emergencias oportunamente y en el momento que ocurren, en base a todas las ventajas de realizar un desarrollo de este tipo se hace necesario el trabajo que se propone y así mejorar el sistema de salud, una vez el servicio este implementado completamente.

A su vez el desarrollo este proyecto se enmarca en la búsqueda del gobierno nacional con su plan "Vive Digital" de ampliar las fronteras de la informática, la información y las comunicaciones ampliando la cobertura del internet para proveer mejores servicios al pueblo Colombiano mejorando la calidad de vida e impulsando un desarrollo auto sostenible, y con esta iniciativa viene también la oportunidad de potenciar la plataforma de servicios basados en localización construida por el grupo RadioGIS (Location-Based Service), agregándole un componente más, en este caso orientado a la telemedicina, a sus servicios ofrecidos dejando este desarrollo para que sea implementado en un futuro, con las bondades en comunicación alcanzadas por Vive Digital y asimismo ofrecer al país una mejora en la atención al ciudadano.

1.4 IMPACTO

1.4.1 Técnico

La implementación de un estándar internacional para el trabajo de ECG permitiría desarrollar trabajos en conjunto con organizaciones de todo el mundo. Esto abriría las puertas para nuevas investigaciones y vincularía a la Universidad Industrial de Santander (U.I.S.) a un nivel internacional, debido a la relación del proyecto con la organización que da soporte al estándar (OpenECG). Asimismo, permitiría la interoperabilidad con gran variedad de dispositivos que se encuentran en el mercado internacional para la toma de señal ECG que manejan el estándar SCP-ECG, el cual podría ser monitoreado y geo-refenciado desde el sistema una vez sea construido y puesto en marcha.

1.4.2 Social

Brindar alternativas de mejoramiento a la atención de la salud de pacientes con enfermedades cardiacas a través de métodos que cuentan con el respaldo de una comunidad internacional, como lo es OpenECG. Esto abriría las puertas para desarrollar

dispositivos y formas de controlar y monitorear a estos pacientes, además de responder oportunamente a cualquier emergencia que llegase a suceder con alguno de ellos teniendo a disposición un servicio en la web que provee la información importante para un paciente cardiaco como lo es el ECG y su ubicación actual.

1.5 ALCANCE

Se trabajará con base al estándar SCP-ECG y se analizará su funcionamiento, planteando su implementación dentro del proyecto a desarrollar. Luego, con el uso del proceso ágil de desarrollo Extreme Programming se construirá software para la ejecución desde el dispositivo móvil, empaquetamiento, transmisión de datos, visualización en el portal web orientado a geo-posicionamiento y registro de los datos recogidos.

Junto a esto se construirá una base de datos (PostGreSQL) diseñada específicamente para las necesidades del software que almacenará los datos enviados por el dispositivo ECG empaquetados, de acuerdo con el estándar SCP-ECG. Asimismo, utilizando la base de datos geo-refenciada (PostGIS) del grupo RadioGis se ubicará la posición de la señal, así se tendrá un sistema ordenado y limpio que permite un manejo eficiente de la información tomada. Estos datos serán luego mostrados en un sitio web, construido específicamente para el desarrollo de este proyecto, el cual cumplirá la función de monitoreo del ECG recibido y mostrará la posición de la señal recibida por GPS usando la API de Google Maps en conjunto con la base de datos y el servidor del grupo RadioGis, de esta forma se desplegarán los datos importantes para el control del ECG en una sola interfaz lo que simplificaría esta tarea.

El enfoque de este proyecto es solo y únicamente hacia el desarrollo del SOFTWARE que compone el servicio como tal, siendo este el desarrollo del software en el dispositivo cliente para empaquetamiento y envío de los datos a través de internet móvil, diseño y construcción de la base de datos para el almacenamiento de toda la información requerida por el servicio (PostgreSQL y PostGIS este último para bases de datos espaciales) y por ultimo desarrollo de un portal web para el monitoreo y visualización de los datos obtenidos y almacenados en el servidor. No es objetivo de este proyecto el desarrollo de dispositivos electrónicos para la captura de datos, sea del ECG, o GPS para la posición, se deja la puerta abierta a futuros desarrollos del grupo RadioGIS y la comunidad universitaria en general para la construcción e implementación de dispositivos que se ajusten a las características del servicio y así complementarlo haciendo de este un desarrollo que promueve la investigación en otros campos de la ingeniería.

2. MARCO TEORICO

Las NGN (*Next Generation Network*) presentan una alternativa de desarrollo para las comunidades de todo el mundo, haciendo que sus servicios, información, aplicaciones y demás recursos de carácter digital estén disponibles para cualquier persona que tenga acceso a la red a cualquier hora del día todos los días de la semana, brindando una cobertura y atención permanente, para esto el gobierno Colombiano por medio del ministerio de TICs presenta el Plan "Vive Digital" el cual tiene como misión promover el acceso, uso efectivo y apropiación masiva de las TIC a través de políticas y programas, para mejorar la calidad de vida de cada Colombiano y el incremento sostenible del desarrollo del país. El MinTIC tiene una visión de Colombia en el 2014 como un referente internacional dado el impacto de penetración y utilización efectivas de las TIC, a nivel sectorial como institucional y gubernamental, se basa en la expansión de las redes actuales de comunicación promoviendo el desarrollo del sector privado multiplicando cuatro veces el número de conexiones a internet actuales, brindando garantías y promoviendo servicios e información convergente para el uso de los Colombianos, razón por la cual plantear un servicio de telemedicina apoyaría a la iniciativa planteada por el gobierno nacional, e impulsaría a la universidad como un foco de desarrollo para el país.

En apoyo a la iniciativa impulsada por este plan "Vive Digital" del MinTIC el grupo RadioGIS ha desarrollado una plataforma para el desarrollo de servicios basados en localización (LBS por sus siglas en ingles Location-based Service) con la cual el grupo pretende aportar a la población variedad de servicios en línea que se puedan usar desde cualquier lugar que tenga conexión a internet con solo acceder al portal web provisto por el grupo. Se podrán usar componentes geoposicionados y este proyecto a desarrollar hará parte de la primera gama de servicios orientados a telemedicina que serán puestos a disposición del grupo para su uso e implementación.

La electrocardiografía consiste en la obtención y registro gráfico de los potenciales eléctricos producidos por el corazón en los diversos momentos de su actividad. Se obtienen mediante instrumentos especiales (electrocardiógrafos) que amplifican los impulsos eléctricos que se obtienen en varios puntos de la superficie corporal; la información obtenida se muestra en un electrocardiograma (ECG), que representa gráficamente la actividad eléctrica del corazón en forma de cinta continua; para unificar la información de los ECG se crearon diversos estándares, debido a la actividad generada por la globalización, la cual obliga a las organizaciones a adoptarlos para competir en mercados de diferentes países que ofrecen productos para el uso en cualquier región del mundo y que certifican su alta calidad.

En esta investigación para el trabajo de tesis, se usará el SCP-ECG (*Standard Communications Protocol for computer assisted ElectroCardioGraphy*) desarrollado en un

principio entre 1989 y 1991 durante el proyecto Europeo AIM R&D, y que está definido en el estándar conjunto ANSI/AAMI EC71:2001 y en el estándar CEN EN 1060:2005[8]. Este estándar maneja huellas de ECG, anotaciones y metadatos que especifica el formato de intercambio y el procedimiento de mensajería para comunicación de ECG *cart-to-host* y para la recuperación de la información de los registros SCP-ECG del host al *ECG cart*. Actualmente el grupo OpenECG da soporte al SCP-ECG suministrando y apoyando implementaciones libres y aplicaciones conformes al estándar, ellos buscan consolidar los esfuerzos de interoperabilidad en la electrocardiografía computarizada a nivel europeo e internacional, fomentando el uso de estándares.

Actualmente el consorcio europeo está compuesto por personal de:

- **ICS-FORTH (Grecia)**
- **BIOSIGNA (Alemania)**
- **Danish Centre for Health Telematics (Dinamarca)**
- **IFC-CNR (Italia)**
- **RGB Medical Devices (España)**

El grupo OpenECG tiene un portal web el cual sirve de punto de encuentro para todos los miembros de la comunidad desde el cual se coordinan y se manejan las evaluaciones de los resultados concurrentes, desde este se da soporte a todos los interesados en la aplicación del estándar computarizado SCP-ECG. También promueven la construcción, evaluación, promoción de herramientas *open-source* para ver archivos SCP-ECG y traducción de soluciones propietarias a estándares computarizados de ECG, adicionalmente una base de datos de ECG digitales conformes a los estándares de ECG computarizados están disponibles para los miembros. La comunidad realiza actividades de diseminación para estar en contacto con actividades de estandarización, organizaciones profesionales, redes de salud regionales, entre otras, buscando participar en conferencias médicas para promover ampliamente el uso de las herramientas estandarizadas.

Para ofrecer un servicio más integral suministrando soluciones de alta calidad en la internet, se utiliza el lenguaje de programación Java, lenguaje que tiene las ventajas de ser libre, independiente de la plataforma y que permite la reutilización de código; además ofrece una plataforma para el desarrollo de aplicaciones empresariales llamada JEE (Java Enterprise Edition) que es un estándar del JCP[17] (*Java Community Process*), definida en la JSR 244[18] (*Java Specification Requests*). Con esta herramienta se pueden desarrollar aplicaciones con componentes modulares en un servidor de aplicaciones que provee servicios con una arquitectura multicapa, utilizando un *web service*.

Un *web service* es una tecnología que permite un método de comunicación interoperable máquina a máquina en una red, tiene una interfaz descrita en un formato procesable para la máquina (WDSL = *Web Services Description Language*). También un web service es típicamente una API (Application Programming Interface) a la que se accede mediante HTTP (*Hypertext Transfer Protocol*) y se ejecuta en un sistema remoto. Otros sistemas que interactúan con un web service lo hacen utilizando mensajes SOAP (*Simple Object Access Protocol*), con serialización en XML en conjunción con otros estándares web.

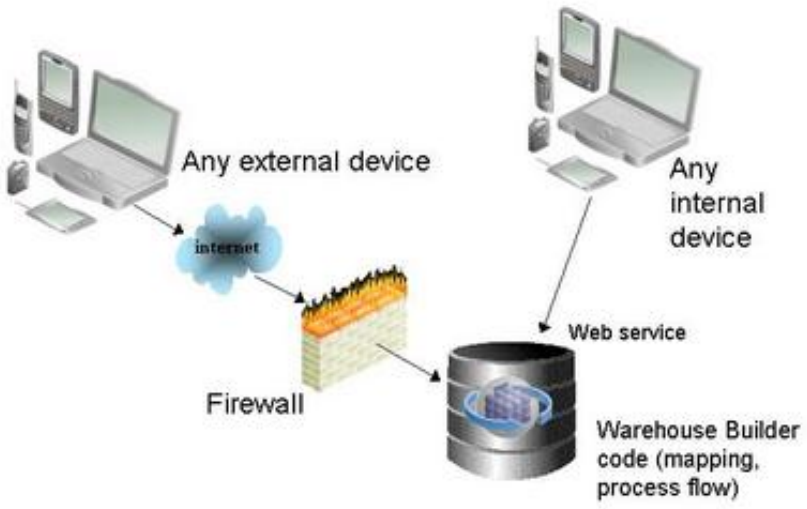

Figura 1: Arquitectura de un servicio web Fuente: http://culturacion.com

Para almacenar la información necesaria se usará el DBMS (Database Management System) PostgreSQL, que es un sistema de gestión de base de datos relacional orientado a objetos y libre, con el módulo PostGIS que añade el soporte de objetos geográficos, de esta manera la base de datos puede manejar información espacial ideal para sistemas de información geográficos y georeferenciados.

Las nuevas tecnologías en comunicación permiten tener contacto permanente con un dispositivo siempre que haya una red de servicio celular disponible, usando el servicio de datos GPRS orientado a móviles el cual provee acceso a internet las 24 horas del día. Este servicio está presente en sistemas de comunicación celular 2G y 3G, se encuentra disponible en más de 200 países en el mundo. GPRS fue originalmente estandarizado por el ETSI (European Telecommunications Standards Institute) en respuesta al anterior CDPD (Cellular Digital Packet Data) y el paquete i-mode de cambio a las tecnologías celulares, es ahora mantenido por el 3rd Generation Partnership Project (3GPP). Este es un servicio de "mejor-esfuerzo" (*best-effort*) y provee tasas de transferencia entre 56-114 kbit/segundo por lo cual brinda una velocidad moderada de transferencia de datos mediante el uso de canales TDMA (Time Division Multiple Access) sin usar en, por ejemplo, el sistema GSM. El GPRS está integrado dentro del GSM Release 97 y los más nuevos que este.

El uso de este servicio se mide por el volumen de datos que se envía o se recibe, mas no por el tiempo que se use, y su funcionalidad es constante si existe una red de servicio celular en la zona, para realizar la transferencia de los datos, lo cual permite un monitoreo de la señal durante las 24 horas del día, haciendo de este una cobertura total de las lecturas tomadas por el dispositivo.

En vista de que se realizará un sistema de monitoreo georeferenciado, existen actualmente tecnologías que juntas pueden ser usadas para mantener la ubicación del dispositivo durante el tiempo que esté activo, usando también los beneficios de la conexión permanente con GPRS; éstas son el GPS y la herramienta de Google Maps. Global Posisitioning System (GPS) es un sistema de navegación global por satélite basado en espacio que provee una localización confiable e información de tiempo bajo cualquier clima, en todo momento y en cualquier lugar en o cerca de la Tierra donde y cuando haya una línea de visión sin obstáculos a cuatro o más satélites GPS. Es mantenido por el gobierno de los Estados Unidos y es de libre acceso para cualquier persona que posea un receptor GPS. Google Maps es una aplicación y tecnología de mapeo vía servicio web proporcionado por Google, gratis (para uso no comercial), que alimenta a varios servicios basados en mapas, incluyendo el sitio web de Google Maps, Google Ride Finder, Google Transit y mapas embebidos en sitios web de terceros de la API de Google Maps. Este ofrece mapas de calles, un planeador de rutas para movilizarse a pie, en automóvil o en transporte público. También provee una vista de satélite de alta definición para la mayoría de las áreas urbanas de los Estados Unidos. Con el lanzamiento de la API de google maps en 2005, los desarrolladores pueden integrar Google Maps a sus sitios web. Éste es un servicio gratis y no contiene publicidad, de forma que mediante el uso de la API de Google maps se puede embeber el sitio de google maps a un sistema de rastreo por GPS y así junto con la señal tomada del GPRS se obtiene la medida del paciente, la ubicación de la señal y su posición actual en el sitio donde se encuentra, lo que permitiría la georeferenciación del usuario en cualquier momento del día, facilitando así su atención en el momento de una emergencia y cuando sea necesario.

3. DESCRIPCIÓN DEL PROBLEMA

Según el Informe sobre la Situación de la Salud en Colombia 2007 del Ministerio de la Protección Social, la enfermedad cardiovascular es la principal causa de muerte en Colombia tanto en hombres como en mujeres mayores de 45 años o más, e inclusive, supera las muertes violentas o por canceres combinados. Muchas de estas suceden incluso antes de presentarse el personal médico para atender al paciente, situación que empeora hacia las áreas rurales y poblaciones lejanas.

Artículos publicados en años recientes muestran un gran avance en cuanto al desarrollo de tecnología portable para la lectura del electrocardiograma, una de estas desarrollada en la University "Mediterranea" of Reggio Calabria, Reggio Calabria, Italia. Donde emplean una arquitectura de web service integrado con una PDA (Personal Digital Assistant) y un circuito que toma la señal cardiaca del paciente; en este, la PDA procesa y analiza la señal ECG capturada por el dispositivo en busca de un diagnóstico acerca de la condición cardiaca del paciente, toda esta información obtenida por los dispositivos se muestra en la PDA, la cual fue programada con diferentes interfaces que permiten al usuario ver el diagnóstico, la señal recogida, un historial personal de las mediciones que se han tomado, imprimir los resultados, enviar un e-mail al especialista o contactar los servicios de emergencia, en caso de presentarse.

Un caso similar a este desarrollo se presenta en Corea, en el artículo investigativo Information and Communications University de Yuseong-gu, se expone el empleo de un grid computing para la realización de identificación avanzada de enfermedades cardiacas. Este propone un sistema de grid para el tratamiento de la señal que accede al historial del paciente con menos diagnósticos erróneos posibles y permite la construcción de gráficas detalladas acerca de la condición del corazón, análisis de ECG y también la generación de imágenes virtuales del corazón en 3D que permite un máximo detalle de la condición cardiaca de la persona que está en el tratamiento tele-médico. Sin embargo, no se aprovechan los avances de las comunicaciones para prevenir riesgos a los pacientes que se encuentran fuera de un hospital, que por cualquier razón, presentan arritmias que requieren atención inmediata.

RadioGis, en apoyo al programa Vive Digital, busca aprovechar la plataforma desarrollada de LBS (Location-Based Service) para ofrecer a los pacientes un servicio que pueda no solo monitorear arritmias sino enviar alertas a una entidad encargada de su vigilancia para que tome medidas. Estas medidas pueden ir desde una simple citación al paciente hasta el envío de una ambulancia al lugar exacto donde se encuentra. Además, los trabajos encontrados carecen de un lenguaje común para manejar la señal, aspecto que hace que las investigaciones estén dependientes al formato desarrollado en el proyecto y limitaría su rango de acción a dispositivos que manejen el mismo tipo de señal ECG que interpreta su software, es decir, no habrá compatibilidad con otras señales. Es por esto que se han

desarrollado normas estándares internacionales de ECG que permiten una gama más amplia de dispositivos para la integración de las diferentes opciones de investigación de los dispositivos de la telemedicina, y así facilita la obtención de elementos para capturar el ECG que será procesado en el sistema e integre en un solo lenguaje la gran variedad de investigaciones y estudios. El seguimiento de esta norma o estandarización de los sistemas de señales de software da un paso más allá de los trabajos realizados al momento, además de resolver y mejorar las tecnologías disponibles y de fácil acceso en la actualidad.

4. ESPECIFICACIÓN DE REQUISITOS PARA EL SERVICIO

4.1 Actores del Sistema

Los usuarios del servicio planteado son los siguientes:

- Administrador: Es el usuario que tiene todos los privilegios y permisos en el sistema, puede agregar, modificar y eliminar usuarios del sistema, asignar permisos, acceder a la base de datos para su administración y acceder a todas las facultades que ofrece el servicio de monitoreo.

- Paciente: Usuario al cual se le brinda el servicio de monitoreo, este, en lo posible, tendrá un dispositivo con el cliente Android instalado en él para la escritura del paquete SCP y él envió de información geo-posicionada al servidor del sistema, sus registros serán almacenados, podrá acceder a sus registros previos de la señal cardiaca.

- Familiar/Acudiente: Usuario asociado con un determinado paciente, este podrá acceder a los registros de ese paciente, así como a un panel de monitoreo donde podrá observar la condición del paciente y también su ubicación, se registra al sistema por solicitud del paciente y solo tendrá privilegios de monitoreo sobre el paciente con el cual está asociado.

- Médico Monitor: es un medico auxiliar asignado para la tarea de monitoreo de pacientes que tienen el servicio. Capacitado para interpretar las señales ECG, puede determinar cuando existen problemas con el paciente para hacer uso del sistema de emergencia y enviar una ambulancia al sitio indicado por el dispositivo para la atención inmediata del paciente, puede acceder a registros previos del paciente para hacer un seguimiento de su condición.

- Cliente Android: Es el software instalado en el dispositivo que se le asigna al paciente para su monitoreo, contiene el escritor de archivos

SCP-ECG, acceso al servicio web REST para el envío de datos, la capacidad de enviar coordenadas de latitud y longitud, los datos del paciente al cual fue asignado, una interfaz gráfica para su manejo y acceso a comunicación con el servidor por medio de internet móvil.

Estas distinciones se realizaron apuntando a cómo debe funcionar el servicio aplicado ya en el campo de la telemedicina, se plantea un modelo de negocios y una arquitectura básica en respuesta a las necesidades expresadas por el grupo RadioGIS, sin embargo, no es un sistema rígido, y tanto los diagramas de casos de uso como los diagramas de clases, el software y el servicio completo son flexibles al cambio. Con la documentación realizada fácilmente se puede adaptar a cualquier necesidad que requiera el cliente o la institución que quiera adoptar este servicio.

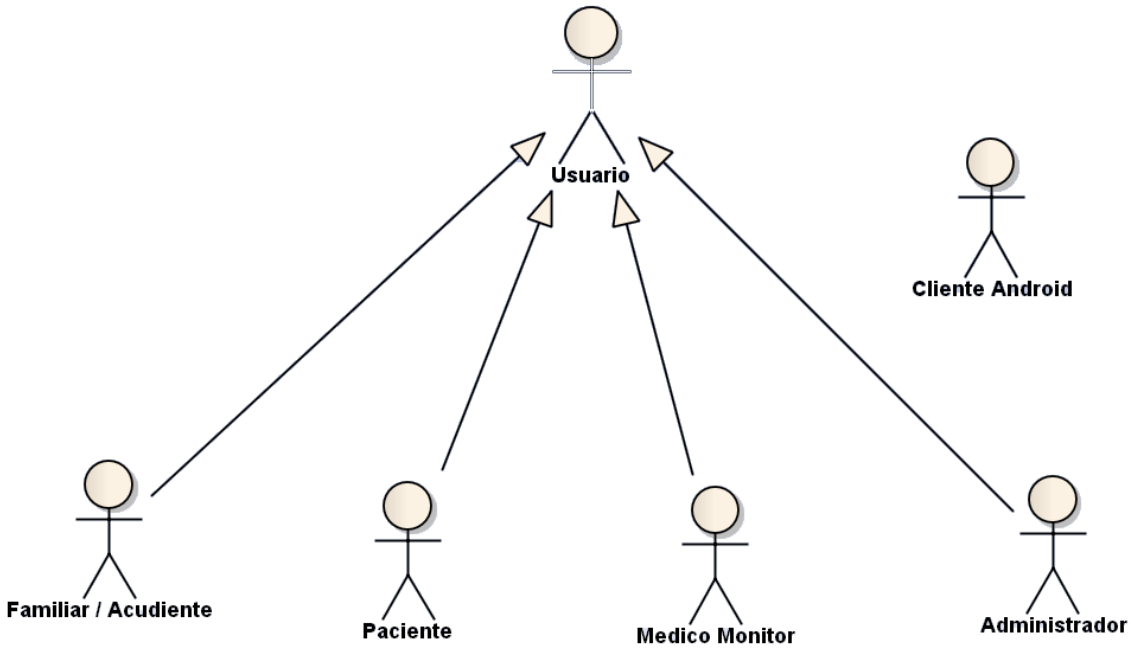

Figura 2. – Actores del Sistema

4.2 MODELO DE CASOS DE USO

Con el servicio se buscó abarcar cierta funcionalidad básica y esencial que permita a un médico estar en permanente monitoreo del estado cardiaco del paciente que hace uso del servicio. Además también se buscó construir una plataforma que sea tanto extensible como segura y robusta, que permita futuros desarrollos para mejorar la calidad del servicio aquí planteado.

También se pretende darle un valor agregado a la aplicación Web con el geo posicionamiento del paciente en una instancia de Google Maps.

Por estas razones se plantearon una serie de casos de usos con los cuales el servició fue construido para garantizar como mínimo la calidad del prototipo y cumplir así los objetivos planteados al iniciar el proyecto

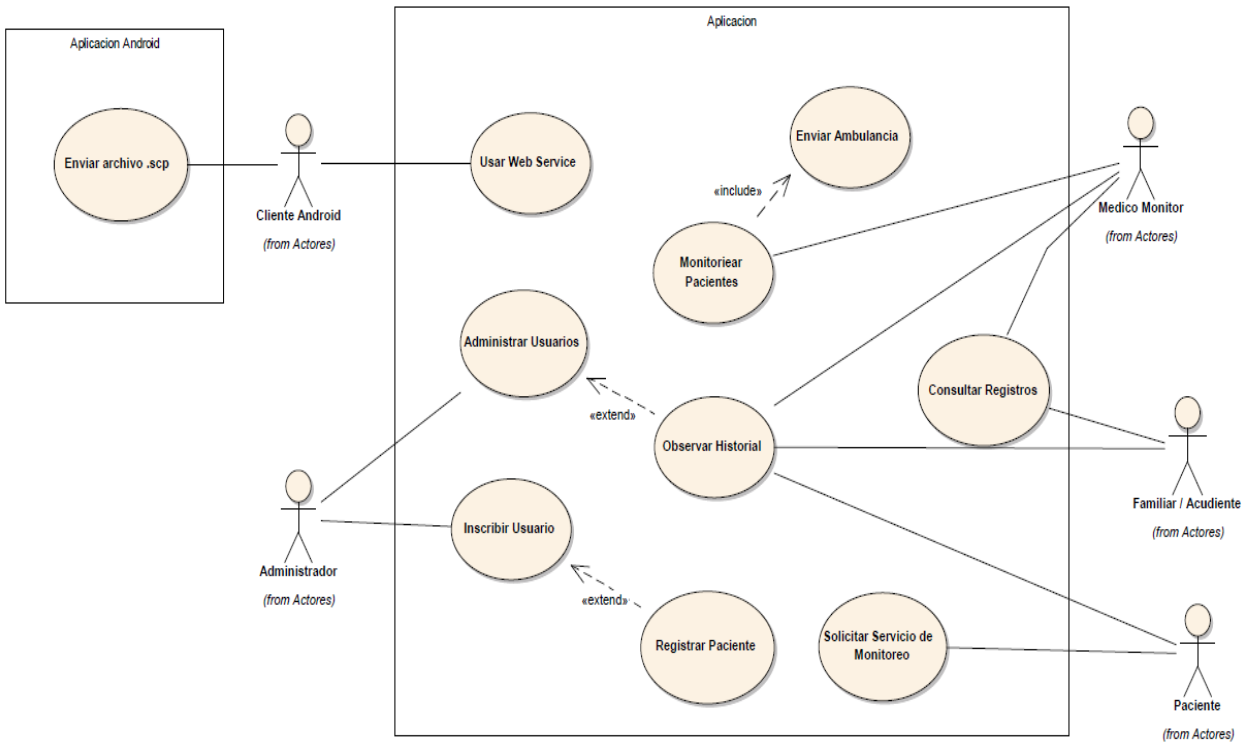

Figura 3. – Modelo de Casos de Uso General

Se describen cada uno de la siguiente manera:
Efectuados por el Actor Cliente Android:

- Enviar Archivo .SCP: El cliente software instalado en el dispositivo que va a usar el paciente para su monitoreo de la señal cardiaca es el encargado de realizar esto, empaqueta y envía los datos obtenidos frecuentemente del paciente en un archivo de formato .SCP, siguiendo el estándar SCP-ECG en su versión 2.2. Para la transmisión el dispositivo debe tener internet móvil con acceso 24/7 al sistema construido para el servicio. El archivo con formato .SCP es almacenado en la base de datos del sistema, relacionado con el ID del paciente del cual proviene la lectura.
- Usar Web Service: El software cliente accede al servidor mediante una solicitud al servicio web REST construido dentro de la aplicación web. Mediante este el servidor de aplicaciones almacena los datos correspondientes en la base de datos

que requiere el servicio para mantener el monitoreo constante del paciente en el entorno de aplicación.

Efectuados por el actor Administrador:

- Administrar Usuarios: Caso usado solo por el administrador de la plataforma, el podrá crear, modificar, borrar, dar privilegios, asignar roles y permisos a todos los que van a acceder al sistema. también puede realizar tareas de monitoreo de la actividad de la plataforma, también administra a nivel de base de datos.
- Observar Historial: Caso por el cual se accede a los registros SCP-ECG de determinado paciente, el administrador del sistema puede acceder a todos los registros guardados en la base de datos, mientras que el Medico Monitor solo tendrá acceso a ver los datos mas no modificarlos, el paciente y su familiar tendrán acceso a sus propios registros. podrán observar la trayectoria de los archivos con fecha y hora de la medición, todo para fines de control sobre el desarrollo de la condición de paciente.
- Inscribir Usuario: el administrador será el encargado de registrar todos los usuarios que van a hacer parte del sistema, incluyendo los pacientes, médicos y familiar o acudiente del paciente. Se registraran los datos pertinentes para cada tipo de usuario y en base al tipo se asignaran privilegios para cada uno de ellos. Este caso extiende el Registrar Paciente, caso aparte ya que es el usuario al que va destinado el servicio.
- Registrar Paciente: Caso destinado al registro del actor al cual va dedicado al realización del servicio, el tendrá asignado un dispositivo móvil con el cliente software instalado para él envió del paquete SCP, el cual contiene la lectura de la señal cardiaca, la posición actual basándonos en los parámetros de latitud y longitud usados por Google Maps. También se registrara él en sistema sus datos personales, información médica relevante, y tendrá asociado un lugar en el almacén SCP que contendrá el historial de sus señales cardiacas, y un familiar o acudiente que puede acceder al sistema para revisar tanto este historial como su estado actual.

Efectuados por el actor Paciente

- Solicitar Servicio de Monitoreo: El paciente por medio de su entidad de prestadora de servicios de salud solicita el servicio para el monitoreo permanente.
- Observar Historial: Caso por el cual se accede a los registros SCP-ECG de determinado paciente, el administrador del sistema puede acceder a todos los registros guardados en la base de datos, mientras que el Medico Monitor solo tendrá acceso a ver los datos mas no modificarlos, el paciente y su familiar tendrán acceso a sus propios registros. podrán observar la trayectoria de los archivos con fecha y hora de la medición, todo para fines de control sobre el desarrollo de la condición de paciente.

Efectuados por el actor Médico Monitor

- Consultar Registros: El sistema permitirá hacer una consulta de los registros almacenados en el almacén SCP, el médico podrá consultar los registros de los pacientes los cuales está asignado a monitorear mientras que el familiar solo podrá consultar los del paciente al cual está relacionado. esta consulta podrá hacerse por día, fecha u hora, parámetros definidos en la base de datos para el Almacén SCP.
- Observar Historial: Caso por el cual se accede a los registros SCP-ECG de determinado paciente, el administrador del sistema puede acceder a todos los registros guardados en la base de datos, mientras que el Medico Monitor solo tendrá acceso a ver los datos mas no modificarlos, el paciente y su familiar tendrán acceso a sus propios registros. podrán observar la trayectoria de los archivos con fecha y hora de la medición, todo para fines de control sobre el desarrollo de la condición de paciente.
- Monitorear Pacientes: El médico encargado de monitorear a determinados pacientes tendrá acceso en tiempo real a la lectura de la señal cardiaca enviada por el dispositivo cliente al servidor. en el panel de monitoreo se mostrara la señal recibida de paciente, los datos personales del mismo, tendrá un vínculo a la historia clínica del paciente si está disponible para él, y tendrá un mapa con la ubicación actual del paciente, este desarrollado usando la API de google maps para Java Server Faces y los datos enviados por el dispositivo que tendrá el paciente (coordenadas latitud y longitud).
- Enviar Ambulancia: Este caso se emplea en situaciones de emergencia, cuando el médico que monitorea al paciente recibe una señal cardiaca que indica un problema grave en el paciente, por lo cual es necesaria la atención inmediata de una ambulancia en el lugar donde se encuentra el paciente. El médico monitor califica la gravedad de la situación, si es necesario envía un servicio de ambulancia a la ubicación indicada por el sistema, o se comunica con el paciente usando los datos almacenados en el sistema en caso de no ser una situación muy grave.

Casos del actor Familiar/Acudiente:

Realiza los mismos casos de uso del actor paciente.

5. HERRAMIENTAS UTILIZADAS

Para el desarrollo del servicio se usaron múltiples herramientas y lenguajes de programación, que fueron de gran ayuda para cumplir con las metas y lograr un software de calidad, robusto y extensible. También se usaron herramientas Software para la documentación de proyecto, dibujo de diagramas y diseños lo que permite la extensibilidad del mismo, puesto que cualquier desarrollador podría tomar el software donde se dejó y agregarle nuevas funcionalidades y mejorar lo ya construido, está siempre ha sido la intención al desarrollar el servicio, ya que se apunta a crear una red de servicios de nueva generación basados en localización.

5.1 Lenguajes de programación:

En primer lugar tenemos los lenguajes de programación utilizados para el desarrollo del servicio:

Java Standard Edition 6

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí toma mucha de su sintaxis de *C* y *C++*, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del *bytecode* por un procesador Java también es posible. La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del *Java Community Process*, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre. Entre diciembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia *GNU GPL*, de acuerdo con las especificaciones del *Java Community Process*, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es).

Java SE 6 es la versión actual principal de la plataforma *Java SE*, lanzada a disposición general en diciembre del 2006. Sun se esfuerza por fomentar el más alto nivel de transparencia y colaboración en la plataforma con la comunidad Java a través de *Project JDK 6*, resultando en mejoras y características claves. La tecnología Java es tanto un lenguaje de programación como una plataforma:

Lenguaje de programación:

Como lenguaje es uno de alto nivel que puede ser caracterizado por las siguientes palabras:

- Simple.
- Orientado a Objetos.
- Distribuido.
- Multihilos.
- Dinámico.
- Arquitectura neutral
- Portable
- Alto Desempeño
- Robusto
- Seguro

Sun reemplazo el nombre J2SE con *Java SE* y quito el ".0" del número de la versión. Esta versión tuvo cambios grandes, que mejoraron dramáticamente el lenguaje y la plataforma e hicieron de este un popular lenguaje de desarrollo, entre estos cambios están:

- Soporte para versiones viejas de *Windows9x* se detuvo, se cree fue debido a los grandes cambios hechos en la actualización 10.
- Mejoras dramáticas en el desempeño del núcleo de la plataforma.
- Soporte a *Web Services* mejorado a través de *JAX-WS*.
- Soporte a *JDBC* 4.0.
- Mejora del *JAXB* a la versión 2.0
- Varias mejoras en la *GUI*, como la integración de *SwingWorker* en la *API*, ordenamiento de tablas y filtrado y un verdadero doble-búfer *Swing* (eliminando el efecto de área gris).
- Mejoras de *JVM* incluyen: optimizaciones en sincronización y desempeño del compilador, nuevos algoritmos y mejoras a algoritmos existentes de manejo de memoria.

En este lenguaje, todo el código fuente es escrito primero en archivos de texto plano terminados con la extensión *.java*. Estos archivos fuentes son luego compilados en archivos *.class* por el compilador *javac.* Un archivo *.class* no contiene código que sea nativo al procesador, en lugar de esto contiene *bytecodes* – El lenguaje de máquina de la Máquina Virtual Java (Java VM). La herramienta de lanzamiento *java* corre la aplicación dentro de una instancia de la Máquina Virtual Java. Y debido a que esta máquina virtual está disponible en diferentes sistemas operativos, el mismo archivo *.class* es capaz de correr en Microsoft Windows, Solaris OS, Linux o Mac OS.

La plataforma Java

Una plataforma es el ambiente *Hardware* y *Software* en el cual corre el programa. La mayoría de las plataformas pueden ser descritas como una combinación del sistema operativo y *Hardware* subyacente. La plataforma *Java* difiere de muchas otras plataformas en que es una de solo *software* que corre sobre otras plataformas basadas en *hardware*.
La plataforma *Java* tiene dos componentes:

- La Máquina Virtual Java.
- La Interfaz de Programación de Aplicaciones Java (API).

La máquina virtual es la base de la plataforma y ha sido portada hacia varias plataformas *hardware*. La *API* es una larga colección de componentes *software* listos-hechos que proveen muchas capacidades útiles. Están agrupados en librerías de clases relacionadas e interfaces, estas librerías se conocen como paquetes. Como una plataforma independiente de ambiente, *Java* puede ser un poco más lento que el código nativo. Sin embargo avances en tecnologías del compilador y la máquina virtual están brindando desempeño cercano a aquel de código nativo, sin amenazar la portabilidad de *java*.

JavaServer Faces (JSF 2.0)

Desarrollado a través del *Java Community Process* bajo el nombre *JSR – 314*, la tecnología JavaServer Faces establece el estándar para la construcción de interfaces de usuario del lado del servidor. Esta tecnología simplifica la construcción de interfaces para aplicaciones de JavaServer, desarrolladores de distintos niveles de habilidad pueden construir rápidamente aplicaciones web mediante: componentes de interfaz de usuario reutilizables en una página; conectar estos componentes a una fuente de datos de la aplicación y cableando eventos generados por el cliente a manejadores de eventos del lado del servidor. El *JSR* especificó ocho metas de diseño para JSF que son: crear un marco de trabajo estándar de componentes UI (*User Interface*) que pueda ser apalancado por herramientas de desarrollo que hagan más fácil a los usuarios tanto crear interfaces de usuario de alta calidad, como administrar las conexiones de la interfaz al comportamiento de la aplicación; Definir un conjunto de simples, livianas clases Java base para los componentes UI, componentes de estado, y eventos de entrada. Estas clases direccionaran temas del ciclo de vida de la UI, administrando notablemente el estado de persistencia de un componente por el tiempo de vida de su página; Proveer un set de componentes UI comunes, incluyendo los elementos de entrada estándar de HTML; Proveer un modelo de *JavaBeans* para el envío de eventos por controles UI del lado del cliente a comportamiento de aplicación por el lado del servidor; Definir API's para la validación de entradas, incluyendo soporte para la validación por el lado del cliente.

SQL – Structured Query Language

Este es el lenguaje de programación diseñado para manejar sistemas de base de datos relacionales, es el lenguaje de bases de datos más usado ampliamente. Aunque muy a menudo es descrito como un lenguaje declarativo, también contiene elementos de procedimiento. Es un estándar para la *American National Standards Institude (ANSI)* y para la Organización Internacional de Estándares *(ISO por sus siglas en ingles)*. Particularmente para este proyecto se usó el motor de base de datos PostgreSQL que será presentado más adelante.

5.2 Herramientas Software

Netbeans IDE 6.9.1

Esta herramienta fue empleada como entorno de desarrollo para todo el servicio realizado en este proyecto, ya que soporta todo tipo de aplicaciones *Java* y es reconocido por algunos como el primer *IDE* original. Provee soporte para varios lenguajes (*PHP, JavaFX, C/C++, JavaScript, etc*) y marcos de trabajo. NetBeans es un proyecto *de Código Abierto* dedicado a proveer productos sólidos de desarrollo software (NetBeans IDE y la plataforma NetBeans) que responden a las necesidades de los desarrolladores, usuarios y

empresas. Este IDE está escrito en Java y puede ejecutarse en cualquier maquina donde esté instalado una Máquina Virtual Java compatible. Este soporta todo tipo de aplicaciones *Java (Java SE incluyendo JavaFX, Java ME, Web, EJB, aplicaciones móviles)* recién instalado. Entre otras características están sistema de proyectos basados en *Ant*, Soporte a *Maven*, refactorizaciones, control de versiones (soportando *CVS, Subversion, Mercurial* y *Clearcase*). Todas las funciones del IDE se proporcionan por módulos, cada módulo proporciona una función bien definida, como un apoyo para el lenguaje *Java*, edición, o soporte para el sistema de versiones *CVS* y *SVN*. NetBeans contiene todos los módulos necesarios para el desarrollo *Java* en una sola descarga, permitiendo al usuario empezar a trabajar inmediatamente, estos también permiten la extensión de NetBeans, como soporte para otros lenguajes de programación.

PostgreSQL 9.0

PostgreSQL es un poderoso, de código abierto, objeto-relacional sistema de base de datos. Tiene más de 15 años de desarrollo activo y una arquitectura probada que ha ganado una fuerte reputación por confiabilidad, integridad de datos y exactitud. Es ejecutable en toda la mayoría de sistemas operativos incluyendo Linux, UNIX, Mac OS X, Solaris, Tru64 y Windows. Es completamente ACID compatible, tiene soporte completo para llaves foráneas, *joins*, vistas, disparadores, y procedimientos almacenados (en múltiples idiomas). Este incluye la mayoría de tipos datos SQL 2008 incluyendo *INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL,* y *TIMESTAMP* así como también soporta el almacenamiento de grandes objetos binarios, incluyendo imágenes, sonidos, o video. Tiene interfaces nativas de programación para C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre otras, y una excelente documentación.

Para la administración de la base de datos se empleó pgAdmin III, software de administración incluido en el paquete de PostgreSQL 9.0. Este ofrece un ambiente gráfico, amigable y fácil de manejar para el usuario que le permite un manejo completo de la base de datos, crear nuevos espacios, modificar tablas, insertar, borrar, importar y exportar bases de datos. Está escrito en C++, usando el marco de trabajo wxWidgets, permitiéndole correr en la mayoría de los sistemas operativos.

Android SDK

Android es un sistema operativo basado en GNU/Linux diseñado originalmente para dispositivos móviles, como teléfonos inteligentes, pero que posteriormente se expandió su desarrollo para soportar otros dispositivos tales como tablets, reproductores MP3, netbooks, PCs, televisores, lectores de e-books e incluso, se han llegado a ver en microondas y lavadoras. Fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Es el principal producto de la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de

servicio. Las unidades vendidas de teléfonos inteligentes con Android se ubican en el primer puesto en los Estados Unidos, en el segundo y tercer trimestres de 2010, con una cuota de mercado de 43,6% en el tercer trimestre. El SDK (*Software Development Kit*) de Android provee las herramientas y APIs necesarias para empezar a desarrollar aplicaciones en la plataforma Android, usando el lenguaje de programación Java, por esta razón es perfectamente compatible con cualquier sistema operativo. Incluye características como un marco de aplicaciones, máquina virtual *Dalvik*, navegador integrado, graficas optimizadas, SQLite para almacenamiento estructurado de datos, soporte multimedia, telefonía GSM, Bluetooth, EDGE, 3G y WiFi, Camara, GPS, brújula y acelerómetro y por ultimo un ambiente rico para el desarrollo, incluyendo un emulador y herramientas para depuración.

Eclipse IDE

Es un entorno de desarrollo software multi-lenguaje que comprende un ambiente de desarrollo integrado (IDE) y un sistema extensible de complementos (plug-in). Está escrito en su mayor parte en Java y puede ser usado para desarrollar aplicaciones en java y, mediante el uso de varios complementos, en otros lenguajes de programación incluyendo Ada, C, C++, COBOL, Perl, PHP, Python, R, Ruby, Scala, Clojure, Groovy y Scheme. En su forma por defecto está dirigido a los desarrolladores Java. Los usuarios pueden extender sus las funcionalidades mediante la instalación de complementos escritos para el marco de trabajo de Eclipse, como diferentes cajas de herramientas para otros lenguajes de programación. Publicado bajo los términos de la licencia publica de *Eclipse Public License*, Eclipse es un software gratis y de código abierto, fue uno de los primeros IDE's en funcionar bajo *GNU Classpath* y se ejecuta sin problemas bajo *IcedTea.*
Este IDE fue seleccionado para este proyecto exclusivamente para el desarrollo del cliente en Android, debido que Google recomienda y desarrolla el complemento que integra el SDK de este sistema operativo al IDE Eclipse, razón por la cual es conveniente emplearlo.

GlassFish Server 3.0.1

GlassFish es un servidor de aplicaciones de software libre desarrollado por Sun Microsystems, compañía adquirida por Oracle Corporation, que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. La versión comercial es denominada Oracle GlassFish Enterprise Server (antes Sun GlassFish Enterprise Server). Es gratuito y de código libre, se distribuye bajo un licenciamiento dual a través de la licencia CDDL y la GNU GPL. GlassFish está basado en el código fuente donado por Sun y Oracle Corporation, éste último proporcionó el módulo de persistencia TopLink. GlassFish tiene como base al servidor Sun Java System Application Server de Oracle Corporation, un derivado de Apache Tomcat, y que usa un componente adicional llamado Grizzly que usa Java NIO para escalabilidad y velocidad.

6. DESCRIPCIÓN DEL ESTANDAR SCP-ECG

SCP-ECG *(Standard Communications Protocol for computer assisted ElectroCardioGraphy)*, desarrollado en un principio entre 1989 y 1991 durante el proyecto Europeo AIM R&D, y que está definido en el estándar conjunto ANSI/AAMI EC71:2001 y en el estándar CEN EN 1060:2005. Este estándar maneja huellas de ECG, anotaciones y metadatos que especifica el formato de intercambio y el procedimiento de mensajería para comunicación de ECG cart-to-host y para la recuperación de la información de los registros SCP-ECG del host al ECG cart. Actualmente el grupo OpenECG da soporte al SCP-ECG suministrando y apoyando implementaciones libres y aplicaciones conformes al estándar, ellos buscan consolidar los esfuerzos de interoperabilidad en la electrocardiografía computarizada a nivel europeo e internacional, fomentando el uso de estándares.

Actualmente el consorcio europeo está compuesto por personal de:

- ICS-FORTH (Grecia)
- BIOSIGNA (Alemania)
- Danish Centre for Health Telematics (Dinamarca)
- IFC-CNR (Italia)
- RGB Medical Devices (España)

El grupo OpenECG tiene un portal web el cual sirve de punto de encuentro para todos los miembros de la comunidad desde el cual se coordinan y se manejan las evaluaciones de los resultados concurrentes, desde este se da soporte a todos los interesados en la aplicación del estándar computarizado SCP-ECG. También promueven la construcción, evaluación, promoción de herramientas *open-source* para ver archivos SCP-ECG y traducción de soluciones propietarias a estándares computarizados de ECG, adicionalmente una base de datos de ECG digitales conformes a los estándares de ECG computarizados están disponibles para los miembros. La comunidad realiza actividades de diseminación para estar en contacto con actividades de estandarización, organizaciones profesionales, redes de salud regionales, entre otras, buscando participar en conferencias médicas para promover ampliamente el uso de las herramientas estandarizadas.

La versión empleada de este estándar para el desarrollo del servicio fue la última existente *(2.2)* tomando como guía el último documento del estándar se realizó la implementación de este formato en lenguaje *Java*. Este formato tiene 14 secciones que pueden componer un archivo SCP, cada una con funciones específicas a desarrollar dentro de las indicaciones planteadas por el estándar, pero dentro el caso del servicio se emplearon solo las secciones requeridas para formar el archivo SCP final, estas son: secciones de encabezado (CRC checksum, tamaño del registro), Sección 0 (punteros a las áreas de datos del registro), Sección 1 (Información de encabezados, Datos generales del paciente), Sección 3

(Definición de las derivaciones ECG), Sección 6 (Datos del ritmo cardiaco). Descripciones más específicas de los contenidos y formatos de las secciones se dan más adelante en este documento, en la parte que se describe la implementación de clases del estándar dentro del servicio para la construcción del lector y escritor de archivos SCP.

7. DISEÑO DE COMPONENTES QUE INTEGRAN EL SERVICIO

7.1 DIAGRAMA DE COMPONENTES

Entre la documentación generada, se encuentra la representación del sistema mediante diagramas de componentes, encargados de mostrar la conformación del software, componentes y dependencias entre estas secciones.

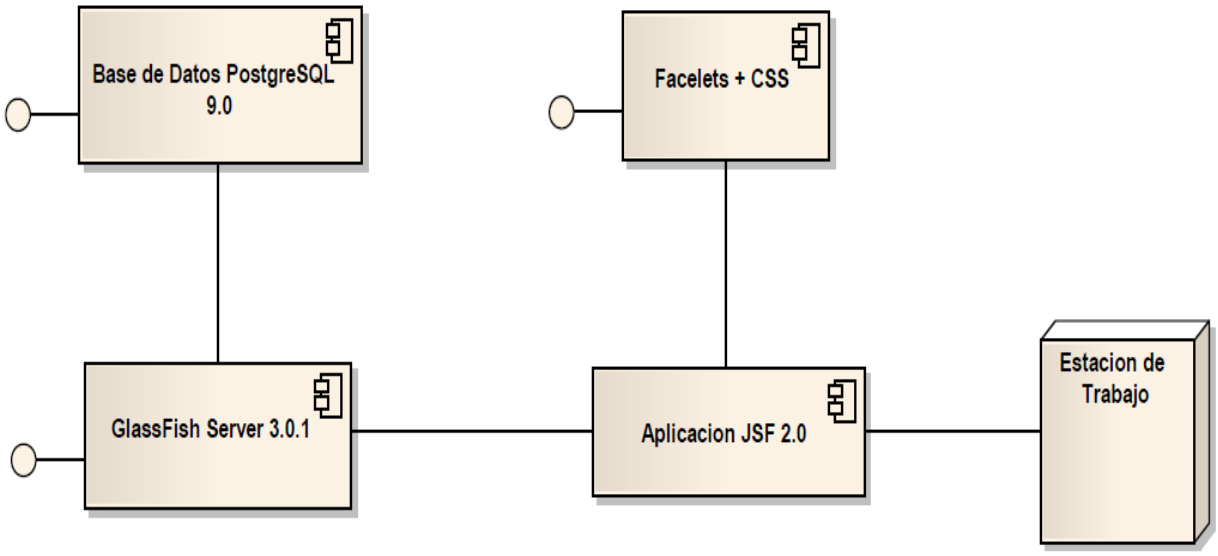

Figura 4 – Esquema de Flujo de Datos

Como se ha mencionado en el documento, los componentes se relacionan entre sí y con otros componentes del servicio, y se observa que antes de llegar a la estación de trabajo, el proceso de manejo y transformación de datos está representado de manera esquemática de tal manera que si hay que cambiar algún componente, se pueda realizar

dicha acción sin ningún daño al servicio completo, siendo necesario solamente configurar los nombres o direcciones adecuadas para mantener la integridad y realizar dicha tarea.

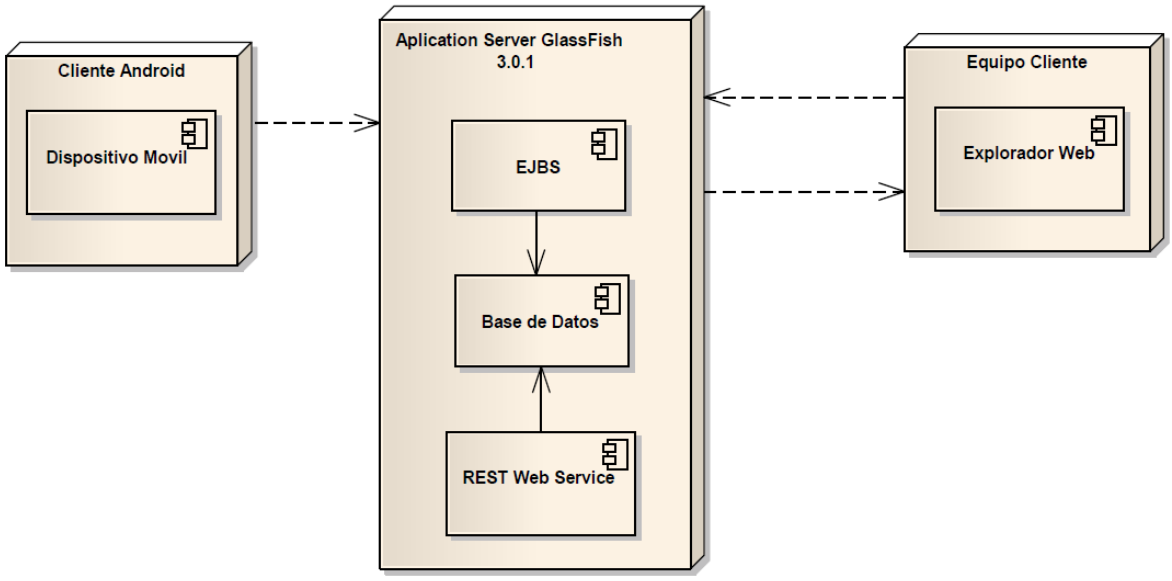

Figura 5. – Estructura del Servicio

El diagrama de componentes que modela la estructura del sistema muestra como está compuesta la plataforma, la aplicación web representa el dispositivo Estación de trabajo que se ejecuta sobre el explorador web e interactúa con el servidor de aplicaciones por medio de protocolos de transferencia de datos http, donde se localiza la lógica del negocio implementada en java sobre EJB y la persistencia de los datos en PostgreSQL. También se representa el software cliente instalado en el dispositivo móvil, encargado de enviar información al servidor de aplicaciones por medio de internet móvil para su procesamiento por parte del servicio web REST encargado de guardar los datos en la base de datos.

7.2 DIAGRAMA DE CLASES

En el estándar UML se trabaja el diagrama de clases con el fin de especificar mucho más a fondo las clases que deberan utilizarse en cada paquete, métodos y operaciones que intervienen en la lógica del servicio. Un informe más detallado de los objetos que se trabajaron en la implementación del estándar SCP-ECG usando *Java SE 6* para la construcción de un lector y escritor de archivos en este formato. Para estos componentes software se presenta una descripción más detallada en el anexo *Java DOCS* generado directamente de la documentación del código empleado.

En el caso de la implementación del estándar SCP-ECG para el Servicio Web planteado fue necesaria la construcción de varias estructuras de clases para manejar en su totalidad la complejidad del contenido que exigía el estándar para la construcción de un archivo siguiendo las reglas de este formato.

Por esta razón se realizaron 3 diagramas de clases que muestran la estructura del software construido solo para el manejo del formato exigido por el estándar, el primero muestra las secciones empleadas para la formación del archivo SCP-ECG, en algunas de ellas se debe escribir datos al nivel de bits, otras bytes y espacios con texto plano, así como algunos donde se combinan las anteriores.

Se observa más detalladamente en el diagrama:

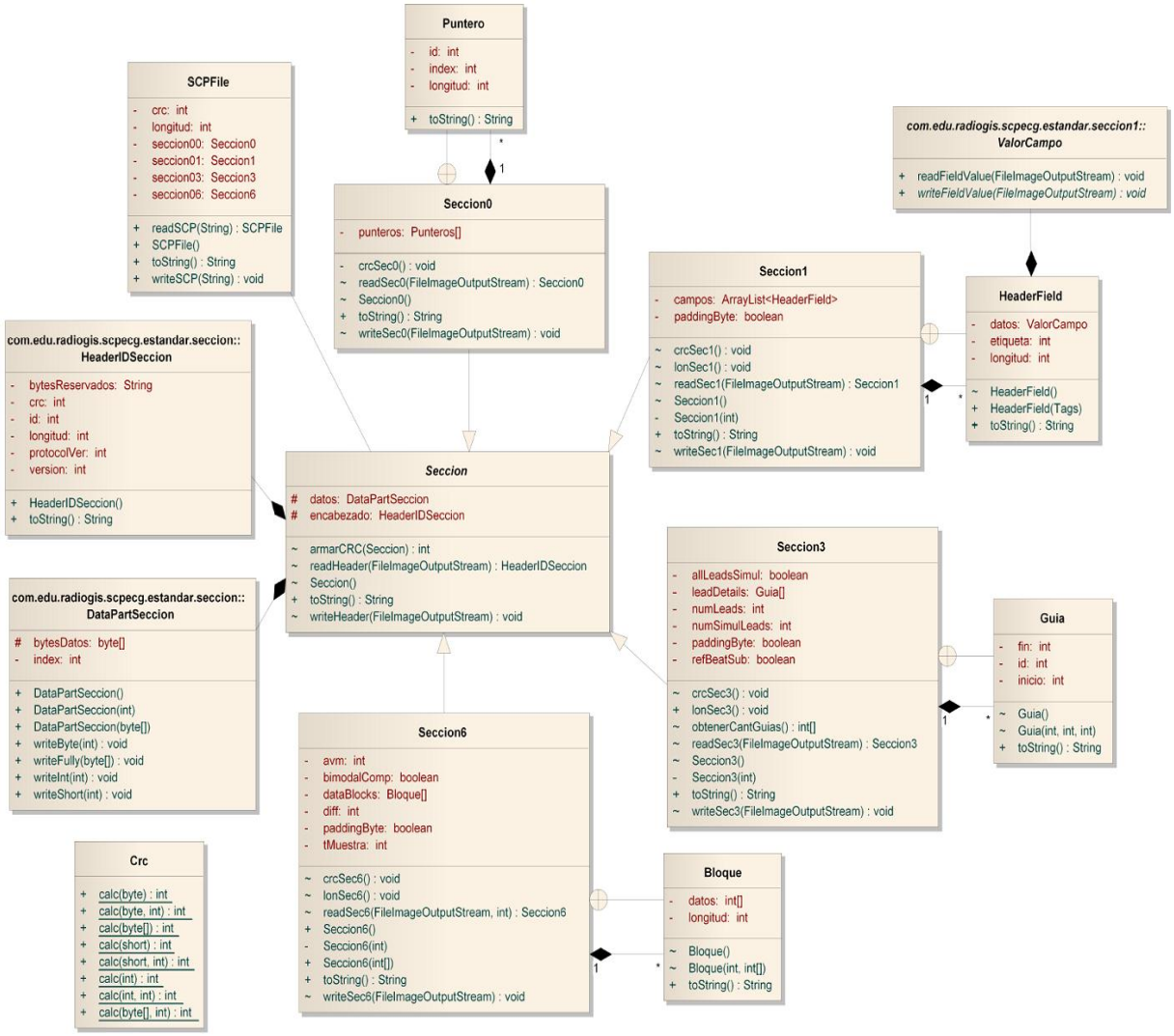

Figura 6. – Diagrama de Clases Estándar SCP-ECG

Este es el diagrama con las clases que representan las secciones requeridas por el estándar que componen solo el archivo SCP-ECG. Dentro de estas secciones se encuentra una sub-sección de encabezado y otra de datos que manejan los mismos campos para todas las secciones ya creadas, estas se construyen para permitir al software de lectura conocer exactamente donde empieza y termina cada una de las secciones, para lo cual requiere campos de punteros, tamaño, numero de bytes y otra información de control cuya utilidad es solo para permitir que el software de lectura sepa exactamente qué hacer en el momento justo.

Como es necesario escribir el archivo con el formato SCP, fue vital la creación de los objetos de tipo Encabezado y Datos los cuales son heredados por todas las secciones del archivo, puesto que son sub-secciones en común para todas ellas compartiendo los mismos campos y es una buena práctica de diseño software ahorrar costos de procesamiento y uso de memoria en el servidor de aplicaciones al momento de ejecutar el servicio, en vista de que debe responder a solicitudes constantes en un tiempo cercano al real. Los objetos creados son:

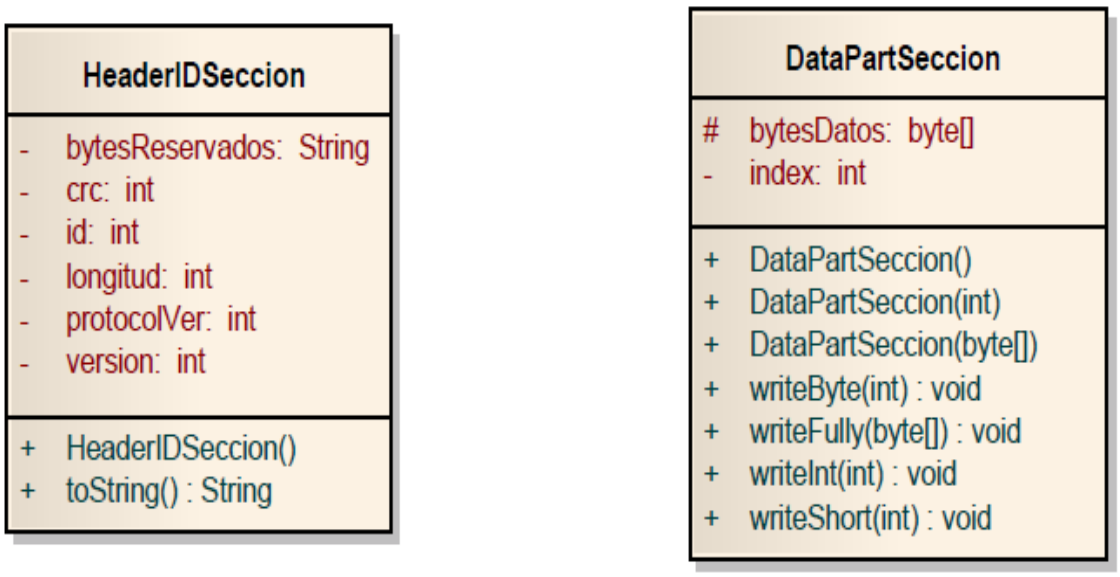

Figura 7. – Diagrama de clases Objetos Comunes

Por último, fue necesaria la construcción de otro modelo de objetos solo para la sección 1.

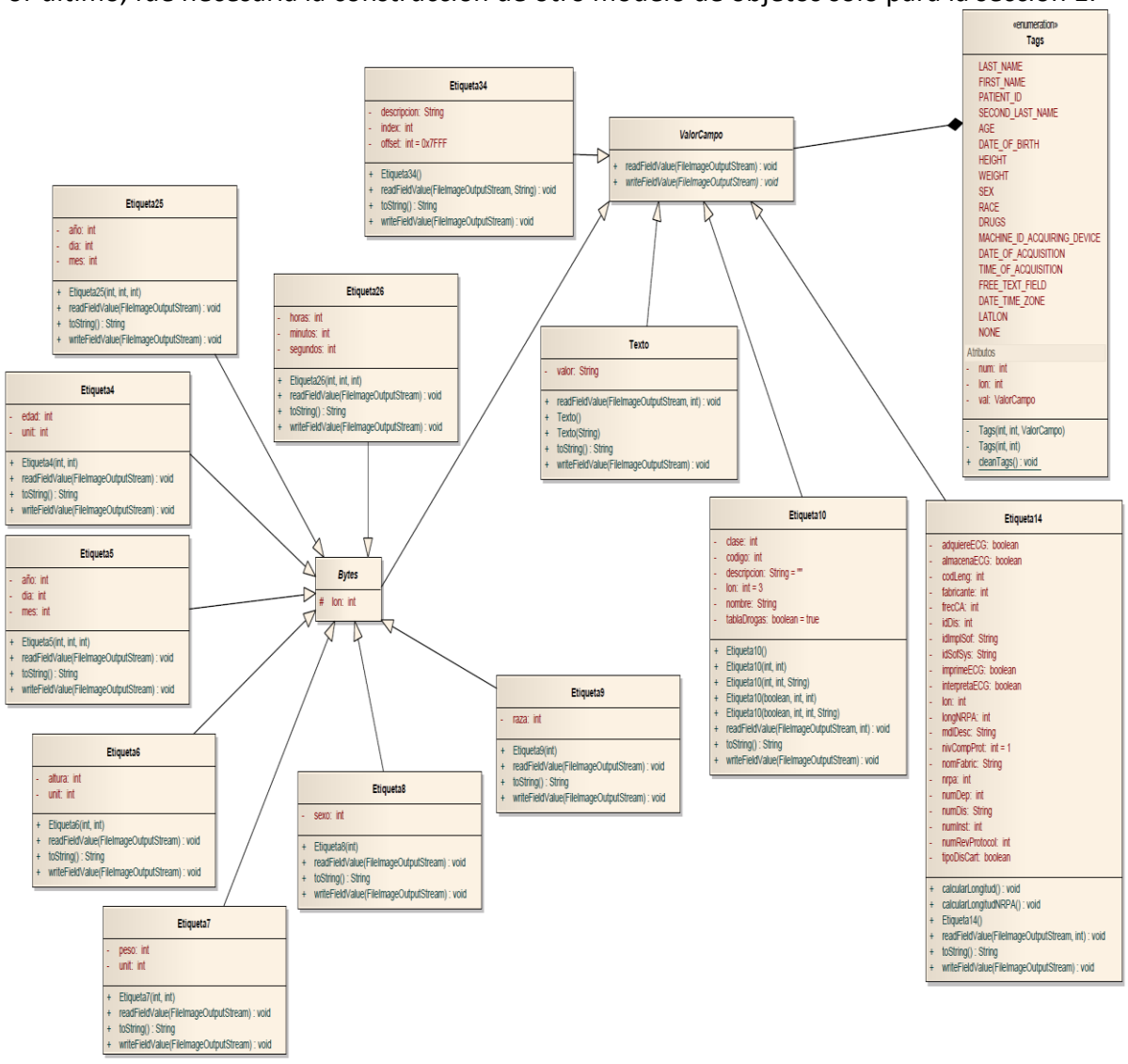

Figura 8. - Diagrama de Clases Sección 1 Estándar SCP-ECG

Esta sección es la encargada de almacenar todos los datos del paciente al cual pertenece el dispositivo que está tomando las mediciones y el cliente del servicio. Estos datos personales como se ha dicho con anterioridad, estarán incrustados en el dispositivo móvil del paciente y serán tomados solo cuando sea necesario, en aras de ahorrar costos de procesamiento para el dispositivo móvil. Como esta sección maneja datos tan específicos y, además, de una forma muy especializada, ya que podemos ver que en algunas etiquetas de esta sección se realizan escrituras de bits específicos junto a cadenas de texto y un orden de bytes determinados así como mezcla de los anteriores, fue de gran importancia

la realización de este diagrama de clases para dominar toda la complejidad de esta sección y cada una de sus etiquetas.

7.3 DIAGRAMA REALACIONAL DE LA BASE DE DATOS

El diagrama relacional de una base de datos es el esquema que se realiza en base a las necesidades planteadas por el Servicio y es la guía con la que se construye el sistema de almacenamiento de la información que va a servir como base para la ejecución del proyecto. Es importante que este mantenga toda integridad referencial, sea segura, robusta y extensible, mirando a futuro en distintas implementaciones que se puedan dar del servicio.

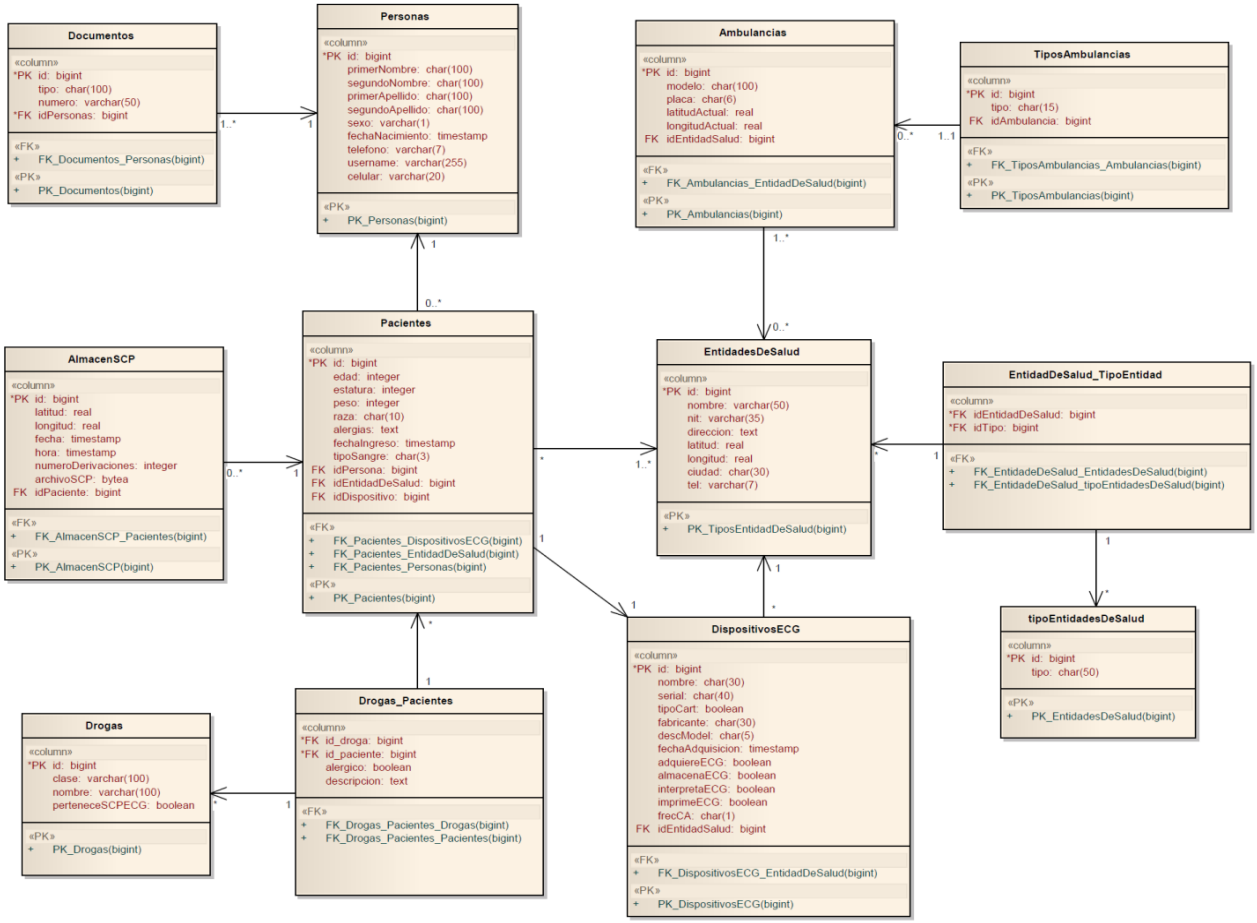

Figura 9. – Diagrama Relacional Base de Datos

Este diagrama relacional fue luego traducido en lenguaje SQL para ingresarlo a la base de datos creada en PostgreSQL 9.0, a través del pgAdminIII. Posteriormente se realizaron inserciones de datos y pruebas en la misma para verificar que la integridad referencial era

la deseada y que la estructura de datos que fue planteada iba a ser la base correcta para desarrollar el Servicio.

Para el control de autenticación y autorización se empleó el siguiente diseño en la base de datos:

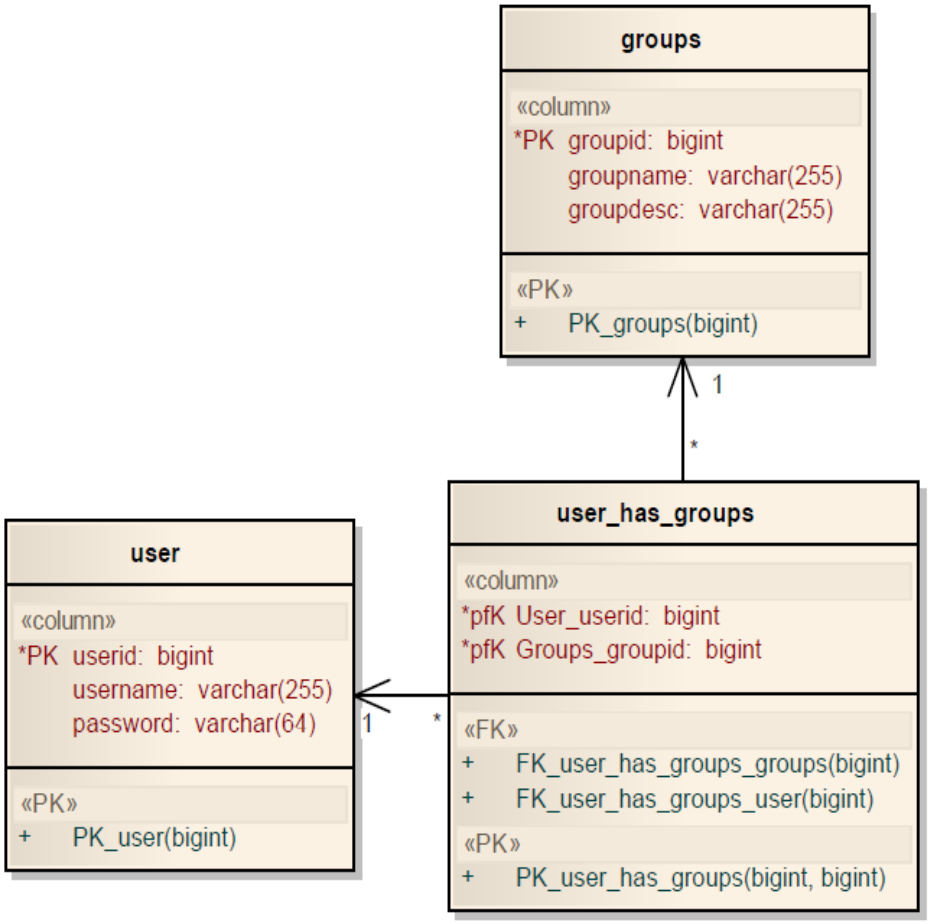

Figura 10. – Diagrama Relacional para el Manejo de Usuarios

Se realizó de esta manera para emplear JAAS (*Java Authentication and Authorization Service*) y Servlet 3.0 *Login,* tecnologías integradas dentro del servidor de aplicaciones Glassfish 3.0.1.

8. DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE CLIENTE

Para la construcción del software cliente se usó el SDK de Android, aprovechando su portabilidad y capacidad de procesamiento, fue ideal para el desarrollo de una aplicación cliente que capturara, empaquetara y enviara los datos al servidor de aplicaciones el cual maneja el servicio, además de tener la ventaja que sus dispositivos cuentan con acceso a internet móvil, lo que permite estar en contacto todo el tiempo con el servidor siempre que haya señal. En aras de seguir la filosofía de Android se simplifico las funciones que ejecuta el cliente, librándolo de carga operacional innecesaria para este dispositivo. Se construyó el siguiente diagrama de clases para ilustrar la estructura del cliente:

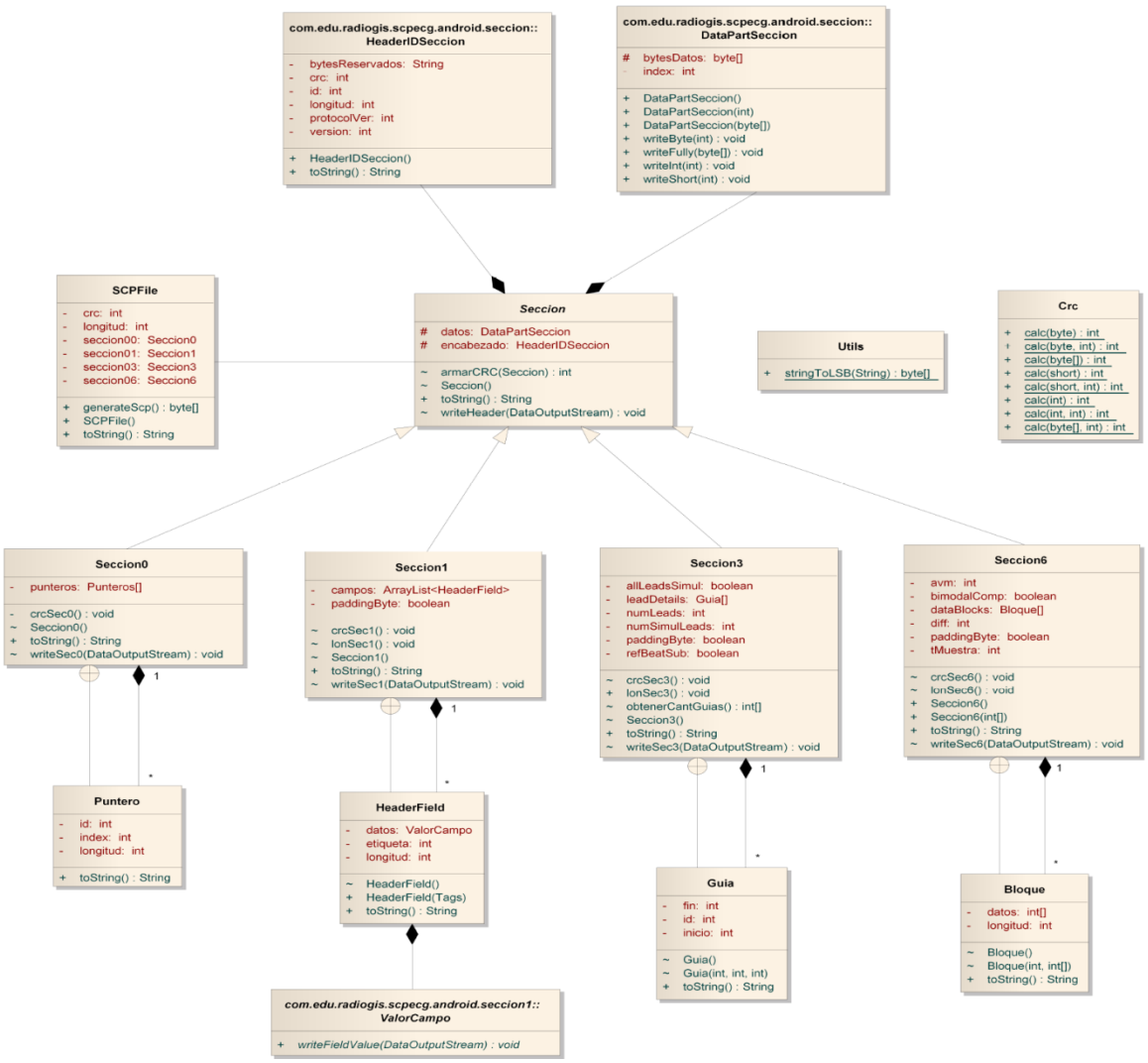

Figura 11 – Diagrama de Clases Cliente Android

Este software se encarga de recibir el ECG del paciente y escribirlo acorde las reglas del Estándar SCP-ECG versión 2.2, luego se conecta con un servicio web REST construido en el servidor y envía todos los datos como una transmisión de *bits*, este servicio en el servidor de aplicaciones recibe esta transmisión de bits y la reconstruye acorde la estructura de clases expuesta anteriormente en la descripción del estándar(ver pag 23) para que cumpla con todas las reglas del estándar SCP-ECG, luego se encarga de guardar la información recibida en la base de datos que se construyó en PostgreSQL 9.0. El cliente Android también posee otras dos estructuras de clases mostradas a continuación:

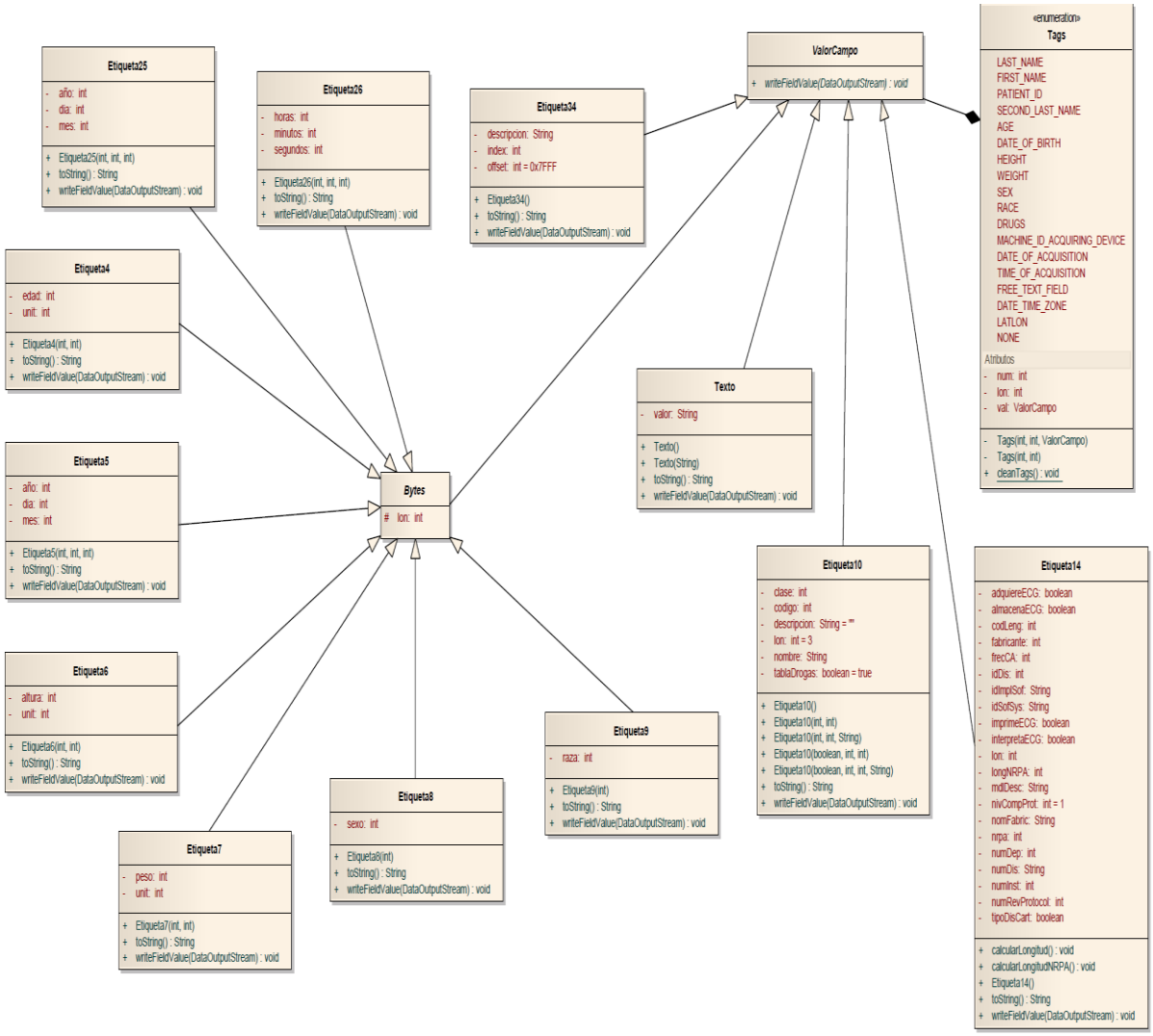

Figura 12. – Diagrama de Clases Sección 1 en Android

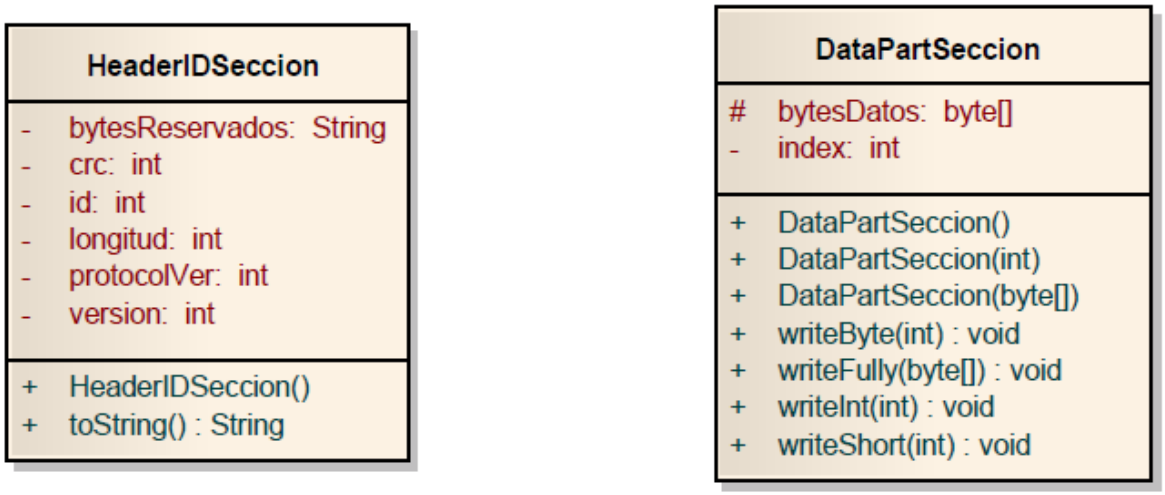

Figura 13. – Diagrama de clases Objetos comunes en Android

Se comportan de la misma manera que el software de empaquetamiento construido enteramente en Java SE6, con una estructura para describir completamente la sección uno y los dos objetos de encabezado y sección de datos para cada una de las secciones, la gran diferencia radica en que el cliente construido para el sistema operativo Android, no tiene la función de leer archivos del formato SCP-ECG y solo cuenta con la facultad de escribir y enviar datos en este formato. Esto fue determinado por la necesidad de ahorrar carga operacional en un dispositivo móvil, puesto que no tiene las mismas prestaciones de una máquina de escritorio y siguiendo la filosofía sobre la cual se construyó Android de usar los menor cantidad de recursos posibles debido a que el hardware es limitado, además de que es innecesario para el cliente móvil leer archivos de este formato, puesto que no recibe información de este tipo, solo se encarga de empaquetar y enviar al servidor los datos del cliente.

Para la construcción del software usando el SDK provisto en la página oficial de Android se empleó el IDE *Eclipse*, esto por recomendación de los desarrolladores del sistema operativo, quienes desarrollan el complemento para trabajar en este IDE. Las clases empleadas para la escritura de los archivos en el cliente se modificaron para que funcionaran correctamente, ya que no todo el API de Java está integrado en Android, en vista de que debe ser más ligero y eficiente. Android emplea una especie de máquina virtual similar a la máquina virtual de Java pero con diferencias en las funciones y aplicativos que tiene integrado, y ya que es un sistema operativo posee una serie de hilos de ejecución sobre los cuales residen las aplicaciones construidas para este, siendo *UIthread* el hilo principal sobre el cual residen todas las propiedades de la interfaz de usuario y la vista del dispositivo.

Sobre este se construyeron prototipos de interfaz, para la interacción del cliente con el software, en esta UI se ven los datos del paciente, las coordenadas y demás información relevante para el servicio, la cual debe ser enviada al servidor por medio de internet móvil, a continuación se muestran imágenes de esta interfaz de usuario, cabe resaltar que todos los datos usados para este prototipo son ficticios y no refieren a un paciente real ni una persona real:

Figura 14. - Interfaz en Android 1

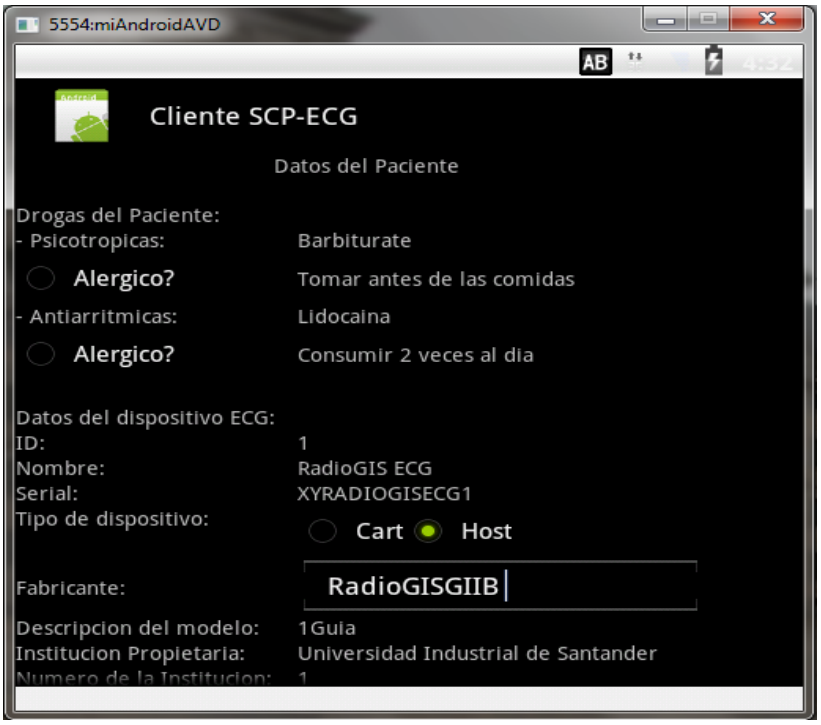

Figura 15. - Interfaz en Android 2

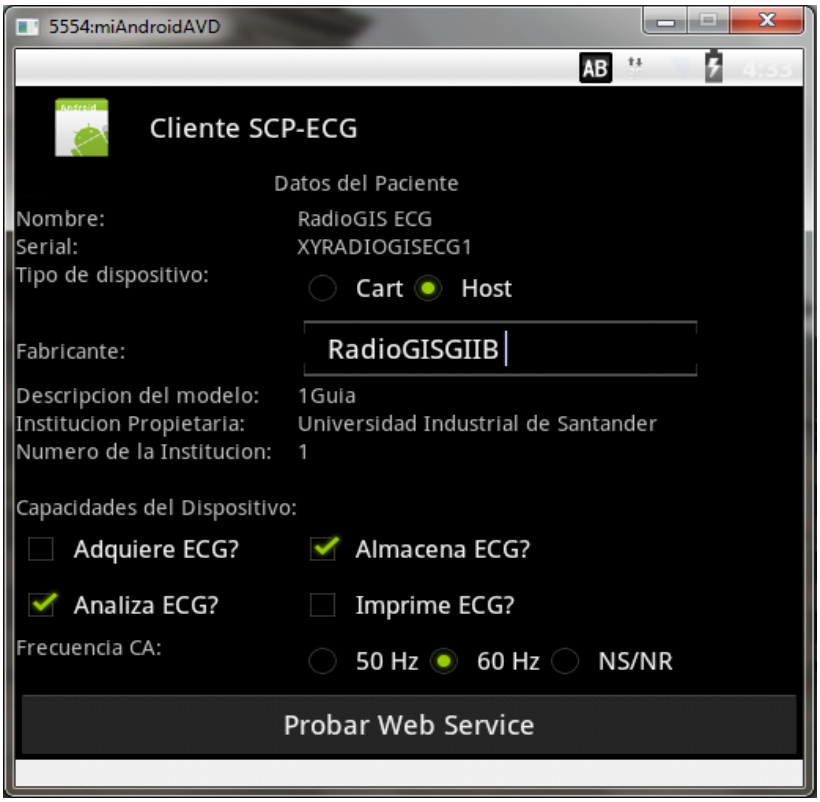

Figura 16. - Interfaz en Android 3

Debido a que Android maneja muchas aplicaciones simultáneamente se debe hacer un manejo eficiente de los hilos y de la memoria disponible para el sistema operativo, para eso, el sistema provee de la clase *AsyncTask* que usa tres genéricos (*Param*, tipo de parámetros enviados a la tarea en ejecución; *Progress*, el tipo de las unidades de avance publicadas en los cálculos de fondo; *Result*, el tipo de resultado de los cálculos de fondo) y es la encargada del manejo de hilos de ejecución. Android permite que se puedan hacer manejadores personalizados, pero no siendo este el caso de estudio para el proyecto se usó la clase propia en el sistema. Además del manejo de hilos Android requiere que se den permisos para la ejecución de las distintas aplicaciones que se vayan a instalar, para este caso en específico se necesitó de un permiso para realizar a la conexión a internet:

<uses-permission android:name="android.permission.INTERNET" />

Este se agrega en el *AndroidManifest.xml* y todas las operaciones sensibles como acceso a internet, sockets, manejo de archivos, entre otras, no se deben hacer en el hilo principal del sistema operativo, por razones de estabilidad del sistema, y para esto hacemos uso del *AsyncTask*. Por último se debe especificar el uso del *Garbage Collector* empleado por la máquina virtual de Java, este se encarga de limpiar cada cierto tiempo la memoria de la máquina de objetos a los cuales la aplicación ya no está apuntando y son obsoletos e innecesarios para la aplicación. Se encuentra activo automáticamente en la JMV del Java SE6 pero debido a que Android no emplea la misma máquina virtual, este no está activo todo el tiempo ni de forma automática, así que se debe ejecutar en el cliente en determinados tiempos para prevenir que el sistema se inunde de objetos inútiles y haga lenta la ejecución del sistema operativo.

Los datos del cliente se encontraran integrados con el software instalado en el dispositivo Android para el monitoreo, este estará conectado con un ECG portable el cual se encarga de tomar la señal cardiaca del paciente, luego el software toma estos datos de la señal y los empaqueta en archivos SCP-ECG, agrega los datos del paciente junto con las coordenadas de su posición actual tomadas del sistema de posicionamiento del dispositivo móvil(Android se ejecuta en *SmartPhones* y estos a su vez tienen funciones para dar al ubicación actual del dispositivo, por eso y al ser desarrollado por Google, se asume que las coordenadas de latitud y longitud se pueden tomar del mismo) y se conecta a un servicio web REST en el servidor de aplicaciones para transmitir los datos.

Para la recepción de los bytes que envía el cliente Android fue necesaria la construcción de un servicio web que recibiera el flujo de bytes enviados y reconstruyera de manera exacta y eficiente el paquete SCP-ECG que fue transmitido. Para lograr esto se

implementó *REpresentational State Transfer (REST)* ya que es un idioma de diseño clave que abarca arquitectura cliente-servidor sin estado en la que los servicios web son vistos como recursos y pueden identificados por sus *URL's (Uniform Resource Locator)*. Esta arquitectura tiene la gran ventaja de ser completamente sin estado, lo que permite que una interacción pueda sobrevivir una reiniciada de servidor lo que es de gran importancia en un sistema de monitoreo como el construido. También funciona con ancho de banda muy limitado, esencial para aplicaciones en dispositivos móviles como PDA's (*Personal Digital Agenda)* o en este caso un dispositivo Android.

9. DISEÑO E IMPLEMENTACION DE LA APLICACIÓN WEB

El servicio de telemedicina consta de una aplicación web la cual cumple la función de mostrar los datos enviados por el cliente Android y permitir el monitoreo de la condición y ubicación de los pacientes que hacen parte del servicio.

Esta aplicación se realizó bajo el marco de trabajo JavaServer Faces 2.0, esta versión por defecto emplea *Facelets*, reemplazando a JSP que era usando en sus versiones 1.x. *Facelets* es una tecnología de vista para JSF, es construida desde cero tomando en cuenta el ciclo de vida del componente JSF. Con *facelets* se producen las plantillas que construyen un árbol de componentes, no un *Servlet*. Es un poderoso sistema de plantillas que permite definir las vistas JSF usando plantillas de estilo HTML, reduce la cantidad de código necesario para integrar componentes dentro de la vista.

9.1 Prototipo de Interfaz

La aplicación web fue construida con Netbeans IDE 6.9.1, esta aplicación se conecta con la base de datos y muestra información que se encuentra almacenada allí.

Inicialmente se tiene la página de inicio de la aplicación:

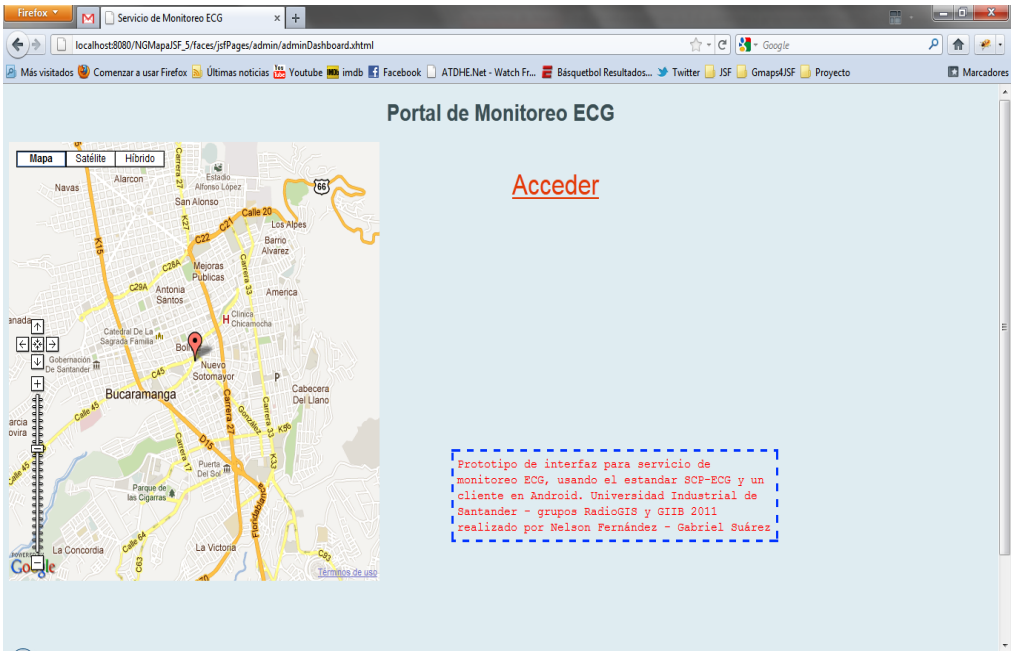

Figura 17. – Prototipo Página de Inicio

Después está el prototipo de página de autenticación de usuario, que se muestra a continuación:

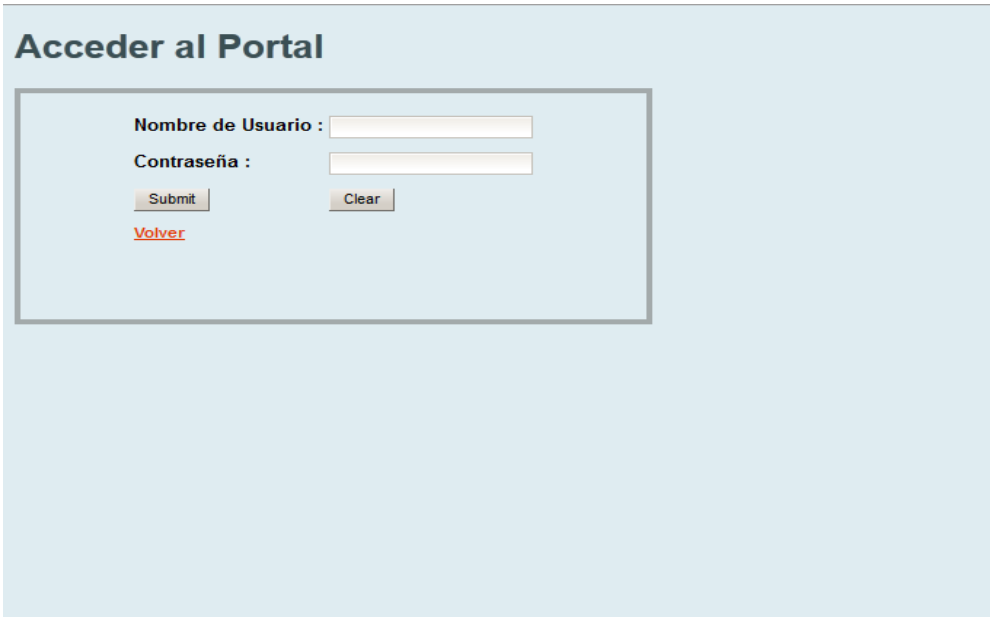

Figura 18. – Página de Autenticación de Usuario.

Para realizar las funciones de autenticación de usuario se empleó JAAS y Servlet 3.0, servicios que están integrados en el servidor de aplicaciones Glassfish 3.0.1 y que son perfectamente compatibles con aplicaciones JavaServer Faces, en la sección 9.3 se profundizara más en estos métodos empleados para garantizar la seguridad e integridad de la aplicación web, pero solo están enfocados al servicio y no se enfocó el trabajo de seguridad a todo el servidor como tal, puesto que este proyecto estará alojado en un servidor del grupo RadioGIS, el cual ya tiene integrado sus métodos de seguridad y protección.

A continuación se muestra el prototipo de interfaz para 3 actores principales de este servicio que son, el administrador, el médico y el paciente:

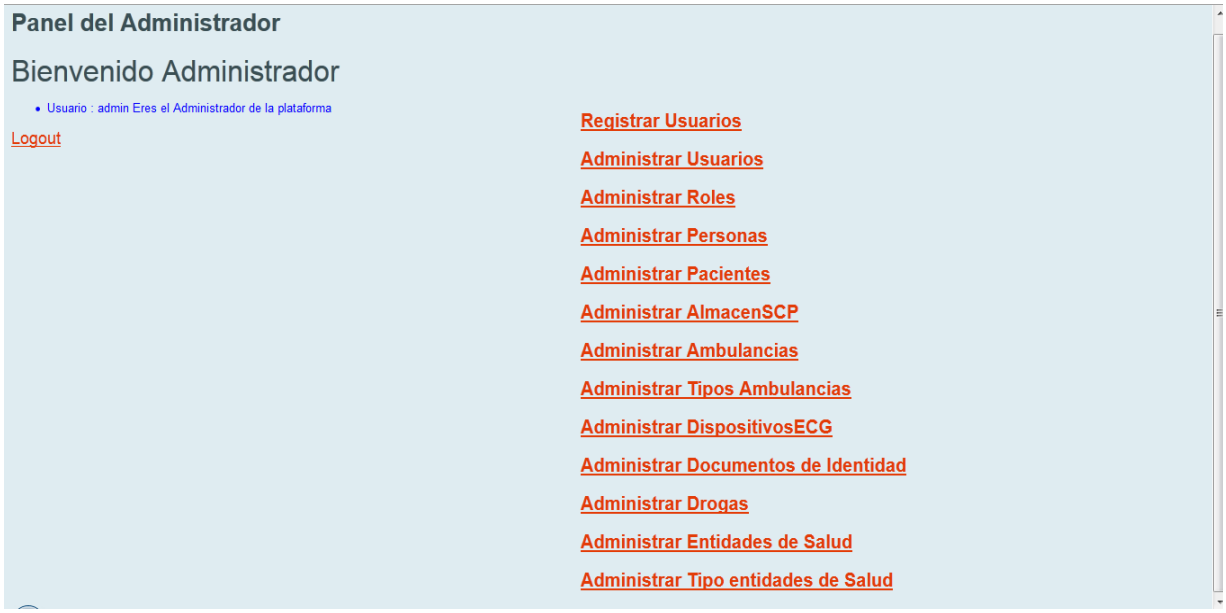

Figura 19. – Panel del Administrador

Como se puede ver en la imagen el administrador del servicio tiene acceso a todas las entidades de la base de datos que interactúan con el sistema, representadas en los enlaces que se muestran en el panel.

Estos enlaces llevan a una página de administración para cada una de estas entidades, como por ejemplo esta de las personas vinculadas al sistema:

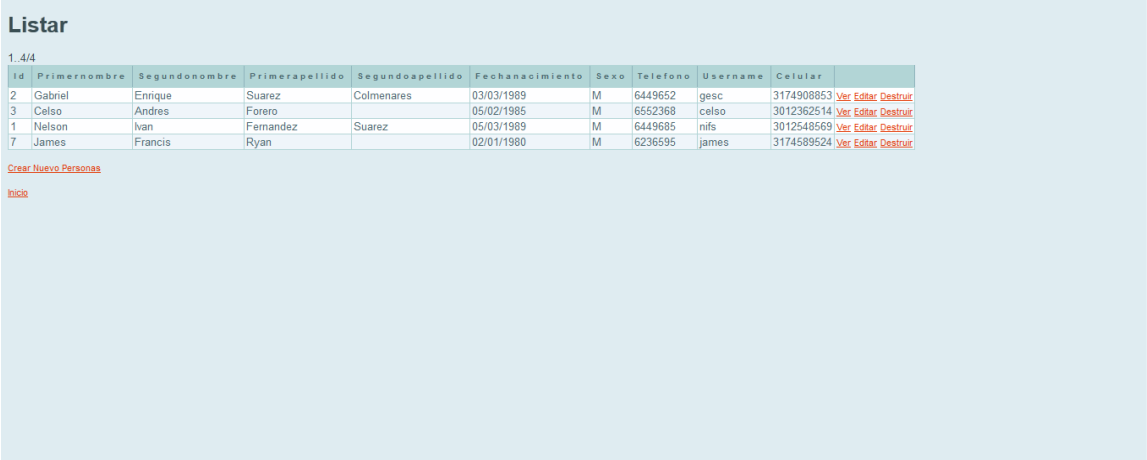

Figura 20. – Lista de Personas, datos de prueba.

Así cada uno de los enlaces lleva a su respectiva vista, exceptuando el primer enlace el cual se especificaran sus detalles más adelante.

De cada una de estas vistas detalladas se destacan los enlaces al final de cada tupla, más claramente los mostrados acá:

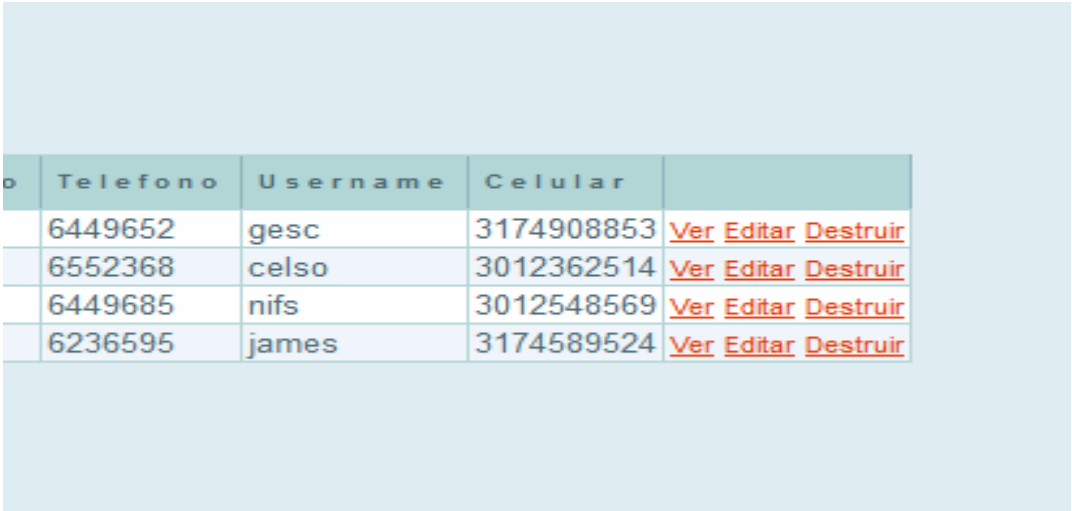

Figura 21. – Enlaces de control en las tuplas

Estos funcionan como herramientas de administración para los elementos del sistema y llevan a otras vistas en las cuales la aplicación despliega datos y escucha eventos del usuario para realizar cambios en la aplicación. Si el administrador escoge ver, se abrirá un página más detallada solo con la información de la tupla que selecciono, editar el permite cambiar datos de esa tupla y destruir simplemente borra el registro de la base de datos.

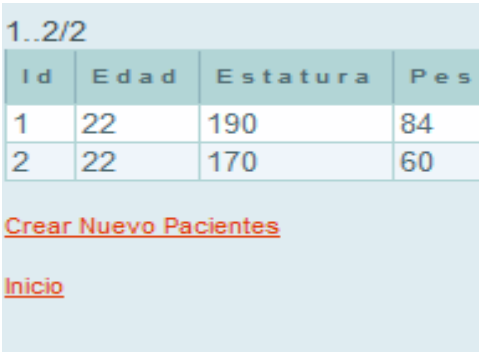

Figura 22. – Controles Adicionales en la vista

También existen estos dos controles adicionales, el cual permite tal y como dice el enlace, crear un nuevo registro para esa entidad o volver a al panel de inicio del usuario.

Otro prototipo de interfaz realizada, y esta también es de suma importancia, es la del médico que estará monitoreando los signos vitales del paciente, y será quien deba tomar decisiones en base a la condición del mismo acorde con los datos que ve en la aplicación web, el prototipo del panel de inicio para el médico se muestra acá:

Figura 23. – Panel de inicio del Medico

Como se observa en la imagen este usuario solo tiene acceso a 3 opciones de monitoreo en la aplicación, estas son las únicas que le concierne al médico ver para desempeñar su función en el servicio. Dentro de estas se encuentran los datos de los pacientes, entidades de salud asociadas a los pacientes y también de las ambulancias que se encuentran registradas a en la base de datos del servicio. Inicialmente como parte de prototipo no se emplean funciones complejas en cuanto a las entidades de salud o las ambulancias, se realizó su persistencia en la base de datos también con su ubicación geográfica en un mapa de Google Maps buscando hacer la aplicación extensible para nuevos desarrollos en estas partes y hacer un servicio mucho más completo en cuanto a eficacia de atención médica y mejoramiento del sistema de salud que emplee este servicio de monitoreo.

El Prototipo de “ver Pacientes” luce de esta forma en la aplicación:

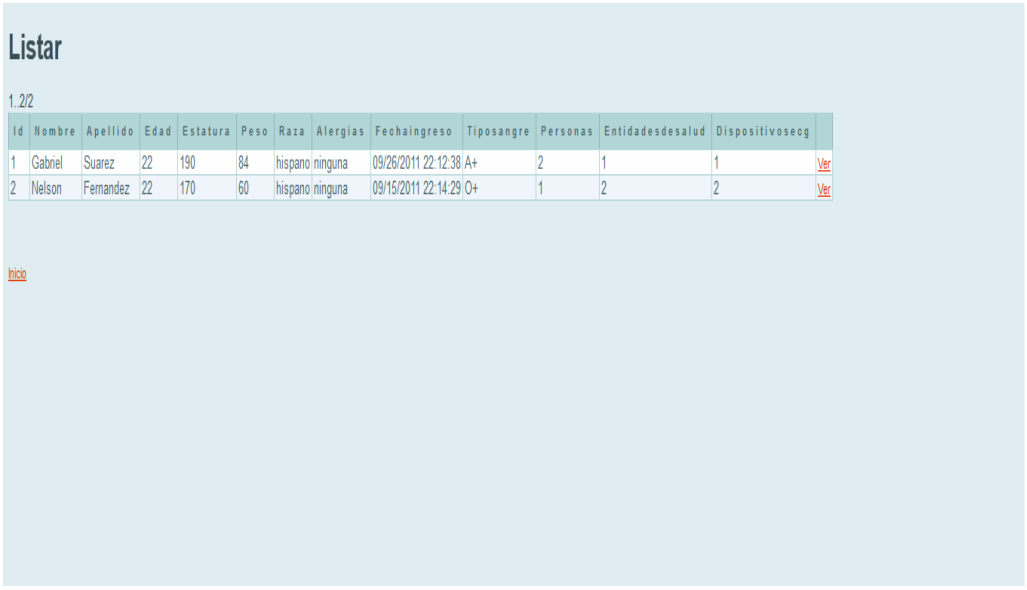

Figura 24. – Ver paciente desde el usuario médico.

Nótese que no tiene facultades de agregar, editar o borrar pacientes del servicio, esto es porque no tiene privilegios de administrador y sus funciones son meramente de monitoreo.

Si después el médico pica en “Ver” se desplegara la siguiente interfaz:

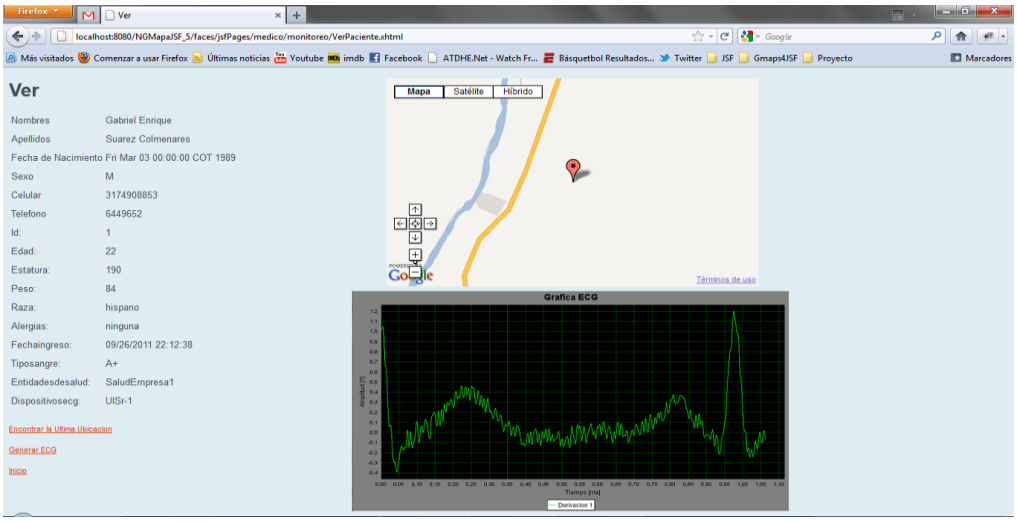

Figura 25. – Detalles del paciente desde el usuario Medico.

Como se puede apreciar en la imagen, la interfaz despliega todos los datos del paciente y se muestra un cuadro con el mapa en el cual está la ubicación del paciente. También hay un enlace en la parte inferior izquierda que se usa para rastrear la última ubicación del paciente, esta información es extraída de la tabla “AlmacenSCP” alojada en la base de datos que está conectada con la aplicación WEB. De la misma forma el cuadro cardiaco se despliega con un enlace, esta información se encuentra empaquetada en formato SCP-ECG dentro de la base de datos, puesto que es enviada por el cliente Android, debido a esto la aplicación debe realizar el proceso de lectura del paquete SCP y extraer solo la información del cuadro cardiaco para mostrar en la vista de monitoreo. Este graficador de la señal cardiaca es realizado en Java con la ayuda de *JFreeChart*, que se encarga de tomar la señal de entrada, procesar los datos con *Spline* para suavizar la curva y mostrar una señal que sea clara y relevante a la hora de realizar un diagnóstico médico, este es necesario debido a que el dispositivo toma miles de pulsos eléctricos que luego son interpretados como la señal cardiaca.

También se puede ver la ubicación de una ambulancia registrada al sistema, estos son solo datos de prueba y no representan una aplicación realizada para localizar ambulancias, sino una funcionalidad que puede ser integrada a este proyecto, para desarrollos futuros:

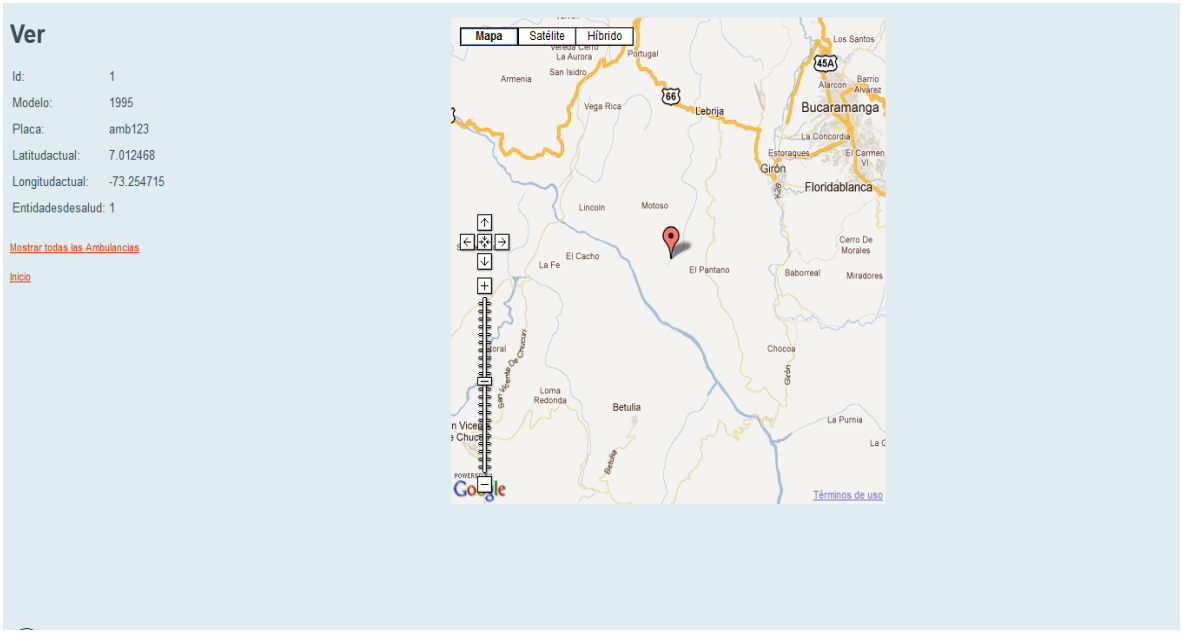

Figura 26. – Ubicación de una Ambulancia en Google Maps.

Esta función también está disponible para las entidades de salud, cuya ubicación no cambia puesto que son lugares fijos en la ciudad.

A continuación se muestra el prototipo de interfaz construido para el paciente, quien también puede acceder a la plataforma para observar su estado actual, se le asigna un nombre de usuario y contraseña para autenticarse y así acceder al servicio:

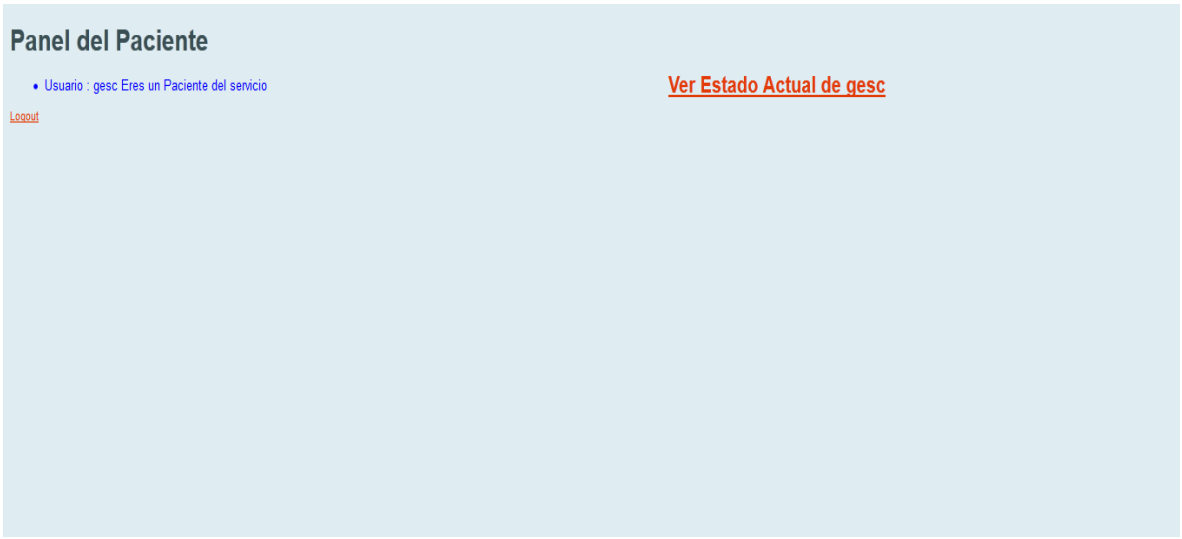

Figura 27. – Panel del Paciente

Nótese que las funciones para el paciente se limitan a la de ver el estado de sí mismo, y no tiene privilegios para ver otros pacientes, editar, agregar o borrar ningún dato en la plataforma, puesto que es solo un paciente y se debe tener un control sobre lo que puede hacer el usuario en la aplicación como buena práctica de desarrollo software.

Cuando el paciente solicita ver su estado actual se despliega otra pantalla, donde ve sus datos personales, ubicación y un detalle de la señal cardiaca que está siendo enviada por el cliente instalado en el dispositivo que lo monitorea. Al contrario del médico, no re direcciona a un panel de selección de pacientes, sino se muestran directamente sus propios datos, todo esto por razones de seguridad y privacidad de los demás pacientes del servicio, como se muestra en la imagen:

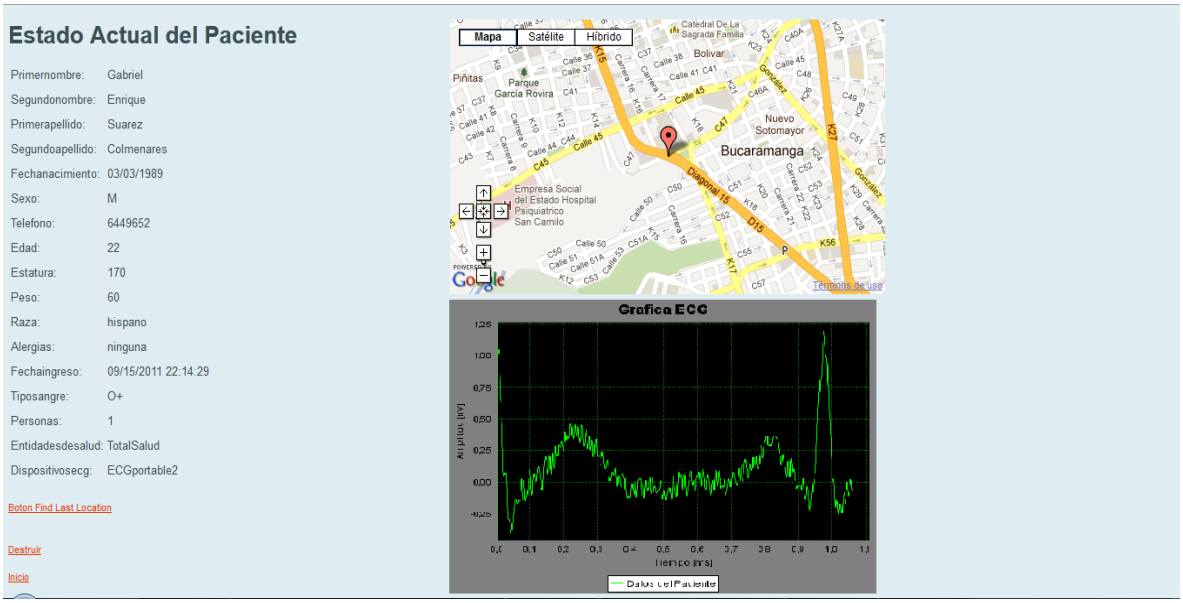

Figura 28. – Estado del paciente desde el panel del Paciente.

Estos son los prototipos de interfaz más relevantes en el servicio, cabe resaltar que el diseño de las paginas se realizó en aras de funcionalidad y puesto que son solo prototipos no presentan gran desarrollo en cuando a estilo y estética, sencillamente se construyeron para presentar los datos relevantes y determinar si el servicio estaba funcionando correctamente o no. Con este proyecto se da vía libre para desarrollar interfaces y vistas mucho más refinadas en el futuro, usando la tecnología JavaServer Faces es posible integrar distintas librerías y complementos que ayudan al desarrollo de interfaces más especializadas y dinámicas, es posible incluir Ajax para actualizar los datos en tiempo real, pero siempre teniendo cuidado de no inundar al servidor con datos.

Los mapas mostrados en todas las interfaces son una integración de Google Maps para JavaServer Faces, esto se logró mediante el uso de librerías específicas para esta tecnología, que se pueden encontrar en el enlace *http://code.google.com/p/gmaps4jsf/*. Esta es de acceso libre y *Open Source* para cualquier integración de Google Maps con JavaServer Faces 2.0.

9.2 Componentes que integran la aplicación.

La aplicación web cuenta con varias tecnologías y componentes integrados que trabajando conjuntamente permiten el funcionamiento del mismo, estos componentes cumplen funciones muy específicas y es una de las razones por las cuales desarrollar proyectos y aplicaciones web usando herramientas Java es tan popular, ya que permiten gran extensibilidad a la vez que ofrecen robustez en los desarrollos.

Específicamente para este caso se emplearon componentes para el geo-posicionamiento, la conexión con base de datos y persistencia, autenticación y autorización de los usuarios y monitoreo de la señal cardiaca. También están las vistas empleadas usando tecnología JavaServer Faces así como el servidor de aplicaciones sobre el cual se ejecuta la aplicación web y el motor de base de datos que se usa para mantener los datos del servicio.

- Geo-posicionamiento

Para este componente se empleó Google Maps integrado a la aplicación mediante la librería *Gmaps4jsf* versión 1.1.4, de uso libre y código abierto, que hace una completa adaptación de la API de Google Maps para aplicaciones JSF 2.0. Esto en conjunto con los datos enviados por el cliente Android los cuales se almacenan en la base de datos permite crear un sistema de geoposicionamiento.

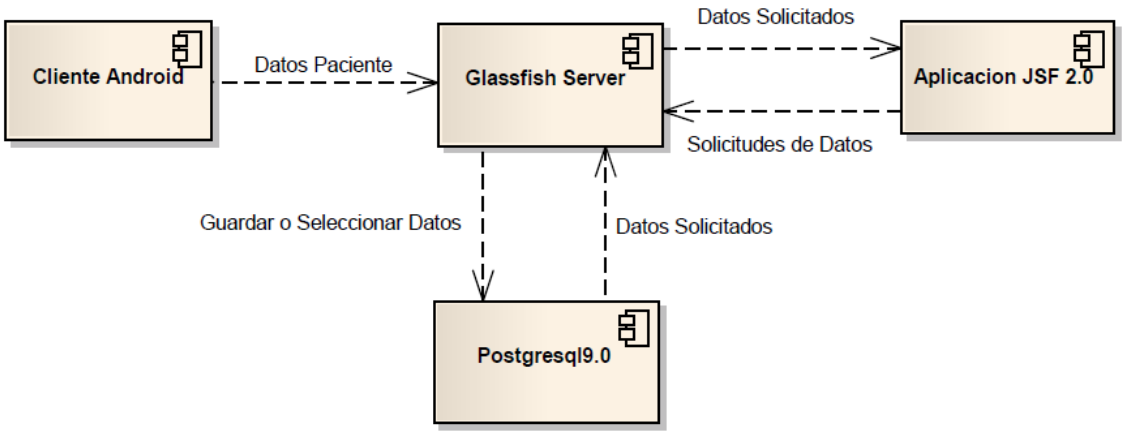

Figura 29. – Componentes con flujo de datos

El flujo de datos presentado muestra cómo funciona el geo-posicionamiento realizado por el servicio en la aplicación Web. Primero el cliente Android envía la información del paciente junto con las coordenadas (Latitud y Longitud) y el servidor de aplicaciones Glassifish las recibe mediante el uso de un servicio web REST, este se encarga de guardar la información en la base de datos (con motor PostgreSQL) luego, cuando es solicitado ubicar a determinado paciente desde la vista, la aplicación Web JSF solicitará al servidor los datos de las coordenadas del paciente en cuestión. Para la comunicación entre vista y servidor JSF emplea JDBC (Java DataBase Connectivity), el mapeo de la base de datos en la aplicación se hace con EJBs de Entidad (Enterprise Java Beans) y para la persistencia JPA (Java Persistence API). Luego de procesada la solicitud el servidor devuelve los datos a la aplicación web, que ingresa los parámetros de latitud y longitud a la instancia de Google Maps para JSF y este ubica automáticamente un marcador en el punto exacto que se solicitó, de esta manera se hace una localización sencilla y rápida, sin necesidad de procesos más complicados o dispositivos muy costosos y sofisticados.

- Base de Datos y Persistencia

Para conexión a la base de datos la aplicación emplea JDBC, que permite realizar operaciones de base de datos desde la aplicación Java, esta funciona sobre cualquier motor de base de datos, pero en este caso particular se empleara el motor Postgresql9.0. en la aplicación las entidades de la base de datos se mapean con EJB 3.0, el cual emplea una unidad de persistencia, utilizada para representar la base de datos en el contexto de

la aplicación, con estos *entity beans* es posible mostrar y guardar datos que son necesarios para el funcionamiento del servicio.

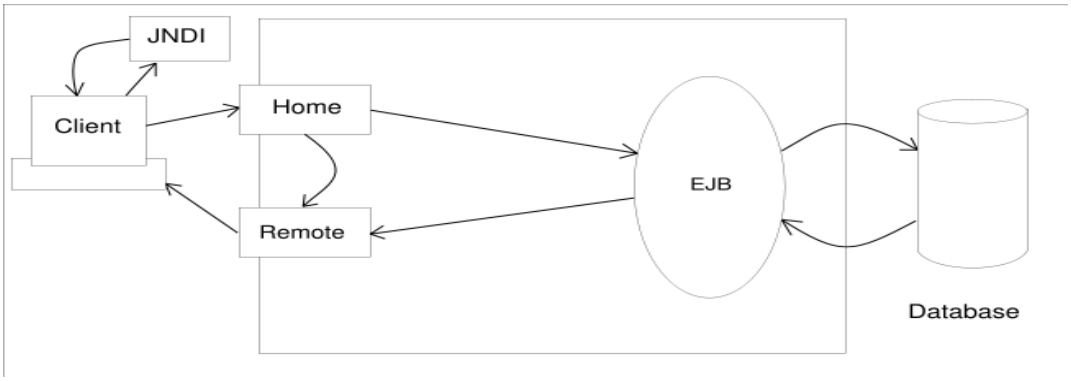

Figura 30. – Arquitectura básica de EJB

- Autenticación y Autorización de los usuarios

Esta se maneja utilizando dominios de seguridad en el servidor de aplicaciones Glassfish 3.0.1, así como recursos JDBC, JAAS (Java Authentication and Authorization Service), JNDI (Java Naming and Directory Interface) y Servlet 3.0. Esta parte será descrita en más detalle en la siguiente sección.

- Monitoreo de la Señal Cardiaca

Este componente se construyó en lenguaje Java SE6. Siempre ha sido una falencia muy grande para Java el uso de interfaces y el manejo de gráficos con estilos dinámicos y llamativos, afortunadamente esto ha ido mejorando con el paso del tiempo y se puede hacer uso de herramientas como los *jfreechart*. Mediante este pequeño aplicativo se puede mostrar los datos de la señal cardiaca tomándolos como entrada y luego procesándolos con *Spline* para suavizar la curva, eliminando el ruido y exceso de datos innecesarios de entrada. Como resultado de este proceso se deja una gráfica ECG clara que puede ser interpretada por el médico que monitorea al paciente, y así hacer más eficiente la tarea de monitoreo por parte del médico.

- Vistas en JSF y Servidor de Aplicaciones GlassFish

Estos son dos componentes esenciales para el funcionamiento del servicio, siendo uno la interfaz de usuario en JSF 2.0 con *facelets* y otro el servidor de aplicaciones *GlassFish* 3.0.1. Juntos se encargan de recibir las entradas por parte del usuario, procesarlas para luego devolver las vistas o los datos que este haya solicitado y en caso de existir una

excepción en este proceso notificarla al usuario para que corrija los datos. La tecnología JSF con *facelets* permite construir interfaces de usuario usando componentes reutilizables de fácil acceso, que son familiares para el usuario y livianos de ejecutar por cualquier navegador Web, a diferencia de otras aplicaciones como Flash que requiere de complementos para cada navegador y emplea gran carga computacional para la maquina cliente. El servidor de aplicaciones Glassfish 3.0.1 es una excelente opción a la hora de hacer aplicaciones web en lenguaje Java, está integrado con el IDE Netbeans y es recomendado por la comunidad Java, también trae incluido rutinas y servicios de seguridad que brindan la protección y robustez que requiere el servicio.

9.3 AUTENTICACIÓN, AUTORIZACIÓN y SEGURIDAD

Esta es parte importante en cualquier aplicación Web y desarrollo de software en general ya que un sistema seguro es confiable, estable y robusto. Para empezar a describir la seguridad de la aplicación desarrollada debemos ahondar en los servicios y aplicaciones que hacen posible esto.

Antes de autenticar se requiere un registro de usuarios en la plataforma que permita mantener un control estricto sobre quienes pueden acceder o no a determinadas partes de la aplicación, para esto se desarrolla un sistema de Usuarios y Grupos que delimita el dominio de la aplicación a los permisos que tenga cada usuario. Para esto se emplea la siguiente estructura de datos para su almacenamiento y administración:

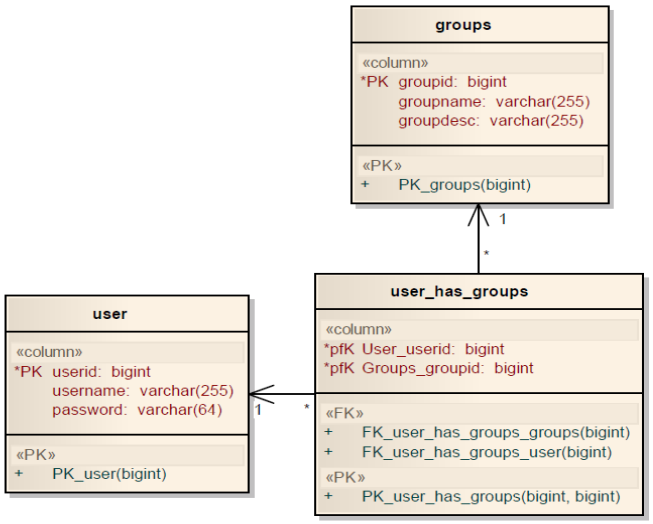

Figura 31. – Diagrama Entidad Relación de los Usuarios

Es necesario usar esta arquitectura de datos para emplear JAAS como servicio de autenticación y autorización. Por esto mismo se realizó en la base de datos una vista que representa el nombre de usuario, su contraseña y el grupo al cual pertenece, ya que GlassFish espera que estos datos estén alojados en la misma tabla, pero esto no es una buena práctica de construcción de base de datos, por lo cual es más práctico realizar una vista que junte estos datos cada vez que se necesiten.

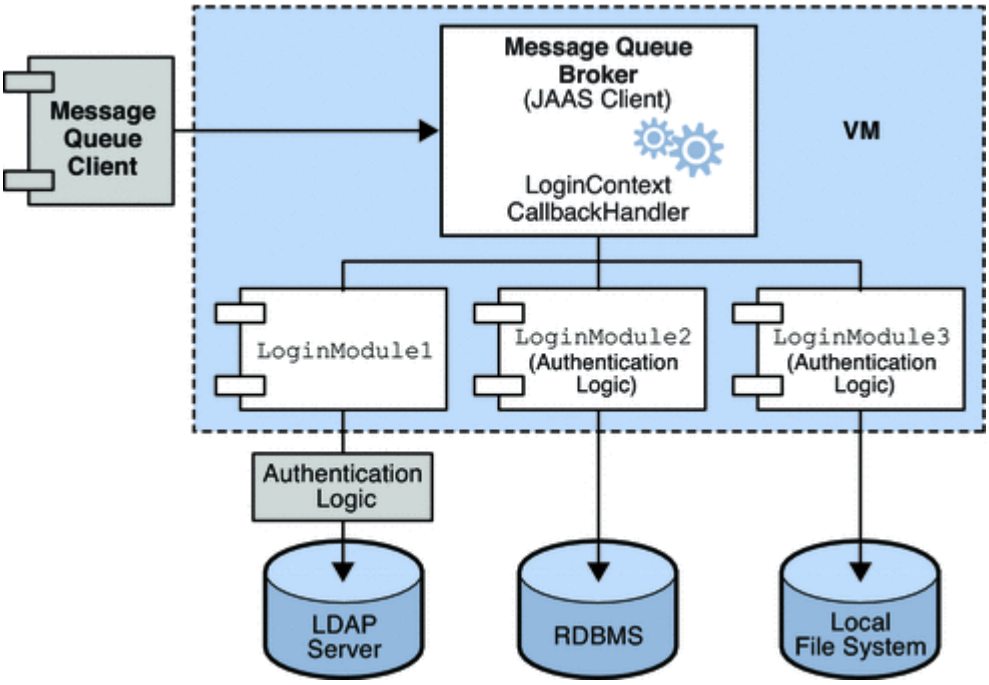

Figura 32. – Funcionamiento Básico de JAAS

El contexto JAAS para la función de autenticación debe ser *jdbcRealm* en la configuración de Glassfish, esto es porque el servidor de aplicaciones ya tiene un módulo de acceso construido con este nombre, se puede construir uno propio pero esto va más allá del dominio de este servicio.

El servicio de autenticación y autorización usa un JNDI, este debe estar previamente configurado en el servidor para permitir que la aplicación realice las operaciones correctamente. Es un recurso JDBC que emplea un conjunto de conexiones a la base de datos para permitir la extracción de datos que se necesita para realizar la autenticación.

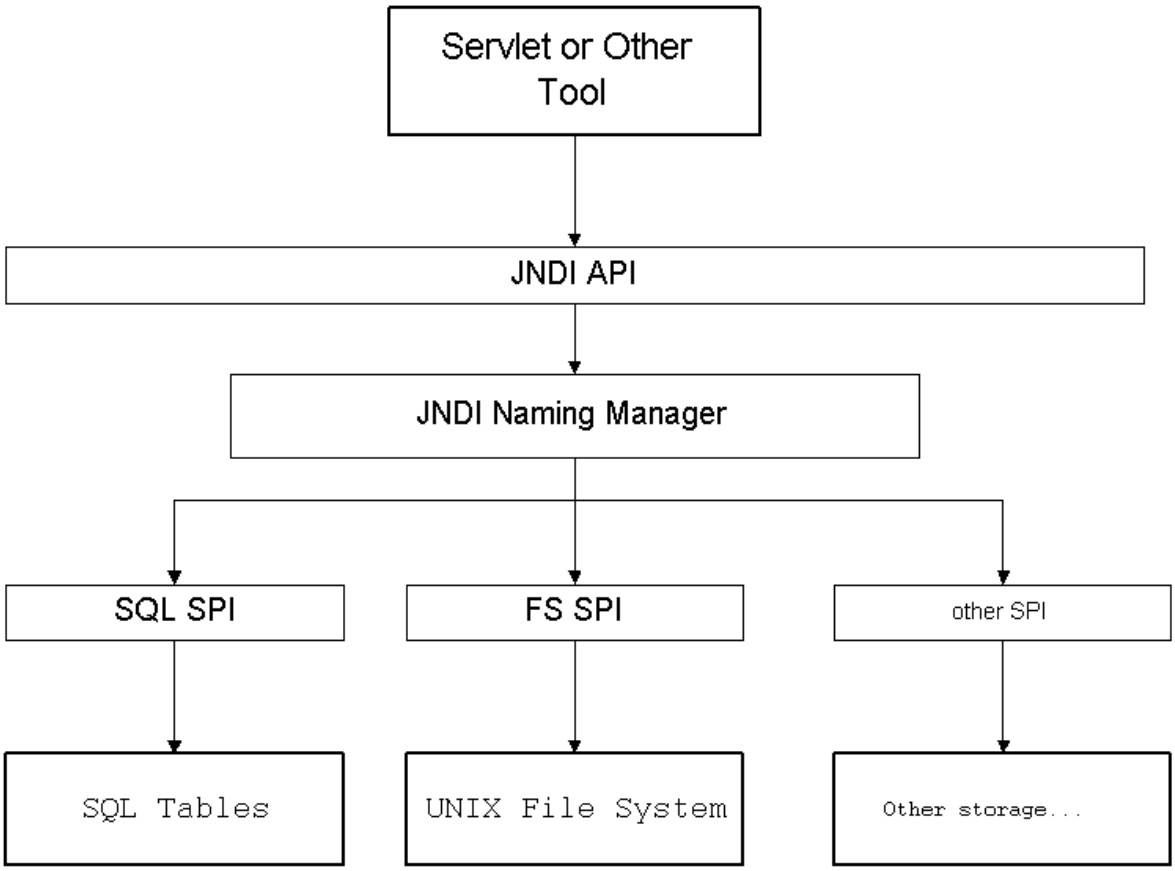

Figura 33. – Esquema básico de JNDI

Con esto se crea un dominio de seguridad en el servidor de aplicaciones que se encarga de poner los límites en la aplicación de acuerdo a los privilegios de los grupos de usuarios y maneja la sesión del usuario. En aras de mantener el sistema seguro este dominio está configurado para encriptar las contraseñas de los usuarios, de forma que esta información no está almacenada en texto plano en la base de datos sino encriptada bajo el algoritmo MD5, usado por defecto al momento de registrar los usuarios por el administrador en la plataforma. Cuando un usuario desea autenticarse en la plataforma, el dominio de seguridad del servidor de aplicaciones aplica el algoritmo de encriptación a los datos de entrada y compara los resultados con lo almacenado en la base de datos, permitiéndole o no el acceso a la aplicación.

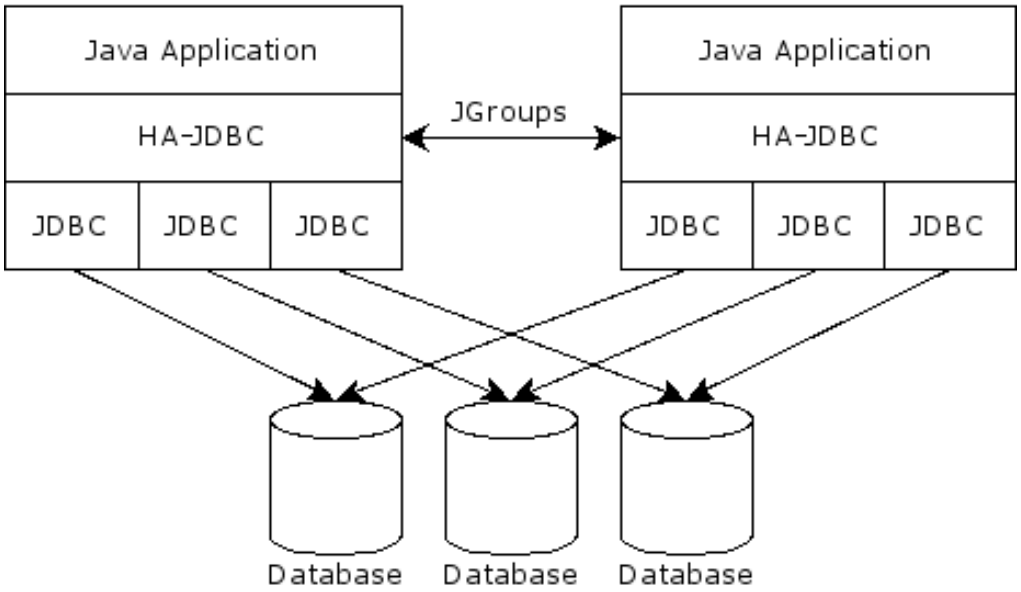

Figura 34. – Esquema básico de JDBC

En la aplicación web se debe agregar el dominio de seguridad encargado de lidiar con la seguridad. Para esto se debe modificar el archivo *web.xml*, añadiendo un método de autenticación por formulario, agregando las páginas que controlan el acceso así como los roles de seguridad que van a estar ligados con esta aplicación. Luego es necesario mapear los roles creados anteriormente con aquellos que residen en la base de datos, para esto se debe modificar el archivo *sun-web.xml*:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, I
3  <sun-web-app error-url="">
4    <context-root>/NGMapaJSF_5</context-root>
5    <security-role-mapping>
6      <role-name>Administrador</role-name>
7      <group-name>Administradores</group-name>
8    </security-role-mapping>
9    <security-role-mapping>
10     <role-name>Medico</role-name>
11     <group-name>Medicos</group-name>
12   </security-role-mapping>
13   <security-role-mapping>
14     <role-name>Usuario</role-name>
15     <group-name>Usuarios</group-name>
16   </security-role-mapping>
17   <security-role-mapping>
```

Figura 35. – Mapeo de los Roles

Y se debe crear una función en el *Backing Bean* de la página de acceso que maneje los datos de entrada y ejecute las solicitudes necesarias para autenticar el usuario:

```
public String login(){
    String message = "";
    String navto = "";
    HttpServletRequest request = (HttpServletRequest) FacesContext.getCurrentInstance().getExternalContext().getRequest();
    try {

        //Login via the Servlet Context
        request.login(username, password);

        //Retrieve the Principal
        Principal principal = request.getUserPrincipal();
```

Figura 36. – Función de Autenticación

Después de autenticar debemos emplear autorización, con esto podemos controlar el acceso a diferentes recursos, básicamente determina lo que un usuario puede ver y lo que no. Para esto se emplearon restricciones de seguridad por patrón URL, mapeadas en el archivo *web.xml*:

```
19      <security-constraint>
20      <web-resource-collection>
21          <web-resource-name>Admin Area</web-resource-name>
22          <url-pattern>/faces/jsfPages/admin/*</url-pattern>
23      </web-resource-collection>
24      <auth-constraint>
25          <role-name>Administrador</role-name>
26      </auth-constraint>
27      </security-constraint>
28      <security-constraint>
29      <web-resource-collection>
30           <web-resource-name>Medic Area</web-resource-name>
31           <url-pattern>/faces/jsfPages/medico/*</url-pattern>
32      </web-resource-collection>
33      <auth-constraint>
34          <role-name>Administrador</role-name>
35          <role-name>Medico</role-name>
36      </auth-constraint>
37      </security-constraint>
```

Figura 37. – Restricciones de Seguridad con Patrón URL

Estas impiden que un usuario que no tenga los privilegios para acceder a determinado recurso lo haga, escribiendo la URL de este en el navegador, ya que al intentarlo disparara la restricción aquí programada y la aplicación mostrara una página de error:

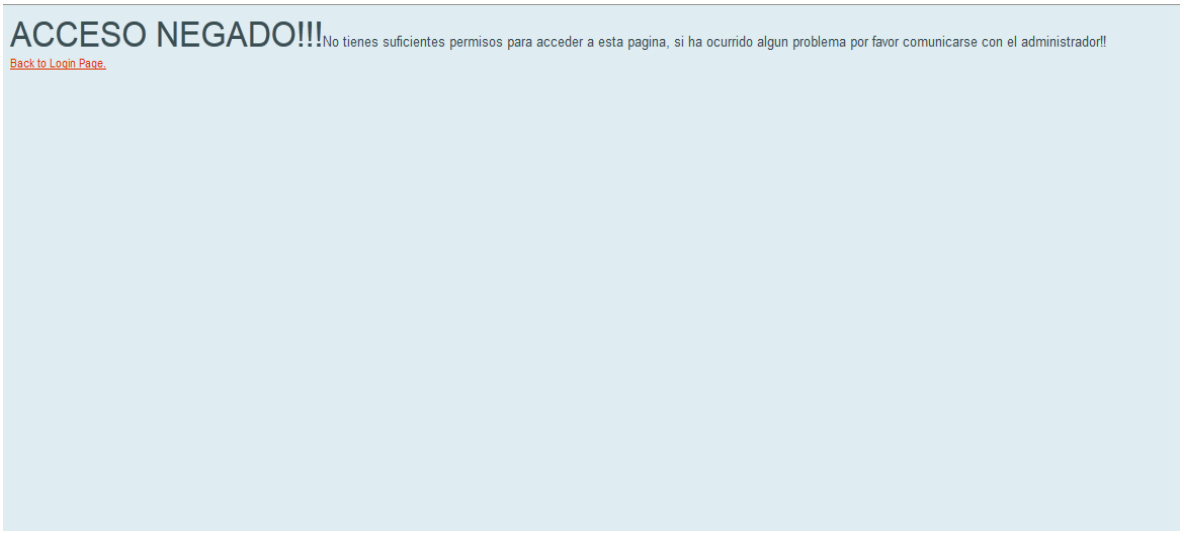

Figura 38. – Página de Acceso Denegado

Debido a que se realiza por patrón URL, los archivos de las vistas deben ser cuidadosamente colocados en carpetas respectivas para cada rol de usuario, con esto se permite el manejo de los permisos en todo el entorno de la aplicación, garantizando la seguridad e integridad del servicio:

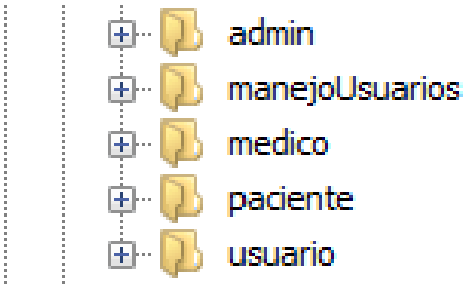

Figura 39. – Árbol de archivos de las vistas

La navegación y el direccionamiento de las vistas se mapea en el archivo *Faces-config.xml*. En este se programa a la aplicación como debe hacer el flujo de las vistas de acuerdo a determinados eventos disparados por datos que ingresa el usuario, como por ejemplo una vez el usuario se haya autenticado hacia a donde debe direccionar la aplicación. Este

archivo administra todos los *Facelets* usados por JSF2.0, también permite agregar nombres a los *backing Beans* para su invocación y agregar plantillas de diseño para simplificar la cantidad de código que debe ser escrito en la construcción de interfaces. En este caso particular fue usado para determinar la navegación de la aplicación:

```
15  <navigation-rule>
16          <display-name>Registration.xhtml</display-name>
17          <from-view-id>/jsfPages/admin/addUsuarios.xhtml</from-view-id>
18          <navigation-case>
19                  <from-outcome>success</from-outcome>
20                  <to-view-id>/jsfPages/admin/RegConfirmation.xhtml</to-view-id>
21          </navigation-case>
22  </navigation-rule>
23  <navigation-rule>
24          <display-name>Registration.xhtml</display-name>
25          <from-view-id>/jsfPages/admin/addUsuarios.xhtml</from-view-id>
26          <navigation-case>
27                  <from-outcome>failure</from-outcome>
28                  <to-view-id>/jsfPages/admin/addUsuarios.xhtml</to-view-id>
29          </navigation-case>
30  </navigation-rule>
31  <navigation-rule>
32      <display-name>Registration.xhtml</display-name>
33      <from-view-id>/jsfPages/admin/addUsuarios.xhtml</from-view-id>
34      <navigation-case>
35          <from-outcome>volver</from-outcome>
```

Figura 40. – Reglas de Navegación en *faces-config.xml*

Este compara valores retornados por funciones de los *Backing Beans* o *Actions* de los enlaces en las vistas y siguiendo la regla programada dirige la aplicación a la vista que haya solicitado el usuario. Esto mantiene la navegación de la aplicación fluida y hace de este un sistema seguro y controlado.

9.4 ARQUITECTURA DE LOS SERVICIOS

La aplicación del lado del cliente pertenece a un conjunto de servicios modulares, cuya tecnología brinda flexibilidad a la hora de añadir, editar o quitar funcionalidades a la aplicación en cualquier momento de su ciclo de vida. El servicio está enfocado para ser usado tanto por ingenieros de sistemas que administren y extiendan el rango de aplicaciones y servicios que puede ofrecer, así como médicos que monitoreen e interpreten las señales cardiacas enviadas por el cliente Android desde el dispositivo que

reside con el paciente y también al mismo paciente quien podría observar su condición accediendo a la aplicación Web desde cualquier lugar con internet.

Este servicio de monitoreo ECG hará parte de la red de servicios web geo-refenciados dispuesta por el grupo RadioGIS cuyo objetivo es la construcción de un gran servidor de aplicaciones y servicios basados en localización, para responder a la iniciativa "Vive Digital" del Gobierno Nacional cuya meta es promover y ampliar el uso de las tecnologías de información en todo el territorio nacional, aumentando la cobertura de los servicios de banda ancha, la calidad de las conexiones a internet y el acceso de la población a educación en el uso de dispositivos electrónicos que permiten el acceso a esta clase de servicios. Siguiendo esta iniciativa el servicio construido estará alojado en el servidor del grupo RadioGIS, razón por la cual se diseño para acoplarse con la arquitectura que maneja este servidor:

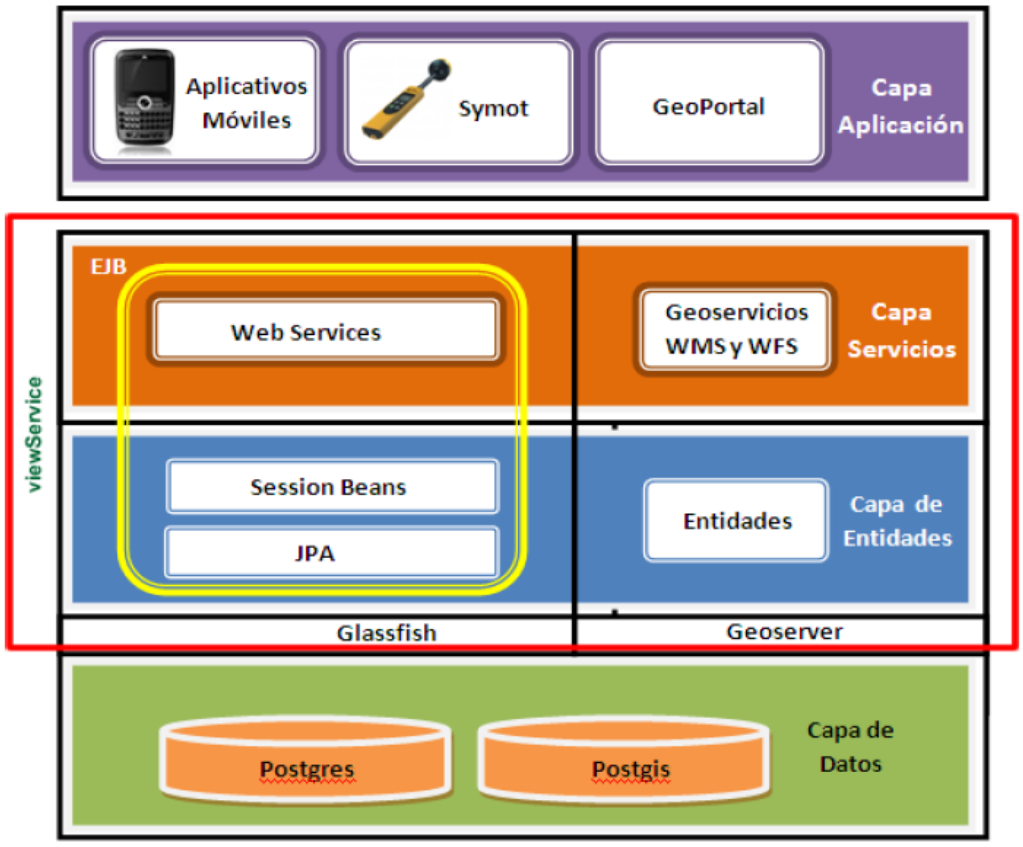

Figura 41. – Modelos de Componentes Alfanuméricos. Fuente RadioGIS

La capa de servicios hace parte de un modelo utilizado en la industria de software conocido como aplicación empresarial, que soporta funcionalidades y exigentes labores sobre datos alfanuméricos. Esta tecnología es la albergada en el servidor de aplicaciones GlassFish 3.0.1 sobre el cual se despliega un conector EJB que contiene la lógica del negocio.

La arquitectura del aplicativo web está basada en tecnología JavaServer Faces v2.0, este usa *facelets* y MVC (*Model View Controller)* para desplegar sus vistas y manejar las solicitudes y respuestas durante su ejecución.

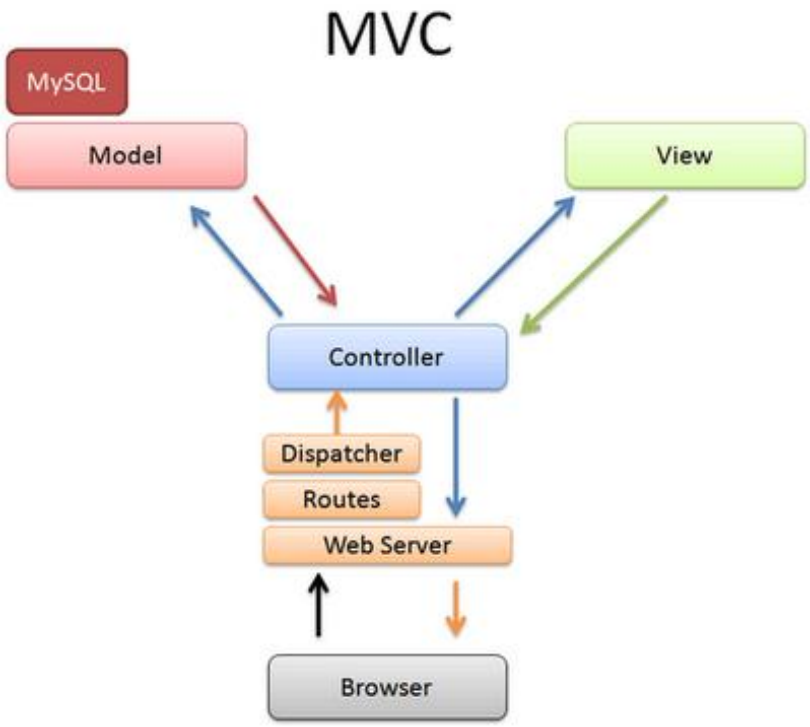

Figura 42. – Modelo Vista Controlador

El servicio también cuenta con una capa de entidad, en la cual se alojan todas aquellas clases que garantizan un mapeo objeto/relacional de los datos usando JPA (*Java Persistence API)* permitiendo que la información almacenada en la base de datos relacional pueda ser utilizada de forma transparente por los desarrolladores que usan objetos.

También existe una capa de servicios Web que cumplen con la función de exponer los métodos del negocio hacia la vista. Esta capa está dividida en servicios básicos y servicios especializados, en los básicos encontramos todas aquellas funcionalidades genéricas que pueden ser utilizadas y reutilizadas por los desarrolladores para construir nuevos servicios, como por ejemplo JAAS para la autenticación y autorización de usuarios o servicios para adaptación de contenidos. El otro grupo de servicios son los específicos que trabajando en asociación con otros servicios y la aplicación ofrece funcionalidades particulares para una aplicación cliente, por ejemplo el servicio construido para el cliente Android y el servicio REST encargado de la recepción de la transmisión hecha por el dispositivo cliente:

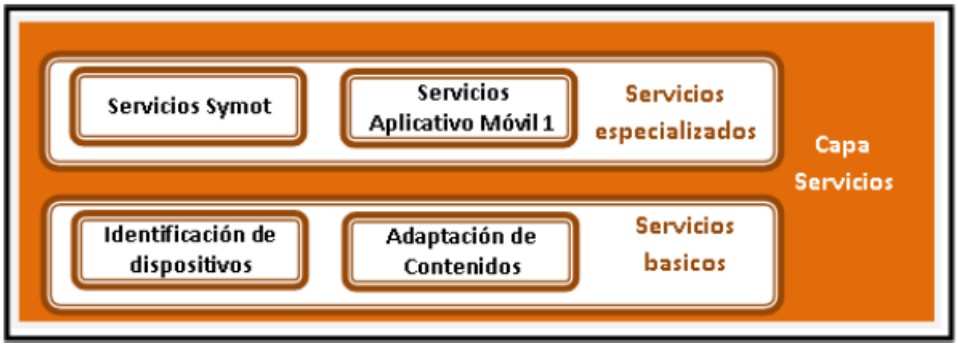

Figura 43. – Capa de Servicios. Fuente RadioGIS

9.5 IMPLEMENTACIÓN

El aplicativo Web utiliza tecnologías Java en todo su dominio de aplicación, y también incluye hojas de estilo en cascada (CSS) las cuales son completamente compatibles con JavaServer Faces y *Facelets*. Estas se implementaron en el IDE Netbeans 6.9.1 para el desarrollo del aplicativo web.

Cabe resaltar que el geo-posicionamiento utiliza el componente del mapa de Google Maps, adaptación de la API para JSF2.0 utilizando una librería de uso gratuito y *open-source*.

Con la librería *Gmaps4jsf* se puede integrar una instancia de Google Maps con funcionalidades propias de este servicio gratuito, solo solicitando una *API KEY* para la

versión JavaScript de la API en su versión 2, ya que la versión 3 no requiere una llave y esta no se encuentra integrada a JSF:

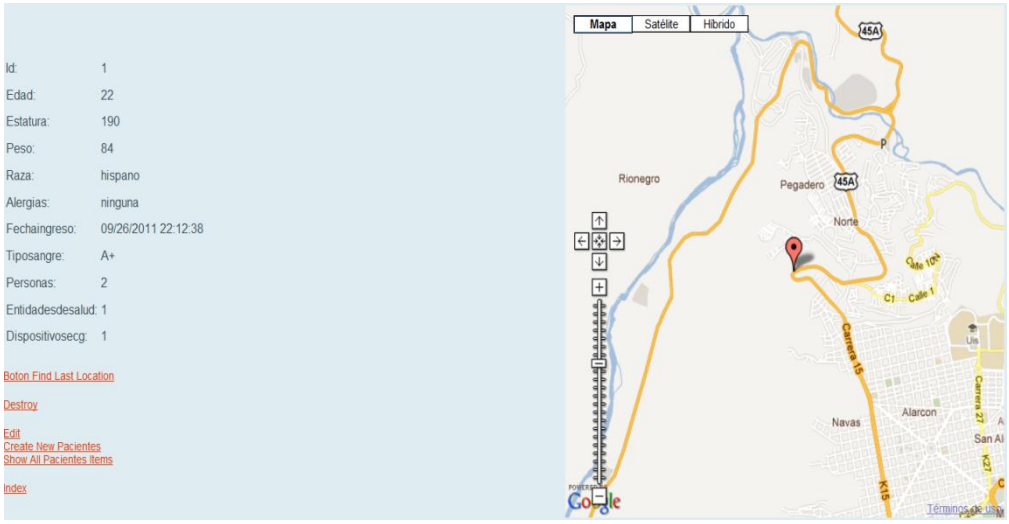

Figura 44. – Instancia de Google Maps API en la vista JSF

En este podemos movernos dentro del mapa, usar los controles dispuestos para acercar o alejar el mapa, así como cambiar entre capas del mapa provistas por Google Maps. Por defecto tiene 3 que son “Mapa” mostrada en la figura. También hay “Satelite” en la siguiente figura:

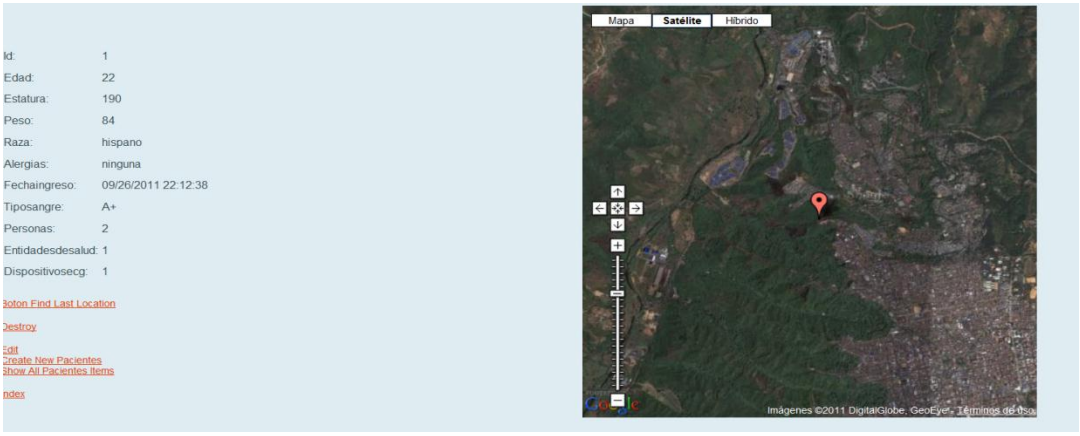

Figura 45. – Mapa de Satelite en Google Maps.

Por último tiene un mapa hibrido, que combina las 2 instancias anteriores y muestra una foto satelital de la zona con la nomenclatura, límites, calles y nombres de lugares sobre ella:

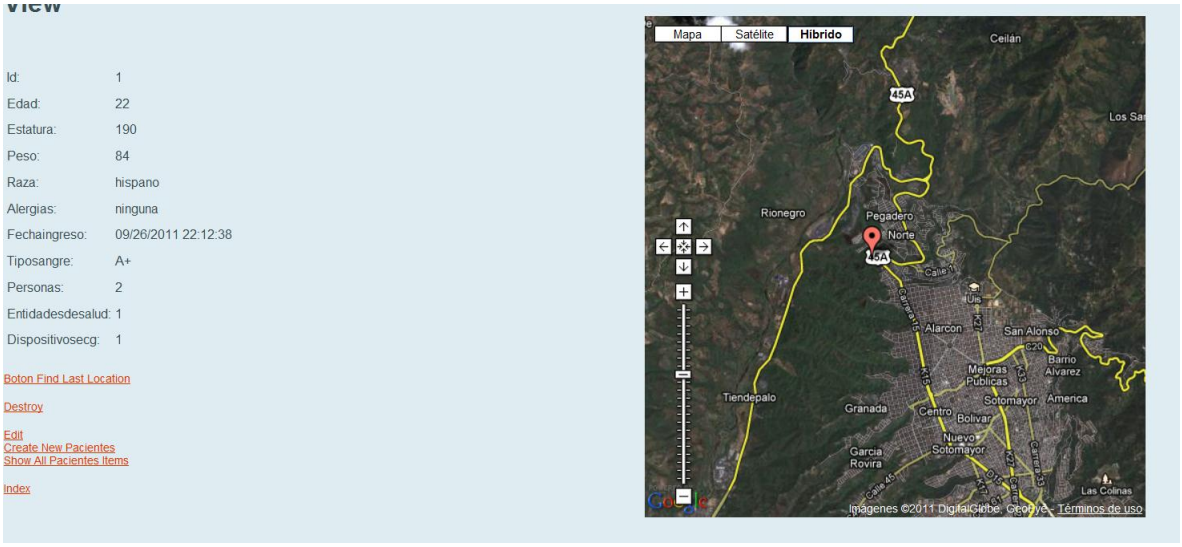

Figura 46. – Mapa Hibrido en Google Maps

Para iniciar una instancia de Google Maps en una aplicación JavaServer Faces 2.0, primero se incluyó la librería (o el *Jar)* que se encarga de lidiar con la solicitud de los recursos al servidor de Google.

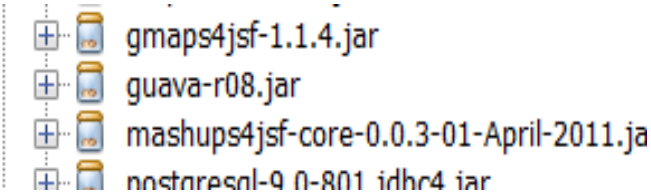

Figura 47. – Librería Gmaps4JSF versión 1.1.4

Luego se agregó el espacio de nombre para incluir elementos de la librería dentro de las vistas JSF y los *facelets*

```
1  <?xml version='1.0' encoding='UTF-8' ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transiti
3  <html xmlns="http://www.w3.org/1999/xhtml"
4        xmlns:m="http://code.google.com/p/gmaps4jsf/"
5        xmlns:h="http://java.sun.com/jsf/html">
6      <h:head>
```

Figura 48. – Declaración del Espacio de Nombre

Esta instancia de Google Maps está hecha en *JavaScript*, por esta razón no solo es muy liviana sino compatible en cualquier navegador disponible actualmente en el mercado,

pero debido a que es la versión 2 del API de Google Maps se debe incluir en el inicio de las vistas un *API KEY* correspondiente a la aplicación en la cual se despliega:

Figura 49. – Api Key de Google Maps.

Con esto ya configurado es posible agregar etiquetas dentro de la vista que determinan los elementos del mapa y donde estarán ubicados dentro de nuestra vista.

Figura 50. – Etiqueta para crear la instancia de Google Maps.

Como se puede ver en la figura, la librería que integra el API de Google Maps posee varias funciones extras, además de la instancia del mapa, con las cuales es posible agregar marcadores, etiquetas HTML, desplegar contenido dentro del mapa, trazar líneas y cargar capas hechas en Google Maps entre otras.

Los atributos dentro de la etiqueta *"map"* determinan parámetros como la ubicación del mapa, el centro, el acercamiento, su tamaño y las posiciones de los controles entre muchas otras.

Nótese que esta instancia se encuentra dentro de una etiqueta *Form*, lo cual permite agregar hojas de estilo al mapa, determinando el aspecto, color, posición, y las muchas otras opciones de estilos disponibles en CSS. Con esto se brinda total y fácil control sobre la vista que deben ver los usuarios.

En el servidor GlassFish se deben implementar las medidas de seguridad mencionadas anteriormente, para esto se debe acceder al panel de Administración y configurar los recursos JDBC que se emplean en la aplicación:

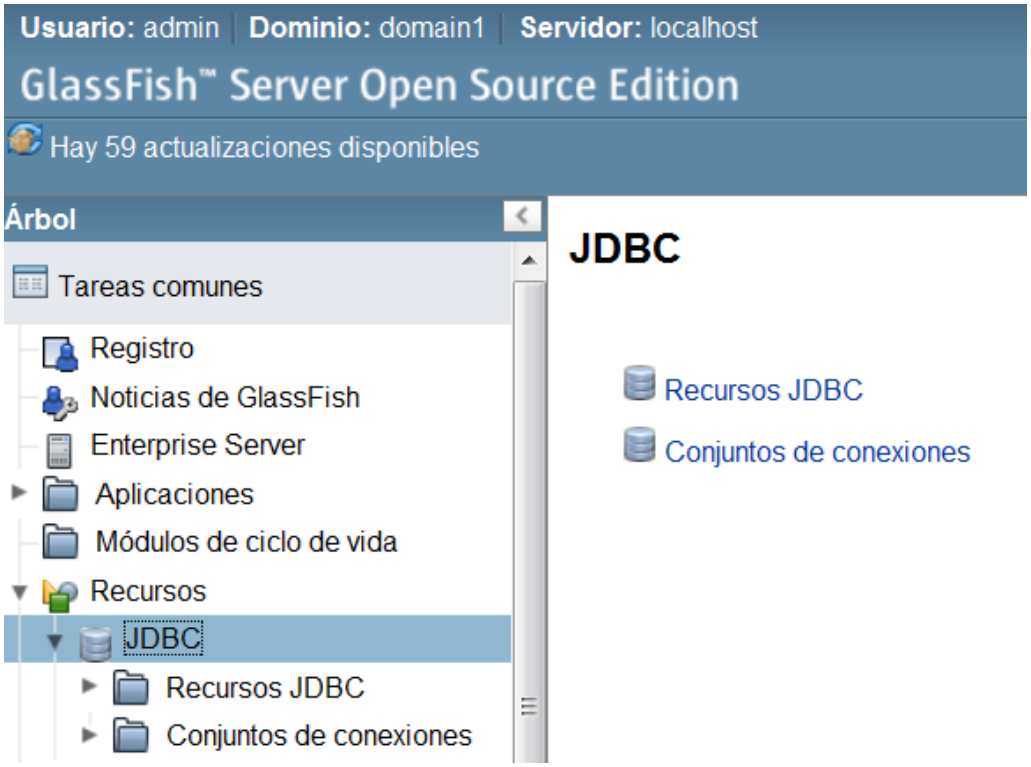

Figura 51. – Recursos JDBC en GlassFish 3.0.1

Con estos podemos determinar la conexión a la base de datos y luego asignarla a un recurso JNDI. También se asigna un dominio de Seguridad para la aplicación web, esto se realiza desde el panel de administración y utiliza los recursos mencionados

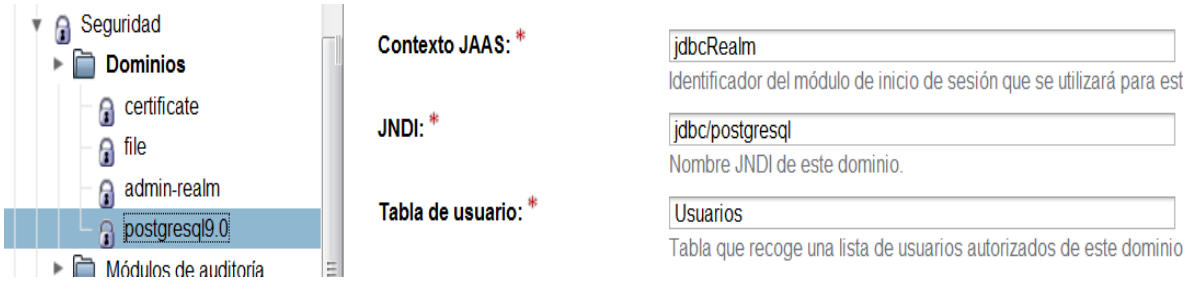

Figura 52. – Dominios de seguridad en GlassFish 3.0.1

Nótese el contexto JAAS asignado, este es una rutina integrada por defecto en el servidor. El nombre del JNDI fue el configurado en los recursos, también como las tablas en la DB.

10. PRUEBAS

La realización de pruebas consistió en verificar el funcionamiento de cada uno de los componentes software que conforman el servicio, tomando datos de prueba y realizando simulaciones de escenarios en el cual fuera necesario el flujo de información a todas las instancias de la aplicación.

Para realizar estas pruebas se tomaron datos de prueba de mediciones ECG almacenadas en el grupo GIIB, con estas se realizó el empaquetamiento en formato SCP-ECG en el cliente Android. Asimismo el dispositivo cliente se emulo usando Eclipse y el SDK de Android en un ordenador personal, esto debido a que no se cuenta con un dispositivo Android real ni un dispositivo que adquiera señales ECG portátil.

La aplicación web fue construida en un PC local, usando el servidor de aplicaciones GlassFish 3.0.1 para ejecutar los servicios web, vistas en JSF2.0 y conexiones a bases de datos necesarias para probar y verificar que la arquitectura planteada para el servicio funcione de la forma esperada. En este mismo equipo fue instalado el motor de base de datos PostgreSQL 9.0.1 en el cual se construyeron las tablas siguiendo el modelo entidad relación que se diseñó y fueron pobladas con datos ficticios para las pruebas, por supuesto siempre se mantuvo la integridad referencial de las tablas y los datos cumplían con las condiciones necesarias para mantener las pruebas lo más cercano posible a la realidad.

Una prueba total del servicio consiste básicamente de 4 partes, en las cuales cada componente de la aplicación debe realizar ciertas funciones específicas en las que se procesa información y finalmente es mostrada en el navegador, las partes son:

10.1 - Primera parte – Cliente Android

En esta primera parte se simula un dispositivo Android en el cual se encuentra instalado el software cliente construido especificamente para el desarrollo del proyecto. Consta de una sencilla interfaz grafica prototipo, el cual muestra datos del paciente y las coordenadas en las cuales esta el dispositivo (asumiendo que se esta usando el localizador integrado en los dispositivos mobiles de ultima tecnologia como *SmarthPhones y Tablets*):

Figura 53 – Prototipo de Interfaz Android

En esta parte de la prueba se ingresan al cliente software los datos del paciente y un conjunto de datos ECG que fueron provistos por el grupo GIIB, tomados con el *BIOPAC* e indicando el número de guías de la señal. Con estos datos el software realizará la escritura del archivo en formato SCP-ECG cumpliendo con los lineamientos de este estándar en su versión 2.2.

Luego, el cliente ejecuta una operación asíncrona en otro hilo del sistema operativo el cual se encarga de conectarse con el servicio web REST construido en la aplicación que reside en el servidor GlassFish y una vez la conexión sea satisfactoria procede a transmitir el paquete en forma de Bits. Terminada la transmisión el software permanece a la espera de nuevos datos de entrada para repetir el proceso anterior.

10.2 - Segunda Parte – Servicio Web REST de recepción y persistencia de Datos.

El servicio REST recibe la transmisión de bytes hecha por el cliente Android y reconstruye el paquete SCP-ECG con los datos capturados, luego, extrae la información de este paquete para reconocer a cual paciente pertenecen y procede a almacenar los datos en la

tabla de “Almacén SCP”. En esta tabla se encuentra el historial de todas las lecturas tomadas por los dispositivos activos para el servicio y con esta se realiza la localización del paciente. Los datos se verificaron en la base de datos para comprobar que toda la transmisión y persistencia funcionara correctamente, manteniendo la integridad de la base de datos y las relaciones correspondientes.

10.3 Tercera Parte – Monitoreo de los Datos enviados en la Aplicación

La aplicación JSF 2.0 se conecta a la base de datos mediante EJB en el servidor de aplicaciones GlassFish y realiza transacciones para obtener los datos relevantes. Accediendo al panel de monitoreo del médico se seleccionó el paciente con el cual se simulo el caso de estudio:

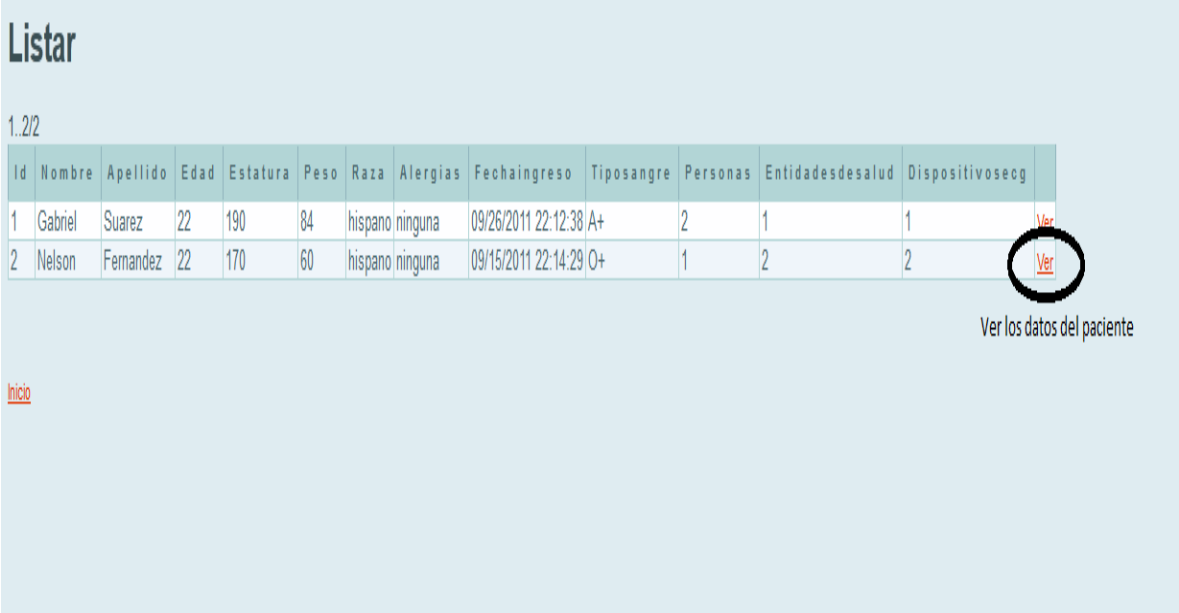

Figura 54. – Prueba de Monitoreo

Al seleccionar “Ver” la aplicación solicita los datos guardados por los pasos anteriores y devuelve otra vista con los datos del paciente y una instancia de Google Maps en la cual se ubicará la posición que fue enviada por el cliente Android.

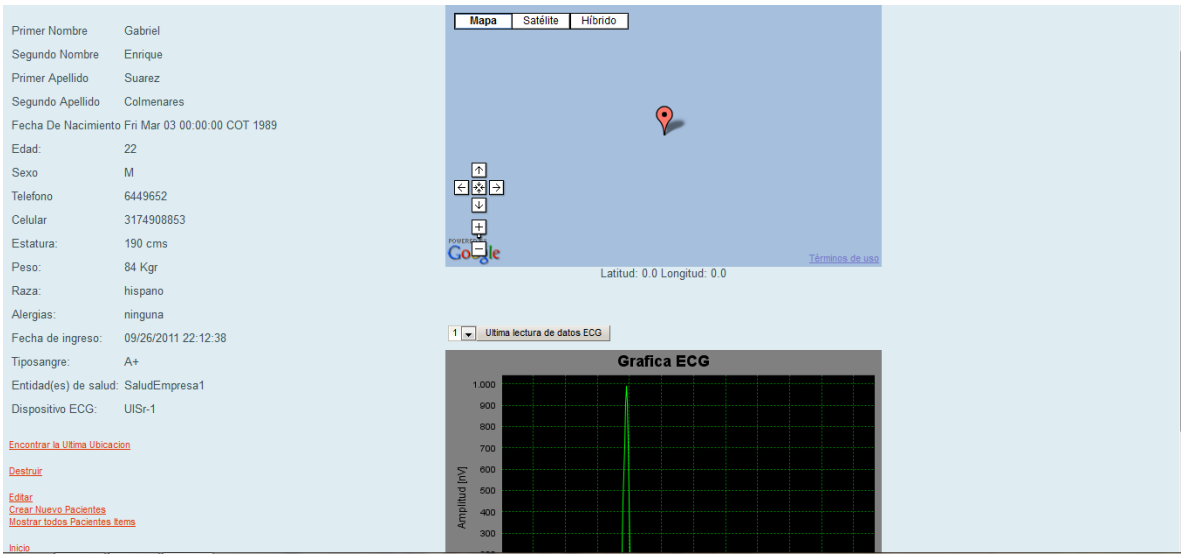

Figura 55. – Datos del Paciente Monitoreado

Como se ve en la figura, inicialmente solo se ven los datos personales y el mapa no apunta a ningún punto en específico y muestra una gráfica por defecto de la condición cardiaca, para generar esta información están los botones señalados.

Escogiendo el botón “Encontrar Ultima Ubicación” se envía una solicitud al servidor pidiendo las últimas coordenadas registradas del paciente monitoreado, la respuesta son la latitud y longitud, que se ubican en el mapa con un marcador:

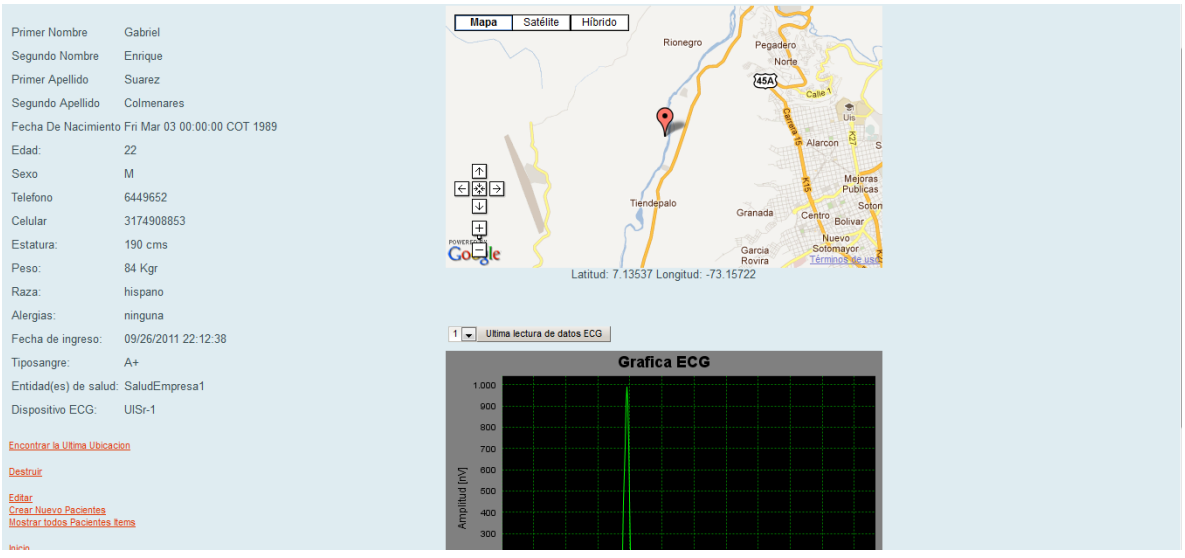

Figura 56. – Paciente ubicado durante monitoreo

Ahora, para observar el cuadro cardiaco del paciente en cuestión se selecciona el botón "Última lectura de datos ECG" este envía una solicitud al servidor por el último archivo ECG almacenado para el paciente en la derivación escogida, y mostrará la gráfica así:

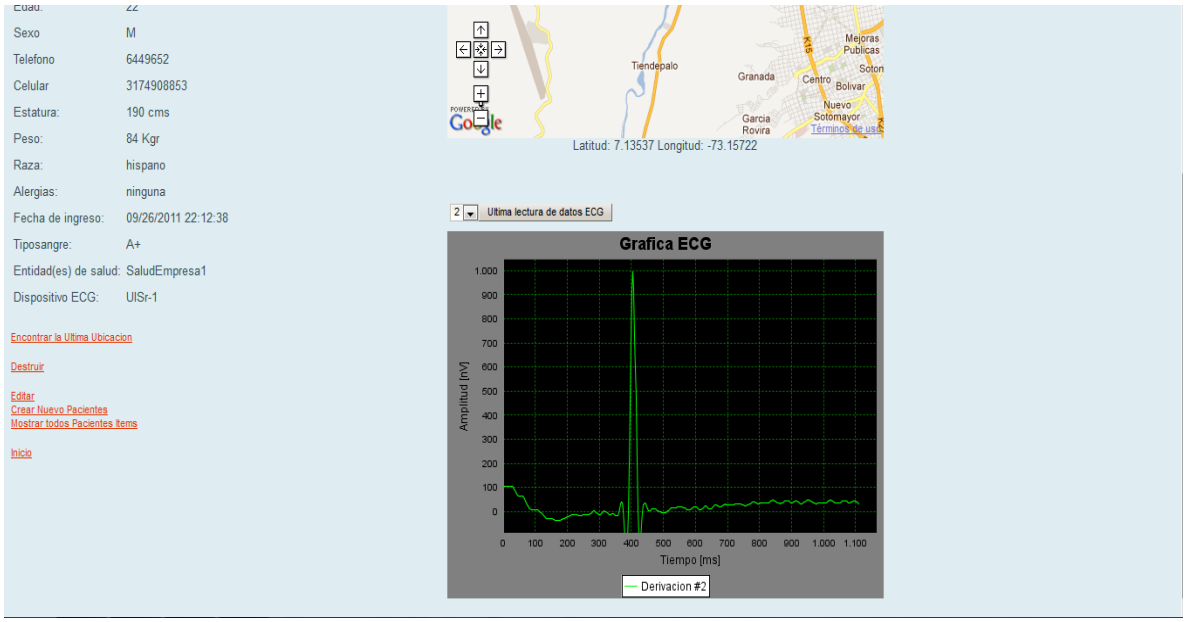

Figura 57. – Grafica ECG generada en el monitoreo

Debido a que el servicio se encuentra en la etapa de prototipo se ahorró en gastos computacionales como lo son usar herramientas dinámicas como *Ajax* o disparadores automáticos para el monitoreo constante, además de que son objetos de mucho cuidado puesto que su uso incorrecto pueden inundar al servidor de solicitudes y datos inservibles.

10.4 Cuarta Parte – Prueba de encriptación.

Esta parte consiste en realizar el registro de un usuario nuevo a la base de datos, comprobar que el *script* de encriptación "MD5" está realizando la tarea correctamente y que en realidad las contraseñas almacenadas en la base de datos no se encuentran en texto plano, sino encriptados con este algoritmo.

Se accede a la aplicación con el rol de Administrador, puesto es este el único que hace registros al servicio en su versión prototipo:

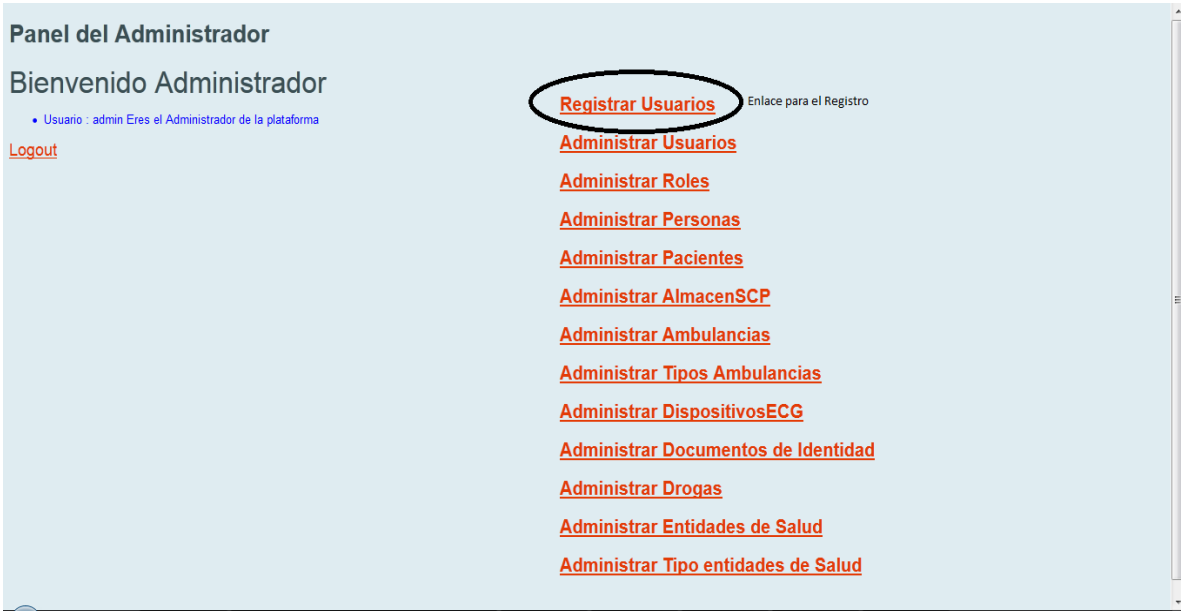

Figura 58. – Enlace de Registrar Usuarios.

Después se muestra un formulario en el cual se deben ingresar todos los datos de usuario, nótese el campo contraseña con su respectivo validador:

Figura 59. – Formulario de Registro de Usuarios

El validador funciona comparando las 2 entradas ingresadas en el campo contraseña, si son iguales las acepta, si no, devuelve la excepción al formulario de inscripción:

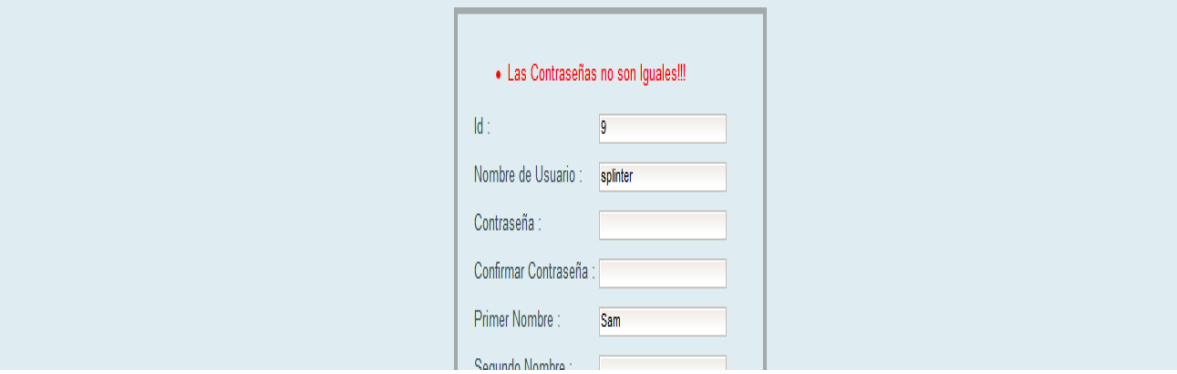

Figura 60. – Excepción al validar la contraseña

La aplicación también realiza validaciones en cuanto al formato de la fecha, campos obligatorios entre otros. Una vez validadas las entradas se permite el registro del usuario en la base de datos:

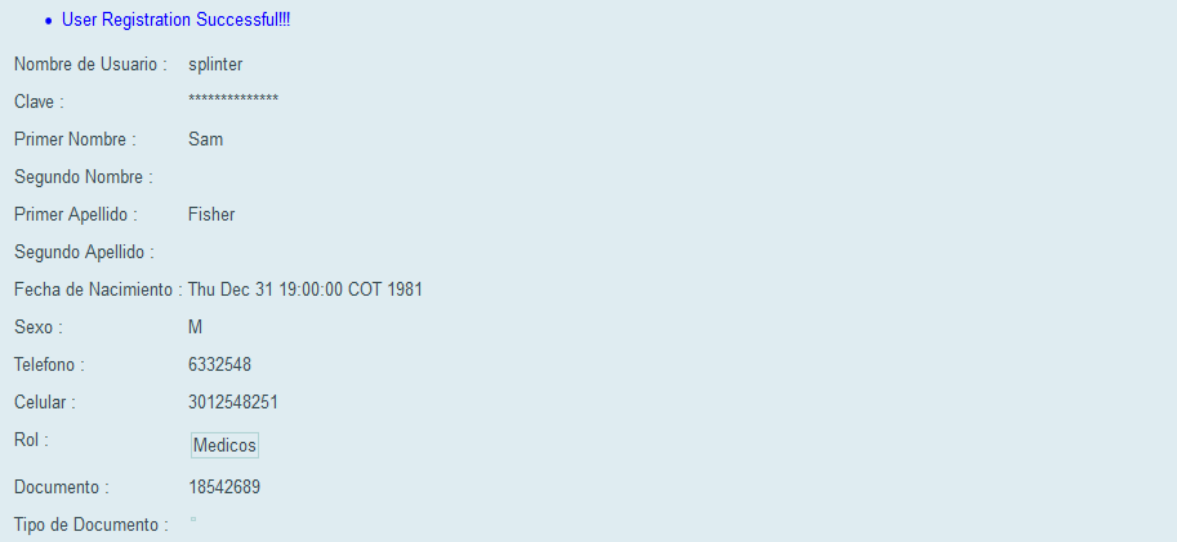

Figura 61. – Confirmación de Registro

Una vez registrado se devuelve una vista con los datos que se ingresaron, esto para confirmar que la información del registro es correctos.

Ahora se debe verificar en la base de datos si efectivamente la encriptación declarada se realizó correctamente y la contraseña no se encuentra almacenada en texto plano:

	userid [PK] bigint	username character varying(255)	clave character varying(64)
1	1	admin	21232f297a57a5a743894a0e4a801fc3
2	2	medico	652044ac6e008761b3e6141748a99880
3	3	usuario	f8032d5cae3de20fcec887f395ec9a6a
4	4	gesc	647431b5ca55b04fdf3c2fce31ef1915
5	5	nifs	a4e360681676c770121e891f8c407572
6	6	celso	d9814c5096326e6cf6dbec0020553bf4
7	7	james	ecda8ff7933831de47cded3bb238b613
8	9	splinter	2163c17ca50bb0fcd227460c4f682a40

Figura 62. – Resaltada, la contraseña ingresada con MD5

En la figura se muestra la tabla de los usuarios, se puede apreciar que la contraseña fue encriptada y guardada efectivamente con el algoritmo MD5, implementado al momento de hacer el registro en la base de datos:

```
45        user.setUserid(registrationBean.getId());
46        user.setUsername(registrationBean.getUsername());
47        user.setClave(util.hash(registrationBean.getPassword()));
48
49
```

Figura 63. Encriptación de la contraseña

Esta le indica a la aplicación que realice la función de encriptación hecha en otro *bean* llamado "*util*", el cual emplea funciones integradas en JAVA para realizarla automáticamente.

Aquí se muestra la parte responsable de realizar la encriptación de la contraseña, localizada en una clase java dentro de la aplicación web:

```
22  public static String hash(String string){
23              try {
24                      //Create MessageDigest and encoding for input String
25                      MessageDigest digest = MessageDigest.getInstance("MD5");
26                      byte[] hash = digest.digest(string.getBytes("UTF-8"));
27
28                      //Hash the Input String
                        StringBuffer sb = new StringBuffer();
30              for (int i = 0; i < hash.length; i++) {
31                sb.append(Integer.toString((hash[i] & 0xff) + 0x100, 16).substring(1));
32              }
33              return sb.toString();
34              } catch (NoSuchAlgorithmException e) {
                        e.printStackTrace();
36              } catch (UnsupportedEncodingException e) {
                        e.printStackTrace();
38              }
39
40              return null;
41      }
```

Figura 64. – Código para la encriptación de la Contraseña

CONCLUSIONES

- En el campo de las telecomunicaciones, las nuevas tecnologías nos permiten realizar aplicaciones avanzadas con el fin de mejorar la calidad de vida, que es en sí una de las razones principales por las que existe la tecnología, mediante la integración y construcción de servicios para beneficiar a la población en general.

- En la actualidad los LBS permiten crear, interactuar y brindar nuevas formas de llevar información al usuario, mostrar una nueva perspectiva en el campo de las TIC, además de dar la posibilidad de mejorar servicios ya creados pero que fallan en agilidad y rapidez.

- El prototipo de servicio web orientado a telemedicina para monitoreo de ECG cumple con las funciones básicas y esenciales que se plantearon como objetivos durante la planeación del proyecto, se puede notar una integración de muchas tecnologías que se han desarrollado en los últimos años para lograrlo y es de vital importancia seguir los desarrollos e investigación en estos campos, puesto que son el futuro de las comunicaciones y servicios.

- Siguiendo los planes del Gobierno Nacional en su proyecto “Vive Digital” este prototipo de servicio se plantea como una clara opción de tanto investigación como desarrollo, apuntando a un mejoramiento de los servicios de salud a pacientes de enfermedades cardiacas mediante el uso de las TIC, buscando seguir la línea de las redes de próxima generación para extender su rango de acción y funciones

- Es de gran importancia seleccionar las tecnologías adecuadas a la hora de buscar la mejor solución a un problema utilizando las TIC de última generación, integrar plataformas, aplicativos y servicios es una tarea de mucho cuidado, además de tener un claro conocimiento sobre todas las funcionalidades de cada uno de los elementos que integran se debe conocer los límites y restricciones que implican, para lograr un desarrollo sólido, robusto, seguro y estable.

RECOMENDACIONES

- Para realizar pruebas más avanzadas con el prototipo del servicio aquí construido es necesario la adquisición tanto de un dispositivo Android como de la construcción o compra de un aparato portable para tomar lecturas de ECG, juntando estos dos elementos se puede hacer una prueba real del flujo de datos del servicio.

- Incluir más trabajos de investigación enfocados en mejorar aspectos claves del servicio como actualización en tiempo real, ampliar el rango de servicios ofrecidos por la aplicación, desarrollo de aplicaciones Android más eficientes que permitan monitoreo más detallado y especializado de signos vitales y trabajo con servicios web que brinden una transmisión de datos más eficiente al mismo tiempo que precisa y oportuna.

- Se recomienda incluir en el servicio un aplicativo de diagnósticos sobre las lecturas de ECG, que reconozca patrones de fallas cardiacas y así activar mecanismos de automáticos de emergencia para la atención inmediata al paciente por parte de personal especializado esto mejoraría enormemente el servicio prestado además de ahorrar costos en el personal que monitorea los datos recibidos.

- Crear espacios de socialización y fomentar la investigación y el desarrollo de servicios de telemedicina orientados a las redes de nueva generación mostrando proyectos como este para incentivar la implementación de nuevos y mejores servicios que permitan no solo mejorar la calidad de vida de la población aprovechando la tecnología actual sino también dar a conocer el nombre de la Universidad Industrial de Santander por sus desarrollos en este campo tanto a nivel nacional como internacional

- Difundir de manera masiva los servicios prestados y en desarrollo del Geoportal del grupo RadioGIS con el fin de darlo a conocer y generar intereses de los sectores del país para generar más inversión en la investigación y desarrollo.

BIBLIOGRAFIA

YANG, Daoqi. Java Persistence with JPA. Outskirts Press, Inc. Denver, Colorado, 2010

BOOCH, Grady. Analisis y Diseño Orientado A Objetos Con Aplicaciones. Addison-Wesley, 1996.

BURKE, Bill y MONSON-HAEFEL, Richard. Enterprise Java Beans 3.0. 5 ED. O'Reilly. 2006

LEE RUBINGER, Andrew y BURKE, Bill. Enterprise Java Beans 31. 6 ED. O'Reilly. 2010

DE CAPUA, Claudio, **ANTONELLA Meduri**, y **MORELLO, Rosario**, "A Smart ECG Measurement System Based on Web-Service-Oriented Architecture For Telemedicine Applications", IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, VOL. 59, NO. 10, pp 2530 – 2538, OCT 2010

HEE HAN, Chang, **YOUN**, **Chan-Hyun**, **JUNG, Wooram**, "Web-based system for Advanced Heart Disease Identification using Grid Computing Technology", 21st IEEE International Symposium on Computer-Based Medical Systems, June 17-19, 2008

Health Informatics, Standard Communication Protocol – Computer-Assisted Electrocardiography, STD Version 2.2, Feb 2007.

Ministerio de Tecnologías de la Información y las Comunicaciones, "Estrategias, Objetivos, Dimensiones Plan Vive Digital", Febrero 2011.

El Espectador.com, "*La enfermedad cardiovascular es la principal causa de muerte en Colombia".* 17 MAR 2009.

About OpenECG, [en línea] http://www.openecg.net/

Android Developer's Guide, [en línea] http://developer.android.com/guide/index.html

RESTful Web Services, [en línea] http://www.oracle.com/technetwork/articles/javase/index-137171.html

BIOSIGNA, Medical Diagnostics, [en línea] http://www.biosigna.de/

Open ECG. Comunidad de soporte para el estándar SCP-ECP. [En línea] http://www.openecg.net/tutorial1/index.html

Eclipse IDE, [en línea] http://www.eclipse.org/org/

Netbeans Community, [en línea] http://netbeans.org/community/index.html

Android Developers, [en línea] http://developer.android.com/index.html

PostgreSQL. 2011 PostgreSQL-es, [en línea]
http://www.postgresql.org.es/sobre_postgresql

JavaServer Faces Technology, [en línea]
http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

Java Naming and Directory Interface (JNDI), [en línea]
http://www.oracle.com/technetwork/java/jndi/index.html

Java Authentication and Authorization Service, [en línea]
http://download.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html

Java DataBase Connectivity API, [en línea]
http://download.oracle.com/javase/7/docs/technotes/guides/jdbc/

The Java EE 6 Tutorial, Part IV Persistence JPA, [en línea]
http://download.oracle.com/javaee/6/tutorial/doc/bnbpy.html

BASSEY-DUKE, Umoh. 2011. User Authentication And Authorization Using JAAS and Servlet 3.0 Login, [en línea] http://www.greenkode.com/2011/09/user-authentication-and-authorization-using-jaas-and-servlet-3-0-login/

Gmaps4JSF, Project Home, [en línea] http://code.google.com/p/gmaps4jsf/

Extreme Programming, Documentacion, [en línea]
http://www.extremeprogramming.org/