DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA EN ELECTRÓNICA REDES Y COMUNICACIÓN DE DATOS

TRABAJO DE TITULACION PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA REDES Y COMUNICACIÓN DE DATOS

TEMA: DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE ESTACIONES GSM, UMTS, LTE CON EQUIPOS NOKIA DE LA RED CELULAR MÓVIL DE LA OPERADORA MOVISTAR A NIVEL NACIONAL

AUTORES: ARELLANO VALLEJO, DAVID ALEJANDRO

SISALIMA ORTEGA, DIEGO ALEJANDRO

DIRECTOR: ING. VEGA MUÑOZ, CHRISTIAN NESTOR

SANGOLQUÍ

AGOSTO 2016

CERTIFICADO

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

INGENIERÍA EN ELECTÓNICA EN REDES Y COMUNICACIÓN DE DATOS

CERTIFICACIÓN

Certifico que el trabajo titulado "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE ESTACIONES GSM, UMTS, LTE CON EQUIPOS NOKIA DE LA RED CELULAR MÓVIL DE LA OPERADORA MOVISTAR A NIVEL NACIONAL", realizado por el Sr. David Alejandro Arellano Vallejo y el Sr. Diego Alejandro Sisalima Ortega, ha sido guiado y revisado periódicamente, adicionalmente se ha analizado en su totalidad por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas – ESPE.

Debido a que se trata de un trabajo de investigación se recomienda su publicación.

Sangolquí, 16 Agosto del 2016

Atentamente,

Ing. Christian Vega
TUTOR DEL PROYECTO

DECLARACION DE RESPONSABILIDAD

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

INGENIERÍA EN ELECTÓNICA EN REDES Y COMUNICACIÓN DE DATOS

AUTORÍA DE RESPONSABILIDAD

DAVID ALEJANDRO ARELLANO VALLEJO
DIEGO ALEJANDRO SISALIMA ORTEGA

DECLARAMOS QUE:

El proyecto de grado denominado "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE ESTACIONES GSM, UMTS, LTE CON EQUIPOS NOKIA DE LA RED CELULAR MÓVIL DE LA OPERADORA MOVISTAR A NIVEL NACIONAL", ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaramos que este trabajo es de nuestra autoría, en virtud de ello, nos declaramos responsables del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, 16 Agosto del 2016

David Alejandro Arellano Vallejo
C.C 1719022699

Diego Alejandro Sisalima Ortega
C.C 0704406396

AUTORIZACIÓN

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

INGENIERÍA EN ELECTÓNICA EN REDES Y COMUNICACIÓN DE DATOS

AUTORIZACIÓN

Nosotros, David Alejandro Arellano Vallejo y Diego Alejandro Sisalima Ortega autorizamos a la Universidad de las Fuerzas Armadas – ESPE la publicación en la biblioteca virtual de la institución, el trabajo titulado "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE MONITOREO DE ESTACIONES GSM, UMTS, LTE CON EQUIPOS NOKIA DE LA RED CELULAR MÓVIL DE LA OPERADORA MOVISTAR A NIVEL NACIONAL", cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y autoría.

Sangolquí, 16 Agosto del 2016

David Alejandro Arellano Vallejo
C.C 1719022699

Diego Alejandro Sisalima Ortega
C.C 0704406396

DEDICATORIA

Dedico el logro de este proyecto de tesis primeramente a Dios que ha estado conmigo a lo largo de todo el transcurso de mi vida, el cual me ha sabido cuidar y brindarme su apoyo, fortaleza y cobijo ante las dificultades y adversidades de la vida, a mis padres que desde el primer día de mi vida han velado por mi bienestar, me han dado su apoyo incondicional tanto en el ámbito educativo, profesional y humano, brindándome su entera confianza durante todos los peldaños que presenta la vida y que gracias a su esfuerzo y sacrificio he logrado superar un objetivo de mi vida, finalmente a mi hermano que a pesar de la distancia, siempre escuche sus palabras de aliento y superación.

David

Dedico este trabajo a mi familia, por su gigantesca paciencia durante todo este trayecto estudiantil para conmigo, por su apoyo incondicional y su infaltable presencia para brindarme su consejo y soporte cuando lo necesitaba.

Diego

AGRADECIMIENTO

Quiero agradecer de la manera más profunda y de todo corazón a mis padres, por ser quienes han dedicado su vida entera en formar a un profesional con valores y responsabilidad social, enseñándome a levantar de las caídas y que sin esfuerzo no hay recompensa.

Agradezco de forma muy emotiva a los maestros, maestras, amigos, compañeros, colegas y familiares que durante toda mi vida estudiantil y humana, me brindaron sus enseñanzas y experiencias para afrontar el diario vivir dentro y fuera de las aulas, cuyos consejos fueron de gran apoyo para superar las adversidades que se presentaron durante mi camino estudiantil.

Al Ing. Christian Vega, director de este proyecto, por su sabia guía y apoyo durante la realización de este trabajo.

David

Agradezco de manera especial a nuestro tutor, que supo darnos su apoyo y guía para la culminación de este proyecto, a nuestros colegas de trabajo que también nos han ayudado para la implementación de nuestras ideas; a todos los docentes que supieron dejarnos una chispa de conocimiento a medida que avanzamos en nuestra carrera cruzando por diversas aulas. Agradezco también a nuestra institución universitaria, ahora conocida como Universidad de las Fuerzas Armadas, la cual nos facilitó el acceso a un mar de conocimientos sin los cuales ahora no podríamos desempeñarnos en este campo profesional tan apasionante.

Finalmente, pero no menos importantes, agradezco a mis padres por ser la guía fundamental para la consecución de esta meta

Diego

INDICE DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

GLOSARIO

GSM. Global System for Mobile Communications, es una tecnología inalámbrica de segunda generación que presta servicio de voz, mensajería de texto corto y datos a menor velocidad.

EDGE. Enhanced Data Rates for GSM Evolution (Tasas de Datos Mejoradas para la evolución de GSM)

UMTS. Universal Mobile Telecommunications System, tecnología inalámbrica de tercera generación, entre sus principales características están: capacidades multimedia, velocidad de datos elevada y mayor calidad en llamadas de voz.

LTE. Long Term Evolution, es una tecnología inalámbrica de cuarta generación que se caracteriza por la alta velocidad en la transmisión y recepción de datos con baja latencia.

Proxy. Mecanismo que permite compartir una única conexión a una red externa entre todos los puertos de una red local.

HTTP. En inglés Hypertext Transfer Protocol. Protocolo de Transferencia de Hipertexto.

URL. Uniform Resource Locator. Sistema de direccionamiento estándar de archivos y funciones en Internet, especialmente en la WWW.

DHTML. Dynamic HTML

BGP. Border Gateway Protocol

Servidor. Es un elemento de software, aplicación que permite atender las peticiones de un cliente, para facilitar el acceso a recursos y aplicaciones de la red.

Programa. Es un software que se crea mediante una secuencia de instrucciones o códigos para la realización de multitareas o tareas específicas.

Base de Datos. Es un conglomerado de información que contiene diversa cantidad de datos categorizados de distinta manera, los mismos que comparten entre sí un vínculo común, los cuales son almacenados sistemáticamente para su posterior uso.

Estación Base. Son equipos de radio frecuencia los cuales se encargan de mantener el enlace radioeléctrico entre la estación móvil y la estación de control de servicio durante las comunicaciones de voz y de datos.

RESUMEN

La operadora de telefonía móvil Telefónica Ecuador hoy en día necesita atender y solucionar los inconvenientes técnicos de su red en el menor tiempo posible, con el fin de mantener a los usuarios con los servicios de voz y datos de forma permanente. La red de telefonía móvil es controlada a través del NOC Externo, el cual es el encargado de realizar el monitoreo constante 24 horas al día, 7 días a la semana, 365 días al año, cuya tarea primordial es atender e interpretar alarmas que se despliegan en los distintos gestores de monitoreo, actualmente se tiene desplegado por todo el país equipos de acceso y transmisión de la marca Nokia los cuales son monitoreados por el gestor NetAct, dicha herramienta no cuenta con todos los detalles que pueden ser valiosos para un diagnóstico temprano de los problemas que pueden suscitarse en la red móvil. Para solucionar este inconveniente se desarrolló una nueva herramienta de software para la gestión, control y monitoreo de afectaciones de fuera de servicio de estaciones base, en tecnologías 2G, 3G y 4G de equipos Nokia. La nueva plataforma de gestión consta de una interfaz multimedia amigable al usuario la cual despliega información en tiempo real de la afectación de estaciones base con su correspondiente nombre, número de celda o sector y ubicación geográfica con lo cual se tendrá la información precisa para el cálculo de la afectación, se levantará un servidor web para que la información se la pueda visualizar en línea en la red de Telefónica Movistar.

PALABRAS CLAVES:

- **NOC EXTERNO**
- **ESTACIONES BASE**
- **PLATAFORMA DE GESTION**
- **INTERFAZ MULTIMEDIA**
- **CALCULO DE AFECTACION**

ABSTRACT

The mobile phone operator Telefonica Ecuador today need to address and solve the technical problems of its network in the shortest time possible, in order to keep users with voice and data permanently. The mobile phone network is controlled by the External NOC, which is responsible for carrying out constant monitoring 24 hours a day, 7 days a week, 365 days a year, whose primary task is to assist and interpret alarms that unfold in the various managers monitoring, currently has deployed across the country access equipment and transmission of the Nokia brand which are monitored by the program NetAct, the tool does not have all the details that can be valuable for early diagnosis the problems that can arise in the mobile network. To solve this problem it developed a new software tool for the management, control and monitoring affectations out of service base stations in technology 2G, 3G and 4G Nokia equipment. The new management platform has a user-friendly multimedia interface to the user which displays real-time information of the affected base station with its corresponding name, number of cell or sector and geographic location with which the information will be required for the calculation of affectation, a web server will rise to that information online can visualize the network of Telefonica Movistar.

KEY WORDS:

- **EXTERNAL NOC**
- **BASE STATION**
- **MANAGEMENT PLATFORM**
- **GEOGRAPHIC LOCATION**

CAPITULO I

INTRODUCCIÓN

1.1 Planteamiento del problema

La red de telefonía móvil de Telefónica Ecuador es controlada a través del NOC Externo, el cual es el encargado de realizar el monitoreo constante 24 horas al día, 7 días a la semana, 365 días al año, es por esto que la principal tarea del NOC Externo es verificar el correcto funcionamiento de la red móvil a través de la atenta interpretación de alarmas que se despliegan en los distintos gestores de monitoreo, generalmente estos son provistos por la empresa tecnológica que implanta la tecnología.

La red movíl de Telefónica Ecuador dispone de varios equipos Nokia de Acceso desplegados por todo el país, estos son monitorizados mediante el gestor de alarmas NetAct. La herramienta NetAct provee en tiempo real los detalles de los eventos de sus equipos a través de todo el país, sin embargo, el principal problema en el monitoreo es la falta de detalles en varios campos que pueden ser valiosos para un diagnóstico temprano de los problemas que pueden suscitarse en la red móvil.

Uno de estos campos es el detalle del nombre, al momento solo se muestra un código de asignado a la estación base el cual debe ser comparado con una base de datos en Excel para obtener el nombre real de la estación, esto resta agilidad al monitoreo ya que en eventos donde se involucran varias estaciones toma mucho tiempo diagnosticar la indisponibilidad de servicio de voz y datos, entorpeciendo la gestión de los incidentes. De igual manera otro campo que no se muestra es la ubicación de la estación, otro campo que no muestra es indicar si la estación se encuentra totalmente por fuera de servicio o si solo se encuentra algún sector específico por lo que actualmente se requiere que el operador se ingrese vía conexión remota a la estación a verificar el estado de la estación monitoreada.

1.2 Formulación resumida del proyecto

Se desarrollará una nueva herramienta de software para la gestión, control y monitoreo de afectaciones de fuera de servicio de estaciones base, de la red nacional de telefonía móvil de la operadora Telefónica Movistar en tecnologías 2G (GSM), 3G (UMTS) y 4G (LTE) de equipos Nokia NSN.

Para que el resultado sea en tiempo real se utilizara el protocolo Telnet el cual permitirán el intercambio de información de administración entre los distintos elementos que conforman la red de telefonía móvil, permitiendo que la nueva herramienta de gestión a ser desarrollada supervise y despliegue la indisponibilidad de servicio que se tenga en la red nacional de telefonía móvil de Telefónica Movistar de forma inmediata, adicional dichos protocolos facilitaran la escalabilidad de la plataforma y planificar su crecimiento, debido al constate aumento de estaciones base.

La nueva plataforma de gestión constará de una interfaz multimedia amigable al usuario la cual despliegue información en tiempo real de la afectación de estaciones base con su correspondiente nombre, número de celda o sector y ubicación geográfica con lo cual se tendrá la información precisa para el cálculo de la afectación a nivel nacional, para lo cual se levantará un servidor web para que la información se la pueda visualizar en línea en la red de Telefónica Movistar.

Se diseñará y creará una base de datos en Mysql la cual contenga la información completa de las estaciones base de quipos Nokia a nivel nacional, dicha base de datos será escalable y contendrá la información actualizada a medida que la red de telefonía móvil siga creciendo.

La interacción entre el servidor web y la base de datos se la realizará por medio de la herramienta de desarrollo de Netbeans de Java, el programa desarrollado en Java se conectará a los distintitos elementos de conmutación de servicios de voz y datos como: Flexis BSC (2G), controladoras RNC (3G) y MME (LTE) y centrales telefónicas para almacenar la información en la base de datos, filtrar la misma y realizar comparaciones para el despliegue de la información mediante el gestor a ser implementado.

Se realizará la evaluación del funcionamiento de la plataforma de monitoreo y gestión, mediante pruebas y análisis de resultados tanto en simulaciones controladas de afectación de servicio, como en escenarios reales tras la implementación de la nueva herramienta se software.

1.3 Objetivos

1.3.1 Objetivo General

- Diseñar e implementar un sistema de monitoreo de estaciones con equipos Nokia que permita una rápida visualización de estaciones y sectores afectados en caso de eventos masivos correspondientes al servicio de telefonía móvil GSM, UMTS y LTE de la red de Telefónica Ecuador a nivel nacional en tiempo real.

1.3.2 Objetivos Específicos

- Describir las características, arquitectura y componentes que integran una red de telefonía móvil celular.
- Levantar información sobre las características de las estaciones GSM, UMTS y LTE de la tecnología Nokia presentes en el Ecuador.
- Detallar las funciones de monitoreo del NOC Externo de Telefónica Movistar Ecuador.
- Crear una interfaz multimedia amigable para el usuario, la cual permita detallar eficientemente la no disponibilidad de estaciones base de la red nacional de telefonía móvil, la misma que deberá ser en línea para la consulta dentro de la red de Telefónica Movistar.
- Diseñar e implantar una base de datos que contenga la información completa de las estaciones base de quipos Nokia a nivel nacional, la cual será escalable y actualizada de forma constante.
- Desarrollar un programa que permita la interacción entre la interfaz web y la base de datos, el mismo que se encargará de filtrar, comparar y desplegar la información de disponibilidad de estaciones base y celdas.

- Hacer una evaluación de funcionalidad y flexibilidad del nuevo gestor, mediante la realización pruebas y análisis, tanto en simulaciones como en escenarios reales.

1.4 Justificación

Debido al crecimiento exponencial en el despliegue de red de telefonía móvil en el país de la operadora Movistar, se ha vuelto cada vez un reto más grande el constante monitoreo de la misma, ya que los tiempos de reacción y atención deben ser cortos para dar una adecuada descripción de los problemas que afectan los servicios de voz y datos a las autoridades de regulación estatales, en este caso la ARCOTEL.

La tecnología Nokia ha estado en constante evolución en cuanto al desarrollo de los equipos de telecomunicaciones y monitoreo, sin embargo no cumple con una de las principales necesidades del NOC Externo de Telefónica Ecuador. Una de esas necesidades es mostrar el nombre de las estaciones relacionadas con un evento, al momento solo presenta un código el cual debe ser relacionada de manera manual en una base de datos en Excel, esto vuelve muy complicado la gestión y el monitoreo de la red en caso de existir problemas con numerosas estaciones.

Otra de las necesidades que se espera a resolver con el desarrollo de este proyecto es informar de manera rápida si una estación de tecnología Nokia se encuentra por fuera de servicio en su totalidad o solo están afectados celdas específicas, al momento para conocer esa información en necesario ingresar de manera remota a cada una de las estaciones afectadas y verificar el estado de las mismas, esto representa un inconveniente debido a la gran cantidad de estaciones que pueden verse afectadas durante un evento de gran magnitud.

Al culminar el proyecto se espera que la herramienta de software desarrollada sirva a los operadores para realizar un correcto análisis de eventos masivos de una manera más rápida y eficiente, de esta manera el NOC Externo puede comunicar a los operadores del grupo de O&M sobre el estado de la red a nivel nacional en menor tiempo y así obtener un diagnóstico eficiente de los problemas, adicional de brindar información en tiempo real de afectación de servicio de voz y datos al ente regulador estatal ARCOTEL tras la pérdida de servicio de estaciones que pertenezcan al Sistema Autónomo de Medición de Redes Móviles (SAMM) del país.

1.5 Antecedentes

El sector de las Telecomunicaciones ha experimentado un gran crecimiento a nivel nacional durante la última década, las operadoras telefónicas asentadas en el país han aumentado sus servicios y la extensión de la disponibilidad de los mismos, debido al incremento de la demanda de servicios de voz y datos que requieren los usuarios. Una de las operadoras más importantes del país como lo es Telefónica Movistar Ecuador ha adaptado la tecnología Nokia para el despliegue de sus equipos a nivel nacional, tanto en 2G, como 3G y 4G.

Al momento los operadores pertenecientes al NOC Externo de Telefónica que controlan el monitoreo no disponen de todas las herramientas necesarias que sean eficaces en el despliegue de los datos necesarios de estaciones afectadas, para su recopilación e información de manera automática en caso de existir incidentes de gran envergadura en el servicio celular con respecto a la tecnología Nokia.

La actual herramienta de monitoreo para tecnologías Nokia llamada Nectact 8, no dispone de funciones que permitan desplegar los nombres asociados a las estaciones con pérdida de servicio, lo cual no permite al operador del NOC saber eficazmente la información de la afectación de disponibilidad de servicio, además de que no permite discernir si una estación se encuentra por fuera de servicio en su totalidad o si se trata de sectores o celdas específicos.

1.6 Alcance del Proyecto

Culminar la implementación de un sistema de monitoreo alternativo para las estaciones de tecnología Nokia tanto 2G como 3G y 4G el cual permita una mejor visualización de los incidentes en tiempo real.

Lograr que el NOC Externo se apoye en la herramienta a implementar para la gestión de Incidentes en la red Nokia a nivel nacional,

Realizar la descripción de la red de telefonía móvil, sus características, componentes y arquitectura.

Describir las funciones del NOC Externo de monitoreo de Telefónica Ecuador con respecto a la gestión de Incidentes.

CAPITULO II

MARCO TEÓRICO

2.1 Evolución de las redes celulares

La telefonía móvil es la comunicación a través de dispositivos que no están conectados mediante cables. El medio de transmisión es el aire y el mensaje se envía por medio de ondas electromagnéticas. Para la comunicación, se utiliza el teléfono móvil, que es un dispositivo inalámbrico electrónico que se usa para tener acceso y utilizar los servicios de la red de telefonía móvil. La telefonía móvil básicamente está formada por dos grandes partes: una red de comunicaciones y los terminales que permiten el acceso a dicha red.

Las tecnologías inalámbricas están teniendo mucho auge y desarrollo en estos últimos años, una de las que ha tenido un gran desarrollo ha sido la telefonía celular, desde sus inicios a finales de los setenta ha revolucionado enormemente las actividades que realizamos diariamente. Los teléfonos celulares se han convertido en una herramienta primordial para la gente común y de negocios, las hace sentir más segura y las hace más productivas.

Martin Cooper fue el pionero en esta tecnología, a él se le considera como "el padre de la telefonía celular" al introducir el primer radioteléfono en 1973 en los Estados Unidos mientras trabajaba para Motorola; pero no fue hasta 1979 en que aparece el primer sistema comercial en Tokio Japón por la compañía NTT (Nippon Telegraph & Telephone Corp.)

La primera generación de la red móvil (1G), como tal, llegó en los años 80, una época en la que ya existían varias redes de móviles, todas ellas basadas en una sola antena. Estas redes contaban con un número limitado de canales y requerían del uso de terminales analógicos, se caracterizó por ser analógica y estrictamente para voz. La calidad de los enlaces de voz era muy baja, baja velocidad [2400 bauds], la transferencia entre celdas era muy imprecisa, tenían baja capacidad [basadas en FDMA, Frequency Divison Multiple Access] y la seguridad no existía. La tecnología predominante de esta generación es AMPS (Advanced Mobile Phone System). Los terminales móviles son equipos demasiado grandes, pesados e incómodos, comparados con los estándares actuales. (Buyya & Pathan, 2008)

No obstante, es importante señalar que en la década de los 80s estos aparatos representaron una evolución sin precedentes dentro de las comunicaciones móviles, y para su época significaron un gran avance, ya que a partir de la denominada primera generación, las terminales se volvieron más pequeños, lo que permitía que los usuarios pudieran trasladar sus equipos de comunicación. (Buyya & Pathan, 2008)

Figura 1 Teléfono móvil primera generación

La 2G arribó hasta 1990 y a diferencia de la primera se caracterizó por ser digital. El sistema 2G utiliza protocolos de codificación más sofisticados y son los sistemas de telefonía celular usados en la actualidad. Las tecnologías predominantes son: GSM (Global System for Mobile Communications); IS-136 (conocido también como TIA/EIA-136 o ANSI-136) y CDMA (Code Division Multiple Access) y PDC (Personal Digital Communications), éste último utilizado en Japón.

Los protocolos empleados en los sistemas 2G soportan velocidades de información más altas para voz pero limitados en comunicaciones de datos. Se pueden ofrecer servicios auxiliares tales como datos, fax y SMS (Short Message Service). La mayoría de los protocolos de 2G ofrecen diferentes niveles de encriptación. En los Estados Unidos y otros países se le conoce a 2G como PCS (Personal Communications Services). (Buyya & Pathan, 2008)

Cabe destacar que el cambio de 1G a 2G significó un importante paso en el mundo de la telefonía móvil, ya que las comunicaciones lograron alcanzar una calidad destacada, gracias a la utilización de las frecuencias de 900 y 1800 MHz.

La generación 2.5G ofrece características extendidas para ofrecer capacidades adicionales que los sistemas 2G tales como GPRS (General Packet Radio System) y EDGE (Enhanced Data Rates for Global Evolution)

Figura 2 Teléfonos móviles de segunda generación

La 3G es tipificada por la convergencia de la voz y datos con acceso inalámbrico a Internet, aplicaciones multimedia y altas transmisiones de datos. Los protocolos empleados en los sistemas 3G soportan más altas velocidades de información enfocados para aplicaciones más allá de la voz tales como audio (MP3), video en movimiento, video conferencia y acceso rápido a Internet, sólo por nombrar algunos. Se espera que las redes 3G empiecen a operar en el 2001 en Japón por NTT DoCoMo, en Europa y parte de Asia en el 2002, posteriormente en Estados Unidos y otros países.

Los sistemas 3G alcanzaran velocidades de hasta 384 Kbps permitiendo una movilidad total a usuarios viajando a 120 kilómetros por hora en ambientes exteriores y alcanzará una velocidad máxima de 2 Mbps permitiendo una movilidad limitada a usuarios caminando a menos de 10 kilómetros por hora en ambientes estacionarios de corto alcance o en interiores. Entre las tecnologías contendientes de la tercera generación se encuentran UMTS (Universal Mobile Telephone Service), cdma2000, IMT-2000, ARIB[3GPP], UWC-136, entre otras. (Buyya & Pathan, 2008)

Gracias a todos estos avances, hoy es posible llegar a velocidades de transmisión de datos superiores a los 7.2 Mbits/s, lo que favoreció el surgimiento de novedosas implementaciones en el celular, tales como la descarga de contenidos de programas, servicios de video llamada, mensajería instantánea y la utilización del correo electrónico, entre muchas otras.

Figura 3 Teléfonos móviles de tercera generación

La tecnología 4G implica un cambio radical en la red de comunicación, ya que gracias a esta nueva plataforma es posible establecer conexiones a una velocidad de 1Gbps, y obtener transferencias de hasta 100 Mbps.

Para lograr esas tasas de velocidades, la cuarta generación de tecnología de telefonía móvil estará basada por completo en IP y su arquitectura la convertirán en una red que se encargará de gestionar otras redes inferiores. (Buyya & Pathan, 2008)

Esto hace posible que la convergencia entre los dispositivos móviles y otros equipos, tales como computadoras de escritorio, sea total y veloz, ya que promete ofrecer una velocidad de acceso de hasta 100 Mbps. descendente y 50 Mbps. en enlace ascendente, siempre manteniendo los estándares de seguridad entre dispositivos.

Es por ello, que 4G surge como una manera de reunir todas las tecnologías disponibles hasta el momento, con las modificaciones necesarias para ofrecer un servicio extremadamente veloz y a bajo costo, dentro del campo de las redes inalámbricas. (Buyya & Pathan, 2008)

Figura 4 Teléfonos móviles de cuarta generación

2.2 Proveedores de telefonía celular en el Ecuador

Hoy en día existen tres operadoras móviles concesionadas en Ecuador como son; Conecel S.A.(Claro); Otecel S.A.(Movistar) y Telecsa S.A.(Alegro-CNT), a continuación, se analizarán algunos puntos relevantes como son: cobertura, tecnología y costo.

Cobertura: La geografía propia de muchas regiones de Ecuador no ofrece una propagación amplia de servicios de telefonía por cable, en consecuencia, la propagación de los servicios de telefonía móvil ha tenido un crecimiento e implantación extraordinario, con una densidad del 101%.

Porta hoy en día conocida como Claro fue el primero en distribuir sus radio bases de conexión (CDMA y GSM) a lo largo y ancho del territorio nacional, lo cual ha significado ganar clientes en regiones donde otras operadoras no tenían cobertura llegando a cubrir esas áreas, se puede decir que Claro tiene una alta propagación de cobertura nacional en ciudades y poblaciones con un nivel de señal propicio para realizar una llamada, pero muchas de ellas se las debe realizar fuera de edificaciones o en espacios abiertos. En el grafico presentado a continuación de describe los usuario de Conecel

RESUMEN MENSUAL LATU - CONECEL S.A.									
FECHA	PREPAGO (HSPA+)	POSTPAGO (HSPA+)	PREPAGO (UMTS)	POSTPAGO (UMTS)	PREPAGO (GSM)	POSTPAGO(GSM)	TOTAL PREPAGO	TOTAL POSPAGO	TOTAL CONECEL S.A.
mar-15	6.593	554.476	364.680	1.231.736	7.732.723	582.511	8.103.996	2.368.723	10.472.719

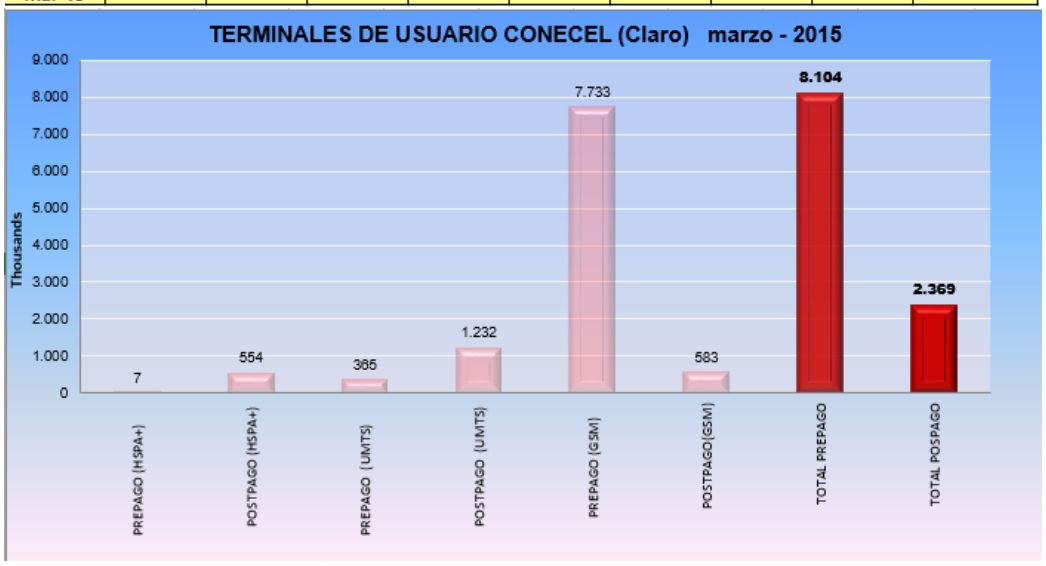


Figura 5 Número de usuarios a nivel nacional Claro

Movistar actualmente está logrando llegar a lugares donde no tenía cobertura en sus inicios, ha realizado un despliegue en la instalación de radio bases en varias zonas del país permitiendo aumentar el número de abonados a su red en los últimos años. En el siguiente grafico se detalla el número de subscriptores que posee la operadora móvil.

RESUMEN MENSUAL LATU - OTECEL S.A.											
FECHA	PREPAGO (HSPA+)	POSTPAGO (HSPA+)	PREPAGO (UMTS)	POSTPAGO (UMTS)	PREPAGO (CDMA)	POSTPAGO (CDMA)	PREPAGO (GSM)	POSTPAGO (GSM)	TOTAL PREPAGO	TOTAL POSPAGO	TOTAL OTECEL S.A.
mar-15	18.262	135.334	64.605	261.133	0	0	3.462.760	733.221	3.545.627	1.129.688	4.675.315

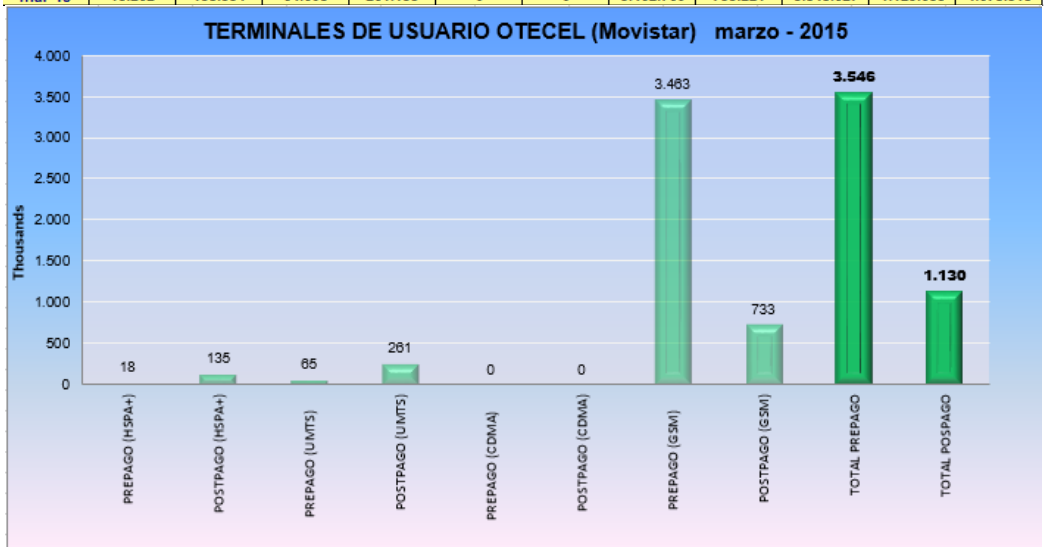
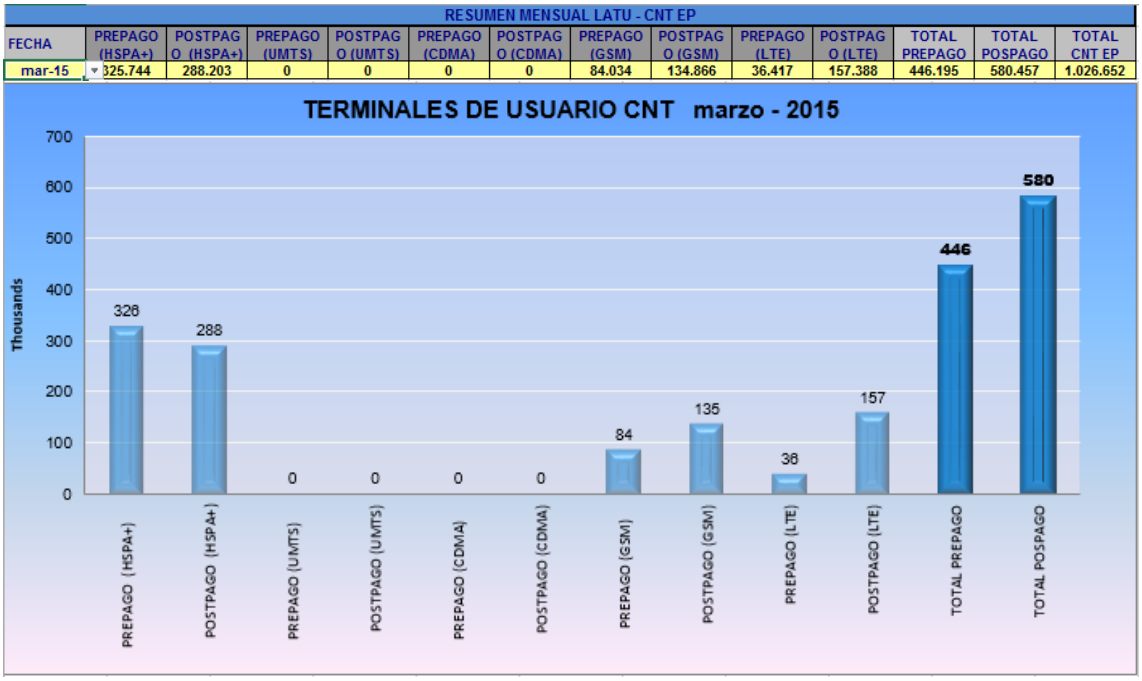


Figura 6 Número de usuarios a nivel nacional Movistar

CNT se encuentra en solo ciertas ciudades principales y poblaciones del país; CNT tiene una red arrendada a Movistar para la red de GSM, donde tiene una cobertura similar a Movistar.



RESUMEN MENSUAL LATU - CNT EP

FECHA	PREPAGO (HSPA+)	POSTPAGO (HSPA+)	PREPAGO (UMTS)	POSTPAGO (UMTS)	PREPAGO (CDMA)	POSTPAGO (CDMA)	PREPAGO (GSM)	POSTPAGO (GSM)	PREPAGO (LTE)	POSTPAGO (LTE)	TOTAL PREPAGO	TOTAL POSPAGO	TOTAL CNT EP
mar-15	325.744	288.203	0	0	0	0	84.034	134.866	36.417	157.388	446.195	580.457	1.026.652

Figura 7 Número de usuarios a nivel nacional CNT

Tecnología: Si bien hoy en día las comunicaciones a nivel mundial ya están usando LTE en gran parte de su red, en nuestro país estamos utilizando en su gran mayoría la red GSM y UMTS por dicha razón las operadoras móviles están desplegando LTE en varias zonas a nivel nacional.

Costos: Para evaluar los costos por minuto de las operadoras se realizará un análisis a los servicios prepago que tiene una demanda del 90% a nivel nacional, en este punto CNT se lleva el mérito.

CNT: misma red ($0.05*) teléfono fijo ($0.12*) otra móvil ($0.18*)

Movistar: misma red ($0.08*) teléfono fijo ($0.15*) otra móvil ($0.25*)

Claro: misma red ($0.05*) teléfono fijo ($0.16*) otra móvil ($0.25*)

*= No incluye impuestos

En el siguiente grafico se puede apreciar la distribución del mercado de telefonía celular con base a los prestadores de servicio CNT, Conecel (Claro) y Otecel (Movistar).

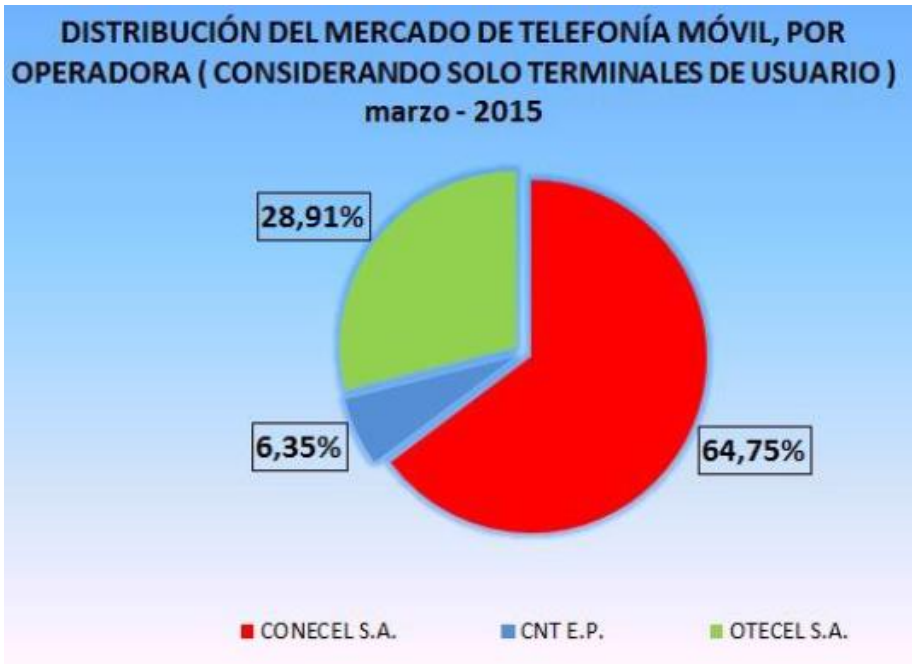


Figura 8 Número de usuarios a nivel nacional CNT

2.3 Arquitectura de la redes celulares

Las redes de telefonía móvil se basan en el concepto de celdas, es decir zonas circulares que se superponen para cubrir un área geográfica.

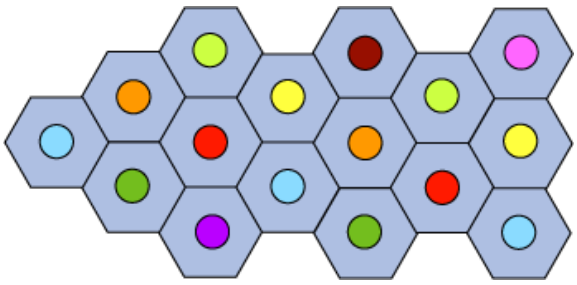


Figura 9 División de celdas en una zona geográfica

Las redes celulares se basan en el uso de un transmisor-reeptor central en cada celda, denominado "estación base" (o Estación base transceptora, BTS).

Cuanto menor sea el radio de una celda, mayor será el ancho de banda disponible. Por lo tanto, en zonas urbanas muy pobladas, hay celdas con un radio de unos cientos de metros mientras que en zonas rurales hay celdas enormes de hasta 30 kilómetros que proporcionan cobertura. En una red celular, cada celda está rodeada por 6 celdas contiguas (por esto las celdas generalmente se dibujan como un hexágono). Para evitar interferencia, las celdas adyacentes no pueden usar la misma frecuencia. En la práctica, dos celdas que usan el mismo rango de frecuencia deben estar separadas por una distancia equivalente a dos o tres veces el diámetro de la celda. [3]

PSTN, ISDN, Data Network Representan Redes externas a la Red Celular por ejemplo una llamada desde un Teléfono Fijo convencional hacia un Teléfono Celular provendría de la PSTN probablemente o una conexión desde un Móvil hacia Internet Tendría que dirigirse hacia la Data Network.

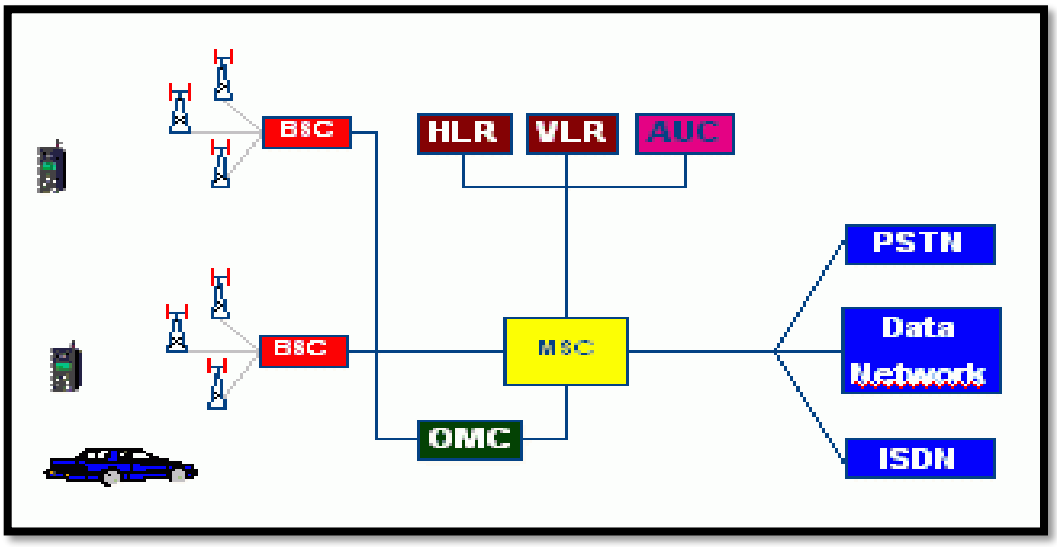


Figura 10 Diagrama básico de una red celular

2.3.1 Arquitectura de red 2G

En GSM se especifican entidades funcionales e interfaces normalizadas. De este modo, siempre que se cumplan las especificaciones normalizadas, los equipos de todos los fabricantes serán interoperables, se podrán comunicar, y un operador podrá comprar sus equipos a distintos fabricantes, evitando así la posibilidad de crear mercados cautivos.

Una red GSM se organiza como un conjunto de células radioeléctricas continuas que proporcionan cobertura completa al área de servicios. Cada una de estas células pertenece a una estación base (BTS), que se denomina también RBS, que opera en un conjunto de canales de radio diferentes a los usados en las células adyacentes y que se encuentran distribuidas según un plan celular de frecuencias. Se compone de varias entidades funcionales físicas y lógicas y cuyas funciones e interfaces se encuentran completamente especificados y definidos. La arquitectura básica de un sistema GSM y su esquema se muestran en las siguientes figuras, en donde podemos distinguir los principales bloques que lo constituyen, incluido el subsistema de la estación base

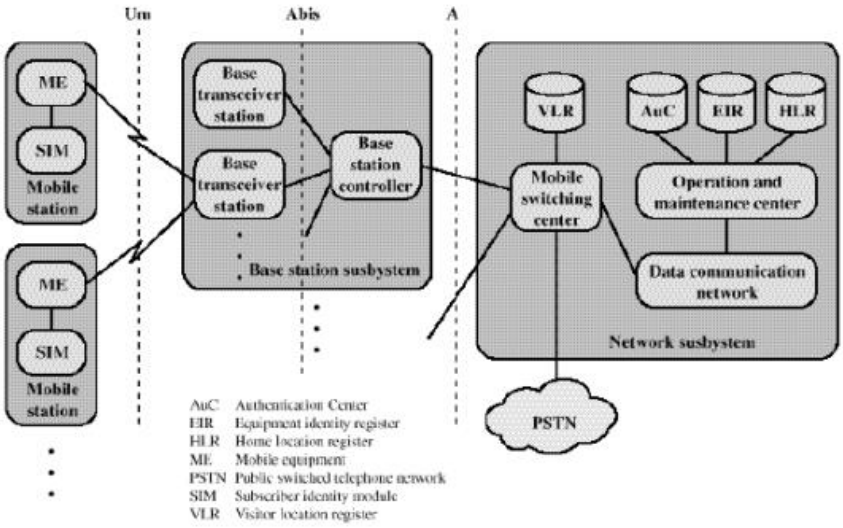


Figura 11 Esquema De La Arquitectura GSM

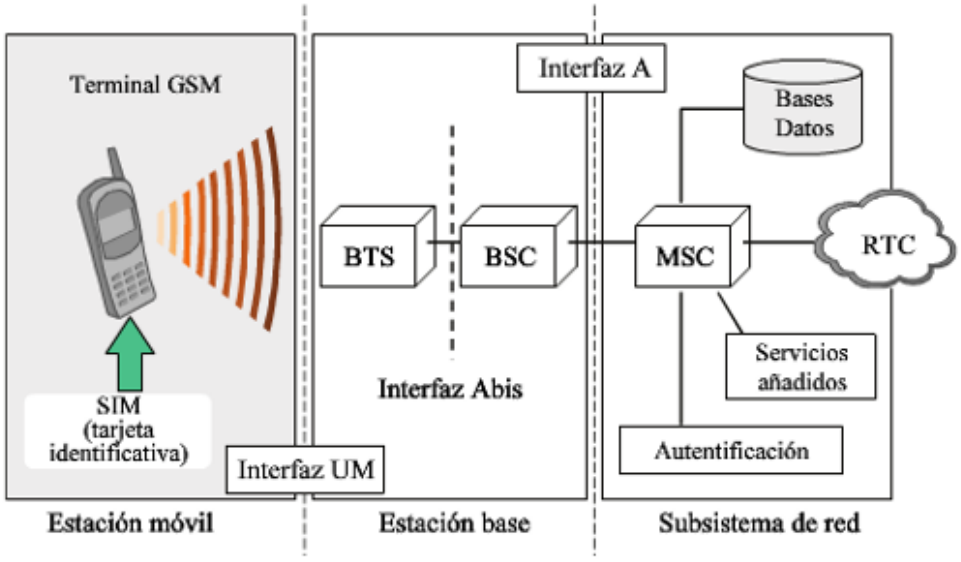


Figura 12 Estructura De Una Red GSM

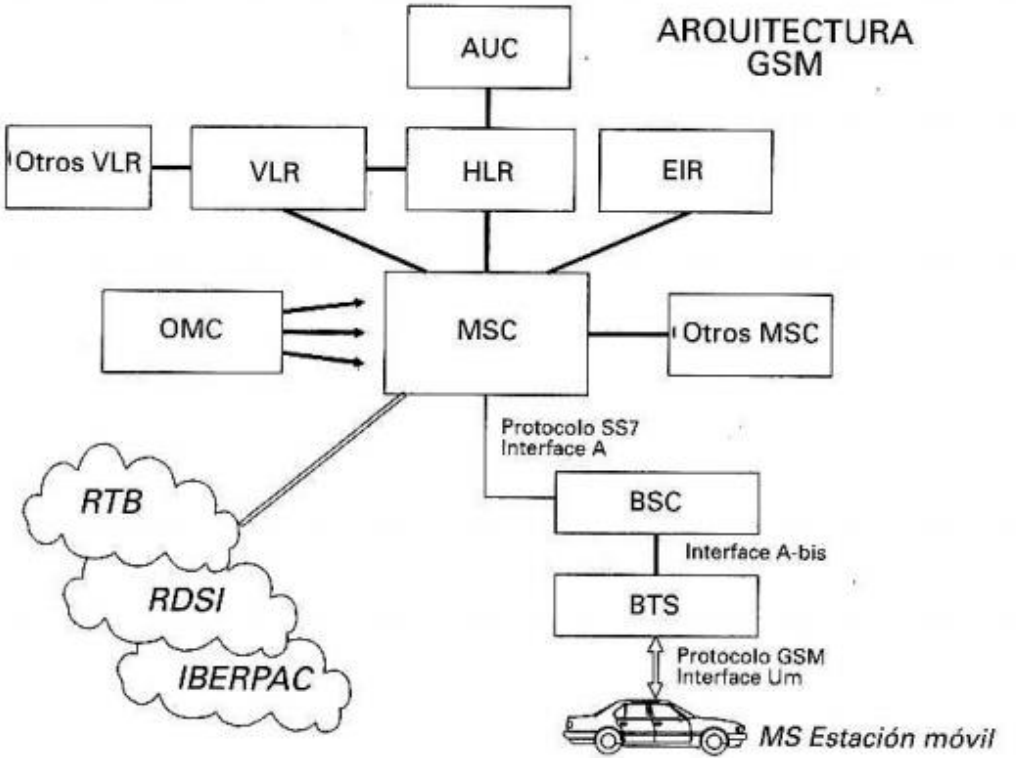


Figura 13 Arquitectura de GSM

A continuación, se describirá los elementos de una red GSM:

Estación Móvil (MS):

Se divide en dos módulos:

ME (Mobile Equipment): terminal en sí. Está identificado por el IMEI (International Mobile Equipment Identity) número de 15 cifras que se puede obtener tecleando *#06# y utilizado por el EIR. [4]

- Dual band: terminal que puede transmitir en dos bandas de frecuencias: GSM 900 (europeo) y DCS1800/PCS1900 (americana)
- Dual mode: capaz de conectarse a redes de tecnología distinta: GSM y DECT.

SIM (Subscriber Identity Module): identificador de usuario. Contiene:

- Identificador de usuario IMSI (International Mobile Subscriber Identity).
- Claves para criptografía.
- Agenda de usuario
- SMSs recibidos y guardados por el usuario.
- Contraseña para restringir el uso del SIM.
- MEs y SIMs son intercambiables. [4]

Estación Base (BSS):

Controla la interface radio, y se divide en:

BTS (Base Transceiver Station): puede haber una o más por BSS, contiene los transmisores / receptores que sirven a una celda. Funciones:

- Interfase física entre MT y BSC.
- Gestión de Diversidad de Antenas.
- FH (Frecuency Hopping).
- Control Dinámico de Potencia.
- Gestión de algoritmos de Clave.
- Monitorización de la conexión.

Controladora Base (BSC)

BSC (Base Station Controller): gobierna los recursos radio para las BTS a él conectadas.

- Gestión y configuración del canal radio: elección de la celda y canal.
- Gestión de los handover.
- Transcodificación de canales radio (16 ó 8kbps) a canales a 64kbps.

Subsistema de red (NSS):

- Permite la interconexión entre BSS y con otras redes públicas (PSTN, ISDN, PSPDN, CSPDN)
- Implementa las funciones de base de datos necesarias para:
  - ✓ Identificación de usuarios y terminales,
  - ✓ Localización de los terminales y conducción de llamadas,
  - ✓ Facturación, etc.
- Formado por:
  - ✓ MSC (Mobile Switching Center).

    Elemento central del NSS se ocupa de la gestión del tráfico de una o más BSS, actuando como un router. Además interconexiona todos los demás elementos del NSS.

    Funciones:

    - Gestión de llamadas:
    - Autenticación de la llamada: localización e identificación del MS.
    - Conmutación entre BSS del mismo NSS o con otros MSC o redes.
    - Funciones de gateway con otras redes (PLMN, ISDN, PSTN,etc.)
    - Proceso de handover:
    - Handover Intra-MSC
    - Handover Inter-MSC
    - Confidencialidad de la identidad de usuario: uso del TMSI en lugar del IMSI en la transmisión. El MSC asocia el IMSI con el TMSI.

- ✓ HLR (Home Location Register).

  Base de datos central o distribuida que contiene información sobre el usuario:

  – IMSI: identificación del usuario en toda la red GSM

  – MSISDN (Mobile Station ISDN Numberk): Similar al número de abonado en una red PSTN.

  – Tipos de servicios contratados.

  – Posición actual del MS: dirección del VLR en el que está.

  • Funciones:

  Seguridad: diálogo con el AuC y el VLR.

  Registro de posición: actualización de los VLRs.

  Coste de llamada: obtenido de la información del MSC.

  Gestión de datos del usuario.

  Gestión de datos estadísticos.

- ✓ VLR (Visitor Location Register).

  Base de datos temporal que contiene información del abonado en el área geográfica bajo su control. Los datos son suministrados por el HLR.

  • Se suele implementar en el MSC para simplificar la señalización ¨el área geográfica del MSC es la del VLR.

  • Almacena:

  – TMSI usado para garantizar la seguridad del IMSI.

  – Estado del MS (standby, ocupado, apagado).

  – Estados de los servicios suplementarios: llamada en espera, llamada diferida, etc.

  – Tipo de servicios suscriptos por el abonado.

  – LAI (Location Area Identity): zona dentro de aquéllas bajo el control del MSC/VLR donde está el MS.

- ✓ AuC (Autentication Center).

  Verifica si el servicio es solicitado por un abonado legítimo.

  • Proceso:

  – Verifica el IMSI sin transmitir información personal del abonado.

  – Genera las claves a partir del IMSI.

  • Las claves son usadas para la encriptación de la información.

  • Las claves y códigos de autentificación también están almacenados en el SIM.

  • La autentificación se produce:

  – Cada vez que el MS se conecta a la red.

  – Cada vez que el MS recibe o efectúa una llamada.

  – Cada vez que se actualiza la posición del MS.

  – Cada vez que se realiza acceso a alguno de los servicios suplementarios.

- ✓ EIR (Equipment Identity Register).

  Verifica si un ME está autorizado para acceder al sistema.

  • Tres Listas:

  – Lista Blanca: contiene los IMEI que pueden acceder a la red.

  – Lista Gris: contiene IMEI marcados y no homologados. Son monitoreados por la red, pudiéndose determinar su localización y el SIM que está siendo usado.

  – Lista Negra: contiene IMEI bloqueados. Formada entre otros por IMEI robados a los cuales se les niega el acceso a la red.

  • Cada vez que un ME intenta acceder a la red la MSC verifica mediante el EIR la lista a la que pertenece el ME, tomando las acciones necesarias.

- ✓ OMC (Operation and Maintenance Center).
  Utilizado para la monitorización y mantenimiento de la red por parte del operador.
  • Funciones:
  – Acceso remoto a todos los elementos del PLMN.
  – Gestión de alarmas y estado del sistema.
  – Recogida de información de tráfico de usuarios facturación.
  – Supervisión del flujo de tráfico, para posibles cambios de arquitectura de red.
  – Reconfiguración de la red mediante acceso remoto.
  – Administración de abonados: altas, bajas, localización dentro de la red. [4]

2.3.2 Arquitectura de red 3G

UMTS (Universal Mobile Telecommunication System): Sistema europeo que surge para la transición suave de las redes 2G que generalmente eran GSM hasta las redes de la Tercera Generación. Utiliza CDMA (Acceso Múltiple por División de Código) proporciona una transmisión de datos de velocidades altas por conmutación de paquetes (384 Kbps) como conmutación de circuitos (2 Mbps)

En las figuras siguientes se muestran la arquitectura general de los sistemas UMTS, se dividen en dos dominios: el dominio del Equipo de Usuario (UE, User Equipment) y el dominio de la infraestructura. Dichos dominios están conectados a través del interfaz radio denominados Uu.

El UE puede incluir una tarjeta inteligente extraíble que puede usarse en diferentes tipos de terminales. El dominio de Equipos de Usuario se divide en: el dominio de Equipo Móvil (ME, Mobile Equipment) y el dominio del Módulo de Identidad de Servicios de Usuario (USIM, User Services Indentity Module). Él ME puede subdividirse en: la Terminación Móvil (MT, Mobile Termination) que realiza

las funciones relacionadas con la transmisión radio, y el Equipo Terminal (TE, Terminal Equipment) que contiene las aplicaciones extremo a extremo. El USIM contiene datos y procedimientos que la identifican de forma segura y sin ambigüedad, y están normalmente incluidos en una tarjeta inteligente.

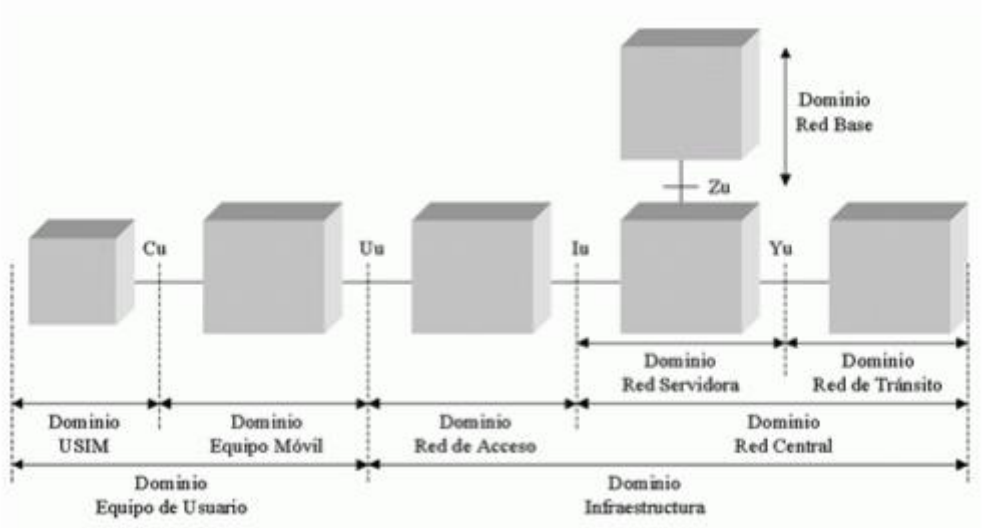


Figura 14 Arquitectura General De Un Sistema UMTS

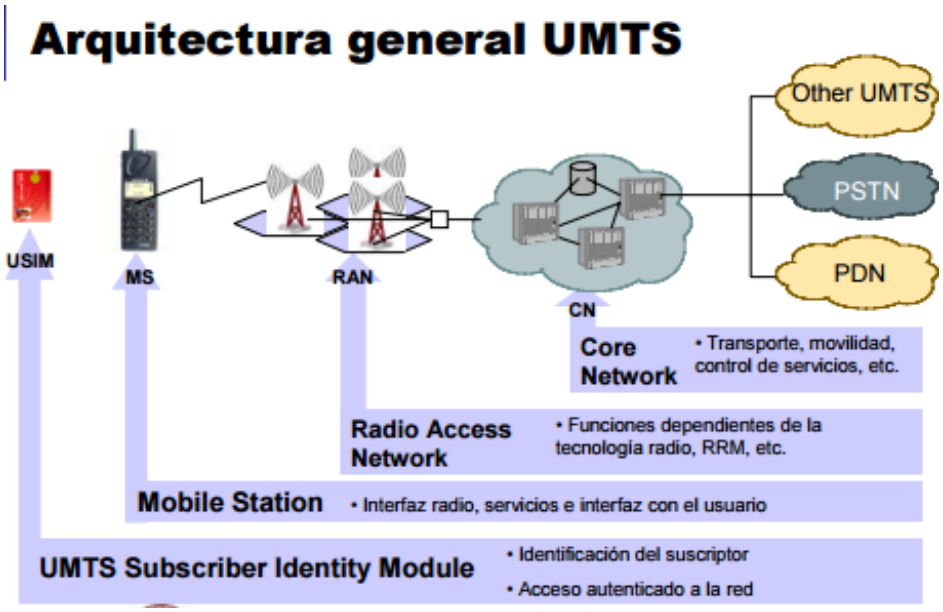


Figura 15 Arquitectura General con descripción UMTS

Con objeto de permitir escenarios de comunicación con interlocutores de otras redes (operadores fijos, otras redes móviles, Internet, etc), tanto si el usuario accede a través de su red propia (“home”) o la de otro operador (itinerancia o “roaming”), dentro del CN se distinguen tres dominios

- Red de Servicio, SN (Service Network): representa la red a la que está accediendo el usuario en un momento dado, pudiendo ser la red propia, o en el caso de “roaming” la red visitada.

- Red Propia, HN (Home Network): es la red del operador al que el usuario está abonado. En el caso de “roaming”, la red visitada debe contactar con la red propia del usuario para diversos aspectos (por ejemplo, para temas de autenticación y tarificación).

- Red de tránsito, TN (Transit Network): es la red destino donde se encuentra el interlocutor con el que desea comunicarse el usuario.

Nótese que en función del escenario concreto considerado, es posible el solapamiento entre los tres dominios del CN. Así, en el caso habitual de un usuario que acceda a su red propia, los dominios SN y HN serán coincidentes. Este escenario conduce a plantear el modelo simplificado de la arquitectura UMTS descrito a continuación.

Una visión simplificada de la arquitectura de una red UMTS estaría compuesta por tres partes fundamentales: los equipos de usuario, la red de acceso y el núcleo de red.

Los equipos de usuario acceden a la red a través de la interfaz radio (Uu), que en UMTS está basa en tecnología WCDMA. El modelo asume el empleo de la interfaz radio terrestre (UTRA, UMTS Terrestrial Radio Access), basada en el despliegue de estaciones bases terrenas. Bajo esta modalidad, la red de acceso recibe la denominación UTRAN (UMTS Terrestrial Radio Acces Network).

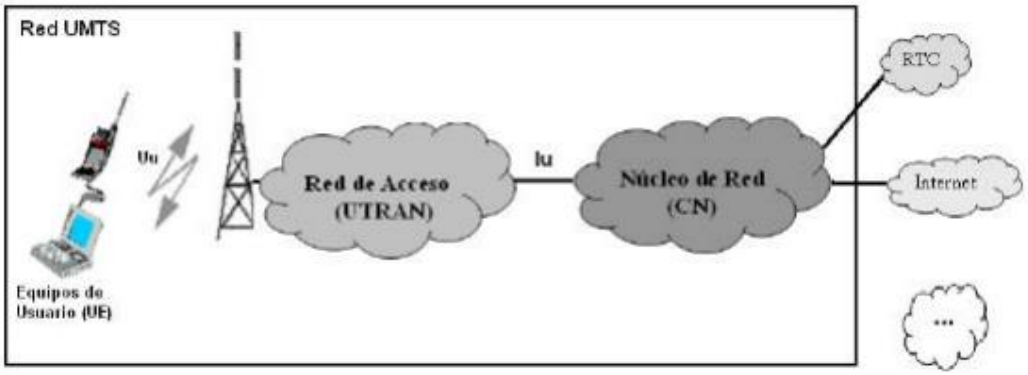


Figura 16 Arquitectura UMTS simplificada

La red de acceso UTRAN se encarga de transportar el tráfico de usuario (voz, datos, señalización móvil-red) hasta el núcleo de red, con el que se comunica a través de la interfaz Iu. Al tratarse de un sistema de comunicaciones móviles, el usuario no dispone de recursos de transmisión asignados de manera estática en la UTRAN. En consecuencia, ésta se encarga también de gestionar la asignación dinámica de dichos recursos cada vez que el móvil utiliza la red.

En el CN se encuentran los recursos de conmutación y transmisión necesarios para completar el trayecto de la comunicación hacia el abonado remoto, que puede pertenecer a la misma red UMTS o a una red externa (otras redes telefónicas, Internet, etc). El CN contiene también funciones relativas a la gestión de los abonados: identidades, claves de autenticación, parámetros de suscripción, etc.

El núcleo de red (CN) de UMTS, se encuentra basada en la topología de la red GSM/GPRS, provee funciones de conmutación, enrutamiento y transporte y bases de datos para el tráfico de la red, posee elementos de conmutación de circuitos, tales como el MSC, el VLR y el GMSC, elementos de conmutación de paquetes, como el SGSN y el GGSN, y elementos que soportan ambos tipos de conmutación, el EIR, el HLR y el AuC.

Controlador de Red Radio (RNC, Radio Network Controller): El RNC es la entidad controlada de un RNS y se encarga del control general de los recursos radio proporcionados por uno o varios nodos B. El RNC es responsable de las decisiones de handover que requieren señalización al MS.

Nodo B (Node B): Es el componente responsable de la transmisión/recepción radio hacia/desde MSs en una o más celdas UMTS. Un nodo B puede soportar el modo FDD, el modo TDD, o una operación en modo dual. Los nodos B se conectan a los RNCs a través de los interfaces Iubis y al UE a través de los interfaces Uu.

Así la Red de Radio Acceso Terrestre UMTS-UTRAN considera la incorporación de dos nuevos elementos: El Controlador de Radio de la Red (Radio Network Controller o RNC) y el Nodo B.

Dentro de la red de acceso, lo más característico es la interfaz radio (Uu) entre el nodo B y el terminal del cliente, puesto que ésta será el principal cuello de botella de velocidad y funcionalidad de todas las comunicaciones que se intentan realizar.

2.3.4 Arquitectura de red 4G

LTE sigue la misma arquitectura de red que los anteriores sistemas especificados por el 3GPP, y abarcan la especificación del equipo de usuario (User Equipment, UE) y de una infraestructura de red que se divide de forma lógica en una infraestructura de red troncal (Core Network, CN) y una de red de acceso (Access Network, AN).

La arquitectura SAE presenta varias ventajas con respecto a las tecnologías que se han desarrollado anteriormente para redes celulares, mejorando la latencia, el throughput y la capacidad de la red, además, el núcleo de la red presenta simplicidad en la arquitectura, optimiza el tráfico de los servicios, los cuales son totalmente basados en IP.

De una manera muy simplificada se puede describir la arquitectura LTE como se puede apreciar en la Figura 15, donde se observa la interacción de los eNodosB, a través de una red IP, con los gateway's que proporcionan comunicación y accesos a diversos servicios o redes, permitiendo una interacción entre la parte móvil y las otras redes con las que cuenta el ISP o incluso externas.

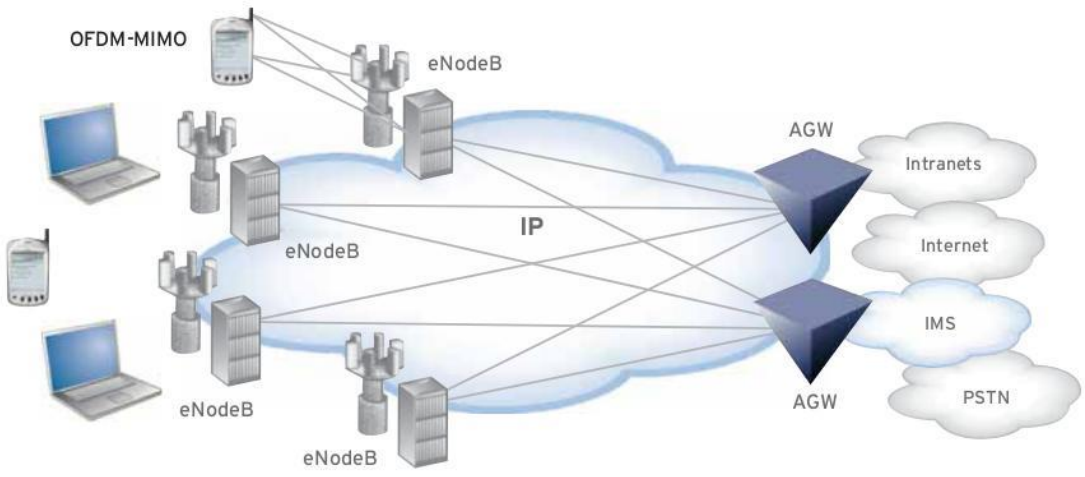


Figura 17 Arquitectura LTE

La arquitectura LTE consta de dos partes como se muestra en la figura 11 la EPC y la EUTRAN (Envolved UTRAN). La EUTRAN es la parte de la red que se encarga de todas la funciones relacionadas a la interfaz de radio y el control de los

móviles, por otro lado, la EPC brinda acceso a otras redes de paquetes IP, además, es aquí donde se gestiona los aspectos relacionados a la seguridad, calidad de servicio, gestión de recursos y movilidad.

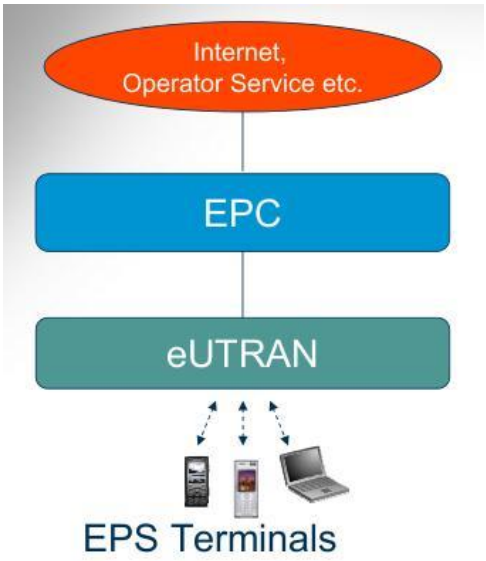


Figura 18 Arquitectura LTE simplificada

La EUTRAN está compuesta por los eNodosB, como lo muestra la figura 12 brindando aspectos de control de móviles y el manejo del medio, estos elementos de red se interconectan entre ellos por medio de interfaces X2, los eNodeB interactúan con la EPC por medio de los MME (“Mobility Management Entity”) con interfaces S1 para el control de la movilidad, gestión y otros aspectos. Ambas son interfaces IP que además usan el protocolo SCTP (“Stream Control Transmission Protocol”) . Por otro lado, algunas de las funciones del eNodeB son:

Funciones de gestión de recursos de radio como conexión, control de admisión de radio, control de movilidad en el plano de usuario.

Compresión de encabezados IP y encripción de datos de usuario.

Enrutamiento en el plano de usuario.

Transmisión de información broadcast.

Reportes de configuración para movilidad.

Las funciones de la MME son las siguientes:

Proveer señalización, seguridad y control de la seguridad.

Proveer señalización entre los nodos para gestionar la movilidad entre nodos.

Es el encargado de administrar tarifas para cobros.

Gestiona la movilidad entre otras redes como 2G y 3G.

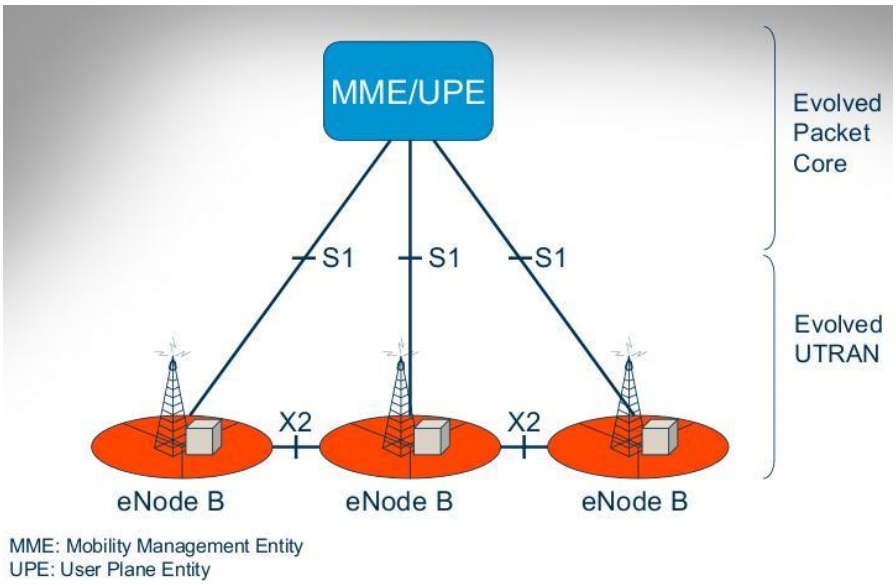


Figura 19 Interfaces entre eNodeB y MME

Una visión más amplia de la arquitectura planteada por LTE se muestra en la figura 13, donde se aprecia la interacción de LTE con diversos tipos de redes como GSM, UMTS, IP y otras. En color naranja se presenta los bloques funcionales que representan los elementos que conforman la SAE, los cuales son: los eNodosB, MME y SAE GW (también denominado SWG, “Serving Gateway”), en este último bloque se han incluido el PGW (PDN Gateway, “Public Data Network Gateway”).

El dicha figura las líneas continuas representan el flujo de datos por medio de las interfaces correspondientes y con líneas discontinuas se representa la señalización entre los diferentes bloques funcionales o equipos.

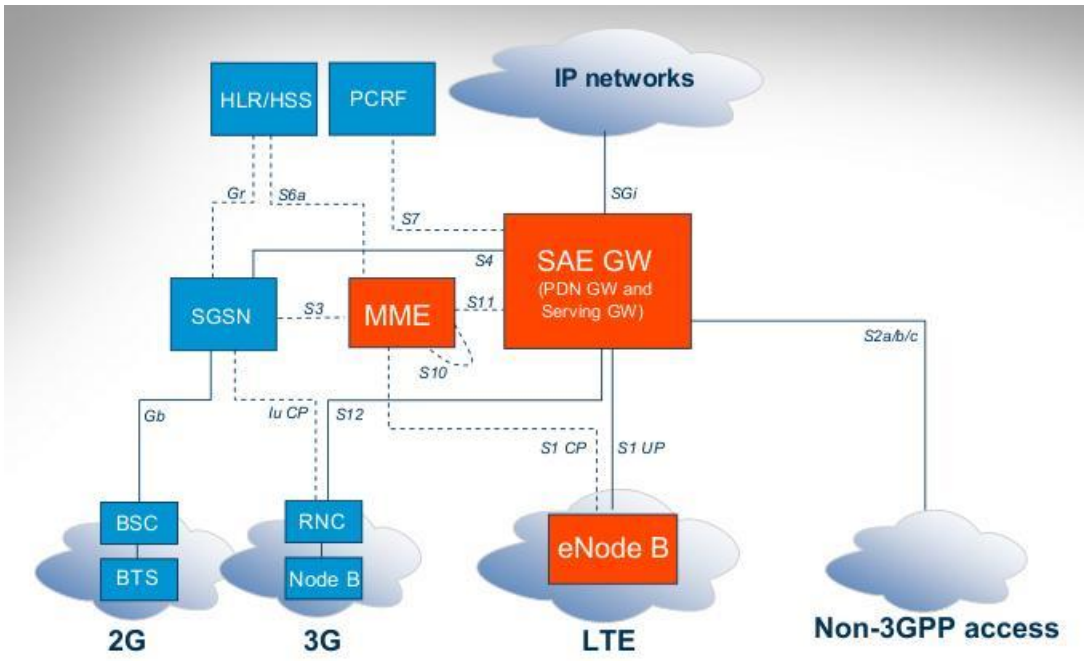


Figura 20 Arquitectura LTE interconectada con otras redes

El MME obtiene información del abonado a través de la información almacenada en el HSS para autorizar al usuario a los servicios a los que tiene acceso. El MME autentica, autoriza y selecciona el PDN (“Public Data Network”) apropiado para establecer el enlace entre el EUTRAN a las redes o servicios externos, al mismo tiempo gestiona la movilidad y obtiene información de cobro. El MME proporciona conectividad entre el eNodoB y una red GSM o UMTS a través del SGSN (“Serving GPRS Support Node”). En general se puede decir que MME tiene toda la responsabilidad por las operaciones concernientes al plano de control, además, es el primer contacto de LTE con GSM o UMTS

El SGW es controlado por el MME, es un punto donde se monitorizan las políticas de conexión y servicio establecidas en el PCRF (“Policy and Charging

Rules Function") para poder administrar QoS, además, es responsable de la organización del tráfico y los buffers para almacenamiento de paquetes. El PGW gestiona la asignación de direcciones IP a los UE ("User Equipment"), tiene que ver con todo lo relacionada a la inspección de paquete IP y realiza las funciones que en GSM realizaba el GGSN ("Gateway GPRS Support Node") pero además tiene la función decontrol de la movilidad. Por otro lado, el HSS almacena y administra todo lo relativo a los datos de suscripciones de los usuarios.

2.4 Características y componentes de la central telefónica

Bajo el esquema basado semi-hash, un servidor local CDN asigna una cierta capacidad de su espacio de disco para almacenar en caché el contenido más popular para los usuarios locales y la parte restante de cooperar con otros servidores CDN a través de una función hash. Como hash puro, semi-hash tiene pequeña sobrecarga aplicación y contenido de alta eficiencia compartir. Además, se ha encontrado que aumenta significativamente la tasa de éxito local de la CDN.

Un régimen basado en la consulta se puede utilizar para inter-cluster de almacenamiento en caché. En este enfoque, cuando un clúster falla para servir a una petición de contenido, consulta otro grupo vecino. Si el contenido se puede obtener de este vecino, responde con un mensaje de "éxito" o si no, reenvía la petición a otros grupos vecinos. Todos los servidores CDN dentro de un uso del clúster hash esquema basado para servir solicitud de contenido y el servidor CDN representante de un clúster sólo consulta al servidor designado de ese grupo para servir a una solicitud de contenido.

Por lo tanto, este sistema utiliza el esquema basado en hash para el enrutamiento de contenido intra-grupo y el esquema basado en consultas para el enrutamiento de contenido inter-cluster. Este enfoque mejora el rendimiento ya que

limita la inundación de tráfico de consultas y soluciona el problema de los retrasos al recuperar contenido desde servidores remotos mediante el uso de un tiempo de espera y TTL con cada mensaje de consulta.

CAPITULO III

DISEÑO E IMPLEMENTACIÓN

3.1 Criterios de Diseño de la Plataforma

La idea principal detrás del desarrollo de la aplicación es la siguiente: Dentro de la red de Telefónica se levantará un servidor para nuestra aplicación. Este servidor constará de los siguientes módulos:

- **Servidor Apache:** Este permitirá montar sobre él una página web que será la interfaz gráfica para el usuario donde se mostrará los datos sobre las estaciones afectadas y su ubicación en un mapa con la API de Google Maps.

- **Programa Java:** Este programa Java será el que permitirá la comunicación entre el servidor y los controles maestros de las estaciones dentro de las tres tecnologías, tanto GSM, UMTS, LTE. El programa Java mediante el protocolo telnet enviará comandos de consulta a estos elementos los cuales responderán en texto plano, el programa será el encargado también de discernir esta información venida desde los elementos de red de las tres tecnologías, hará una consulta a la base de

- datos unificada para obtener el nombre real basándose en el código que obtenga como resultado de consulta a estos elementos de red, así mismo realizará consultas para saber si la estación afectada se encuentra totalmente fuera de servicio o si tan solo un sector se encuentra sin servicio. Una vez obtenida esta información el programa se encargará de guardar una información en una base de datos MySQL.

- **Base de Datos MySQL:** La base de datos servirá para el almacenamiento de la información recolectada por el programa Java ejecutado en el servidor. Por otro lado, la página web se encargará de realizar las consultas a esta base de datos para desplegarlos en su interfaz y que el usuario final pueda observarlos.

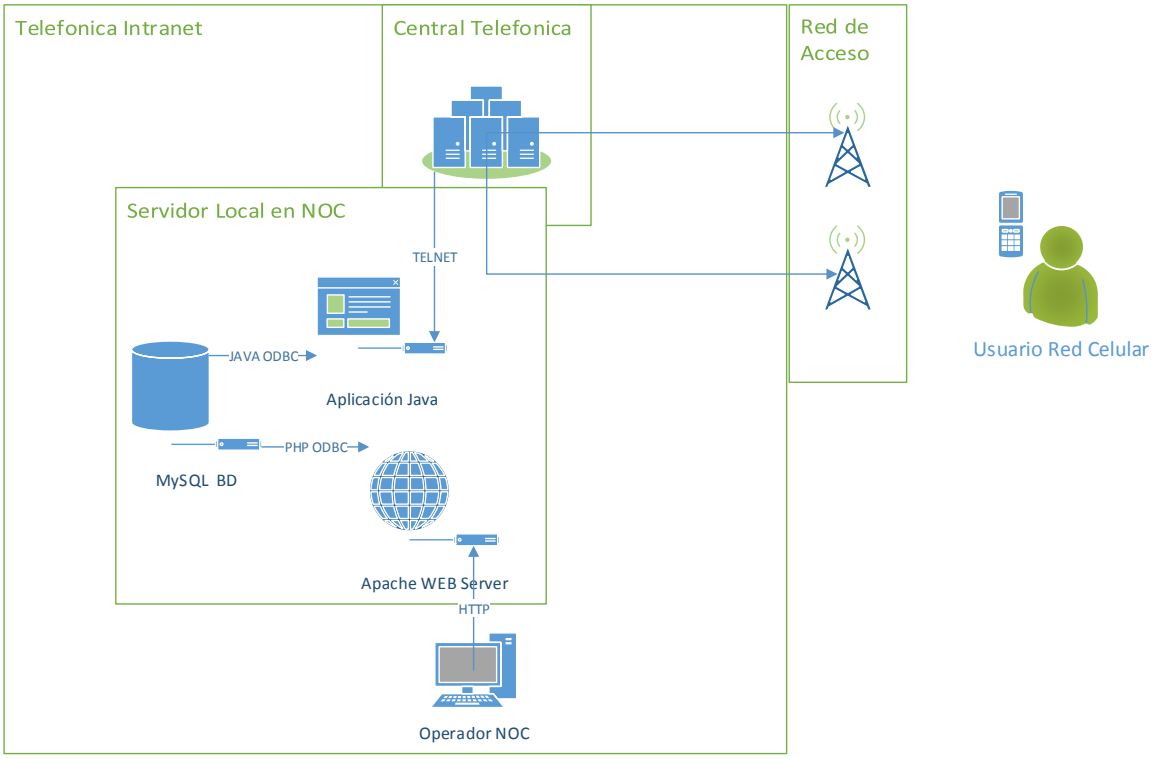


Figura 21 Esquema de la Infraestructura de la solución Implementada

Estos tres modulo implementados se comunicarán entre sí para así de esta forma dar al usuario, en este caso el operador del NOC, la información necesaria sobre las estaciones Nokia afectadas en ese momento.

El software necesario en el servidor para el funcionamiento de cada de los módulos en el servidor son los siguientes:

- **AppServ v2.5.10:** Aplicación necesaria para el montado del servidor Apache v, además instala los componenetes necesarios para la ejecución de PHP v4 en el servidor y un servidor MySQL v5.1 necesario para la base de Datos. Esta aplicación permite así de una manera rápida los componentes necesarios para levantar nuestro servidor web además de incluir herramientas de administración gráficas para el servidor MySQL y PHP como es el componente PhpMyAdmin.

- **Java JDK 7.79:** Necesario para la programación de nuestra aplicación que se encargará de recoger la información desde los elementos de la central. Así mismo se agrega el componente Java ODBC que permite la comunicación entre el código java y el servidor de base de datos MySQL que se ha instalado de manera local con el software AppServ.

Tanto para la instalación del AppServ, como del JDK no se hacen configuraciones especiales en la selección de puertos y/o ubicación de archivos en los directorios, dejando así las configuraciones por defecto.

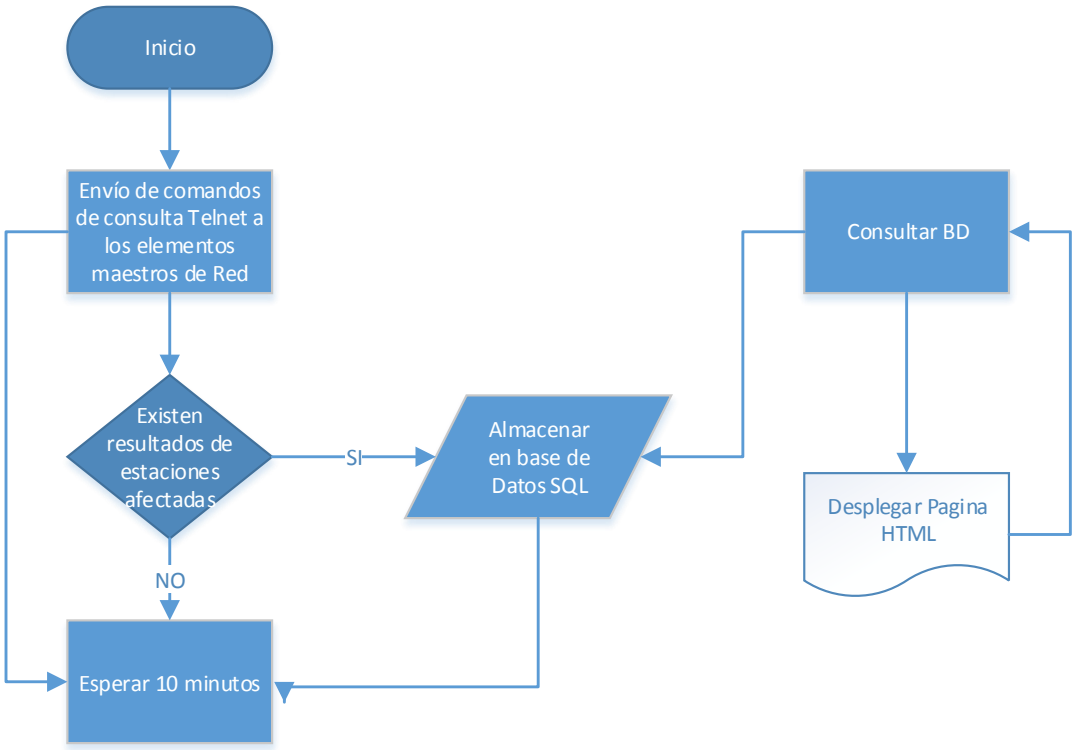


Figura 22 Diagrama de Flujo de la Plataforma

El hardware utilizado será una de los equipos utilizados por los operadores del NOC que cuentan con las siguientes características:

- Sistema Operativo Windows 7
- Memoria RAM 4GB DDR3
- Procesador Intel Core i5 32bits
- Disco Duro 1TB

Estas características son suficientes ya que la cantidad de usuarios en el NOC que utilizarán la aplicación será de máximo dos personas accediendo a ella simultáneamente.

3.2 Protocolo de Comunicación Telenet

Para la obtención de la información de las estaciones sin servicio se ha decidido escoger el protocolo de comunicación Telnet, esto debido a que el NOC solo tiene acceso a esa vía de comunicación para obtener información mediante consola de los elementos de red. Sin embargo, aunque esta parezca una opción insegura, toda la comunicación esta confinada dentro de una Intranet con altos estándares de seguridad, y ya que, nuestra aplicación será solo usada dentro de la intranet en la cual también se hayan agregados los equipos del NOC se optó por la utilización de este protocolo para la obtención de la información necesaria.

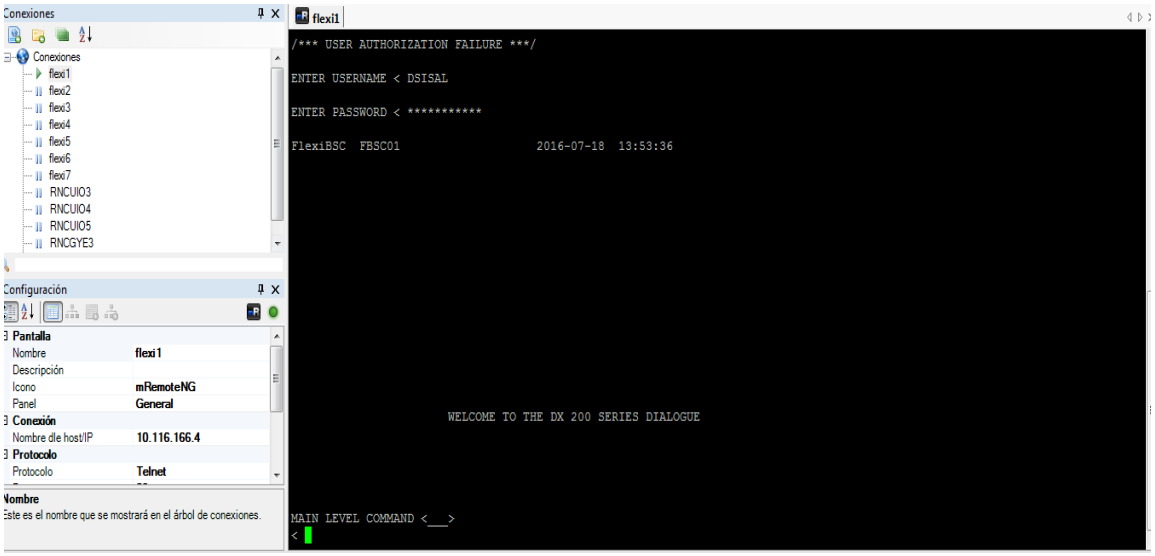


Figura 23 Conexión TELNET a una estación central GSM

Los elementos de la red Nokia, siendo estos RNCs para UMTS, Flexis para GSM o MME para LTE, manejan el sistema operativo propietario de Nokia para su manejo en consola. El operador del NOC tiene acceso a ciertos comandos los cuales permiten obtener la información del estado de la estación, los cuales se describen a continuación:

- **ZEOL:** Comando utilizado en los elementos Flexis para obtener las estaciones GSM fuera de servicio
- **ZEEI:** Comando utilizado en los elementos Flexis para obtener los sectores fuera de servicio dentro de una estación en particular
- **ZAAP:** Comando utilizado en los elementos RNC para las estaciones UMTS fuera de servicio
- **ZQRX:** Comando utilizado en los elementos MME para las estaciones LTE fuera de servicio

A estos comandos se les tiene que enviar como parámetros los números de alarma que identifican la indisponibilidad para una tecnología. Los números de alarma que describen la indisponibilidad del servicio son:

- 7786: Fuera de Servicio de datos UMTS.
- 7771: Perdida de conexión a la estación UMTS.
- 7653: Perdida de tráfico de voz en estación UMTS
- 7750: Niveles altos de caídas de llamadas en la estación UMTS
- 7767: Perdida de servicio en estación GSM

Por otro lado, el comando ZQRX realiza una prueba de ping a la estación LTE para verificar la conectividad y disponibilidad de la estación.

Obteniendo ya estos datos podemos realizar una consulta mediante telnet a los elementos de la central para saber que estaciones se encuentra fuera de servicio con los siguientes comandos:

Tabla 1

Comandos utilizados para revisión por fuera de servicio

Tecnología	Comando
GSM	ZEOL::NR = 7767;
	ZEEI:BCF=BCF_NUMBER;
UMTS	ZAAP::NR=7786;
	ZAAP::NR=7771;
	ZAAP::NR=7653;
	ZAAP::NR=7750;
LTE	ZQRX:IPDU,1::PING;

3.3 Aplicación de monitorización

3.3.1 Diseño de la aplicación Java

La aplicación Java será la encargada de realizar conexiones recurrentes a los elementos de central de la red y hacerles consultas mediante el protocolo Telnet, en base a la información generada por estos elementos el programa determinará que estaciones se encuentra fuera de servicio en que tecnología, y si estas se encuentran por fuera de servicio solo un sector y toda la estación.

El programa se ha desarrolla usando el JDK 2.7 al cual además se le ha agregado la librería *commons-net-3.3* la incluye las clases necesarias para realizar las conexiones Telnet con los elementos de las centrales. Además, se añadió la librería *mysql-connector-java-5.1.22* esta incluye clases que permiten las comunicaciones del programa Java con las bases de datos MySQL.

A continuación se realizara la descripción de las clases programa java que son las siguientes:

- **NOKIA_DS.class**

Esta es la clase principal de la aplicación Java. Esta clase permite que se haga un continuo llamado a las centrales mediante Telnet para ir almacenado la información en variables temporales. La información recibida será procesada para ser almacenada en la base de Datos MySQL

```
public class NOKIA_DS {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here

        Connection miConexion;
        miConexion=ConexionBD.GetConnection();
        if(miConexion!=null)
            System.out.println("Listado de Estaciones\n");

        //while(true){
        Conexion_Nokia A = new Conexion_Nokia();
        String[] S = new String [22];
        Datos_Conexion B = new Datos_Conexion();
        Estaciones E = new Estaciones (miConexion);
        int tam = 300;

        MME_Thread MME_Conexion = new MME_Thread();


            Data_Estaciones[] UQ3 = new Data_Estaciones[tam];
            Data_Estaciones[] UQ4 = new Data_Estaciones[tam];
            Data_Estaciones[] UQ5 = new Data_Estaciones[tam];
            Data_Estaciones[] UG3 = new Data_Estaciones[tam];

            Data_Estaciones[] UQ3_C = new Data_Estaciones[tam];
            Data_Estaciones[] UQ4_C = new Data_Estaciones[tam];
            Data_Estaciones[] UQ5_C = new Data_Estaciones[tam];
            Data_Estaciones[] UG3_C = new Data_Estaciones[tam];

            Data_Estaciones[] UQ3_CF = new Data_Estaciones[tam];
            Data_Estaciones[] UQ4_CF = new Data_Estaciones[tam];
            Data_Estaciones[] UQ5_CF = new Data_Estaciones[tam];
            Data_Estaciones[] UG3_CF = new Data_Estaciones[tam];

            Data_Estaciones[] UQ3_CB = new Data_Estaciones[tam];
            Data_Estaciones[] UQ4_CB = new Data_Estaciones[tam];
            Data_Estaciones[] UQ5_CB = new Data_Estaciones[tam];
            Data_Estaciones[] UG3_CB = new Data_Estaciones[tam];

            Data_Estaciones[] F1 = new Data_Estaciones [tam];
            Data_Estaciones[] F2 = new Data_Estaciones [tam];
            Data_Estaciones[] F3 = new Data_Estaciones [tam];
            Data_Estaciones[] F5 = new Data_Estaciones [tam];
```

Figura 24 Clase NOKIA_DS. Primera Parte

```
        E.vaciarBD();

        UQ3 = E.separarAlarmas(S[0]); // rncuio3
        UQ4 = E.separarAlarmas(S[1]); // rncuio4
        UQ5 = E.separarAlarmas(S[2]); // rncuio5
        UG3 = E.separarAlarmas(S[3]); // rncgye5

        UQ3_C = E.separarAlarmasC(S[16]); // rncuio3
        UQ4_C = E.separarAlarmasC(S[17]); // rncuio4
        UQ5_C = E.separarAlarmasC(S[18]); // rncuio5
        UG3_C = E.separarAlarmasC(S[19]); // rncgye5

        F1 = E.separarAlarmas2G(S[4]); //flexi1
        F2 = E.separarAlarmas2G(S[5]); //flexi2
        F3 = E.separarAlarmas2G(S[6]); //flexi3
        F5 = E.separarAlarmas2G(S[7]); //flexi5
        F6 = E.separarAlarmas2G(S[20]); //flexi5
        F7 = E.separarAlarmas2G(S[21]); //flexi5

        UQ3_CF = E.separarAlarmasSt(S[8]); //rncuio3 7771
        UQ3_CB = E.separarAlarmasSt(S[9]); //rncuio3 7653
        UQ4_CF = E.separarAlarmasSt(S[10]); //rncuio4 7771
        UQ4_CB = E.separarAlarmasSt(S[11]); // rncuio4 7653

        E.comprobarEst();
        E.limpiarSectores();
        E.resumirAlarmas();
        E.resumirAlarmas2G();

        try {
            Thread.sleep(100000);                //1000 milliseconds is one second.
        } catch(InterruptedException ex) {
            Thread.currentThread().interrupt();
        }
    }
    // }
}
```

Figura 24. Clase NOKIA_DS. Segunda Parte

- **Cliente_Telnet.class**

La clase Cliente_Telnet tiene las funciones necesarias para la comunicación Telnet de la aplicación con las centrales GSM, UMTS, LTE.

Esta incluye los comandos a ser enviados y recibe como parámetros la IP de la central a la que debe conectarse para realizar la consulta. La salida de esta clase es un texto plano donde se incluye toda la información resultante de la ejecución de los comandos de consulta.

```
public class Cliente_Telnet {

    private Connection Conector;
    private TelnetClient telnet = new TelnetClient();
    private BufferedReader reader;
    private InputStream in;
    private PrintStream out;
    public static String promptComplete;

    public Cliente_Telnet(){

    }

    String conectar(String ip_servidor, String Usuario, String Pass, String tipo) {

        String preData = new String();
        String comando = new String();

        try {
            promptComplete = "[" + ip_servidor + "][" + Usuario + " =>";
            telnet.connect(ip_servidor, 23);

            in = telnet.getInputStream();
            out = new PrintStream(telnet.getOutputStream());
            reader = new BufferedReader(new InputStreamReader(in));

            readUntil("ENTER USERNAME <");
            write(Usuario);

            readUntil("ENTER PASSWORD <");
            write(Pass);

            if (tipo.equalsIgnoreCase("RNC"))
                comando = "ZAAP::NR=7786;";
            else if(tipo.equalsIgnoreCase("Flexi"))
                comando = "ZEOL::NR=7767,:;";
            else if (tipo.equalsIgnoreCase("RNC_CF"))
                comando = "ZAAP::NR=7771;";
            else if (tipo.equalsIgnoreCase("RNC_CB"))
                comando = "ZAAP::NR=7653;";
            else if (tipo.equalsIgnoreCase("RNC_C"))
                comando = "ZAAP::NR=7750;";
            else {
                comando = "ZEEI:BCF="+tipo+";";
            }
```

Figura 25 Clase Cliente_Telnet. Primera Parte

```
            write(comando);
            write("Z;");
            write("Z;");
            preData = readUntil(promptComplete);

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                //Liberamos recursos
                out.close();
                reader.close();
                in.close();
                telnet.disconnect();

            } catch (IOException ex) {
                Logger.getLogger(Cliente_Telnet.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        return (preData);
    }
```

```
    String[] conectar_MME() {

        String Hora_MME = new String();
        String Fecha_MME = new String();
        Connection miConexion;
        miConexion=ConexionBD.GetConnection();

        Estaciones E = new Estaciones(miConexion);
        if(miConexion!=null)
            System.out.println("Conexion LTE. BD\n");
        this.Conector = miConexion;

        Datos_Conexion Datos = new Datos_Conexion();

        ResultSet Resultado = null;
        Statement comando_sql = null;
        String preData = new String();
        String[] results = new String[300];
```

```
        try {
            promptComplete = "[" + Datos.getIp_MME()+ "][" + Datos.getUsuario_MME() + " =>";
            telnet.connect(Datos.getIp_MME(), 23);
            in = telnet.getInputStream();
            out = new PrintStream(telnet.getOutputStream());
            reader = new BufferedReader(new InputStreamReader(in));

            readUntil("ENTER USERNAME <");
            write(Datos.getUsuario_MME());

            readUntil("ENTER PASSWORD <");
            write(Datos.getPass_MME());

            preData = this.readUntil("< ");
            //System.out.println(preData);
            Fecha_MME = preData.substring(51, 61);
            Hora_MME = preData.substring(63, 71);
            E.guardaHoraElemento("MME", Fecha_MME+"  "+Hora_MME);
            //aqui va la magia donde se va a solicitar un comando para barrido de todas las i
            //String
```

Figura 25. Clase Cliente_Telnet. Segunda Parte

```
try {
    promptComplete = "[" + Datos.getIp_MME()+ "][" + Datos.getUsuario_MME() + " =>";
    telnet.connect(Datos.getIp_MME(), 23);
    in = telnet.getInputStream();
    out = new PrintStream(telnet.getOutputStream());
    reader = new BufferedReader(new InputStreamReader(in));

    readUntil("ENTER USERNAME <");
    write(Datos.getUsuario_MME());

    readUntil("ENTER PASSWORD <");
    write(Datos.getPass_MME());

    preData = this.readUntil("< ");
    //System.out.println(preData);
    Fecha_MME = preData.substring(51, 61);
    Hora_MME = preData.substring(63, 71);
    E.guardaHoraElemento("MME", Fecha_MME+"  "+Hora_MME);
    //aqui va la magia donde se va a solicitar un comando para barrido de todas las ip
    //String
```

```
    String codigoSql = "SELECT * from sitios_nokia WHERE SECTORS = 0";
    int i = 0;
    try{
        comando_sql = Conector.createStatement();
        Resultado = comando_sql.executeQuery(codigoSql);
        while (Resultado.next()){
            write(Datos.getComando_MME(Resultado.getString("UBICACION")));
            preData = this.readUntil("< ");
            //System.out.println(preData);
            if(preData.contains("100% packet loss")){
                System.out.println("Entro por: "+Resultado.getString("NOMBRES"));
                //System.out.println(preData);
                codigoSql = "INSERT INTO estaciones_4g "
                        + "(`ID`, `UBICACION`, `NOMBRE`, `HORA`, `FECHA`,`ACK`) "
                        + "VALUES ('"+Resultado.getString("ID")+"', '"+
                        Resultado.getString("UBICACION")+"', '"+
                        Resultado.getString("NOMBRES")+"', '"+
                        Hora_MME+"', '"+Fecha_MME+"','NO');";
```

```
                try{
                    comando_sql = Conector.createStatement();
                    comando_sql.executeUpdate(codigoSql);
                }catch (SQLException ex) {
                    //System.out.println("ERROR AL GRABAR LTE");
                }
                  i++;
            }
            else if(preData.contains(" 0% packet loss")){
                //System.out.println("Entro por: "+Resultado.getString("NOMBRES"));
                //System.out.println(preData);
                codigoSql = "DELETE from estaciones_4g WHERE ID='"+Resultado.getString("ID")+"';";
                //System.out.println("codigo SQL"+codigoSql);
                try{
                    comando_sql = Conector.createStatement();
                    comando_sql.executeUpdate(codigoSql);
                }catch (SQLException ex) {
                    //System.out.println("ERROR AL BORRAR LTE"+Resultado.getString("NOMBRES"));
                }
                i++;
            }
```

Figura 25. Clase Cliente_Telnet. Tercera Parte

- **ConexionBD.class**

Esta clase permite la conexión con la base de Datos MySQL, contiene la información necesaria para la conexión y los comandos SQL a ejecutarse para guardar información en la misma.

```
public class ConexionBD {

    public static Connection GetConnection()
    {
        Connection conexion=null;

        try
        {
            Class.forName("com.mysql.jdbc.Driver");
            String servidor = "jdbc:mysql://localhost/estaciones";
            String usuarioDB="root";
            String passwordDB="root";
            conexion= DriverManager.getConnection(servidor,usuarioDB,passwordDB);
        }
        catch(ClassNotFoundException ex)
        {
            JOptionPane.showMessageDialog(null, ex,
                    "Error1 en la Conexión con la BD "+ex.getMessage(),
                    JOptionPane.ERROR_MESSAGE);
            conexion=null;
        }

        catch(SQLException ex)
        {
            JOptionPane.showMessageDialog(null, ex,
                    "Error2 en la Conexión con la BD "+ex.getMessage(),
                    JOptionPane.ERROR_MESSAGE);
            conexion=null;
        }
        catch(Exception ex)
        {
            JOptionPane.showMessageDialog(null, ex,
                    "Error3 en la Conexión con la BD "+ex.getMessage(),
                    JOptionPane.ERROR_MESSAGE);
            conexion=null;
        }
        finally
        {
            return conexion;
        }
    }
}
```

Figura 26 Clase ConexionBD

- **Conexión_Nokia.class**

Esta clase contiene la conexión a cada una de las centrales, esto se realizó para no hacer repetitivo la creación de una clase Cliente_Telnet por cada central en la clase principal, ya que de otra manera de debería especificar los datos de conexión uno por uno

```
public class Conexion_Nokia {

    String [] preDatos;

    public String[] Conectar (Datos_Conexion A){

        Datos_Conexion Datos = new Datos_Conexion();
        Cliente_Telnet Cliente = new Cliente_Telnet();
        preDatos = new String[22];

        //RNCs
        //Quito 3
        preDatos[0] = Cliente.conectar(Datos.getIp_RNCUIO3(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC");
        //Quito 4
        preDatos[1] = Cliente.conectar(Datos.getIp_RNCUIO4(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC");
        //Quito 5
        preDatos[2] = Cliente.conectar(Datos.getIp_RNCUIO5(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC");
        //Gye 3
        preDatos[3] = Cliente.conectar(Datos.getIp_RNCGYE3(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC");

         preDatos[16] = Cliente.conectar(Datos.getIp_RNCUIO3(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_C");
```

```
         preDatos[16] = Cliente.conectar(Datos.getIp_RNCUIO3(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_C");
        //Quito 4
        preDatos[17] = Cliente.conectar(Datos.getIp_RNCUIO4(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_C");
        //Quito 5
        preDatos[18] = Cliente.conectar(Datos.getIp_RNCUIO5(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_C");
        //Gye 3
        preDatos[19] = Cliente.conectar(Datos.getIp_RNCGYE3(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_C");

        //flexis
        //flexi 1
        preDatos[4] = Cliente.conectar(Datos.getIp_flexi1(), Datos.getUsuario_flexi(), Datos.getPass_flexi(), "Flexi");
        //flexi 2
        preDatos[5] = Cliente.conectar(Datos.getIp_flexi2(), Datos.getUsuario_flexi(), Datos.getPass_flexi(), "Flexi");
        //flexi 3
        preDatos[6] = Cliente.conectar(Datos.getIp_flexi3(), Datos.getUsuario_flexi(), Datos.getPass_flexi(), "Flexi");
        System.out.println("PreDatos*****************"+preDatos[6]);
        //flexi 5
        preDatos[7] = Cliente.conectar(Datos.getIp_flexi5(), Datos.getUsuario_flexi(), Datos.getPass_flexi(), "Flexi");

        preDatos[20] = Cliente.conectar(Datos.getIp_flexi6(), Datos.getUsuario_flexi(), Datos.getPass_flexi(), "Flexi");

        preDatos[21] = Cliente.conectar(Datos.getIp_flexi7(), Datos.getUsuario_flexi(), Datos.getPass_flexi(), "Flexi");
```

Figura 27 Clase Conexión_Nokia

```
        preDatos[8] = Cliente.conectar(Datos.getIp_RNCUIO3(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_CF");
        //Quito 4
        preDatos[10] = Cliente.conectar(Datos.getIp_RNCUIO4(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_CF");
        //Quito 5
        preDatos[12] = Cliente.conectar(Datos.getIp_RNCUIO5(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_CF");
        //Gye 3
        preDatos[14] = Cliente.conectar(Datos.getIp_RNCGYE3(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_CF");

        //comando RNC 7653
        //Quito 3
        preDatos[9] = Cliente.conectar(Datos.getIp_RNCUIO3(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_CB");
        //Quito 4
        preDatos[11] = Cliente.conectar(Datos.getIp_RNCUIO4(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_CB");
        //Quito 5
        preDatos[13] = Cliente.conectar(Datos.getIp_RNCUIO5(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_CB");
        //Gye 3
        preDatos[15] = Cliente.conectar(Datos.getIp_RNCGYE3(), Datos.getUsuario_RNC(), Datos.getPass_RNC(), "RNC_CB");

        return (preDatos);
    }
}
```

Figura 27. Clase Conexión_Nokia. Segunda Parte

- **Data_Estaciones.class**

Aquí se guardará de manera temporal la información de las estaciones afectadas antes de ser enviados a almacenar en la base de datos. Las informaciones de las estaciones se las obtienen luego de validar el texto plano del resultado de los comandos de consulta enviado por la aplicación.

```
public class Data_Estaciones {

    private String ID;
    private String cellID;
    private String Nombre;
    private String Fecha;
    private String Hora;
    private String Ubicacion;
    private String Tecnologia;
    private String Latitud;
    private String Longitud;


    public void Imprimir (){
        System.out.println("ID: "+this.ID);
        System.out.println("Cell ID: "+this.cellID);
        System.out.println("Ubicacion: "+this.Ubicacion);
        System.out.println("Nombre: "+this.Nombre);
        System.out.println("Hora: "+this.Hora);
```

Figura 28 Clase Data_Estaciones

```
        System.out.println("Hora: "+this.Hora);
        System.out.println("Fecha: "+this.Fecha);
        System.out.println("Tecnologia: "+this.Tecnologia + "\n");
        System.out.println("Lat: "+this.Latitud + "\n");
        System.out.println("Lng: "+this.Longitud + "\n");

    }

    public String getID() {
        return ID;
    }

    public void setID(String ID) {
        this.ID = ID;
    }

    public String getCellID() {
        return cellID;
    }

    public void setCellID(String cellID) {
        this.cellID = cellID;

    public void setNombre(String Nombre) {
        this.Nombre = Nombre;
    }

    public String getFecha() {
        return Fecha;
    }

    public void setFecha(String Fecha) {
        this.Fecha = Fecha;
    }

    public String getHora() {
        return Hora;
    }

    public void setHora(String Hora) {
        this.Hora = Hora;
    }

    public String getLongitud() {
        return Longitud;
    }

    public void setLongitud(String Longitud) {
        this.Longitud = Longitud;
    }
}
```

Figura 28. Clase Data_Estaciones. Segunda Parte

- **Datos_Conexion.class**

Contiene la información necesaria para realizar las conexiones a las diferentes centrales mediante los comandos Telnet. Entre la información necesaria se incluye la IP del elemento y el comando a ejecutar en cada elemento.

```
public class Datos_Conexion {

    private final String comando_RNC;
    private final String comando_FLEXI;
    private final String ip_flexi1;
    private final String ip_flexi2;
    private final String ip_flexi3;
    private final String ip_flexi4;
    private final String ip_flexi5;
    private final String ip_flexi6;
    private final String ip_flexi7;
    private final String ip_RNCUIO3;
    private final String ip_RNCUIO4;
    private final String ip_RNCUIO5;
    private final String ip_RNCGYE3;
    private final String ip_MMECLD; //ok
    private final String usuario_flexi;
    private final String usuario_RNC;
    private final String usuario_MME;
    private final String pass_flexi;
    private final String pass_RNC;
    private final String pass_MME;
```

```
    public String getComando_RNC() {
        return comando_RNC;
    }

    public String getComando_FLEXI() {
        return comando_FLEXI;
    }

    public String getComando_MME(String IP_eNodeB){
        return ("ZQRX:IPDU,1::PING:IP=\""+IP_eNodeB+"\",SRC=\"10.116.80.2\",SIZ=1500;");
    }

    public String getIp_flexi1() {
        return ip_flexi1;
    }

    public String getIp_flexi2() {
        return ip_flexi2;
    }

    public String getIp_flexi3() {
        return ip_flexi3;
```

Figura 29 Clase Data_Conexion.

- **Estaciones.class**

Esta clase es la que validará el texto recibido como respuesta de la ejecución del comando Telnet. Aquí se realizará un análisis de las cadenas de caracteres para extraer el código de la estación afectada, así como la tecnología a la que pertenece, la hora a la que se presentó la alarma, la hora de la lectura de la alarma y el elemento central a la que se conecta directamente.

```
public class Estaciones {

    private Connection Conector;

    public Estaciones(Connection Conector) {
        this.Conector = Conector;
    }

    public Data_Estaciones [] separarAlarmas (String preData){

        String hora = null;
        if (preData.contains("EXCEDED")==false)
            hora = preData.substring(53,73);

        String Ubicacion = preData.substring(preData.indexOf("RNC")+10, preData.indexOf("RNC")+17);

        if (hora == null)
            System.out.println("No se actualiza hora en el elemento");
        else
            this.guardaHoraElemento(Ubicacion, hora);
```

```
        Data_Estaciones[] Estacion = new Data_Estaciones[300];
        String[] alarmas = new String [200];
        preData = preData.substring(296); //300 es el tamaño de la cabecera
        int tam_alarma = 349; //es el tamañan de cada bloque de alarma
        int total = preData.indexOf("TOTAL");

        if (total == -1){
            Estacion[0] = null ;
            return (Estacion);
        }

        String T = preData.substring(total+15, total+20).trim();
        int num = Integer.parseInt(T); //numero de alarmas obtenidos

        preData = preData.substring(preData.indexOf("("),preData.indexOf("NUMBER"));
        preData = preData.trim();
        preData = "    ".concat(preData);
        preData = preData.concat(" \n\n");

        int cursor = 0;
```

Figura 30 Clase Estaciones

```
        for (int i = 0; i < num; i++){
            alarmas[i] = preData.substring(cursor, tam_alarma*(i+1)-2);
            alarmas[i] = alarmas[i].trim();
            cursor = tam_alarma*(i+1);
            Estacion[i] = this.convertir3G(alarmas[i], Ubicacion, "F");
            Estacion[i].Imprimir();
        }

        return (Estacion);
    }

    public Data_Estaciones [] separarAlarmas2G (String preData){

        String hora = null;
        if (preData.contains("EXCEDED")==false){
            //hora = preData.substring(53,73);
            hora = "zzzz";
            //System.out.println("Esta es la hora: "+hora);
        }

        String Ubicacion = preData.substring(preData.indexOf("FBS"), preData.indexOf("FBS")+6);

        if (hora == null)
            System.out.println("No se actualiza hora en el elemento");
        else
            this.guardaHoraElemento(Ubicacion, hora);


        Data_Estaciones[] Estacion = new Data_Estaciones[300];
        String[] alarmas = new String [200];
        int tam_alarma = 218; //es el tamañan de cada bloque de alarmas
        int num_caidas = 0;

        if (preData.indexOf("END OF BTS") == -1 ){
            Estacion [0] = null;
            return (Estacion);
        }

        preData = preData.substring(preData.indexOf("LISTING")+11,preData.indexOf("END OF BTS")-2);
        num_caidas = preData.length()/tam_alarma;

        int cursor = 0;

        for (int i = 0; i < num_caidas; i++){
            alarmas[i] = preData.substring(cursor, tam_alarma*(i+1)-2);

        preData = preData.substring(preData.indexOf("LISTING")+11,preData.indexOf("END OF BTS")-2);
        num_caidas = preData.length()/tam_alarma;

        int cursor = 0;

        for (int i = 0; i < num_caidas; i++){
            alarmas[i] = preData.substring(cursor, tam_alarma*(i+1)-2);
            alarmas[i] = alarmas[i].trim();
            cursor = tam_alarma*(i+1);
            Estacion[i] = this.convertir2G(alarmas[i], Ubicacion);
            Estacion[i].Imprimir();
        }

        return (Estacion);
    }
```

Figura 30 Clase Estaciones. Segunda Parte

```
public Data_Estaciones [] separarAlarmasC(String preData){

    String hora = null;
    if (preData.contains("EXCEDED")==false){
        hora = preData.substring(53,73);
        //System.out.println("Esta es la hora: "+hora);
    }

    String Ubicacion = preData.substring(preData.indexOf("RNC")+10, preData.indexOf("RNC")+17);

    if (hora == null)
        System.out.println("No se actualiza hora en el elemento");
    else
        this.guardaHoraElemento(Ubicacion+"_S", hora);

    Data_Estaciones[] Estacion = new Data_Estaciones[300];
    String[] alarmas = new String [200];
    preData = preData.substring(296); //300 es el tamaño de la cabecera
    int tam_alarma = 434; //es el tamañan de cada bloque de alarma
    int total = preData.indexOf("TOTAL");

    if (total == -1){
        Estacion[0] = null ;
        return (Estacion);
    }

    String T = preData.substring(total+15, total+20).trim(); //numero total de alarmas desc
    int num = Integer.parseInt(T); //numero de alarmas obtenidos

    preData = preData.substring(preData.indexOf("("),preData.indexOf("NUMBER")-4);
    preData = "    ".concat(preData);
    preData = preData.concat(" \n\n");

    int i = 0;
    int bandera = 0;
    while (bandera == 0){
        alarmas[i] = preData.substring(preData.indexOf("("), preData.indexOf("WBTS")+10);
        alarmas[i] = alarmas[i].trim();
        preData = preData.substring(preData.indexOf("(")+1);
        if (preData.indexOf("(")==-1)
            bandera = 1;
```

```
public Data_Estaciones [] separarAlarmasC(String preData){

    String hora = null;
    if (preData.contains("EXCEDED")==false){
        hora = preData.substring(53,73);
        //System.out.println("Esta es la hora: "+hora);
    }

    String Ubicacion = preData.substring(preData.indexOf("RNC")+10, preData.indexOf("RNC")+17);

    if (hora == null)
        System.out.println("No se actualiza hora en el elemento");
    else
        this.guardaHoraElemento(Ubicacion+"_S", hora);

    Data_Estaciones[] Estacion = new Data_Estaciones[300];
    String[] alarmas = new String [200];
    preData = preData.substring(296); //300 es el tamaño de la cabecera
    int tam_alarma = 434; //es el tamañan de cada bloque de alarma
    int total = preData.indexOf("TOTAL");
```

Figura 30 Clase Estaciones. Tercera Parte

```
    String T = preData.substring(total+15, total+20).trim(); //numero total de alarmas desc
    int num = Integer.parseInt(T); //numero de alarmas obtenidos

    preData = preData.substring(preData.indexOf("("),preData.indexOf("NUMBER")-4);
    preData = "    ".concat(preData);
    preData = preData.concat(" \n\n");

    int i = 0;
    int bandera = 0;
    while (bandera == 0){
        alarmas[i] = preData.substring(preData.indexOf("("), preData.indexOf("WBTS")+10);
        alarmas[i] = alarmas[i].trim();
        preData = preData.substring(preData.indexOf("(")+1);
        if (preData.indexOf("(")==-1)
            bandera = 1;
        else
            preData = preData.substring(preData.indexOf("("));

        Estacion[i] = this.convertir3G(alarmas[i], Ubicacion, "C");
        Estacion[i].Imprimir();
        i++;
    }
```

```
public Data_Estaciones [] separarAlarmasSt(String preData){

    String hora = null;
    if (preData.contains("EXCEDED")==false){
        hora = preData.substring(53,73);
        //System.out.println("Esta es la hora: "+hora);
    }

    String Ubicacion = preData.substring(preData.indexOf("RNC")+10, preData.indexOf("RNC")+17);

    if (hora == null)
        System.out.println("No se actualiza hora en el elemento");
    else
        this.guardaHoraElemento(Ubicacion+"_S", hora);

    Data_Estaciones[] Estacion = new Data_Estaciones[300];
    String[] alarmas = new String [200];
    preData = preData.substring(296); //300 es el tamaño de la cabecera
    int tam_alarma = 434; //es el tamañan de cada bloque de alarma
    int total = preData.indexOf("TOTAL");
```

```
    if (total == -1){
        Estacion[0] = null ;
        //System.out.println("No hay alarmas");
        return (Estacion);
    }

    String T = preData.substring(total+15, total+20).trim(); //numero total de alarm
    int num = Integer.parseInt(T); //numero de alarmas obtenidos

    preData = preData.substring(preData.indexOf("("),preData.indexOf("NUMBER")-4);
    preData = "    ".concat(preData);
    preData = preData.concat(" \n\n");

    int cursor = 0;

    int i = 0;
```

Figura 30 Clase Estaciones. Cuarta Parte

```
        while (bandera == 0){
            alarmas[i] = preData.substring(preData.indexOf("("), preData.indexOf("WCEL")+10);
            alarmas[i] = alarmas[i].trim();
            preData = preData.substring(preData.indexOf("(")+1);
            if (preData.indexOf("(")==-1)
                bandera = 1;
            else
                preData = preData.substring(preData.indexOf("("));

            Estacion[i] = this.convertirSt(alarmas[i], Ubicacion);
            Estacion[i].Imprimir();
            i++;
        }


        return (Estacion);
    }
```

```
    public Data_Estaciones convertir2G (String alarma, String Ubicacion){


        Data_Estaciones Dato = new Data_Estaciones();

        Dato.setID(alarma.substring(18, 21));
        Dato.setCellID(alarma.substring(28, 31)); //!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        Dato.setFecha(alarma.substring(45, 55));
        Dato.setHora(alarma.substring(57, 68));
        Dato.setTecnologia("GSM");
        Dato.setUbicacion(Ubicacion);
        Dato.setNombre(this.setNombre(Dato));
        Dato.setLatitud(this.setlat(Dato)); // añadido
        Dato.setLongitud(this.setlng(Dato)); //añadido

        // !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
        //this.comprobarEst(Dato);

        this.guardarEstacion2G(Dato);

        return (Dato);
    }
```

```
    public void comprobarEst (){
        String Ubicacion = new String();
        String ID = new String();
        String Nombre = new String();
        Datos_Conexion Datos = new Datos_Conexion();
        Cliente_Telnet Cliente = new Cliente_Telnet();
        String preData = new String();
        ResultSet Resultado = null;

        Statement comando = null;
        String codigoSql = new String();

        //codigoSql = "INSERT INTO estaciones_3gt (`ID`, `UBICACION`, `NOMBRE`, `HO
        codigoSql = "SELECT UBICACION, BCF, NOMBRE FROM estaciones_2g";
        System.out.println("codigo SQL: "+codigoSql);
        try {
            comando = Conector.createStatement();
            Resultado = comando.executeQuery(codigoSql);
            while(Resultado.next()){

                Ubicacion = Resultado.getString("Ubicacion");
                ID = Resultado.getString("BCF");
```

Figura 30 Clase Estaciones. Quinta Parte

```
            Nombre = Resultado.getString("NOMBRE");
            System.out.println("ubicacio--"+Ubicacion+"BCF----"+ID+"Nombre-----"+Nombre);
            if(Ubicacion.equalsIgnoreCase("FBSC01"))
                preData = Cliente.conectar(Datos.getIp_flexi1(), Datos.getUsuario_flexi(), Datos.getPass_flexi(), ID);
            else if(Ubicacion.equalsIgnoreCase("FBSC02"))
                preData = Cliente.conectar(Datos.getIp_flexi2(), Datos.getUsuario_flexi(), Datos.getPass_flexi(), ID);
            else if(Ubicacion.equalsIgnoreCase("FBSC03"))
                preData = Cliente.conectar(Datos.getIp_flexi3(), Datos.getUsuario_flexi(), Datos.getPass_flexi(), ID);
            else if(Ubicacion.equalsIgnoreCase("FBSC05"))
                preData = Cliente.conectar(Datos.getIp_flexi5(), Datos.getUsuario_flexi(), Datos.getPass_flexi(), ID);

            System.out.println("Nombre: "+Nombre+"Valor = "+this.comprobarTrx(preData));
            if(this.comprobarTrx(preData)){
                codigoSql = "DELETE FROM estaciones_2g WHERE NOMBRE = '"+Nombre+"';";
                System.out.println("codigo SQL: "+codigoSql);
                try {
                    comando = Conector.createStatement();
                    //Resultado = comando.executeQuery(codigoSql);
                    comando.executeUpdate(codigoSql);
                    }
```

```
                catch (SQLException ex) {
                    Logger.getLogger(Estaciones.class.getName()).log(Level.SEVERE, null, ex);
                    System.out.println("Ha ocurrido un error al grabar la estacion");
                }
            }
        }
    } catch (SQLException ex) {
        Logger.getLogger(Estaciones.class.getName()).log(Level.SEVERE, null, ex);
        System.out.println("Ha ocurrido un error al grabar la estacion");
    }
}
```

```
public boolean comprobarTrx (String preData){
    boolean Resultado = false;
    String Aux;
    int contador=0;

    preData = preData.substring(preData.indexOf("RADIO NETWORK"));
    while(preData.contains("TRX")) {
        //System.out.println("Esta es el preData de sector\n"+preData);
        preData = preData.substring(preData.indexOf("TRX")+8);
        Aux = preData.substring(0, 8);
        if (Aux.contains("U WO")){
            Resultado = true;
            contador++;
        }
    }
    //System.out.println("Esta es el preData de sector\n"+preData);

    return(Resultado);
}
```

Figura 30 Clase Estaciones. Sexta Parte

```
public Data_Estaciones convertir3G (String alarma, String Ubicacion, String Tipo_Alarma){
    Data_Estaciones Dato = new Data_Estaciones();

    Dato.setID(alarma.substring(95, 99));
    Dato.setCellID("TODA");
    Dato.setFecha(alarma.substring(54, 64));
    Dato.setHora(alarma.substring(65, 76));
    Dato.setTecnologia("UMTS");
    Dato.setUbicacion(Ubicacion);
    Dato.setNombre(this.setNombre(Dato));
    Dato.setLatitud(this.setlat(Dato)); // añadido
    Dato.setLongitud(this.setlng(Dato)); //añadido
    this.guardarEstacion(Dato, Tipo_Alarma);

    return (Dato);
}
```

```
public Data_Estaciones convertirSt (String alarma, String Ubicacion){
    Data_Estaciones Dato = new Data_Estaciones();

    Dato.setID(alarma.substring(95, 99));
    Dato.setCellID(alarma.substring(105,110));
    Dato.setFecha(alarma.substring(54, 64));
    Dato.setHora(alarma.substring(65, 76));
    Dato.setTecnologia("UMTS");
    Dato.setUbicacion(Ubicacion);
    Dato.setNombre(this.setNombre(Dato));
    Dato.setLatitud(this.setlat(Dato)); // añadido
    Dato.setLongitud(this.setlng(Dato)); //añadido
    this.guardarEstacion3GS(Dato);
    return (Dato);
}
```

```
public String setNombre (Data_Estaciones Est){

    String Nombre = new String ();
    ResultSet resultado = null;
    Statement comando = null;
    String codigoSql = new String();

    if (Est.getUbicacion().contains("RNC"))
        codigoSql = "SELECT NOMBRES FROM sitios_nokia WHERE ID = '"+Est.getID()+"';";
    else
        codigoSql = "SELECT NOMBRES FROM sitios_nokia WHERE ID = '"+Integer.parseInt(Est.getID())

    try {
        comando = Conector.createStatement();
        resultado = comando.executeQuery(codigoSql);
        while (resultado.next())
            Nombre = resultado.getString(1);
        resultado.close();
    } catch (SQLException ex) {
        Nombre = "Error al obtener el Nombre";
        return (Nombre);
    }
```

. **Figura 30.** Clase Estaciones. Séptima Parte

```
public String setlat (Data_Estaciones Est){

    String Nombre = new String ();
    ResultSet resultado = null;
    Statement comando = null;
    String codigoSql = new String();

    if (Est.getUbicacion().contains("RNC"))
        codigoSql = "SELECT lat FROM sitios_nokia WHERE ID = '"+Est.getID()+"';";
    else
        codigoSql = "SELECT lat FROM sitios_nokia WHERE ID = '"+Est.getID()+"' AND UBICACION = '"+Est.getUbicacion()·

    try {
        comando = Conector.createStatement();
        resultado = comando.executeQuery(codigoSql);
        while (resultado.next())
            Nombre = resultado.getString(1);
        resultado.close();
    } catch (SQLException ex) {
        Nombre = "Error al obtener lat";
        return (Nombre);
    }
```

```
public String setlng (Data_Estaciones Est){

    String Nombre = new String ();
    ResultSet resultado = null;
    Statement comando = null;
    String codigoSql = new String();

    if (Est.getUbicacion().contains("RNC"))
        codigoSql = "SELECT lng FROM sitios_nokia WHERE ID = '"+Est.getID()+"';";
    else
        codigoSql = "SELECT lng FROM sitios_nokia WHERE ID = '"+Est.getID()+"' AND UBICACION = '"+Est.getUbicacion()·

    try {
        comando = Conector.createStatement();
        resultado = comando.executeQuery(codigoSql);
        while (resultado.next())
            Nombre = resultado.getString(1);
        resultado.close();
    } catch (SQLException ex) {
        Nombre = "Error al obtener lng";

    return (Nombre);
}

public void guardarEstacion (Data_Estaciones Est, String Tipo_Alarma){

    Statement comando = null;
    String codigoSql = new String();

    try {
        codigoSql = "INSERT INTO estaciones_3gt (`ID`, `UBICACION`, `NOMBRE`, `HORA`, `FECHA`, `ACK`, `ALARMA
        System.out.println("codigo SQL"+codigoSql);
        comando = Conector.createStatement();
        comando.executeUpdate(codigoSql);
    } catch (SQLException ex) {
        //Logger.getLogger(Estaciones.class.getName()).log(Level.SEVERE, null, ex);
        System.out.println("Ha ocurrido un error al grabar la estacion");
        try {
            codigoSql = "UPDATE `estaciones_3gt` SET `ALARMAS` = 'F/C' WHERE `ID` = '"+Est.getID()+"';";
            comando = Conector.createStatement();
            comando.executeUpdate(codigoSql);
        }
```

Figura 30. Clase Estaciones. Octava Parte

```
            catch (SQLException e){
                System.out.println("Error al actualizar");
            }
        }
    }

    public void guardarEstacion3GS (Data_Estaciones Est){

        Statement comando = null;
        String codigoSql = new String();

        codigoSql = "INSERT INTO estaciones_3gs (`ID`, `CELL_ID`, `UBICACION`, `NOMBRE`, `HORA`, `FECHA`, `ACK`, `lat`,
        System.out.println("codigo SQL"+codigoSql);
        try {
            comando = Conector.createStatement();
            comando.executeUpdate(codigoSql);
        } catch (SQLException ex) {

            //Logger.getLogger(Estaciones.class.getName()).log(Level.SEVERE, null, ex);
            System.out.println("Estacion Duplicada");
        }
```

```
    public void guardarEstacion2G (Data_Estaciones Est){

        Statement comando = null;
        String codigoSql = new String();

        try {
            codigoSql = "INSERT INTO estaciones_2G (`BCF`, `BTS`, `UBICACION`, `NOMBRE`, `HORA`, `FECHA`, `ACK`,
            System.out.println("codigo SQL"+codigoSql);
            comando = Conector.createStatement();
            comando.executeUpdate(codigoSql);

            //this.comprobarEst(Est);
        } catch (SQLException ex) {
            //Logger.getLogger(Estaciones.class.getName()).log(Level.SEVERE, null, ex);
            System.out.println("2G DUPLICADO");
        }
```

```
        try {

            codigoSql = "INSERT INTO estaciones_2GS (`BCF`, `BTS`, `UBICACION`, `NOMBRE`, `HORA`, `FECHA`, `ACK`,
            System.out.println("codigo SQL"+codigoSql);
            comando = Conector.createStatement();
            comando.executeUpdate(codigoSql);
            //this.comprobarEst(Est);
        } catch (SQLException ex) {
            //Logger.getLogger(Estaciones.class.getName()).log(Level.SEVERE, null, ex);
            System.out.println("2G DUPLICADO");
        }

    }
```

```
public void guardaHoraElemento(String Elemento, String Hora){

    Statement comando = null;
    String codigoSql = new String();

    try {
        codigoSql = "INSERT INTO hora_actualizacion_elementos (`ELEMENTO`, `HORA`) VALUES ('"+Elemento+"', '"+Hora+"')
        System.out.println("codigo SQL"+codigoSql);
        comando = Conector.createStatement();
        comando.executeUpdate(codigoSql);
    } catch (SQLException ex) {
        //Logger.getLogger(Estaciones.class.getName()).log(Level.SEVERE, null, ex);
        //System.out.println("Ha ocurrido al cargar la hora al elemento");
        try {
            codigoSql = "UPDATE `hora_actualizacion_elementos` SET `HORA` = '"+Hora+"' WHERE `ELEMENTO` = '"+Elemento+
            comando = Conector.createStatement();
            comando.executeUpdate(codigoSql);
        }
        catch (SQLException e){
            System.out.println("Error al actualizar");
        }
```

. **Figura 30.** Clase Estaciones. Novena Parte

```
    public void resumirAlarmas (){
    //para 3G
        Datos_Conexion Datos = new Datos_Conexion();
        ResultSet Resultado = null;
        ResultSet Resultado_2 = null;
        ResultSet Resultado_3 = null;
        Statement comando = null;
        String codigoSql = new String();
        int aux = 0;
        String Sectores = new String();
        String Est = new String();

        codigoSql = "SELECT * from estaciones_3gs";

        try{
            System.out.println("Pruba");
            comando = Conector.createStatement();
            Resultado = comando.executeQuery(codigoSql);
```

```
    while (Resultado.next()){
        aux = 0;
        System.out.println(Resultado.getString("ID"));
        codigoSql = "SELECT * from estaciones_3gs WHERE ID = '"+Resultado.getString("ID")+"';";

        try {
            System.out.println("Prueba 3");
            comando = Conector.createStatement();
            Resultado_2 = comando.executeQuery(codigoSql);
            while (Resultado_2.next()){
                if (aux == 0){
                    Sectores = Resultado_2.getString("CELL_ID").substring(1, 2);
                    Est = Resultado_2.getString("CELL_ID");
                }
                else
                    Sectores = Sectores+","+Resultado_2.getString("CELL_ID").substring(1, 2);
                aux=1;
            }
            System.out.println("Sectores: " +Sectores);
            codigoSql = "UPDATE estaciones_3gs SET SECTORES ='"+Sectores+"' WHERE CELL_ID = "+Est+";";
```

```
                System.out.println(codigoSql);
                try{
                    comando = Conector.createStatement();
                    comando.executeUpdate(codigoSql);
                }
                catch (SQLException ex3){
                    System.out.println("EX3 :"+ex3);
                }

            }catch (SQLException ex2){
                System.out.println("2. "+ex2);
            }
        }
    }catch (SQLException ex){
        System.out.println(ex);
    }

    //--------------------------------------------------------------------
}
```

Figura 30. Clase Estaciones. Décima Parte

```
public void resumirAlarmas2G (){
//para 2G
    Datos_Conexion Datos = new Datos_Conexion();
    ResultSet Resultado = null;
    ResultSet Resultado_2 = null;
    ResultSet Resultado_3 = null;
    Statement comando = null;
    String codigoSql = new String();
    int aux = 0;
    String Sectores = new String();
    String Est = new String();

    codigoSql = "SELECT * from estaciones_2gs";

    try{
        System.out.println("PruEba");
        comando = Conector.createStatement();
        Resultado = comando.executeQuery(codigoSql);

        while (Resultado.next()){
            System.out.println("While");
            aux = 0;
```

```
        System.out.println(Resultado.getString("ID_"));
        codigoSql = "SELECT * from estaciones_2gs WHERE BCF = '"+Resultado.getString("BCF")+"';";

        try {
            System.out.println("Prueba 3");
            comando = Conector.createStatement();
            Resultado_2 = comando.executeQuery(codigoSql);
            while (Resultado_2.next()){
                if (aux == 0){
                    Sectores = Resultado_2.getString("BTS");
                    Est = Resultado_2.getString("ID_");
                }
                else
                    Sectores = Sectores+","+Resultado_2.getString("BTS");
                aux=1;
            }
            System.out.println("Sectores: " +Sectores);
            codigoSql = "UPDATE estaciones_2gs SET SECTORES ='"+Sectores+"' WHERE ID_ = "+Est+";";
            System.out.println(codigoSql);
            try{
                comando = Conector.createStatement();
```

```
                }
                catch (SQLException ex3){
                    System.out.println("EX3 :"+ex3);
                }

            }catch (SQLException ex2){
                System.out.println("2. "+ex2);
            }
        }
    }catch (SQLException ex){
        System.out.println(ex);
    }

    //---------------------------------------------------------------------
}

}
```

. **Figura 30.** Clase Estaciones. Décima Primera Parte

3.3.2 Diseño de la página Web

La página web será diseñada mediante lenguaje PHP para la interacción con la base de datos y lenguaje JavaScript para el despliegue de mapas y cargas automáticas de la página. En la actualidad existen varios lenguajes de programación que permiten la interacción entre páginas web y conexión de bases de datos, PHP es uno de los lenguajes más utilizados al momento para este tipo de tareas debido a su simplicidad y capacidad de adaptación a los diferentes navegadores. Por otro lado, JavaScript ofrece una gran versatilidad a la hora de añadir componentes que se ejecuten en el navegador cliente, tales como funcionalidades de Google Maps y recargas automáticas de páginas. (Hein, 2013).

La plataforma web está dividida en dos páginas, la primera página muestra un listado de los sitios afectados divididos por la tecnología a la que corresponden, GMS, UMTS o LTE, de igual manera indica si la afectación es en toda la estación, o si se hace referencia a un sector en específico. La segunda página corresponde el mapa donde se mostrarán en tiempo real, la ubicación de la estación afectada, el nombre de la misma y la tecnología a la que corresponde.

La primera página web realiza una carga automática cada 10 minutos, al comenzar cada auto carga realizará una consulta a la base de datos del servidor sobre el listado de estaciones que se encuentren afectadas en ese instante, en las tres tecnologías, así, la información de las estaciones afectadas se desplegará en la página del cliente con una demora de máximo 10 minutos en la página web, aunque esta funcionalidad puede cambiarse a tiempo menores, pero el tiempo de actualización siempre dependerá de los tiempos de duración de ejecución del software cliente Java.

El código de consulta a las bases de datos fue realizado en PHP debido a su sencillez de codificación y la compactibilidad absoluta con las bases de datos MySQL. Por otro lado, la recarga automática fue hecha mediante un componente JavaScript que permite la ejecución de este tipo de tareas en el lado del cliente.

```
<!DOCTYPE html>
<html><head><!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
        <meta charset="UTF-8">
        <title></title>

        <style>
            table, td, th {
                border: 1px solid white;
            }

            th {
                background-color: green;
                color: white;
            }

            td {
                background-color: peachpuff;
                color: black;
            }
        </style>
    </head>
<body background="redes.jpg">
        <br><div style="text-align: center;"><big><big style="color: white;"><big>
            MONITOREO ESTACIONES NOKIA</big></big>
                <br></br>
<A HREF = "./map.php">Abrir Mapa</A>
            <?php

                $link = mysql_connect("localhost", "root", "root");

                mysql_select_db("estaciones", $link);
                //BORRADO DE LAS DE PRUEBA
                $result = mysql_query("DELETE FROM estaciones_3gt WHERE ID='9000' OR ID='3066' OR ID='3165' OR ID ='3172';", $link);

                mysql_select_db("estaciones", $link);
                $result = mysql_query("SELECT * FROM hora_actualizacion_elementos ORDER BY HORA ASC", $link);
                if ($row = mysql_fetch_array($result)){
                  //  echo "<br><div style='text-align: center;'><style='color: white;'>";
                    echo "<font color = 'white'>ULTIMA ACTUALIZACION: ".$row["HORA"]."</font><br></br>";
                }
```

Figura 31 Código Pagina Web

```
            mysql_select_db("estaciones", $link);
            $result = mysql_query("SELECT * FROM estaciones_3gt ORDER BY FECHA DESC, HORA DESC", $link);

            if ($row = mysql_fetch_array($result)){
                //echo "<div style='text-align:center;'>";
                echo "<b>UMTS (Estaciones)</b><br></br>";
                echo "<table border = '1' style='margin: 0 auto;'> \n";
                echo "<tr><th>ID</th><th>UBICACION</th><th>NOMBRE</th><th>HORA</th><th>FECHA</th><th>ALARMA</th></tr> \n";
                do {
                    echo "<tr><td>".$row["ID"]."</td><td>".$row["UBICACION"]."</td><td>".$row["NOMBRE"]."</td><td>".$row["HORA"]."</td><td>"
                } while ($row = mysql_fetch_array($result));
                //echo "</font>";
                echo "</table> \n";
                //echo "</div";
            } else {
                echo "<br></br>No existen estaciones UMTS totalmente afectadas <br></br>Volviendo a intentar";
            }


            $result = mysql_query("SELECT * FROM estaciones_3gs ORDER BY FECHA DESC, HORA DESC", $link);
```

```
            if ($row = mysql_fetch_array($result)){
                //echo "<div style='text-align:center;'>";
                echo "<b><br></br>UMTS (Sectores)</b><br></br>";
                echo "<table border = '1' style='margin: 0 auto;'> \n";
                echo "<tr><th>ID</th><th>CELL ID</th><th>UBICACION</th><th>NOMBRE</th><th>HORA</th><th>FECHA</th></tr> \n";
                do {
                    echo "<tr><td>".$row["ID"]."</td><td>".$row["CELL_ID"]."</td><td>".$row["UBICACION"]."</td><td>".$row["NOMBRE"].
                } while ($row = mysql_fetch_array($result));
                //echo "</font>";
                echo "</table> \n";
                //echo "</div";
            } else {
                echo "<br></br>No exiten sectores UMTS afectados <br></br>Volviendo a intentar";
            }

            $result = mysql_query("SELECT * FROM estaciones_2g ORDER BY FECHA DESC, HORA DESC", $link);

            if ($row = mysql_fetch_array($result)){
                //echo "<div style='text-align:center;'>";
                echo "<br></br><b>GSM (Estaciones)</b><br></br>";
                echo "<table border = '1' style='margin: 0 auto;'> \n";
                echo "<tr><th>BCF</th><th>UBICACION</th><th>NOMBRE</th><th>HORA</th><th>FECHA</th></tr> \n";
```

```
                do {
                    echo "<tr><td>".$row["BCF"]."</td><td>".$row["UBICACION"]."</td><td>".$row["NOMBRE"]."</td><td>".$row["HORA"]."</td><td>
                } while ($row = mysql_fetch_array($result));
                //echo "</font>";
                echo "</table> \n";
                //echo "</div";
            } else {
                echo "<br></br>No existen estaciones GSM totalmente afectadas <br></br>Volviendo a intentar";
            }

            $result = mysql_query("SELECT * FROM estaciones_2gs ORDER BY FECHA DESC, HORA DESC", $link);

            if ($row = mysql_fetch_array($result)){
                //echo "<div style='text-align:center;'>";
                echo "<br></br><b>GSM (Sectores)</b><br></br>";
                echo "<table border = '1' style='margin: 0 auto;'> \n";
                echo "<tr><th>BCF</th><th>BTS</th><th>UBICACION</th><th>NOMBRE</th><th>HORA</th><th>FECHA</th></tr> \n";
                do {
                    echo "<tr><td>".$row["BCF"]."</td><td>".$row["BTS"]."</td><td>".$row["UBICACION"]."</td><td>".$row["NOMBRE"]."</td><td>
                } while ($row = mysql_fetch_array($result));
                //echo "</font>";
                echo "</table> \n";
                //echo "</div";
            } else {
                echo "<br></br>No existen sectores GSM afectados <br>";
            }

        ?></big></div>
        <script>setTimeout('document.location.reload()',10000); </script>

</body></html>
```

. **Figura 31.** Código Pagina Web. Segunda Parte

```
<body background="redes.jpg" onload="load()">
        <br><div style="text-align: center;"><big><big style="color: white;"><big>
            MAPA ESTACIONES AFECTADAS NOKIA</big></big>
                <br></br>
                <script type="text/javascript">
                //<![CDATA[
                var customIcons = {
                  restaurant: {
                    icon: 'http://labs.google.com/ridefinder/images/mm_20_blue.png',
                    shadow: 'http://labs.google.com/ridefinder/images/mm_20_shadow.png'
                  },
                  bar: {
                    icon: 'http://labs.google.com/ridefinder/images/mm_20_red.png',
                    shadow: 'http://labs.google.com/ridefinder/images/mm_20_shadow.png'
                  }
                };
                function load() {
                  var map = new google.maps.Map(document.getElementById("map"), {
                    center: new google.maps.LatLng(-1.65639, -78.4921),
                    zoom: 7,
                    mapTypeId: 'roadmap'
```

```
                  });
                  var infoWindow = new google.maps.InfoWindow;
                  // Change this depending on the name of your PHP file
                  downloadUrl("phpsqlajax_genxml.php", function(data) {
                    var xml = data.responseXML;
                    var markers = xml.documentElement.getElementsByTagName("marker");
                    for (var i = 0; i < markers.length; i++) {
                      var name = markers[i].getAttribute("name");
                      var address = markers[i].getAttribute("address");
                      var type = markers[i].getAttribute("type");
                      var point = new google.maps.LatLng(
                          parseFloat(markers[i].getAttribute("lat")),
                          parseFloat(markers[i].getAttribute("lng")));
                      var html = "<b>" + name + "</b> <br/>" + address;
                      var icon = customIcons[type] || {};
                      var marker = new google.maps.Marker({
                        map: map,
                        position: point,
                        icon: icon.icon,
                        shadow: icon.shadow
                      });
                      bindInfoWindow(marker, map, infoWindow, html);
                    }
```

```
                    bindInfoWindow(marker, map, infoWindow, html);
                  }
                });
              }
              function bindInfoWindow(marker, map, infoWindow, html) {
                google.maps.event.addListener(marker, 'click', function() {
                  infoWindow.setContent(html);
                  infoWindow.open(map, marker);
                });
              }
              function downloadUrl(url, callback) {
                var request = window.ActiveXObject ?
                    new ActiveXObject('Microsoft.XMLHTTP') :
                    new XMLHttpRequest;
                request.onreadystatechange = function() {
                  if (request.readyState == 4) {
                    request.onreadystatechange = doNothing;
                    callback(request, request.status);
                  }
                };
                request.open('GET', url, true);
                request.send(null);
```

Figura 31. Código Pagina Web. Graficado en Google Maps

3.3.3 Diseño de la base de datos

Se han diseñado la base de datos de tal forma que permita guardar la información obtenida del programa cliente Java y que la página web sea capaz de realizar consultas continuas a la base de datos para desplegar la información de los sitios con afectación.

A continuación, se describen las tablas creadas en la base de datos de la plataforma.

Tabla 2

“estaciones _2g”

estaciones_2g		
Campo	**Tipo**	**Función**
BCF	VARCHAR(10)	Número de BCF de la estación GSM
UBICACION	VARCHAR(15)	Nombre del elemento maestro de la estación
NOMBRE	VARCHAR(50)	Nombre real de la estación
FECHA	DATE	Fecha del evento
HORA	TIME	Hora del evento
LAT	FLOAT(10,6)	Latitud de la estación
LNG	FLOAT(10,6)	Longitud de la estación

En la tabla “estaciones_2g” se almacena toda la información referente a las estaciones GSM afectadas en su totalidad, por otro lado, en la tabla “estaciones_2gs” se almacenará la información referente a sectores individuales de afectados.

Tabla 3

Esquema de la tabla "estaciones_2gs"

estaciones_2gs		
Campo	**Tipo**	**Función**
BCF	VARCHAR(10)	Código de la estación GSM afectada
BTS	VARCHAR(10)	Número de sector de la estación afectada
SECTORES	VARCHAR(10)	Conteo de los sectores afectados en total de la estación
UBICACION	VARCHAR(15)	Nombre del elemento maestro de la estación
NOMBRE	VARCHAR(50)	Nombre real de la estación
FECHA	DATE	Fecha del evento
HORA	TIME	Hora del evento
LAT	FLOAT(10,6)	Latitud de la estación
LNG	FLOAT(10,6)	Longitud de la estación

Para guardar la afectación de las estaciones y sectores en UMTS se ha decidido seguir un esquema similar.

Tabla 4

Esquema de a tabla "estaciones_3g"

estaciones_3g		
Campo	**Tipo**	**Función**
ID	VARCHAR(10)	Código de la estación UMTS
UBICACION	VARCHAR(15)	Nombre del elemento maestro de la estación
NOMBRE	VARCHAR(50)	Nombre real de la estación
FECHA	DATE	Fecha del evento
HORA	TIME	Hora del evento
LAT	FLOAT(10,6)	Latitud de la estación
LNG	FLOAT(10,6)	Longitud de la estación

Como en el caso anterior se ha divido en dos tablas, una para guardar las estaciones totalmente afectadas y otra para guardar estaciones solo con sectores afectados.

Tabla 5

Esquema de la tabla "estaciones_3gs"

estaciones_3gs		
Campo	**Tipo**	**Función**
ID	VARCHAR(10)	Código de la estación UMTS
CELL_ID	VARCHAR(10)	Código de celda afectada
UBICACION	VARCHAR(15)	Nombre del elemento maestro de la estación
NOMBRE	VARCHAR(50)	Nombre real de la estación
FECHA	DATE	Fecha del evento
HORA	TIME	Hora del evento
LAT	FLOAT(10,6)	Latitud de la estación
LNG	FLOAT(10,6)	Longitud de la estación

Sin embargo, para el caso de estaciones LTE afectadas solo se ha diseñado una tabla, ya que mediante la información telnet solo podemos obtener el dato si la estación se encuentra afectada o no, con lo que se debe realizar una conexión al sitio mediante una interfaz propietaria Nokia para verificar que porcentaje de la estación se encontraría afectado, esto permite saber si la estación esta por fuera de servicio en su totalidad o a nivel de celda, si se encuentra por fuera de cobertura la estación base en su totalidad las misma desplegará en su totalidad, y si esta por fuera de servicio a nivel de sector se mostrara el número de la celda afectada con lo cual se tendrá una apreciación real de la afectación.

Tabla 6

Esquema de la tabla "estaciones_4g"

estaciones_4g		
Campo	**Tipo**	**Función**
ID	VARCHAR(10)	Código de la estación LTE
UBICACION	VARCHAR(15)	Nombre del elemento maestro de la estación
NOMBRE	VARCHAR(50)	Nombre real de la estación
FECHA	DATE	Fecha del evento
HORA	TIME	Hora del evento
LAT	FLOAT(10,6)	Latitud de la estación
LNG	FLOAT(10,6)	Longitud de la estación

La consulta de los nombres a partir de los códigos se lo realizará a través de una tabla que almacenará toda la información de las estaciones las cuales disponemos previamente en archivos en formato xls. La migración de los datos de las estaciones donde se puede encontrar el nombre de la estación a través del respectivo código será realizada la primera vez de manera manual, luego será que el programa cliente java realizará las consultas de manera automática cada vez que obtenga un cogido nuevo de alguna estación con afectación.

Tabla 7

Esquema tabla de consulta "sitios_nokia"

sitios_nokia		
Campo	**Tipo**	**Función**
CODIGO	VARCHAR(10)	Código maestro de la estación
ID	VARCHAR(15)	ID de único del registro (Llave principal)
UBICACION	VARCHAR(10)	Nombre del elemento maestro de la estación
NOMBRE	VARCHAR(50)	Nombre real de la estación

3.3.4 Variable de entrada y salida

Los datos de entrada que obtendremos son los detalles de la alarma de las diferentes tecnologías. Las alarmas disponen de variada información dependiendo de la tecnología y versión del software que manejan, sin embargo tenemos siempre elementos que será comunes en las distintas tecnologías, como son: la hora en que presentó la afectación, la fecha del evento, el código de la estación y nombre del elemento maestro al que pertenece.

A continuación se detallaran las variables de entrada que se utilizaron para la elaboración de la aplicación:

La alarma de la estación tendrá los siguientes elementos dentro de su información, este es un texto plano de los cual se realizará el análisis mediante la programación del cliente java.

- Código de la estación o sector afectado
- Hora de la afectación
- Fecha de la afectación
- Nombre del elemento controlador de la estación

A continuación se detallaran las variables de salida que se utilizaron para la elaboración de la aplicación:

Una vez la plataforma realice el análisis de la estación obtendrá los siguientes datos referentes a las estaciones para su posterior despliegue en la plataforma web que se ha montado en el servidor.

- Código de la estación o sector afectado
- Hora de la afectación
- Fecha de la afectación

- Nombre del elemento controlador de la estación
- Nombre de la estación afectada
- Longitud y latitud de la estación afectada
- Validador que indicará que porcentaje de la estación se encuentra afectada.

Estos datos serán suficientes para que el operador del NOC pueda realizar un troubleshooting más eficiente durante un evento que implique a varias estaciones que se afecten al mismo tiempo.

3.4 Conexión con las bases de datos de las centrales telefónicas

La conexión será realizada mediante conexiones telnet, estas conexiones están habilitadas solamente para clientes que se encuentren dentro de la intranet del NOC, donde se ha asignado usuarios específicos para el monitoreo de la red los cuales solo tienen habilitados los comandos necesarios para realizar consultas sobre el estado de las estaciones, todo esto pensando en la seguridad de la red.

4.5 Pruebas de funcionalidad en simulaciones y escenarios reales

Debido a la envergadura de la red celular de Telefónica la gran mayoría del tiempo observamos que existen al menos unas pocas estaciones afectadas en algún lugar del país. Esto representa una ventaja al momento de realizar las pruebas ya que siempre dispondremos de alguna estación que nos ayude generando datos de indisponibilidad.

Para las pruebas ejecutamos el software cliente en nuestro servidor virtual, procedemos a abrir la plataforma y verificamos las estaciones que se encuentran afectadas al momento y comparamos con las alarmas que se presenta en en NetAct en ese mismo instante.

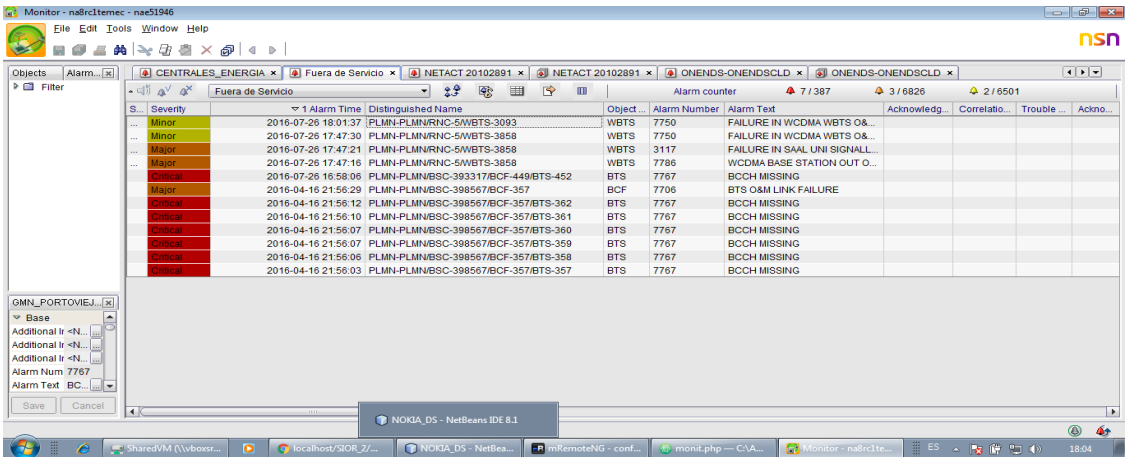


Figura 32 Vista de las alarmas generadas en la plataforma NetAct

Figura 33 Vista de la Plataforma WEB. Primera parte

Figura 34 Vista de la Plataforma WEB. Segunda parte

De la tabla descrita resumimos entonces las siguientes estaciones afectadas para la fecha 26 Julio del 2017 a las 18h03.

Tabla 8

Resumen de las estaciones afectadas en la prueba

Nombre	Tecnología	Afectación
TULCAN	UMTS	Toda la estación afectada
PUYO OBRERO	UMTS	Toda la estación afectada
PAPALLACTA	UMTS	Afectado 1 sector de la estación
QUEVEDO_SUR	UMTS	Afectado 1 sector de la estación
TRUBUNA_SHYRIS	UMTS	Afectados 2 sectores de la estación
SUPERMAXI_ATAHUALPA	GSM	Toda la estación con afectación

Pasamos a verificar que las estaciones se hayan graficado correctamente en la opción del mapa de google.

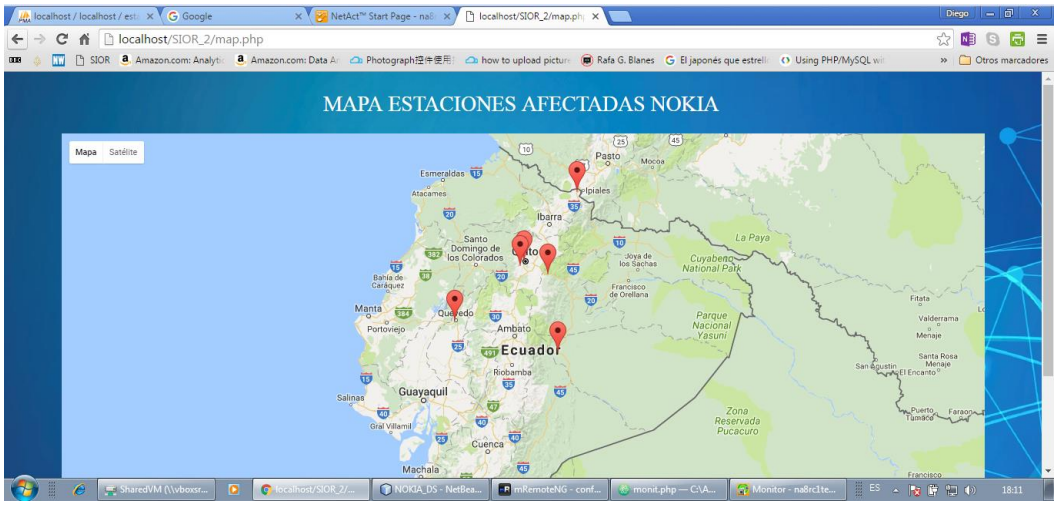


Figura 35 Gráfico de las estaciones afectadas en el Mapa de Ecuador

Se puede observar entonces que las estaciones se han graficado, procedemos a verificar que cada uno de los Ítems correspondan a las estaciones afectadas. El usuario podrá hacer verificar el nombre de la estación al realizar un clic en cada ícono graficado en el mapa.

Figura 36 Gráfico en el mapa de la estación TULCAN

Figura 37 Gráfico estación SUPERMAXI_ATAHUALPA

Figura 38 Gráfico en el mapa de la estación TRIBUNA_SHYRIS

Figura 39 Gráfico en el mapa de la estación PAPALLACTA

Figura 40 Gráfico en el mapa de la estación PUYO_OBRERO

Figura 41 Gráfico de la estación QUEVEDO_SUR

Esta información la podemos comparar con los datos de las alarmas obtenidas en el gestor NetAct.

Alarm Number	Alarm Type	Severity	Alarm Time	Alarm Text	Distinguished Name	Object Class
7767	Quality Of Service	Critical	16/04/2016 21:56	BCCH MISSING	PLMN-PLMN/BSC-398567/BCF-357/BTS-357	BTS
7767	Quality Of Service	Critical	16/04/2016 21:56	BCCH MISSING	PLMN-PLMN/BSC-398567/BCF-357/BTS-358	BTS
7767	Quality Of Service	Critical	16/04/2016 21:56	BCCH MISSING	PLMN-PLMN/BSC-398567/BCF-357/BTS-360	BTS
7767	Quality Of Service	Critical	16/04/2016 21:56	BCCH MISSING	PLMN-PLMN/BSC-398567/BCF-357/BTS-359	BTS
7767	Quality Of Service	Critical	16/04/2016 21:56	BCCH MISSING	PLMN-PLMN/BSC-398567/BCF-357/BTS-361	BTS
7767	Quality Of Service	Critical	16/04/2016 21:56	BCCH MISSING	PLMN-PLMN/BSC-398567/BCF-357/BTS-362	BTS
7706	Communication	Major	16/04/2016 21:56	BTS O&M LINK FAILURE	PLMN-PLMN/BSC-398567/BCF-357	BCF
7767	Quality Of Service	Critical	26/07/2016 16:58	BCCH MISSING	PLMN-PLMN/BSC-393317/BCF-449/BTS-452	BTS
7750	Communication	Minor	26/07/2016 17:47	FAILURE IN WCDMA WBTS O&M CONNECTION	PLMN-PLMN/RNC-5/WBTS-3858	WBTS
7786	Quality Of Service	Major	26/07/2016 17:47	WCDMA BASE STATION OUT OF USE	PLMN-PLMN/RNC-5/WBTS-3858	WBTS
3117	Quality Of Service	Major	26/07/2016 17:47	FAILURE IN SAAL UNI SIGNALLING LINK ACTIVATION	PLMN-PLMN/RNC-5/WBTS-3858	WBTS
7750	Communication	Minor	26/07/2016 18:01	FAILURE IN WCDMA WBTS O&M CONNECTION	PLMN-PLMN/RNC-5/WBTS-3093	WBTS

Figura 42 Alarmas obtenidas en el gestor NetAct

Para obtener la información comparamos el Código de la estación, que corresponde al “Distinguished Name” con la data de la cual disponemos en nuestros archivos xls.

Distinguished Name	Name
PLMN-PLMN/BSC-398567/BCF-357/BTS-357	SUPERMAXI_ATAHUALPA_1
PLMN-PLMN/BSC-398567/BCF-357/BTS-358	SUPERMAXI_ATAHUALPA_2
PLMN-PLMN/BSC-398567/BCF-357/BTS-360	SUPERMAXI_ATAHUALPA_4
PLMN-PLMN/BSC-398567/BCF-357/BTS-359	SUPERMAXI_ATAHUALPA_3
PLMN-PLMN/BSC-398567/BCF-357/BTS-361	SUPERMAXI_ATAHUALPA_5
PLMN-PLMN/BSC-398567/BCF-357/BTS-362	SUPERMAXI_ATAHUALPA_6
PLMN-PLMN/BSC-398567/BCF-357	SUPERMAXI_ATAHUALPA_0
PLMN-PLMN/BSC-393317/BCF-449/BTS-452	TRIBUNA_SHYRIS
PLMN-PLMN/RNC-5/WBTS-3858	PUYO_OBRERO
PLMN-PLMN/RNC-5/WBTS-3858	PAPALLACTA
PLMN-PLMN/RNC-5/WBTS-3858	QUEVEDO_SUR
PLMN-PLMN/RNC-5/WBTS-3093	TULCAN

Figura 43 Alarmas con su estación correspondiente

Se observa entonces que las alarmas corresponden a las estaciones obtenidas a partir de nuestra plataforma WEB con lo cual se valida el correcto funcionamiento de la aplicación.

CAPITULO IV

ANÁLISIS DE RESULTADOS

4.1 Análisis de las Alarmas

Las alarmas obtenidas tienen el siguiente esquema:

Alarm Number	Alarm Type	Severity	Alarm Time	Alarm Text	Distinguished Name	Object Class
7767	Quality Of Service	Critical	16/04/2016 21:56	BCCH MISSING	PLMN-PLMN/BSC-398567/BCF-357/BTS-357	BTS

Figura 44 Formato de la Alarma Nokia

A partir de la alarma obtenida el programa cliente en java discernirá la información necesaria para lo cual se extraen los siguientes campos

- Hora de la Alarma
- Fecha de la Alarma
- Numero de Alarma
- ID de la estación

El número de alarma nos permite discernir qué tipo de afectación es la que se presenta en la estación, es a partir entonces del número de alarma que identificamos si una estación Nokia se encuentra con problemas.

El ID de la estación será la que nos permitirá identificar comparándola con nuestra Base de Datos sobre cuál es el nombre real de la estación. Este nombre sirve para que tanto el operador como el personal técnico se orienten en la ubicación de la misma.

En el caso especial de las alarmas GSM, no existen un número de alarma específico que nos indique si un sector de la estación esta caída o si se trata de toda la estación que presenta la afectación. Para solucionar este problema se ha elaborado un algoritmo que envía un comando a las estaciones que presenten previamente alarmas en indisponibilidad en alguno de sus sectores, este comando obtener el resultado de todos los TRXs de la estación, si se encuentran trabajando normalmente o si por el contrario se encuentran con errores de funcionamiento, para eso nos vamos ayudar de la siguiente nomenclatura

Basándonos en la nomenclatura de la gráfica, el algoritmo lo que hará es reconocer si algún TRX se encuentra en estado WO, de ser así el caso eso significa que algún sector está operativo por lo que las alarmas obtenidas previamente indican los sectores afectados, caso contrario indica que toda la estación se encuentra indisponible.

4.1.1 Estaciones Fuera de Servicio

La lista de estaciones totalmente fuera de servicio en tecnología GSM serán validadas de acuerdo a si se tiene activo un TRX de la estación, para estaciones UMTS y LTE en cambio nos podremos ayudar discerniendo por el número de la alarma que esté presente al momento.

La prioridad de atención de las estaciones se da por lo general verificando si hay una estación cercana que pueda cubrir la cobertura que una estación ha dejado de brindar. En entornos urbanos mayoritariamente existen estaciones que siempre permitirán cubrir con la cobertura en caso de que alguna deje de funcionar, sin embargo en entornos rurales no se suele dar eso caso por lo que el NOC deberá estar atento a estos casos específicos.

Tener la herramienta de graficado en el mapa de Google Maps es de gran ayuda al usuario que se encuentre en el monitoreo de la red ese instante, ya que, de esta manera puede establecer si en caso de afectación al servicio esta amerita atención inmediata en el momento de que muchas estaciones concentradas en un sector del país específico se encuentran fuera de servicio

4.1.2 Sector de la Estación

Para obtener el número de sector afectado en la estación se procederá de la siguiente manera: Para el caso de las estaciones GSM el código en el cliente Java realizará una resta entre el número de BTS y el número de BCF que se obtiene en la alarma.

Para el caso de las alarmas UMTS el número de sector se lo obtiene a través del valor del CELL ID, dentro de este código se identifica el segundo digito que es el que indica a que número de sector hace referencia la alarma presentada

En el caso de LTE aún no se ha implementado la identificación por sectores por lo que lo única información disponible es saber si la estación está operativa o no. Para verificar si solo parte de la estación está operativa será necesario ingresar manualmente vía una interfaz Nokia de código propietario a la estación.

4.2 Tiempos de Espera

Los tiempos de espera dependerán del tiempo de respuesta a los comandos Telnet de los diferentes elementos maestros de la red de acceso. Para no saturar con peticiones constantes a los elementos de red se ha configurado en el código del programa Java un tiempo de 10 minutos entre ejecución de comandos Telnet a los elementos de red. Es decir, después de realizar una consulta mediante Telnet a un elemento maestro de la estación se esperará 10 minutos para realizar otra consulta a la estación.

La página WEB por otro lado está de tal manera para que haga una recarga automática cada 5 minutos, con lo que se hará una consulta a las bases de datos generadas por el código del cliente Telnet automáticamente pasado este tiempo.

Una vez ejecutado el código Java podemos observar mediante el mismo IDE Netbeans cuanto tiempo demora en realizar una consulta a todos los elementos configurados en el programa, el resultado de la ejecución tiene un tiempo de 14 segundos según se observa en la gráfica

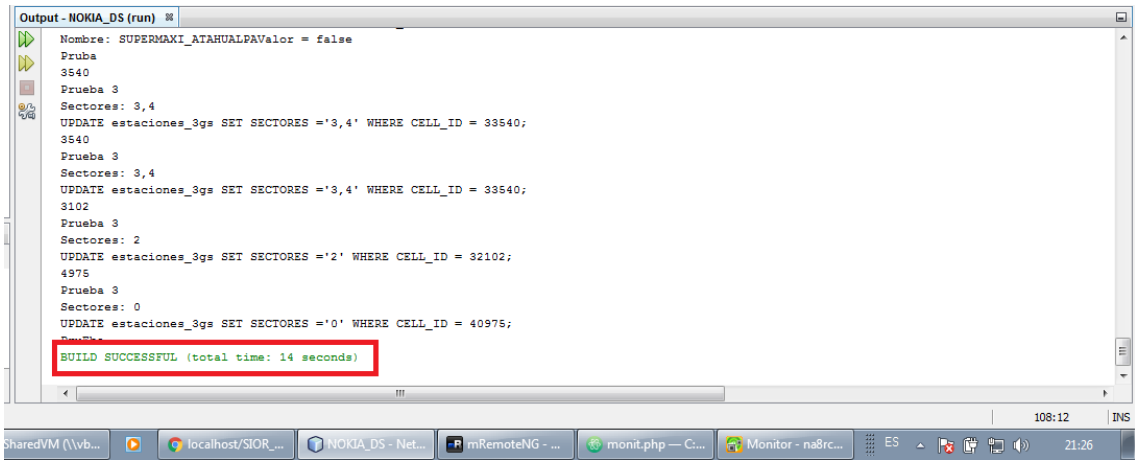


Figura 45 Verificación de tiempo de ejecución en el IDE Netbeans

Sumando todos estos tiempos obtenemos un total máximo de 15 minutos 14 segundos en el peor de los escenarios. Este tiempo el usuario lo tiene como aceptable ya que la detección de una alarma debe de realizarse dentro de un rango de 30 minutos en caso de que continúe la afectación para proceder con su reporte a técnico de campo.

4.3 Cálculo de indisponibilidad

Una vez obtenido las estaciones afectadas para proceder con el cálculo de indisponibilidad con lo referente a tecnología Nokia se procede con una regla de tres simples de acuerdo al parque de estaciones Nokia en el país.

El número de estaciones Nokia al momento en la red móvil es de 820 estaciones. Por lo que al tener es nuestra prueba 6 estaciones afectadas calcularíamos que el nivel de indisponibilidad es del 0.007% a nivel Nacional

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

Debido a los exigentes tiempo de SLA entre el cliente y el proveedor en este caso, es necesario seguir buscando oportunidades de mejoramiento de los tiempos de respuesta de los operadores para brindar una oportuna información al cliente en caso de existir incidentes con gran afectación, ya que ha medido de que crece el número de estaciones afectadas más crítico se vuelve el tiempo en que debe ser atendidos estos eventos, para esto debemos apoyarnos en nuestros conocimientos de ingeniería obtenidos a lo largo de la carrera para implementar soluciones inteligentes a los problemas que iremos encontrando día a día en nuestras labores diarias.

En la actualidad existen muchas herramientas libres que permiten la automatización de procesos en prácticamente todas las ramas de la industria. Al momento las tecnologías convergen hacia el punto de ahorro de tiempo y optimización de procesos. Desarrollando nuestra aplicación observamos que existen muchas otras posibilidades en cuanto a automatización de procesos, por lo que solo es necesario un buen análisis de los problemas que causan cuellos de botella en la operación del día a día para obtener provecho de la gran variedad de herramientas de desarrollo existentes en el cual debemos aprovechar la gran variedad de información disponible en la web para su implementación según sean nuestras necesidades

5.2 Recomendaciones

Para el caso particular del desarrollo de aplicaciones se recomienda una buena documentación de los pasos para la implementación de las soluciones. De esta forma no nos ataremos a una solución específica y que cualquier persona que realice la debida revisión de la documentación elaborada pueda plantear mejoras y resolver problemas que podrían ocurrir a lo largo de la vida en el ambiente de producción de las aplicaciones implementadas.

Se recomienda usar herramientas libres para el desarrollo plataformas web, esto con el fin de evitar costos a menos que los requerimientos de la aplicación lo ameriten. En nuestro caso, la aplicación no requería de mayores requerimientos de hardware y de software por lo que se recurrió al uso de freeware, como es el caso de Java, Mysql, PHP y las APIs gratuitas de google para el gráfico de mapas. Google además posee varias otras herramientas que permiten personalizar el gráfico de puntos en coordenadas específicas. Esto servirá de gran ayuda en caso de que deseen realizarse cambios solicitados por los usuarios de la Plataforma.

BIBLIOGRAFÍA

Huidobro, J.M, (2012). Comunicaciones Móviles, sistemas GSM, UMTS y LTE, RA-MA Editorial.

Huidobro, J.M, (2014). Telecomunicaciones. Tecnologías, Redes y Servicios, RA-MA Editorial.

Lara, J.C, (2006). Conceptos básicos de telefónica celular. Hidalgo – México

Eveliux, E, (2001). La evolución de la telefonía móvil.
Recuperado de:
http://www.eveliux.com/mx/La-evolucion-de-la-telefonia-movil.html

Informática Hoy (2007). La historia del teléfono celular.
Recuperado de:
http://www.informatica-hoy.com.ar/telefonos-celulares/La-historia-del-Telefono-Celular.php

Ranchal, J, (2014). Inicios, evolución y futuro del teléfono móvil. Recuperado de: http://www.muycanal.com/2014/01/31/futuro-del-telefono-movil

Farez, D, (2007). Estándar GSM (Sistema global de comunicaciones móviles).
Recuperado de :
http://es.ccm.net/contents/681-estandar-gsm-sistema-global-de-comunicaciones-moviles

Soarez, R, (2015). UMTS:Arquitectura. Recuperado de: http://www.teleco.com.br/es/tutoriais/es_tutorialumts/pagina_2.asp

Sequeira, L, (2015). Características de la arquitectura LTE/SAE.
Recuperado de: http://www.telecomsharing.com/es/biblioteca/lte-4g/item/33-caracteristicas-de-la-arquitectura-lte-sae

Programación en Castellano, 08 12 2014. [En línea]. Available: http://programacion.net/articulo/desarrollo_de_aplicaciones_web_con_tapestry_334.