ESCUELA POLITÉCNICA DEL EJÉRCITO

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE ALARMAS COMUNITARIAS A TRAVÉS DE UN OPERADOR MÓVIL

LUIS DAVID BRAVO ROSERO

SANGOLQUÍ – ECUADOR
2011

CERTIFICACIÓN

Certificamos que el presente proyecto de grado titulado: "DISEÑO E IMPLEMENTACIÓN DE ALARMAS COMUNITARIAS A TRAVÉS DE UN OPERADOR MÓVIL", ha sido desarrollado en su totalidad por el señor LUIS DAVID BRAVO ROSERO con CC: 171642796-6, bajo nuestra dirección.

Ing. Darío Duque Cajas.
DIRECTOR

Ing. José Robles Salazar.
CODIRECTOR

RESUMEN

El presente proyecto contempla el diseño e implementación de un SERVlet para recibir notificaciones de alarmas provenientes de un MIDlet desarrollado con plataforma J2ME el cual será instalado en los dispositivos móviles de los moradores de la comunidad a los alrededores de la Unidad de Policía Comunitaria (UPC) del Barrio Central de Sangolquí mediante la utilización de mensajes de texto SMS a través de un operador móvil.

El SERVlet establece la conexión mediante la utilización de comandos AT&AT+ con un modem GPRS/GSM que será encargado de hacer la recepción de alarmas para así desplegarlas en un ambiente gráfico con el cual un operador en la UPC se deberá familiarizar para su correcta manipulación al momento de establecer que la notificación de alarma necesita ser tratada, es decir saber si es una falsa alarma, en proceso o atendido, además tendrá la potestad de realizar un registro previo de datos de los moradores del barrio para así poder interactuar con una base de datos MySQL que permitirá realizar búsquedas, reportes, etc.

Finalmente, se ha desarrollado el SERVlet para que cuando una notificación de alarma arribe a la central no se pierda si no es atendida, el modem GPRS/GSM realiza automáticamente el desvío de alarma hacia el celular del jefe de la UPC para evitar el inconveniente.

DEDICATORIA

A Dios todopoderoso
quien siempre ha sido mi único amigo fiel,
a mis amados padres Bolívar y Margoth

quienes me han guiado y me han brindado su apoyo desde pequeño para lograr cosas grandes e importantes en mi vida,

a mis queridos hermanos, Bolivar, Adriana y Cynthia

que han estado en momentos alegres y difíciles,

a mis queridos Amigos

quienes siempre fueron mi pedestal y gran compañía.

Luis David Bravo Rosero

AGRADECIMIENTO

Agradezco a Dios por darme valiosos años de vida, llenos de gracia, felicidad y tranquilidad.

A mí amada madre Margoth quien supo sostener una hermosa familia, me llenó de sus sabios consejos y me brindó confianza incondicional.

A mí amado padre Bolívar quien hizo de mí una persona de bien formando muchos de los valores de la vida esencialmente la responsabilidad y respeto.

A mi querido hermano Bolivar quien ha sido mi amigo incondicional, con el que siempre he compartido cosas hermosas y sueños increíbles.

A mis hermanas Adriana y Cynthia por cuidarme y por haber dejado sus huellas en el camino para aprender de sus experiencias.

A mis queridos y amados amigos que desde el colegio no se han separado de mí y han sido mi alma de alegría.

A María Augusta, Julio, Henry y compañeros de universidad con quienes disfrute todos estos cinco años de carrera y no dejaron de apoyarme.

A los ingenieros Darío Duque y José Robles por abrirme las puertas y guiarme en buen camino para la realización y cumplimiento de objetivos de este proyecto.

A los ingenieros con quienes pude compartir conocimientos y experiencias en los cinco años de carrera para construir una formación integral.

Gracias con mucha sinceridad a todas aquellas personas quienes contribuyeron de alguna u otra manera en mi vida y han sabido conservar su lazo con mi persona.

Luis David Bravo Rosero

PROLOGO

Las redes celulares GSM, como tecnología han alcanzado una gran cobertura al nivel mundial, principalmente por sus ventajas comerciales, además de aportar para el desarrollo de aplicaciones que hacen uso de algunos de sus servicios.

El presente proyecto tiene como objetivo hacer uso de las redes celulares GSM y especialmente del servicio de mensajes de texto SMS para desarrollar e implementar un sistema de alarmas comunitarias, para así brindar más seguridad a comunidades que poseen una Unidad de Policía Comunitaria (UPC).

La UPC será la encargada de administrar todos los datos de los moradores mediante el uso de una aplicación y un modem para poder así enlazarlos y actuar sobre las notificaciones de alarma que sean recibidas.

Este trabajo hace referencia a una revisión teórica y práctica de todos los temas relacionados con las tecnologías usadas para el uso del prototipo las cuales son: sistema GSM, modem GPRS/GSM, comandos AT&AT+, bases de datos MySQL, MIDP/CLDC y aspectos relevantes del lenguaje J2ME. Posteriormente se describe el diseño e implementación del prototipo haciendo referencia a la teoría pautada anteriormente y tratando de ser lo más implícito en el desarrollo del sistema, para así finalmente obtener pruebas y resultados los cuales serán analizados en aspectos, tecnológicos, sociales y financieros para conocer si la integración del sistema es rentable.

INDICE GENERAL

CAPITULO I
INTRODUCCION

CAPITULO II
FUNDAMENTOS DE LA TELEFONIA CELULAR

CAPITULO III
DISEÑO E IMPLEMENTACION DEL PROTOTIPO

CAPITULO IV
PRUEBAS Y RESULTADOS EXPERIMENTALES

CAPITULO V
ANALISIS FINACIERO

CAPITULO VI
CONCLUSIONES Y RECOMENDACIONES

ANEXOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

GLOSARIO

API: Application Programming Interface

AuC: Authentication Center

BSC: Base Station Controller

BSS: Base Station Subsystem

BTS: Base Transceiver Station

CDC: Connected Device Configuration

CLDC: Connected Limited Device Configuration

CSD: Circuit Switch Data

CTS: Clear To Send

DCD: Data Carrier Detect

DSR: Data Set Ready

DTMF: Dual Tone Multifrequency

DTR: Data Terminal Ready

EDGE: Enhanced Data Rated for GSM Evolution

EIR: Equipment Identity Registry

ETSI: European Telecommunications Standards Institute

FDMA: Frequency Division Multiple Access

GMSK: Gaussian Minimum Shift Keying

GND: Common Ground

GPRS: General Packet Radio Service

GSM: Global System for Mobile Communication

HLR: Home Location Register

IDE: Integrated Development Environment

J2ME: Java 2 Micro Edition

JAD: Java Application Descriptor

JAR: Java Archive

JDBC: Java Database Connectivity

JDK: Java Development Kit

JRE: Java Runtime Environment

MD5: Message Digest algorithm 5

MIDP: Mobile Information Device Profile

MS: Mobile Station

MSC: Mobile Services Switching Center

NSS: Network and Switching Subsystem

PC: Personal Computer

PDU: Protocol Data Unit

RAM: Random Access Memory

RTS: Request To Send

RXD: Received Data

SIM: Subscriber Identity Module

SME: Short Messaging Entity

SM MO: Short Message Mobile Originated Point-to-Point

SM MT: Short Message Mobile Terminated Point-to-Point

SMS: Short Message Service

SMSC: Short Message Service Center

SMS-GMSC: Gateway MSC for Short Message Service

SMS-IWMSC: Internetworking MSC for Short Message Service

TDMA: Time Division Multiple Access

TIO: Tasa de Interes de Oportunidad

TIR: Tasa Interna de Retorno

TXD: Transmitted Data

URL: Uniform Resource Locator

USB: Universal Serial Bus

VAN: Valor Actual Neto

VLR: Visitor Location Register

WMA: Wireless Messaging API

CAPITULO 1

INTRODUCCION

La falta de seguridad ha pasado a ser en los últimos tiempos uno de los temas centrales de preocupación de la ciudadanía y, por lo tanto, una de las cuestiones a resolver por los responsables de seguridad en el Ecuador. La inseguridad ha ido incrementando potencialmente por motivos que no son razonables en estos días, en efecto, la crisis de la seguridad ha conducido a que, en ocasiones los ámbitos de gobierno se formulen cosas simples para actuar sobre los síntomas pero no sobre las causas.

La sociedad se ha aferrado a muchas causas, que a veces el gobierno no puede controlar y hace que seamos más propensos a ataques de la delincuencia, y sobre todo a que NO tengamos forma de alarmar situaciones sin hacer alguna denuncia delante de la ley, además la gran demanda de productos de seguridad ha ido incrementando los últimos años, que al parecer dan resultados, pero no los deseados por los usuarios.

El proyecto que se presenta en esta documentación, tiene el fin de acercarse potencialmente a la ciudadanía que es objeto de ataques de delincuencia domiciliaria frecuentes. Dicho proyecto se constituye principalmente de dos aplicaciones tecnológicas innovadoras, que cumplen las necesidades de ayuda comunitaria directa, vía celular, hacia una central que será la encargada de recibir notificaciones de alarmas de las personas que necesiten la asistencia de la Policía o de cualquier otra institución de seguridad cercana.

1.1. ANTECEDENTES

En el país se vive con cierto nerviosismo en cuanto a cuidar bienes materiales y humanos dado que la seguridad ciudadana ha ido disminuyendo. Los gobiernos de turno han creado los UPC's (Unidades de Policía Comunitaria) con el objetivo de precautelar la seguridad en los diferentes barrios. Sin embargo no disponen de infraestructura tecnológica y peor de armamento sofisticado.

1.2. JUSTIFICACION E IMPORTANCIA

Justificación

La seguridad pública representa un rol importante dentro de la nación, pero los ciudadanos no han encontrado respuesta positiva con relación al alto porcentaje de inseguridad que el país está atravesando. Los barrios actualmente disponen de un sistema de alarma doméstico a través de un botón de pánico instalado en los hogares. Este sistema se instala previo un acuerdo con los moradores del barrio que soliciten dicho servicio. Este sistema ha servido para solventar problemas de robos, alertas u otra emergencia; a pesar de ello la eficiencia del mismo, depende de la adecuada instalación eléctrica que se disponga en cada domicilio.

Hace un par de décadas era prácticamente imposible pensar en hacer llegar información en cuestión de segundos. La aparición de las redes celulares permitieron el intercambio de información y datos en forma rápida y confiable.

El presente proyecto pretende el aprovechamiento de las tecnologías modernas para que realicen el mismo efecto que tiene el botón de pánico.

A través de las redes de telefonía celular es posible mediante el servicio de mensajes cortos (SMS) alertar un posible robo o atentado que pueda suceder en el domicilio de algún morador del

sector, comunicándose con un centro de gestión que esté ubicado en las UPC's y tener una respuesta inmediata de dichas unidades de policía.

Importancia

La importancia del presente proyecto es que serán beneficiadas dos entidades; la policía definida como una UPC y los habitantes de cada uno de los barrios.

En la UPC se instalará una base de datos de todos los moradores del sector que deseen adquirir este sistema de alarma. Debe existir un administrador el cual gestionará los requerimientos de los usuarios, en el que se definirán los diferentes tipos de procedimientos que pueden ser: robo, falsa alarma, en progreso, atendido, etc. Sí el administrador está ausente la alarma se re-direcciona al número telefónico celular del jefe de turno de la UPC.

Para los usuarios del sistema, en los celulares se instalará la aplicación para que pueda interactuar con el sistema base central. Con la implementación del sistema de alarma propuesto se pretende en conjunto con las UPC's brindar mayor seguridad y tranquilidad a los moradores del barrio.

1.3. OBJETIVOS

Objetivo General

Diseñar e implementar un sistema de alarmas comunitarias a través de un operador móvil.

Objetivos Específicos

- Investigar alternativas tecnológicas utilizadas para la recepción y envío de mensajes cortos para implementar el prototipo.

- Conocer, procesar y analizar los datos que entrega el receptor GSM para que así puedan ser familiarizados con el protocolo de comunicación que maneja.

- Diseñar y desarrollar el SERVlet a través del software necesario para controlar el receptor GSM, para así poder automatizar la comunicación entre las estaciones móviles y un computador central.

- Diseñar y desarrollar la base de datos para que interactúe con la MIDlet y el SERVlet.

- Diseñar y desarrollar la MIDlet tal que permita la interacción de un equipo móvil con el usuario para así poder enviar las notificaciones de alarmas usando mensajes cortos SMS.

- Realizar la instalación y pruebas de evaluación requeridas para el prototipo de alarmas comunitarias.

- Documentar los resultados.

CAPITULO II

FUNDAMENTOS DE LA TELEFONIA CELULAR

La telefonía móvil básicamente está formada por dos grandes partes: una red de comunicaciones y los terminales que permiten el acceso a dicha red, dentro de la telefonía móvil existen varias tecnologías que han sido aplicadas en la vida actual tales como GSM, CDMA, EDGE, etc. Las cuales han situado su vestigio en todo el mundo.

Una de las tecnologías aplicadas a la telefonía móvil más utilizadas es GSM, dado que sus terminales así como su red tienen bases económicas muy bajas y equipos tecnológicos que permitieron que esta sea la tecnología de auge para celulares.

A continuación se describirán brevemente los conceptos básicos del sistema GSM, especificaciones, arquitectura de red, características y servicios, que serán el preámbulo para detallar el Servicio de Mensajes Cortos (SMS), en el cual se basa el presente proyecto.

2.1. GSM

2.1.1.Definición de GSM

GSM (*Global System for Mobile Communication*) es el nombre de un grupo de estandarización establecido en 1982 para la creación de un estándar habitual de telefonía móvil europeo que manifestaría detalles y ciertas especificaciones para un sistema móvil celular de radio que opere a 900 MHz; además de ser un estándar íntegramente admitido para

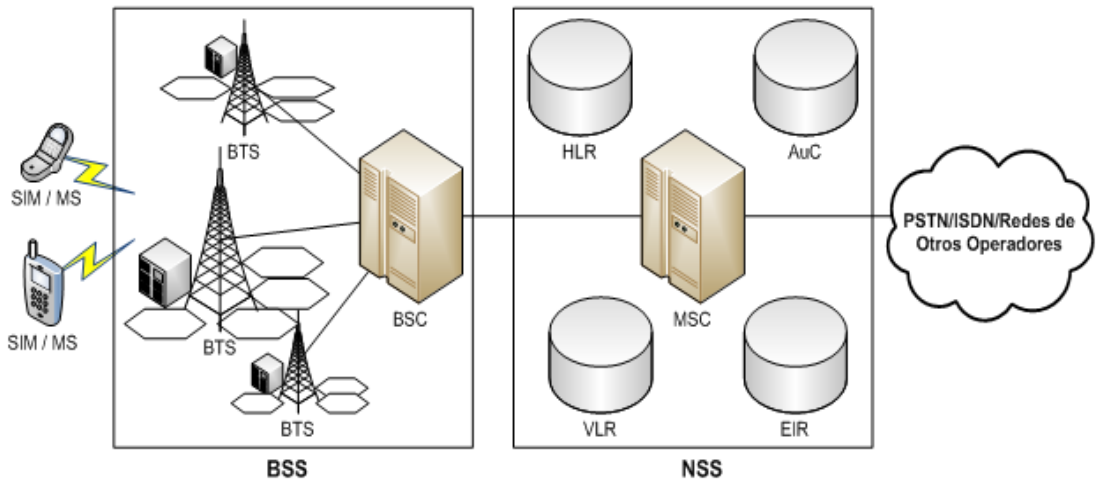
comunicaciones celulares digitales. En 1989 el compromiso de GSM se trasladó al ETSI (*European Telecommunications Standards Institute*) y las primeras especificaciones se publicaron en 1990.

GSM emprendió su servicio de forma comercial a mediados de 1991, y en 1993 ya había 36 redes GSM en 22 países. Actualmente es el estándar más popular del mundo para sistemas de telefonía móvil.

La asociación GSM (GSMA) estima que el 80% del mercado móvil mundial utiliza el estándar. GSM es utilizada por más de 1,5 millones de personas en más de 212 países y territorios. Esta omnipresencia significa que los suscriptores pueden usar sus teléfonos en todo el mundo, habilitado por acuerdos internacionales de "*roaming*" entre operadores de redes móviles. GSM difiere de las tecnologías en cuanto a que la señalización y los canales de expresión son digitales, y por lo tanto se considera una segunda generación (2G), además fue el primer estándar en implementar bajos costos dentro de la telefonía celular emplazando el servicio de mensajes cortos SMS, también llamada mensajería de texto.

2.1.2.Arquitectura de la red GSM

La demanda por parte de los usuarios de comunicaciones móviles que les admitan a éstos moverse a través de edificios, ciudades o países, ha llevado al desarrollo de nuevas redes de comunicaciones móviles que cumplan ciertas características que se acoplen ciertamente a ser robustas, escalables, fiables entre otras. El estándar de telefonía celular GSM es totalmente responsable de proporcionar cobertura a través de un territorio particular, llamado región de cobertura o mercado y se ajusta a tener una interconexión la cual se define en una red inalámbrica capaz de proporcionar servicios a los usuarios móviles a través de un país o continente determinando así una arquitectura de la red. En la Figura 2.1 se muestra de forma resumida la arquitectura de la red GSM,



SIM Modulo Identificador del Cliente
MS Equipo Móvil
BTS Estación Base Transceptora
BSC Controlador de Estación Base
HLR Registro de localización de Clientes
AuC Centro de autenticación
VLR Registro de localización de Visitas
EIR Registro de Identificación de Móviles
MSC Centro de conmutación de Móviles

Figura. 2.1. Arquitectura de red GSM

esta arquitectura es más compleja y dispone de más elementos que los presentados en la Figura 2.1. El objetivo de este proyecto es describir el servicio de mensajes de texto cortos SMS a nivel de aplicación, sin entrar en demasiados detalles de la red subyacente. La arquitectura está constituida por:

1. **Estación Móvil** (**MS**, *Mobile Station*) Son terminales digitales los cuales pueden ser portables e incluso portátiles incorporan un dispositivo **SIM** (*Subscriber Identify Module*) que es básicamente la típica Tarjeta que proporciona la información de servicios e identificación en la Red.
2. **Subsistema de Estaciones Base** (**BSS**, *Base Station Subsystem*) Es una colección de dispositivos que soportan la interface de radio de redes de conmutación. Los principales componentes del BSS son:

   a) **Estación Transceptora de Base** (**BTS**, *Base Transceiver Station*) Consta de modems de radio y el equipo de antenas.
   b) **Controlador de Estación Base** (**BSC**, *Base Station Controller*) - Gestiona las operaciones de radio de varias BTS y conecta a un único NSS.

3. **Subsistema de red y conmutación** (NSS, *Network and Switching Subsystem*) Suministra la conmutación entre el subsistema **GSM** y las redes externas junto con las bases de datos utilizadas para la gestión adicional de la movilidad y de los abonados. Los componentes son:

   a) **Centro de Conmutacion de Servicios Móviles** (**MSC,** *Mobile Services Switching Center*) Describe al centro de conmutación entre otras funciones.
   b) **Bases de Datos** (**HLR** *Home Location Register***, VLR** *Visitor Location Register***, EIR** *Equipment Identity Registry***, AuC** *Authentication Center*).

4. **Centro de mantenimiento y operaciones** (**OCC,** *Operations and Maintenance Center*) Controla la operación del sistema e inicialización de la red, además de la gestión de los equipos móviles y cobro de cuota.

2.1.3.Características Técnicas

Las bandas asignadas por la UIT (*International Telecommunication Union*) para GSM en la mayoría del mundo son de 900MHz y 1800MHz. Algunos países en América, incluyendo Ecuador, usan las bandas de 850MHz y 1900MHz, partiendo de estos datos las principales características de GSM son:

- **Acceso al medio:** TDMA/FDMA (*Time Division Multiple Access / Frequency Division Multiple Access*).
- **Canales:** 124 canales, cada canal da un servicio de 8 a 16 usuarios a la vez.
- **Bandas:** Usa dos bandas de 25MHz para transmisión y recepción de información.
- **Subida:** Usa la banda de 824-849MHz.
- **Bajada:** Usa la banda de 869-894MHz.
- **Ancho de banda:** 200KHz
- **Modulación:** GMSK (*Gaussian Minimum Shift Keying*).
- **Velocidad máxima del canal de radio:** 270.833Kbps
- **Duración de bit:** 3,692msec.
- **Longitud de trama:** 4,615msec

- **Longitud de slot de tiempo:** 577usec.
- **Codificación de voz:** RPE-LPT 13Kbps (*Regular Pulse Excitation-Long Term Prediction*).
- **Potencia de salida:** de 20mW a 20W

2.1.4.Servicios

2.1.4.1. Servicios Básicos

Hay dos tipos de servicios que se pueden ofrecer a través de GSM los cuales son la telefonía y los datos.

- Los servicios de telefonía son principalmente servicios de voz que proveen a los suscriptores la capacidad total para comunicarse con otros, incluyendo el equipo terminal necesario y otros servicios además de la telefonía normal, llamadas de emergencia los cuales son DTMF (*Dual-Tone Multifrequency*), Fax, Buzón de Voz y el más importante el cual esta estipulado en el proyecto; SMS (*Short Message Service*).

- Los servicios de datos tienen la capacidad necesaria para transmitir datos entre puntos de acceso, creando una interfaz de red.

2.1.4.2. Servicios Suplementarios

GSM soporta un amplio conjunto de servicios suplementarios los cuales complementan a los servicios básicos mencionados anteriormente, a continuación se listan algunos de ellos

- Marcación abreviada

- Identificación de llamadas.

- Conferencias.
- Bloqueo de llamadas entrantes y salientes
- Facturación de servicios y llamadas.
- Transferencia de llamadas.
- Claves de seguridad
- Mantenimiento de llamadas
- Llamadas en espera
- Terminación de llamadas de usuarios ocupados
- Grupos cerrados de usuarios
- Tarifación

2.2. MENSAJES CORTOS SMS

2.2.1.Servicio SMS

El servicio SMS permite trasladar un mensaje de texto entre una Estación Móvil (MS) y otra entidad denominada SME (*Short Messaging Entity*) a través de un centro de servicio SMSC (*Short Message Service Center*) como se muestra en la Figura 2.2. La entidad que recibe el mensaje de texto puede ser otro terminal MS o puede estar situada en una red fija. Cuando se envía un mensaje para solicitar algún tipo de servicio, un extremo es una MS y la otra es un servidor que atiende las peticiones, el proyecto en marcha se basa en el servicio de SMS mencionado finalmente como muestra la Figura 2.3.

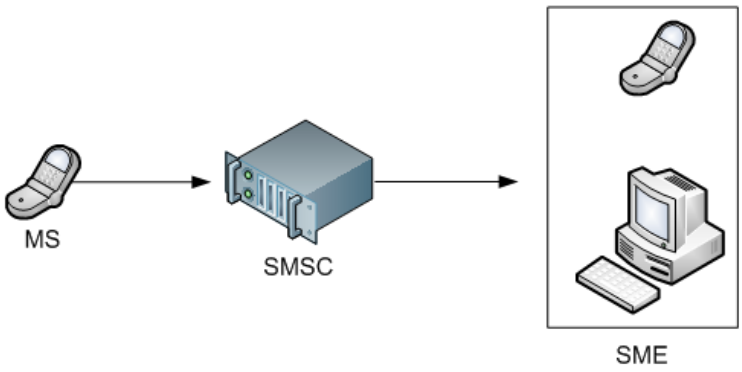


Figura. 2.2. Servicio SMS

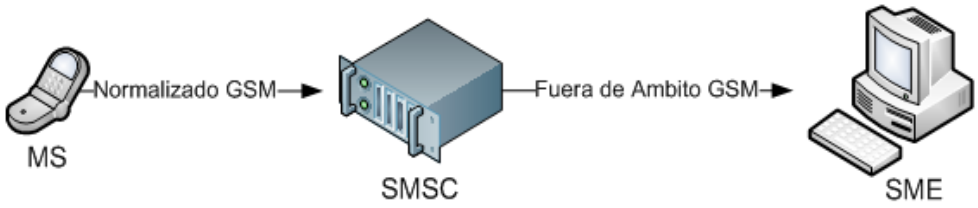
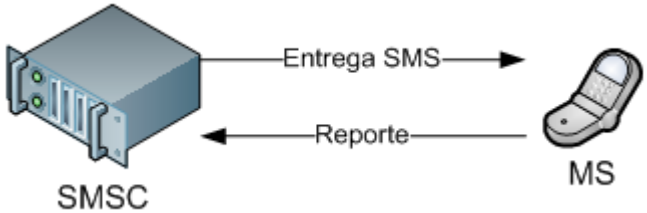


Figura. 2.3. Envío de un SMS entre un MS y una entidad Fija/Servidor

El servicio de mensajes de texto SMS se divide en dos servicios básicos los cuales se detallan a continuación y se muestran en la Figura 2.4.

a) **SM MT** (*Short Message Mobile Terminated Point-to-Point*) Es un servicio de entrega de mensaje que transita desde el SMSC hasta un MS, obteniendo de dicha transacción un reporte de lo ocurrido.
b) **SM MO** (*Short Message Mobile Originated Point-to-Point*) Es un servicio de envío de mensaje que transita desde el MS hasta el SMSC, obteniendo de dicha transacción un reporte de lo ocurrido.



a) Entrega de SMS / SM MT

b) Envío de SMS / SM MO

Figura. 2.4. Servicios básicos – SM MT / SM MO

2.2.2.Formatos del SMS

Dentro de las especificaciones de mensajes cortos SMS existe la posibilidad de hacer el envío de mensajes de dos maneras las cuales se detallan

a) Modo PDU.- Es aquella estructura de mensaje que lleva consigo bits de información específica además de funciones de control para la presentación del mensaje.

b) Modo Texto.- Es aquella estructura de mensaje que está conformada por caracteres de texto, números y símbolos, cabe recalcar que es un modo de gama media o alta codificación que no se encuentra en todos los terminales.

2.2.3.Arquitectura de red SMS

Para la estructura básica de la red SMS se presentan varias entidades las cuales se explican de manera vertiginosa y se hacen presentes en la Figura 2.5

- **MS**. Estación Móvil.
- **MSC**. Centro de Conmutación
- **SMS-GMSC** (*Gateway MSC for Short Message Service*) Es una puerta de enlace para que el MSC resuelva el servicio del SMS
- **SMS-IWMSC** (*Internetworking MSC for Short Message Service*) Puerta para hacer la interconexión entre el MSC para que resuelva el servicio SMS.
- **SMSC**. Centro de servicios

- **HLR**. Base de datos para registro de localización de clientes.
- **VLR**. Base de datos para registro de localización de visitas.

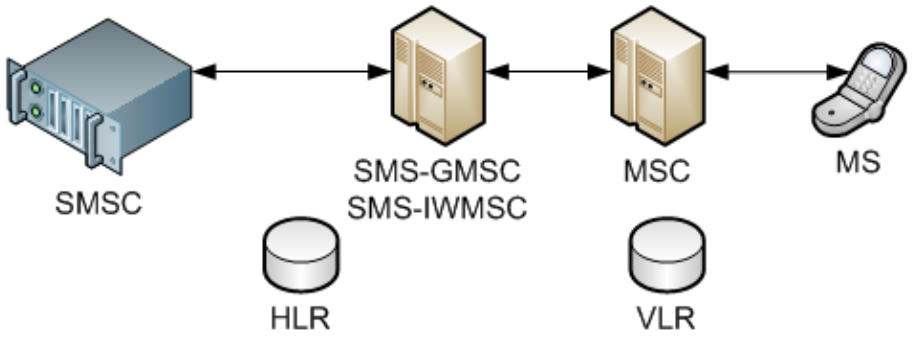


Figura. 2.5. Arquitectura de red SMS

2.3. MODEM GSM Y COMANDOS AT

2.3.1.Modem GSM

Es un dispositivo electrónico que acepta un modulo de identidad para el suscriptor (SIM) programado para transmitir datos de forma inalámbrica usando la red GSM, con el cual se puede conectar cualquier sistema digital usando sus servicios; entre estos los mensajes de texto SMS. Se observa a continuación el diagrama lógico de la conexión del modem GSM.

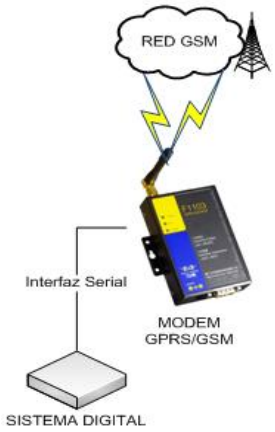


Figura. 2.6. Diagrama lógico de conexión de modem GPRS/GSM

2.3.2.Interfaz con Módems: Comandos AT&AT+

La comunicación del modem GSM se la realiza con interfaces de diferentes tipos, en la figura 2.6 se especifica la indicada para la realización de este proyecto, la cual establece una conexión dedicada con el modem para poder transmitir de manera fiable y robusta los datos de control basados en comandos AT *Hayes* comúnmente conocidos como comandos AT. Una vez establecida la conexión el modem interpreta mediante su interfaz que existe una conexión y está listo para recibir órdenes del dispositivo al cual esté conectado, entrando así en *modo comando*.

Las respuestas del modem GSM a los comandos AT pueden ser: **códigos de resultado** ó **códigos de error** como se detalla en la tabla 2.1

Tabla. 2.1. Códigos de resultado de la operación de comandos AT

CODIGO	DETALLE
OK	Indica que el comando y cualquier parámetro especificado fueron válidos
ERROR	Indica que ha ocurrido un error en el proceso de ejecución del comando o que la sintaxis del comando ejecutado es incorrecta.

Los comandos AT&AT+ son cadenas ASCII las cuales son procesadas por el modem GSM para entregar un resultado que normalmente es otra cadena ASCII.

A continuación se describirán algunos comandos AT, que se usan específicamente para la realización del proyecto. La ejecución de dichos comandos se realiza a través de una conexión serial a un computador utilizando el programa predeterminado *HyperTerminal* con los parámetros de comunicación que se detallan en la figura 2.7.

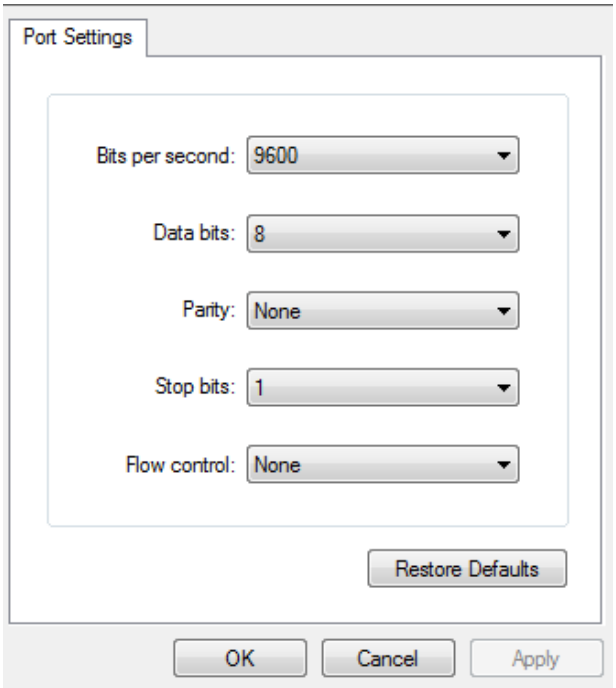


Figura. 2.7. Parámetros de comunicación para conexión entre Modem y otro dispositivo

1. **AT+CPMS** (*Preferred SMS Message Storage*) Selecciona el espacio de memoria de almacenamiento para la ejecución de mensajes de texto SMS, ya sea lectura, escritura etc. La figura 2.8 muestra la ejecución de este comando.

Formato: AT+CPMS=<mem1><mem2><mem3>

Donde:

- *mem1:* memoria desde la cual los mensajes son leídos y borrados
- *mem2:* memoria desde donde se realizan las opciones de escritura y envió
- *mem3:* memoria a la cual se prefiere almacenar los SMS recibidos

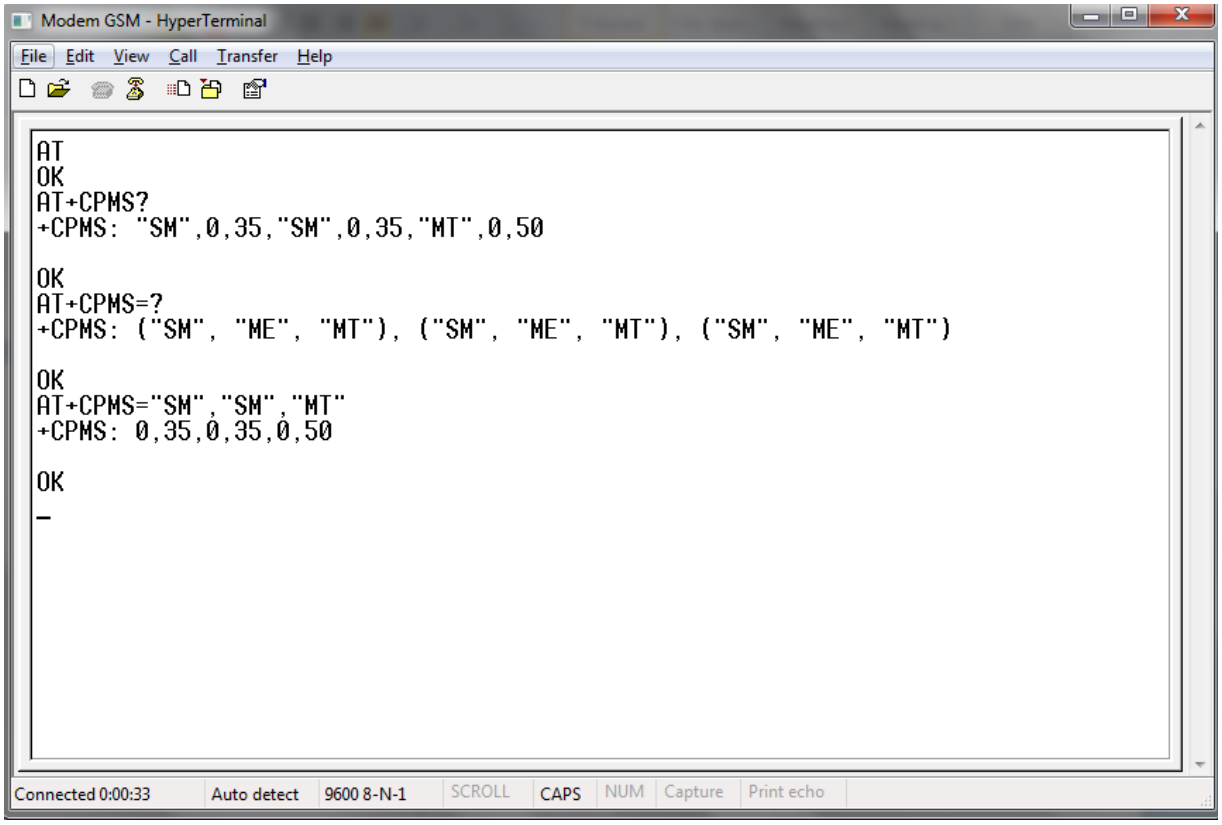


Figura. 2.8. Ejecución de comando AT+CPMS

La respuesta del Modem GSM referente al comando AT+CPMS indica las siglas "SM", "ME" ó "MT" las cuales se detallan a continuación:

SM: Referente al almacenamiento en la tarjeta SIM.

ME: Referente al almacenamiento en el Modem GSM.

MT: Las opciones SM y MT combinadas.

2. **AT+CMGF** (Select SMS *Message Format*) Selección del formato de entrada y salida para los mensajes de texto SMS. La figura 2.9 muestra la ejecución de este comando.

Formato: AT+CMGF=<modo>

Donde:

- *modo:* puede ser 0 para modo PDU ó 1 para modo TEXTO.

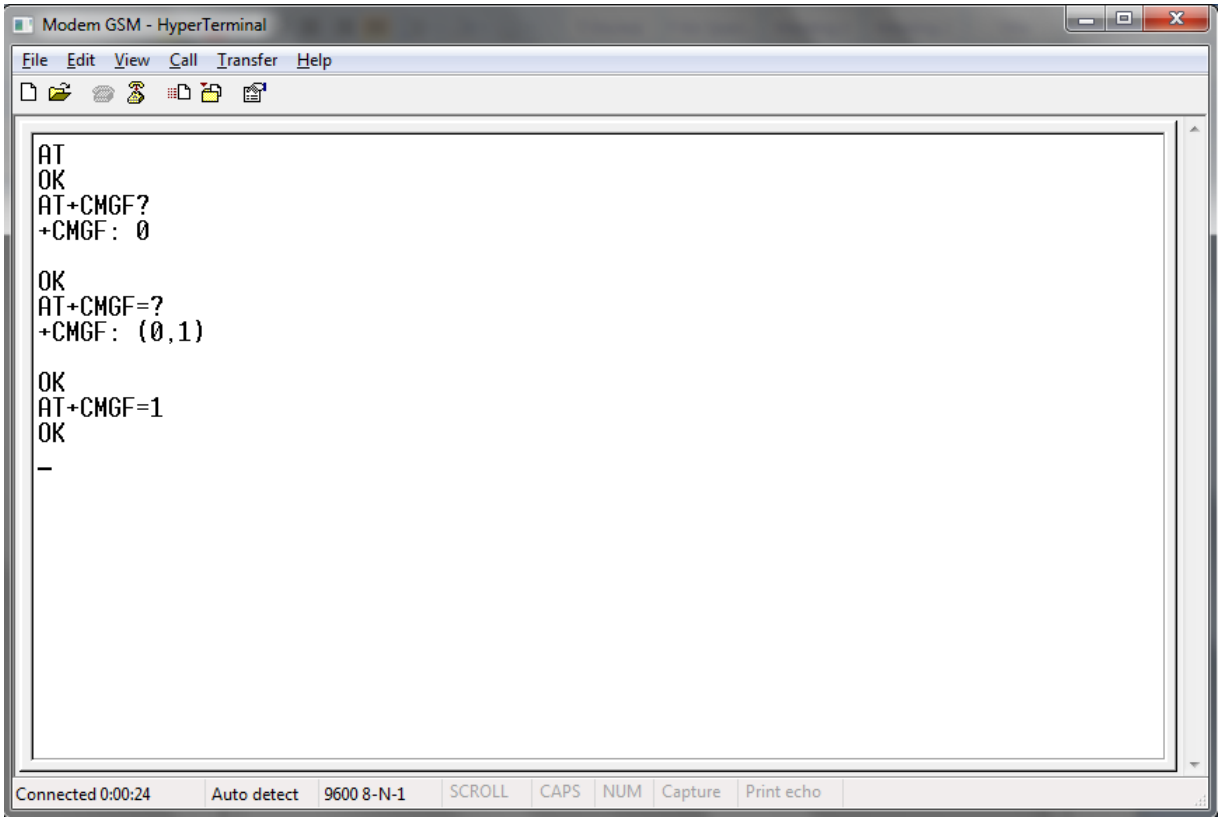


Figura. 2.9. Ejecución de comando AT+CMGF

3. **AT+CMGL** (*List SMS Messages From Preferred Stored*) Devuelve los mensajes que han sido almacenados con el estado indicado desde la memoria *mem1*. La figura 2.10 muestra la ejecución de este comando.

Formato: AT+CMGL=<estado>

Donde:

- *estado:* identifica la categoría de mensajes a listar, los cuales pueden ser:

"REC UNREAD": mensajes recibidos pero no leídos.

"REC READ": mensajes recibidos y leídos.

"STO UNSEND": mensajes escritos y almacenados pero no enviados

"STO SEND": mensajes enviados.

"ALL": todos los mensajes.

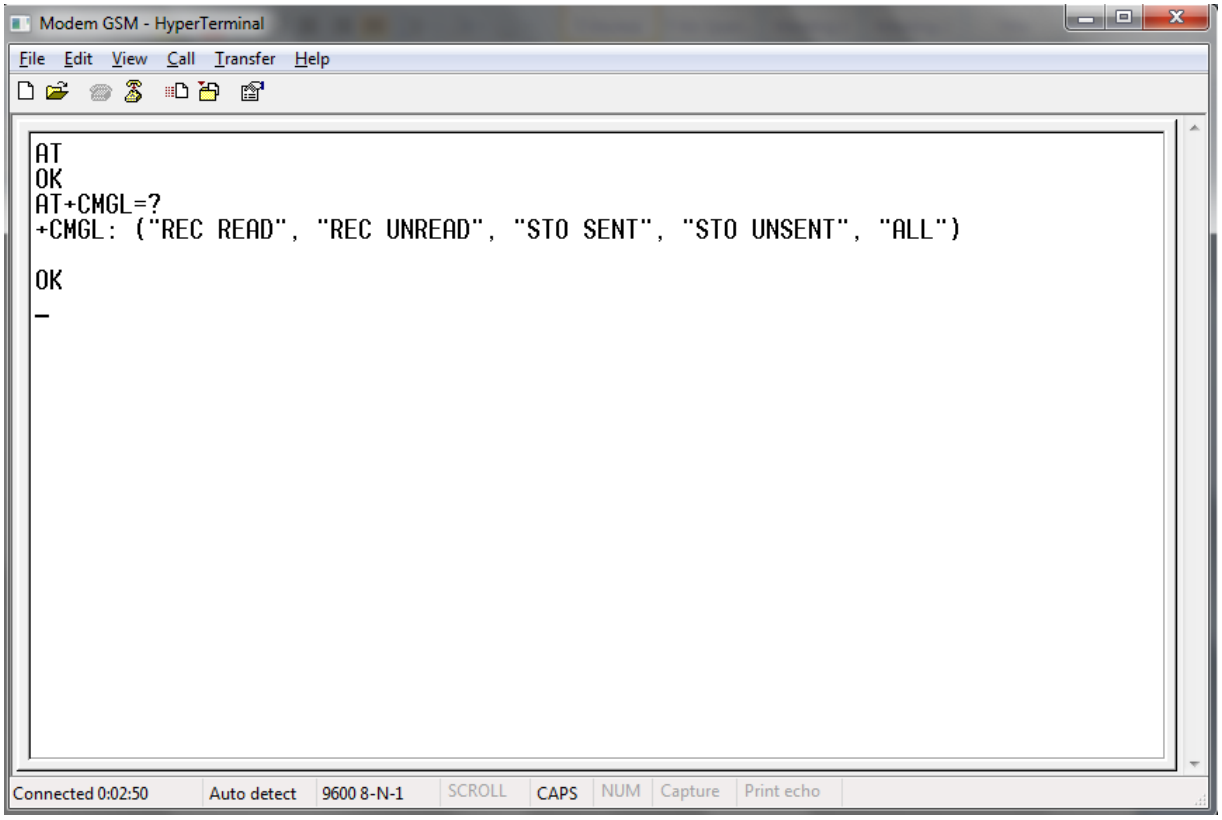


Figura. 2.10. Ejecución del comando AT+CMGL

4. **AT+CMGS** (*Send SMS Message*) Permite realizar el envío de un mensaje SMS desde un equipo terminal a otro, depende del modo en el cual se trabaje. La figura 2.11 muestra la ejecución de este comando.

Formato: AT+CMGS=<número><CR>

Donde:

- *número:* El número telefónico de destino para el envió del SMS.
- *CR:* carácter de retorno (Enter).

El modem responde enviando un carácter “>” que indica que se puede escribir el mensaje a ser enviado, una vez escrito el mensaje se debe ingresar el carácter de envío <ctrl+z> el cual es el 0x26 en el código ASCII.

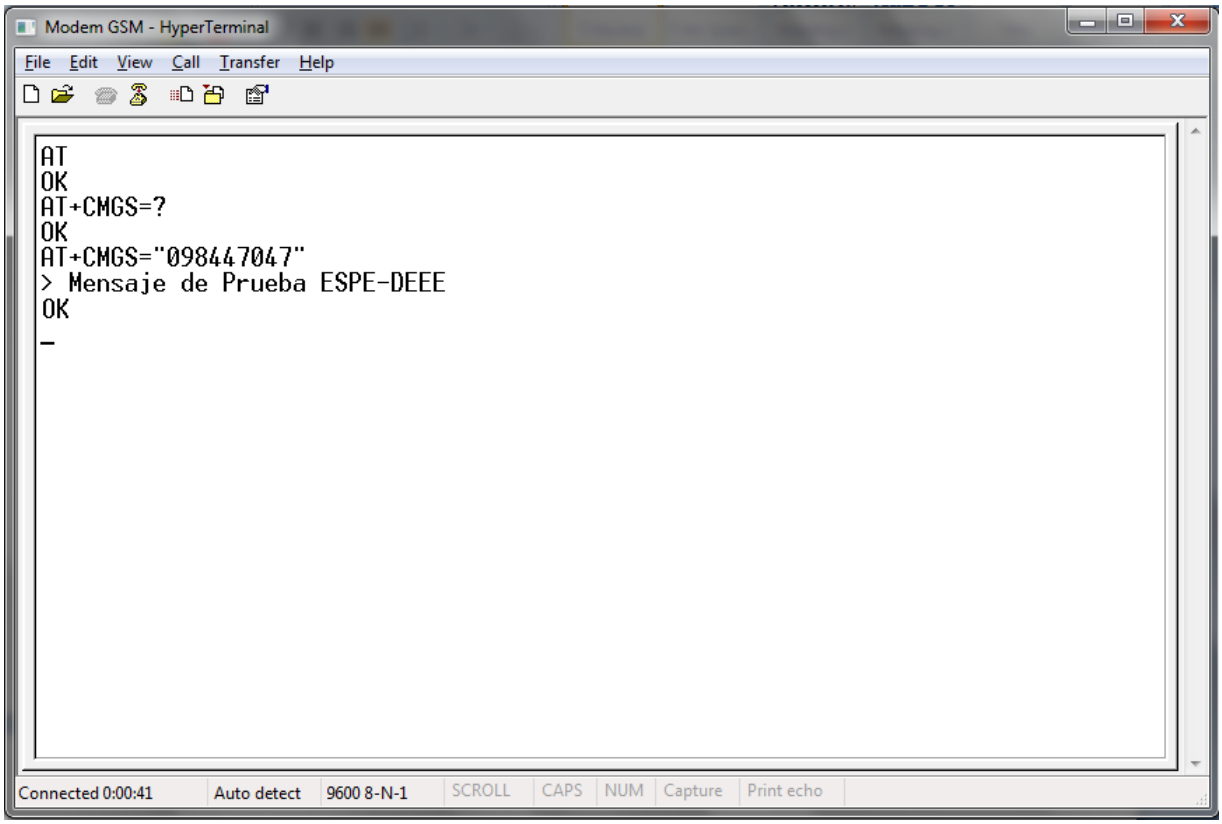


Figura. 2.11. Ejecución de comando AT+CMGS

5. **AT+QMGDA** (*Delete all SMS*) Elimina los mensajes de texto, dependiendo del tipo y modo en el que este configurado el modem GSM. La figura 2.12 muestra la ejecución de este comando.

Formato: AT+QMGDA=<tipo>

Donde:

- *tipo:* identifica la categoría de mensajes a borrar, los cuales pueden ser:

"DEL READ": elimina los mensajes leídos.

"DEL UNREAD": elimina los mensajes no leídos.

"DEL SENT": elimina los mensajes enviados.

"DEL UNSET": elimina los mensajes que no han sido enviados.

"DEL INBOX": elimina la bandeja de entrada.

"DEL ALL": elimina todos los mensajes.

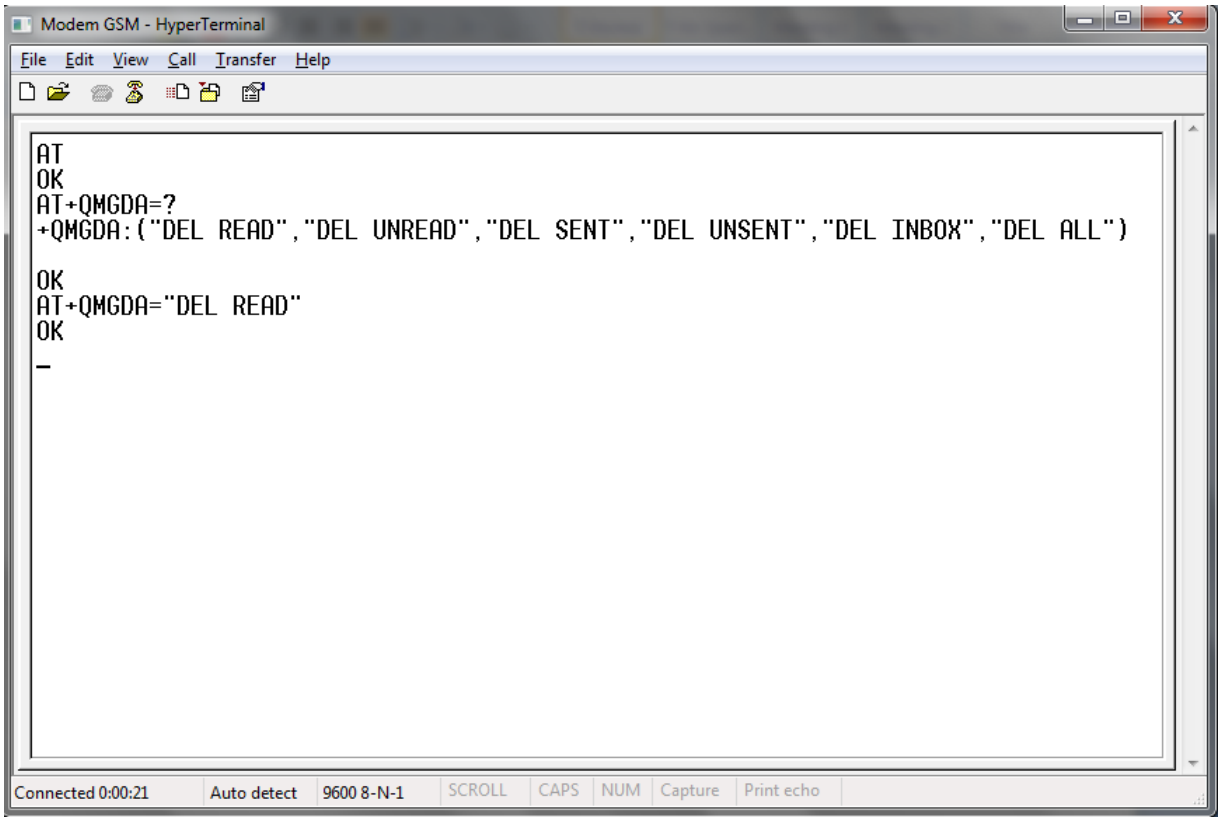


Figura. 2.12. Ejecución de comando AT+QMGDA

2.4. Java y MySQL

2.4.1.Instalación y configuración del servidor MySQL

El proyecto a seguir en este documento se basa en un ambiente usuario – servidor en donde el SERVlet será el encargado de tener un registro para administrar una base de datos, es por esta razón que es necesaria la instalación y configuración de dicho servidor.

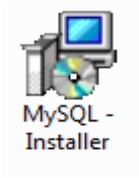


Figura. 2.13. Asistente de Instalación MySQL

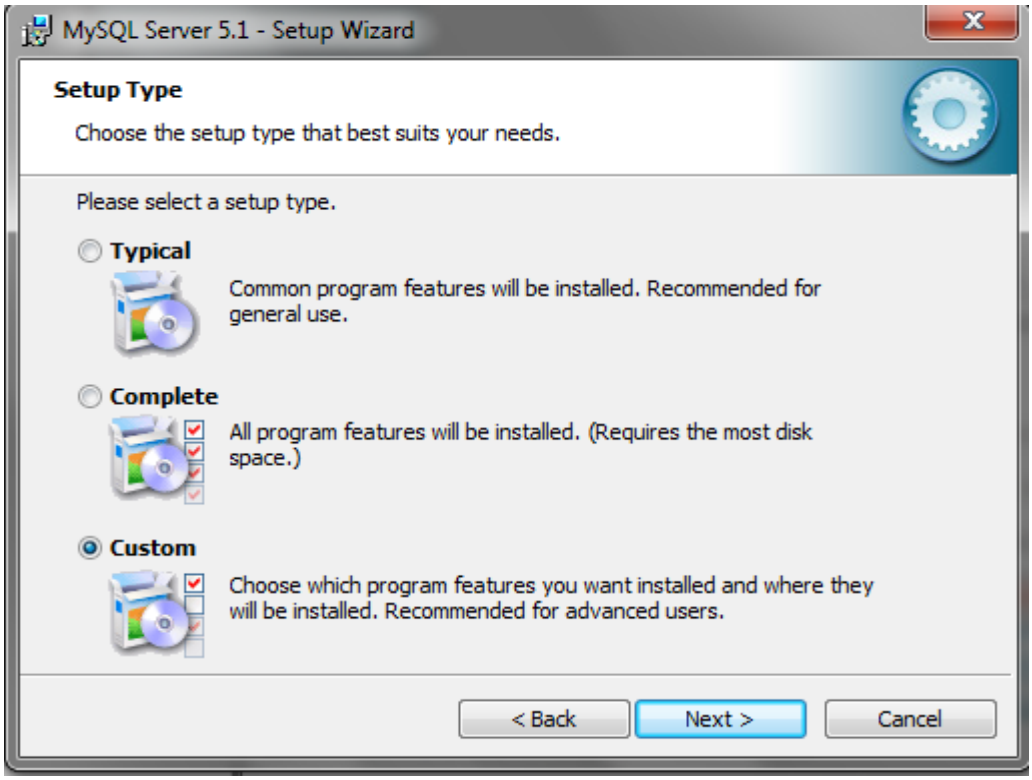
El asistente de instalación de MySQL, en combinación con el asistente de configuración, le permite a un usuario instalar y configurar un servidor MySQL que esté listo para el uso inmediatamente a continuación de la instalación. Después de haber obtenido el instalador se debe seguir algunos sencillos pasos.

Figura. 2.14. Página principal para la instalación de MySQL

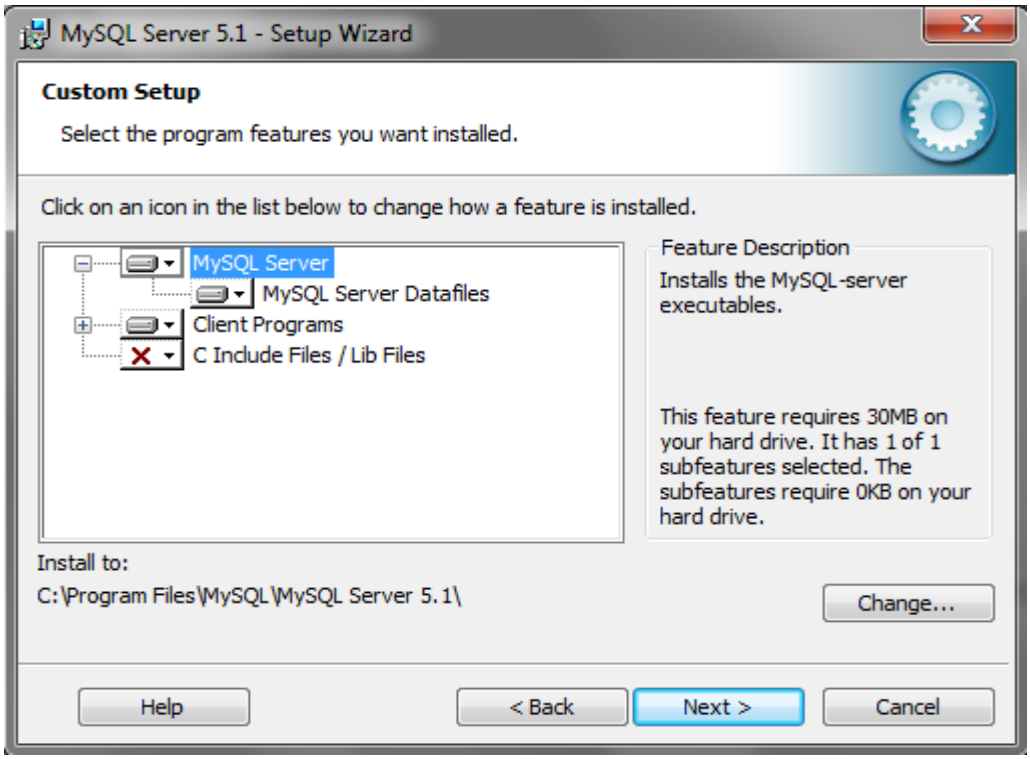
a) **Escoger tipo de instalación:** Hay tres tipos de instalación: típica, completa y personalizada. Se recomienda la instalación personalizada para tener más control sobre los paquetes que se desea instalar como detalla en la figura 2.15 a y b.

b) **Confirmación:** Una vez que hayamos seleccionado los paquetes y carpetas en donde se va a instalar el servidor, aparecerá la pantalla de confirmación para instalar como se detalla en la figura 2.16

c) **Finalización:** El asistente será el encargado de finalizar de manera satisfactoria la instalación para a continuación poder configurar y usar de manera confiable y fiable el servidor. La figura 2.17 muestra la pantalla de finalización de instalación del servidor e inicialización de configuración del mismo.



a) Selección de tipo de instalación



b) Opciones del paquete MySQL

Figura. 2.15. Instalación personalizada MySQL

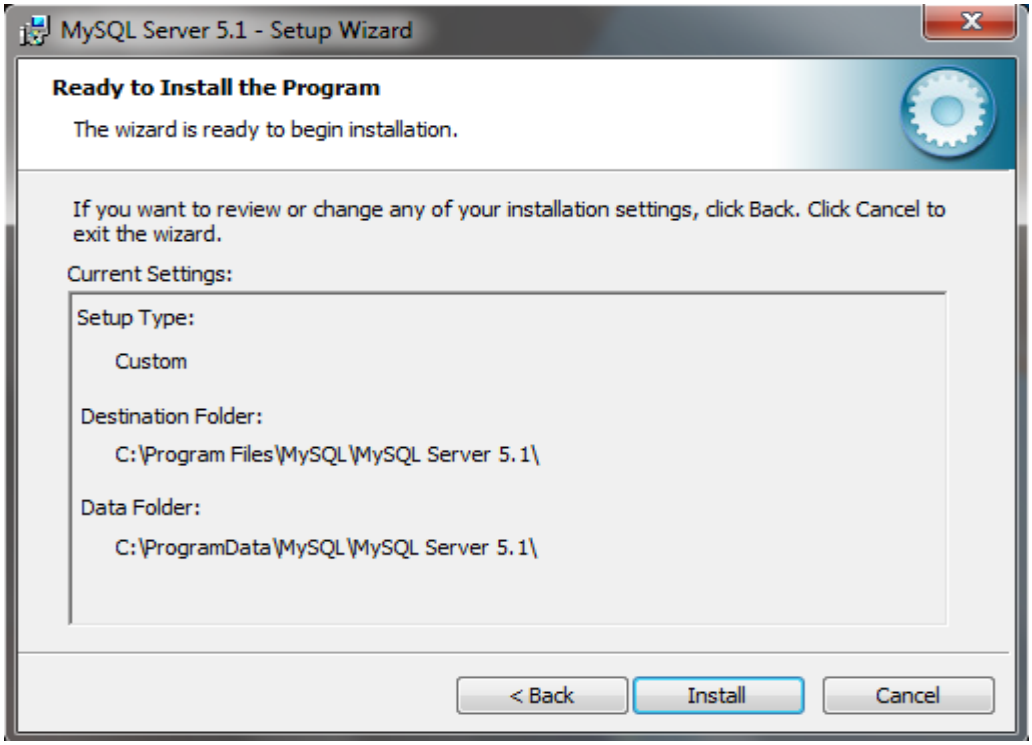


Figura. 2.16. Pagina de confirmación para instalar MySQL

Figura. 2.17. Finalización de asistente MySQL

Una vez que el servidor ha sido instalado satisfactoriamente entrará al modo de configuración el cual se detalla a continuación:

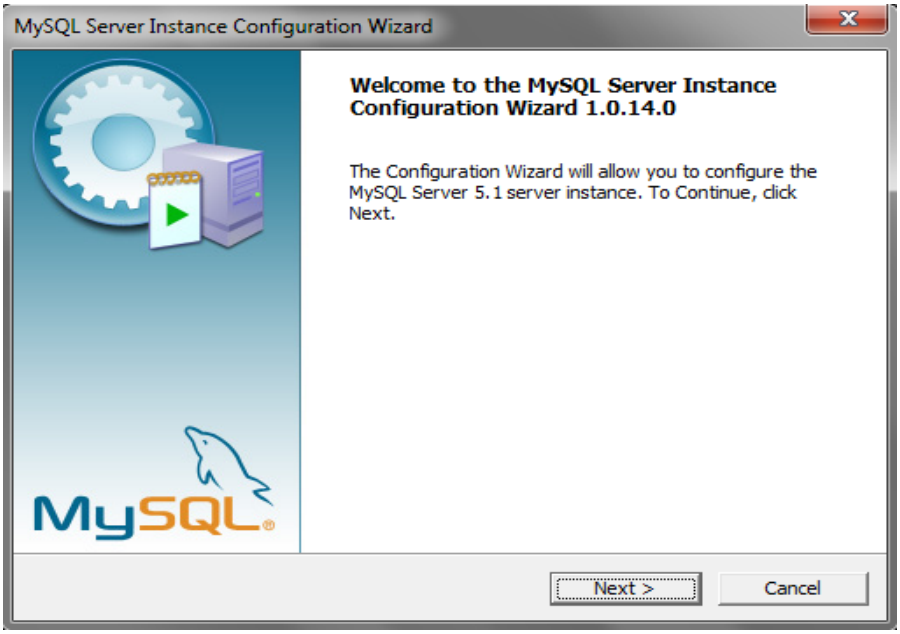


Figura. 2.18. Página principal para configuración de MySQL

a) **Tipo de configuración:** Para obtener una configuración más detallada y mejor estructurada del servidor, es necesario escoger la opción "*Detailed Configuration*" para crear un sentido de optimización entre la máquina y el servidor, en la figura 2.19 se muestra la selección del tipo de configuración.

b) **Tipo de Servidor:** Depende de qué tipo de procesamiento y transacciones realiza el servidor. Se recomienda escoger la opción "*Server Machine*" para que el consumo de memoria sea medio, además de poder ejecutar otro tipo de aplicaciones que estén instaladas en la maquina en donde el servidor ha sido instalado, se detalla la selección del tipo de servidor en la figura 2.20.

c) **Uso de Base de datos:** Se presenta las opciones para conformar que tipos de base de datos y que uso van a tener en el servidor, se debe usar bases de datos genéricas y usualmente multifuncionales por lo que se aconseja escoger la opción "*Multifunctional Database*" la cual va a optimizar las transacciones del servidor orientando a que el motor de MySQL funcione en alta velocidad. La figura 2.21 detalla de mejor manera lo descrito en este punto.

d) **Disco para InnoDB:** InnoDB es un motor de bases de datos que actúa directamente con MySQL y dentro de la configuración representa el uso de disco para el almacenamiento de datos de la máquina para su utilización como se muestra en la figura 2.22.

e) **Conexiones aproximadas al servidor:** Describe las conexiones concurrentes que el servidor va a tener para realizar transacciones y búsquedas en la base de datos, se detalla en la figura 2.23 la selección correcta para que concuerde con las transacciones aproximadas de los usuarios.

f) **Opciones de Red:** Cuando el servidor está en una red usa puertos e interfaces para que la comunicación se realice, el puerto predeterminado es el 3306 el cual puede ser cambiado

dentro de la configuración, además puede añadir excepciones para que el cortafuegos no interrumpa la comunicación, se observa en la figura 2.24 la configuración.

g) **Opciones de seguridad:** Se debe ingresar una clave para que cualquier tipo de transacción que se vaya a realizar se deba conectar con un usuario y clave, para así evitar cualquier tipo de amenaza contra los datos que están guardados como se observa en la figura 2.25.

h) **Ejecutar:** Finalmente se ejecutan todas las configuraciones para que surtan efecto en el servidor como se muestra en la figura 2.26.

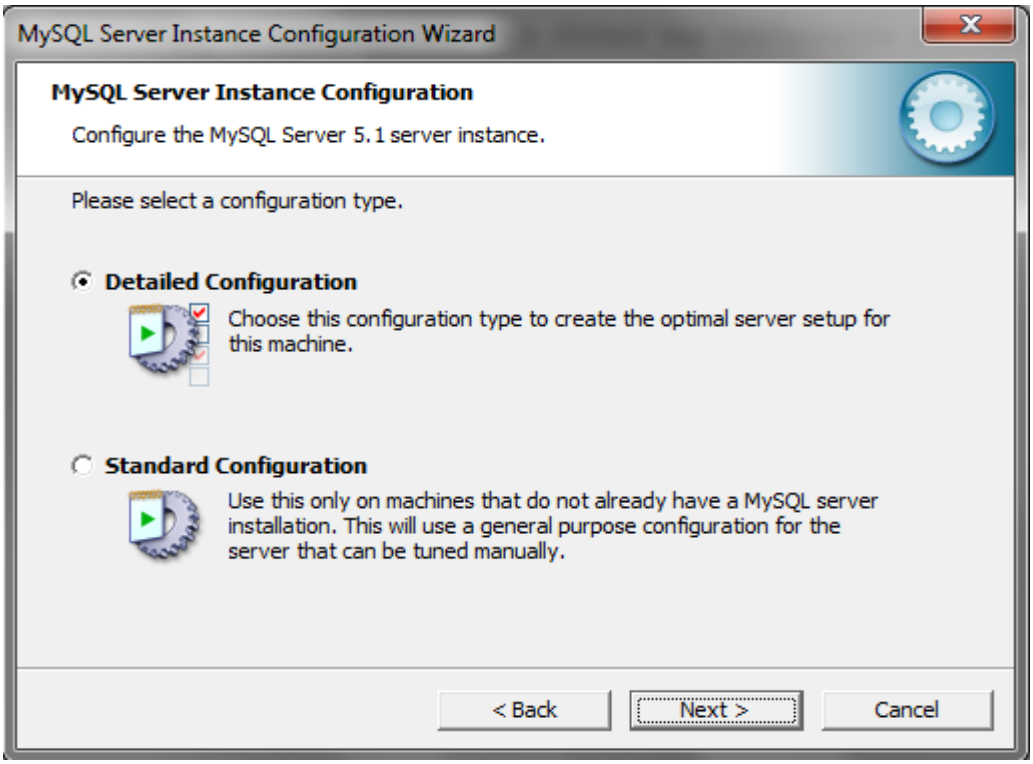


Figura. 2.19. Tipo de configuración de MySQL

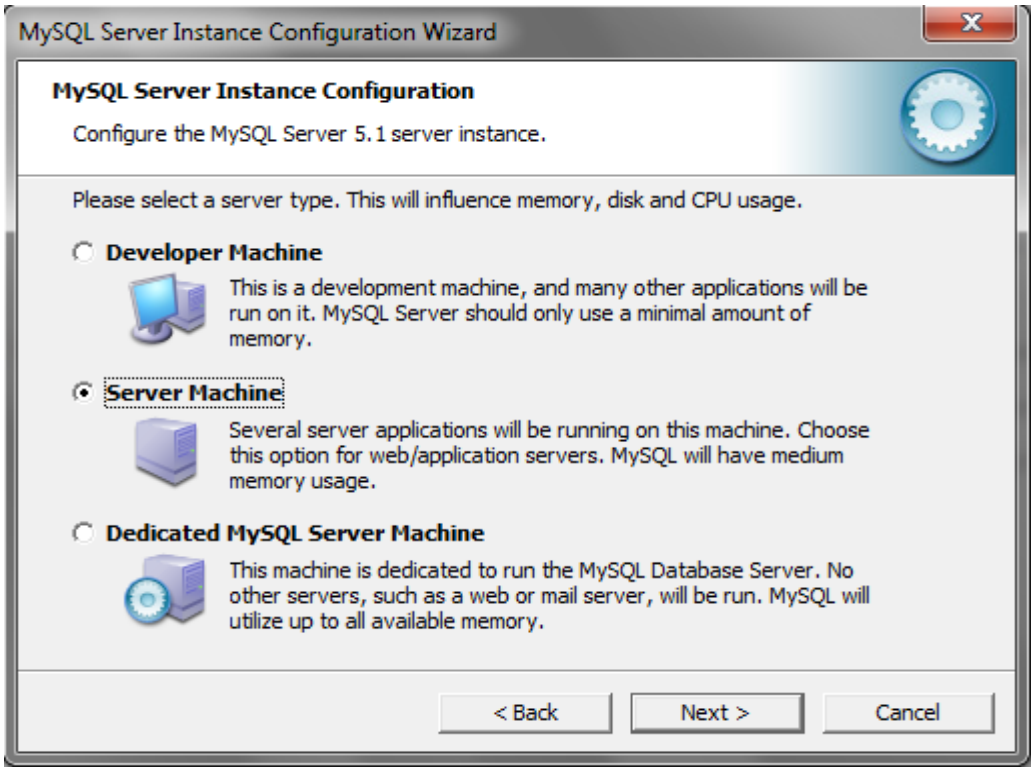


Figura. 2.20. Tipo de servidor de MySQL

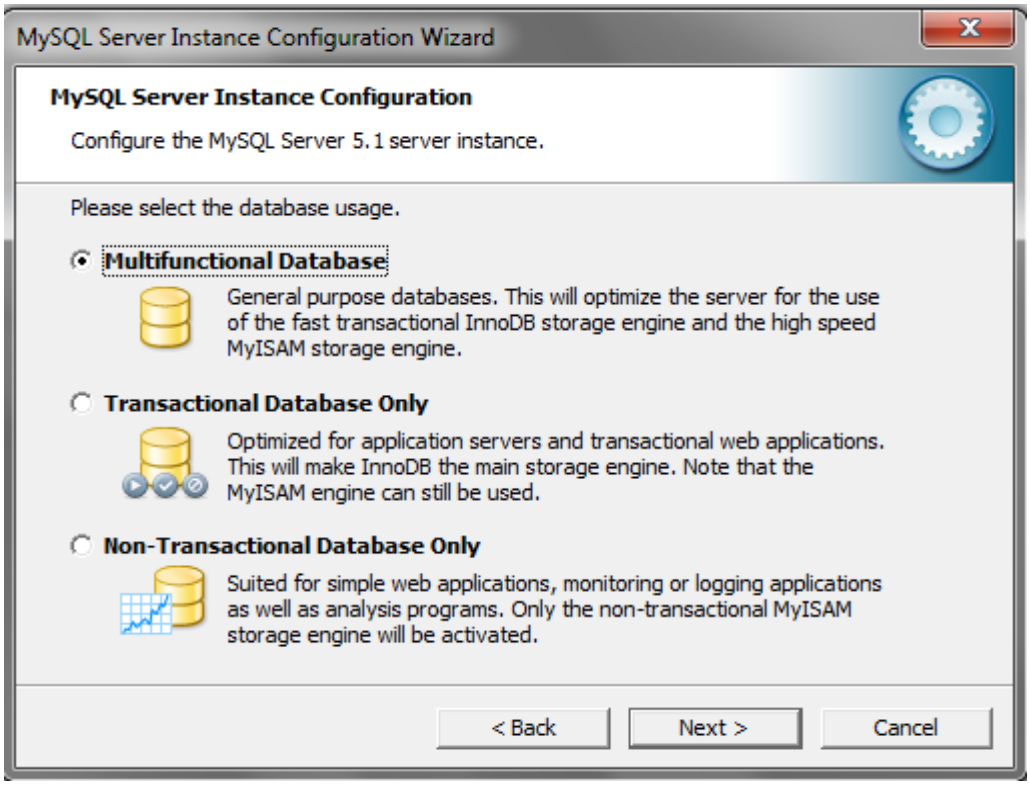


Figura. 2.21. Uso de bases de datos MySQL

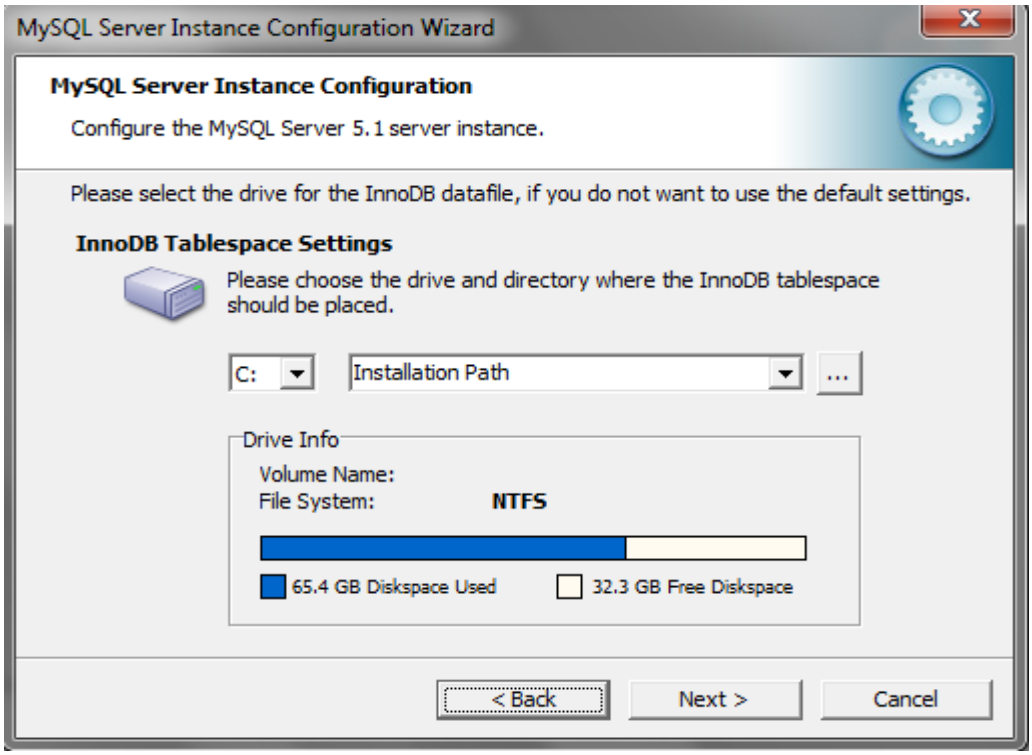


Figura. 2.22. Disco para InnoDB MySQL

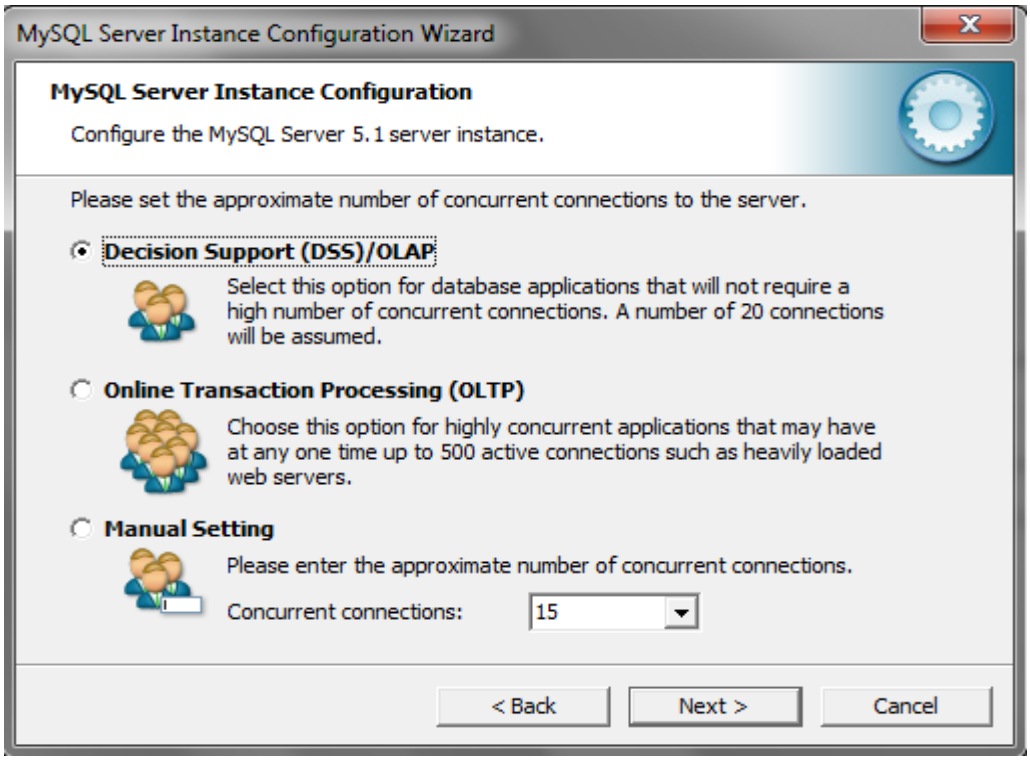


Figura. 2.23. Conexiones aproximadas al servidor MySQL

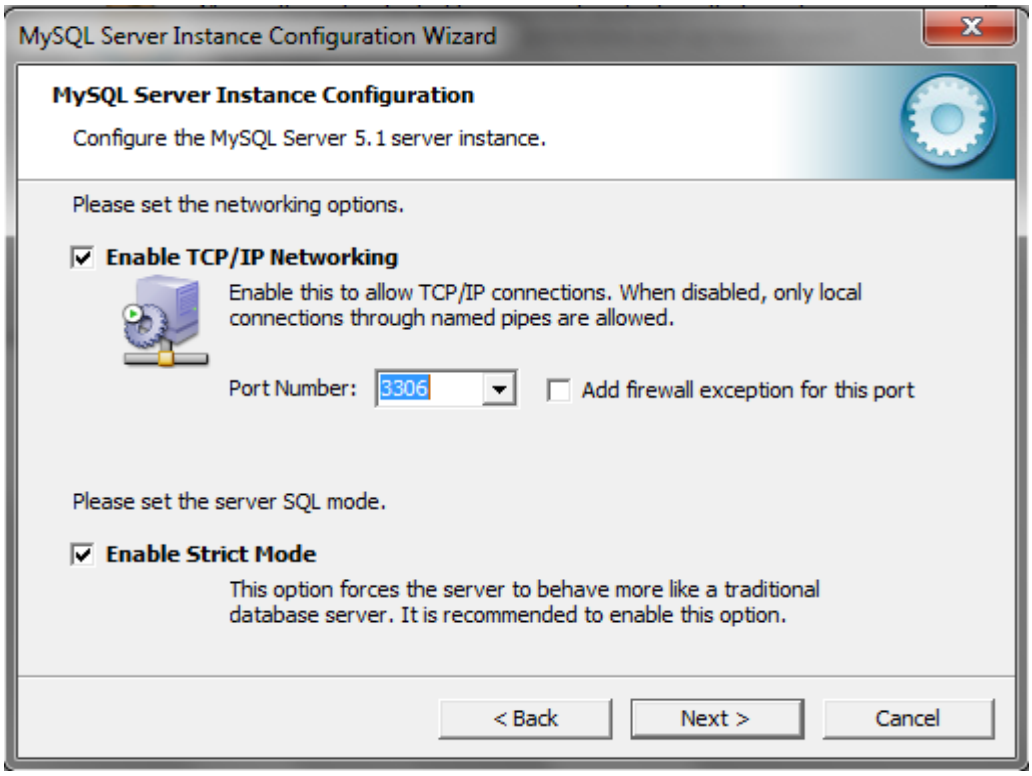


Figura. 2.24. Configuración de red para MySQL

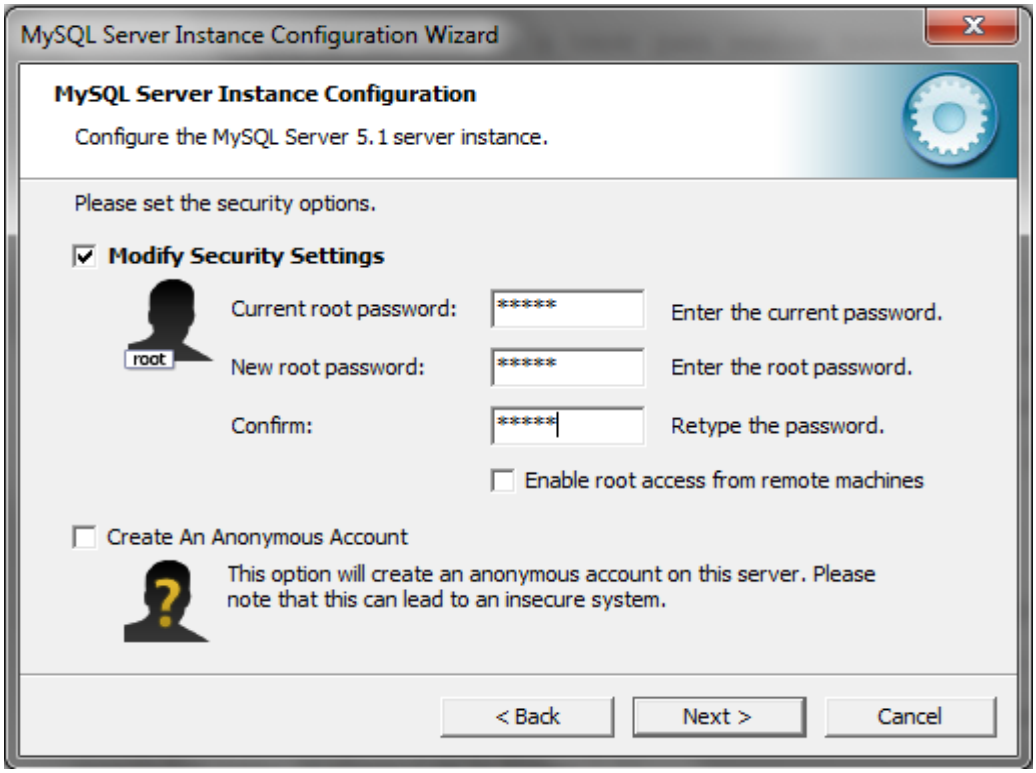


Figura. 2.25. Configuración seguridad para MySQL

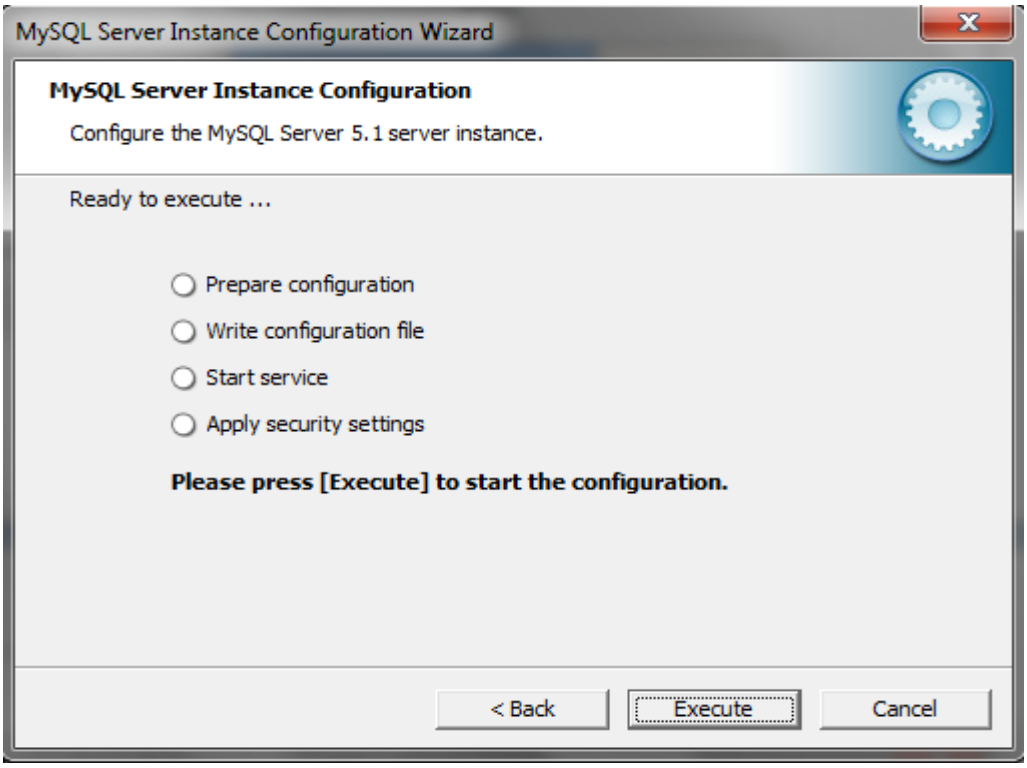


Figura. 2.26. Ejecutar configuración de MySQL

2.4.2.Conexión de Java y Bases de datos MySQL

Para el presente proyecto se debe considerar que la aplicación se desarrolla en lenguaje Java, es por eso que se aprovecha de mejor manera todas sus librerías para poder hacer una aplicación multiplataforma, es decir, que sirva para diferentes sistemas operativos. Se debe recalcar también que para la realización del proyecto se ha usado el IDE desarrollador *NetBeans* el cual tiene incluido conectores de bases de datos MySQL.

En el programa *NetBeans* se debe hacer la implementación de librerías que permitan realizar una conexión satisfactoria con el servidor MySQL instalado, es por eso que se usa las librerías embebidas al programa para obtener buenos resultados de conexión. En la siguiente figura se muestra la adición de una librería que usa el controlador JDBC para establecer conexión entre aplicación y servidor para un proyecto o archivo que se esté creando en el centro de desarrollo *NetBeans.*

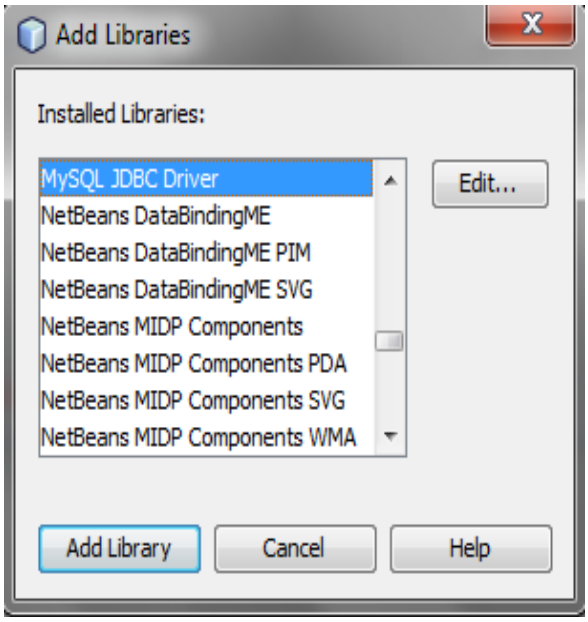


Figura. 2.27. Adición de librería MySQL

Con esta adición de librería la aplicación quedaría totalmente lista para realizar conexiones satisfactorias con el servidor MySQL.

2.5. JAVA 2 MICRO EDITION

2.5.1.Introducción a J2ME

J2ME es la versión de Java orientada a aplicaciones en dispositivos electrónicos con capacidades computacionales y graficas muy reducidas, como ejemplo: teléfonos móviles, PDAs o electrodomésticos inteligentes.

J2ME sujeta una mínima parte de las APIs de Java, esto es debido a que la edición estándar de APIs de Java ocupa más cantidad de memoria y los dispositivos mencionados anteriormente disponen de una cantidad mínima de memoria.

2.5.2.MIDlets

Los MIDlets son aplicaciones desarrolladas mediante un perfil pensadas básicamente para que puedan ser descargadas a través de internet, o simplemente dentro de una red local, puertos infrarrojos, bluetooth o cable de datos.

El MIDlet se compone de un archivo *.jar (Java Archive)* en el cual se empaquetará la aplicación y un archivo denominado manifestó *.jad (Java Application Descriptor)* el cual obtiene algunas configuraciones y pequeños archivos de ejecución.

2.5.2.1. Ciclo de vida de un MIDlet

El MIDlet tiene cinco fases en su ciclo, las cuales se describen a continuación y se muestran en la figura 2.28.

a) **Localización:** Es la etapa previa a la instalación del MIDlet la cual realiza la descarga del mismo ya sea por cualquiera de los medios mencionados anteriormente y localiza el registro de instalación para su ejecución.

b) **Instalación:** Es la etapa en donde se controla el proceso de instalación informando al usuario de manera gráfica la evolución de la misma.

c) **Ejecución:** Etapa cuyo funcionamiento es ejecutar todos los estados del MIDlet en función a eventos externos al mismo.

d) **Actualización:** Es detectado siempre y cuando el MIDlet haya provenido de una descarga vía web, para así notificar que existe una actualización del mismo.

e) **Borrado:** Fase de borrado del MIDlet que se liga a la decisión del usuario para que sea eliminado del dispositivo.

Figura. 2.28. Ciclo de vida de un MIDlet

2.5.2.2. Desarrollo de Aplicaciones

Para el desarrollo de aplicaciones J2ME se debe considerar que no todas las arquitecturas de los dispositivos móviles son exactamente iguales, es por eso que se debe desarrollar la aplicación de la forma más genérica posible para así abarcar la mayoría de dispositivos en el mercado. A continuación se detalla algunos conceptos necesarios para hacer de nuestra aplicación la más genérica posible.

2.5.2.2.1.CLDC (*Connected, Limited Device Configuration*)

La configuración de dispositivos conectados Limitada (CLDC) define la base de un conjunto de interfaces de programación de aplicaciones y librerías para dispositivos con

recursos limitados como fueron algunos mencionados anteriormente. Cuando se combina con un perfil como el MIDP (*Mobile Information Device Profile*), proporciona una consistente plataforma Java para desarrollar aplicaciones para ejecutarse en dispositivos con memoria limitada.

2.5.2.2.2.MIDP (*Mobile Information Device Profile*)

Permite que la información del MIDlet sea específica en el ámbito de marcar una ruta de desarrollo, es decir permite escribir aplicaciones descargables, que contengan servicios y hasta aplicaciones para poder conectarse a una red u otros dispositivos de manera sencilla.

2.5.2.2.3.Creación de aplicaciones MIDP/CLDC

La creación de aplicaciones MIDP/CLDC permiten la facilidad que se le pueda dar al desarrollador para que implemente y fragmente sus ideas de mejor manera, van muy de la mano con ambientes gráficos tales como diagramas de flujo y analizadores de errores para que su realización no sea imprecisa. En la siguiente figura se muestra algunas de las características al momento de crear aplicaciones MIDP/CLDC con el programa desarrollador *NetBeans.*

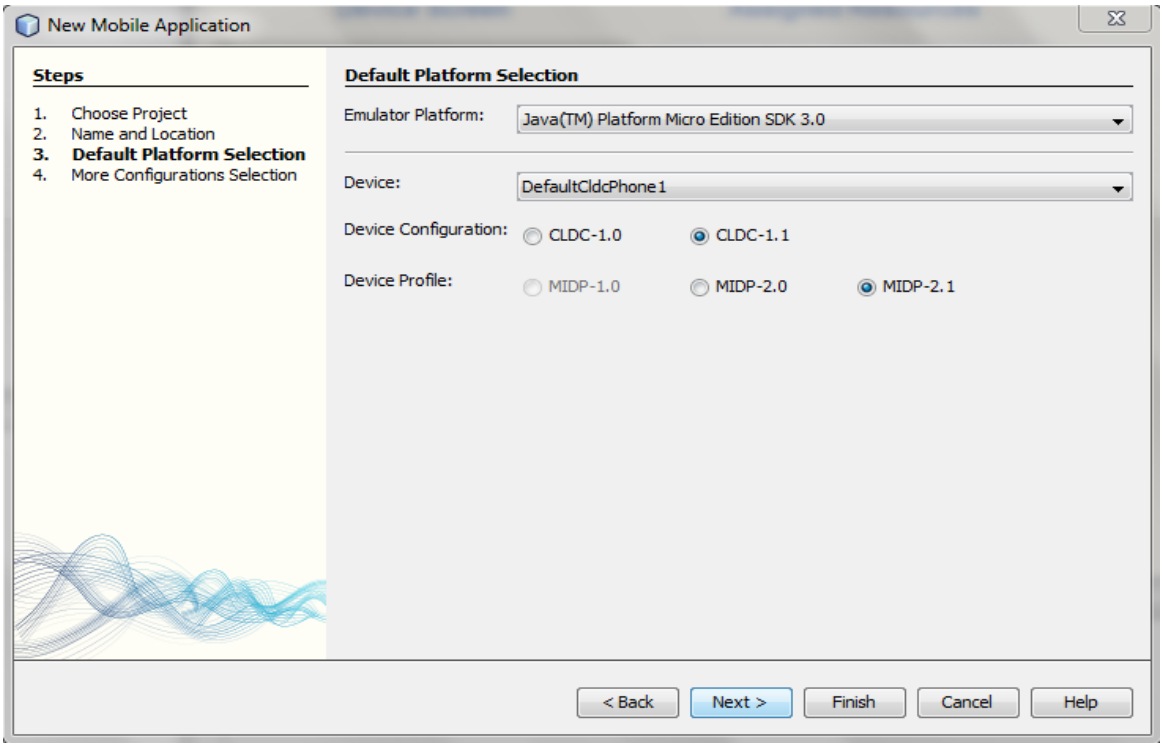


Figura. 2.29. Creación de aplicaciones MIDP/CLDC

Una vez creada la aplicación se denotará que se tienen más opciones de visualización y creación para el MIDlet, a continuación se ejemplariza algunas herramientas que el software de desarrollo *NetBeans* proporciona para la manipulación y correcta creación de la aplicación.

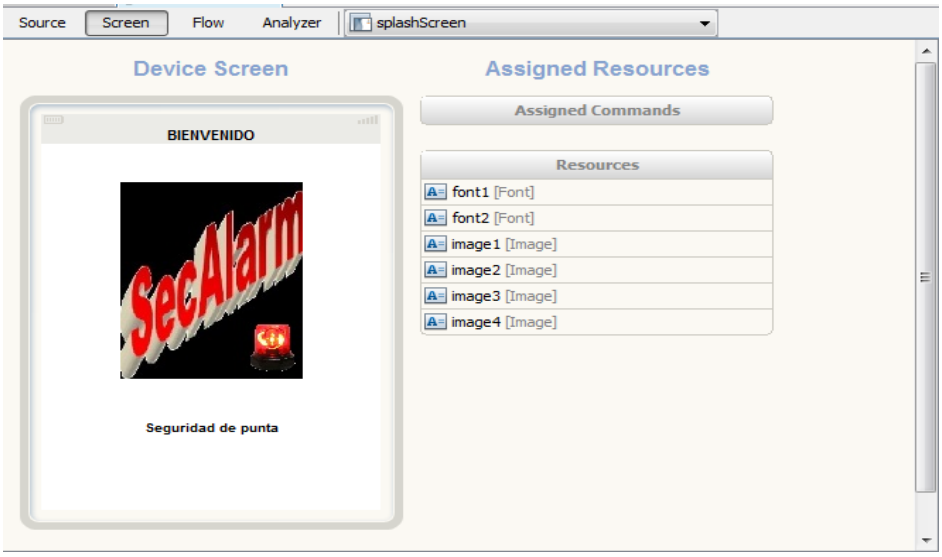


Figura. 2.30. Presentación de pantalla de aplicación MIDP/CLDC

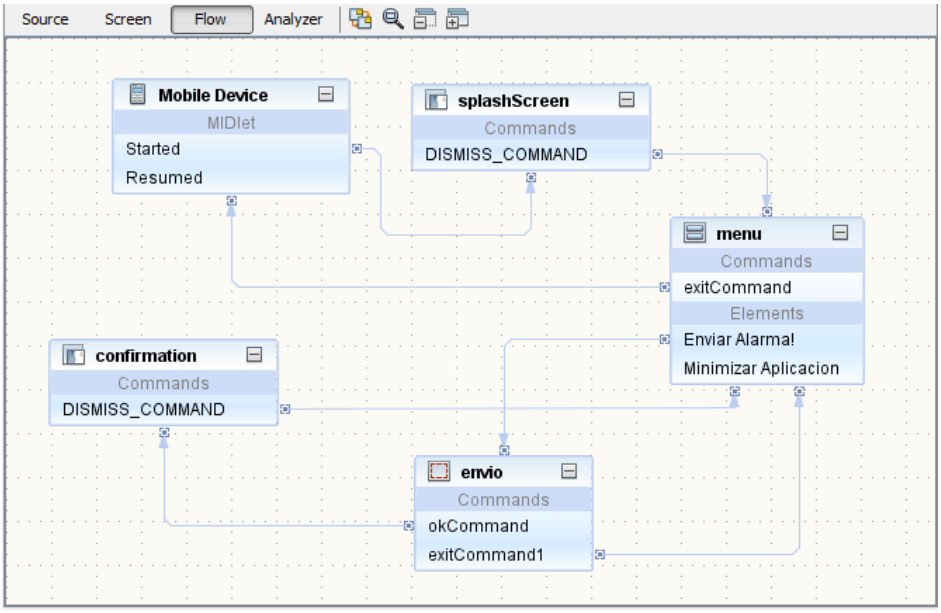


Figura. 2.31. Diagrama de flujo de aplicación MIDP/CLDC

2.5.3.Java Wireless Message API (WMA)

Es un API de java que está embebido en la aplicación el cual es una extensión de las especificaciones MIDP/CLDC para el envió, recepción y gestión de mensajes de texto SMS desde MIDlets. WMA trabaja bajo el paquete *javax.wireless.messaging* el cual contiene interfaces con métodos de comunicación los cuales se detallan en la siguiente tabla.

Tabla. 2.2. Interfaces y métodos de WMA

Interfaz	Descripción	Métodos
Message	Interfaz de comunicación base de la cual derivan *TextMessage* y *BinaryMessage*	getAddress() getTimestamp() setAddress()
BinaryMessage	Fija y detecta el *Payload* binario	getPayloadData() setPayloadData()
TextMessage	Fija y detecta el *Payload* de texto	getPayloadText() setPayloadText()
MessageConection	Interfaz por la cual se realiza el envío	newMessage()

	y recepción de mensajes	receive() send() setMessageListener() numberOfSegments()
MessageListener	Implementa la notificación asíncrona de objetos	notifyIncomingMessage()

Con los métodos especificados en la tabla 2.2 se consigue que la implementación de comunicación para mensajes cortos de texto SMS se realice de manera satisfactoria. En el capítulo 3 se analiza con más detalle la implementación de dichos métodos de WMA.

CAPITULO III

DISEÑO E IMPLEMENTACION DEL PROTOTIPO

3.1. HARDWARE

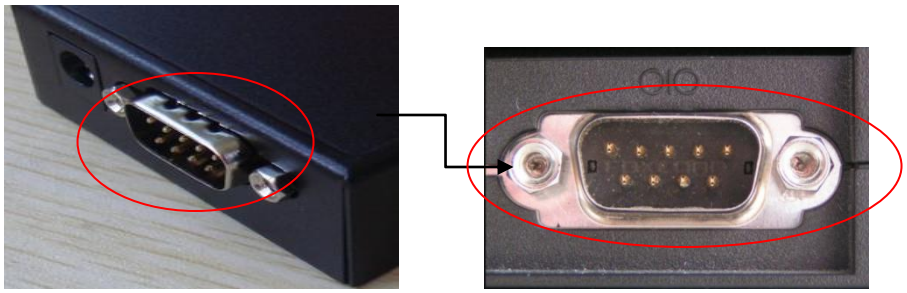
En esta sección se describirá el hardware del prototipo, el cual se compone de lo siguiente:

- Modem GSM (Receptor)
- Cable de transmisión USB-SERIAL GUC232A
- Configuración de Modem GSM (Receptor)

3.1.1.Modem GSM receptor

Con el fin de cumplir con el objetivo del proyecto se opta como iniciativa reutilizar equipos terminales que tengan bajo costo y sean aptos para el uso de la gama de comandos AT que se requieren, es por eso que se ha utilizado para la implementación del mismo un modem GSM/GPRS modelo F1103 de marca Four-Faith que se presenta en la Figura 3.1a.

a) Terminal Receptor



b) Puerto de comunicación Serial RS232

Figura. 3.1. Modem GSM/GPRS F1103

El modem GSM/GPRS y la tarjeta SIM tienen las siguientes características técnicas:

Tabla. 3.1. Características del Modem GSM/GPRS F1103

Modem GSM/GPRS F1103	
Especificaciones Inalámbricas	
Bandas de Operación	QuadBand EGSM 850/900/1800/1900Mhz
Soporte redes	GSM / EDGE
Fase GSM	ETSI GSM Phase 2+
Soporte	DTMF, SMS, CSD y funciones de datos
Interface estándar	Estándar de comandos AT&AT+
Interface	
Puerto	Serial RS232
Velocidad	110~230400bits/s
Luces indicadoras	“Power”, “ACT”, “Online”
Interfaz de antena	Estándar SMA hembra, 50ohm
Interfaz SIM	SIM/UIM interfaz estándar universal
Interfaz de Encendido	Conector estándar de 3 Pines
Alimentación Eléctrica	
Voltaje AC	110~120 V – AC
Voltaje DC	12V/0.5A – DC
Rango de Voltaje DC Permitido	5~35V
Otros	
Temperatura de operación	-25 ~ +65°C

Rango de Temperatura máxima	-40 ~ + 85°C
Humedad permitida	95%
Dimensiones	91x58.5x22 mm
Peso	630g
Tarjeta SIM	
Memoria	250MB
Memoria dinámica	4MB
Memoria SMS	4MB
Mensajes cortos	Enviar / Recibir / Difusión
Mensajes cortos concatenados	Si

La Figura 3.1b muestra el puerto de comunicación el cual es el medio directo para acceder al modem GSM/GPRS del cual cabe recalcar que solo basta conocer los pines 2 (Rx), 3 (Tx) y 5 (GND) para poder comunicarse ya sea con la PC u otro dispositivo, la Tabla 3.2 muestra detalladamente la descripción de cada uno de los pines del modem.

Tabla. 3.2. Descripción de pines modem GSM/GPRS F1103

Número de PIN	Nombre de PIN	Descripción
1	DCD (Data Carrier Detect)	Señal de control para comunicar al equipo terminal conectado que se ha hecho la conexión con una línea telefónica.
2	RXD (Received Data)	Recibe los datos desde el terminal
3	TXD (Transmitted Data)	Transmite los datos hacia el terminal
4	DTR (Data Terminal Ready)	Señal de control para comunicar al equipo terminal que la conexión está lista.
5	GND (Common Ground)	Tierra para Rx y Tx de datos
6	DSR (Data Set Ready)	Señal de control que comunica al equipo terminal que está listo para recibir datos o comandos
7	RTS (Request To Send)	Señal de control que comunica al equipo

		terminal que va a enviar datos
8	CTS (Clear To Send)	Señal de control que reconoce a RTS y permite que se transmita datos.
9	RI (Ring Indicator)	Señal de control que comunica al terminal que existe una señal de timbrado en la línea telefónica

3.1.2.Cable de transmisión USB-SERIAL GUC232A

Para la conexión del modem GSM/GPRS con computadores o dispositivos nuevos es recomendable usar un cable USB-SERIAL de modelo GUC232A o simplemente si se tiene la interfaz serial hacer la conexión directa sin utilizar el cable mencionado anteriormente. El cable USB-SERIAL GUC232A cumple con la descripción de pines dada anteriormente en la Tabla 3.2 para poder acceder al Modem GSM/GPRS siendo esta la vía para el envió y recepción de datos.

Podemos encontrar en el mercado diferentes series de cables, sin que necesariamente sean del modelo mencionado, a continuación se muestra en la Figura 3.2 la apariencia del cable USB-SERIAL GUC232A

Figura. 3.2. Cable de datos USB-SERIAL GUC232A

3.1.3.Configuración de Modem GSM (Receptor)

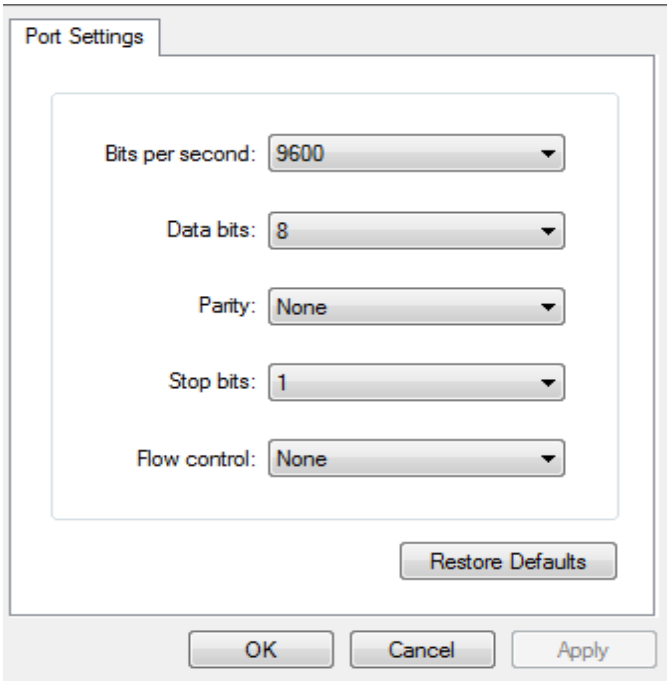
El Modem GSM debe estar configurado por los siguientes parámetros detallados en la Tabla 3.3 (ver Figura 3.3b)

Tabla. 3.3. Parámetros de configuración del Modem GSM

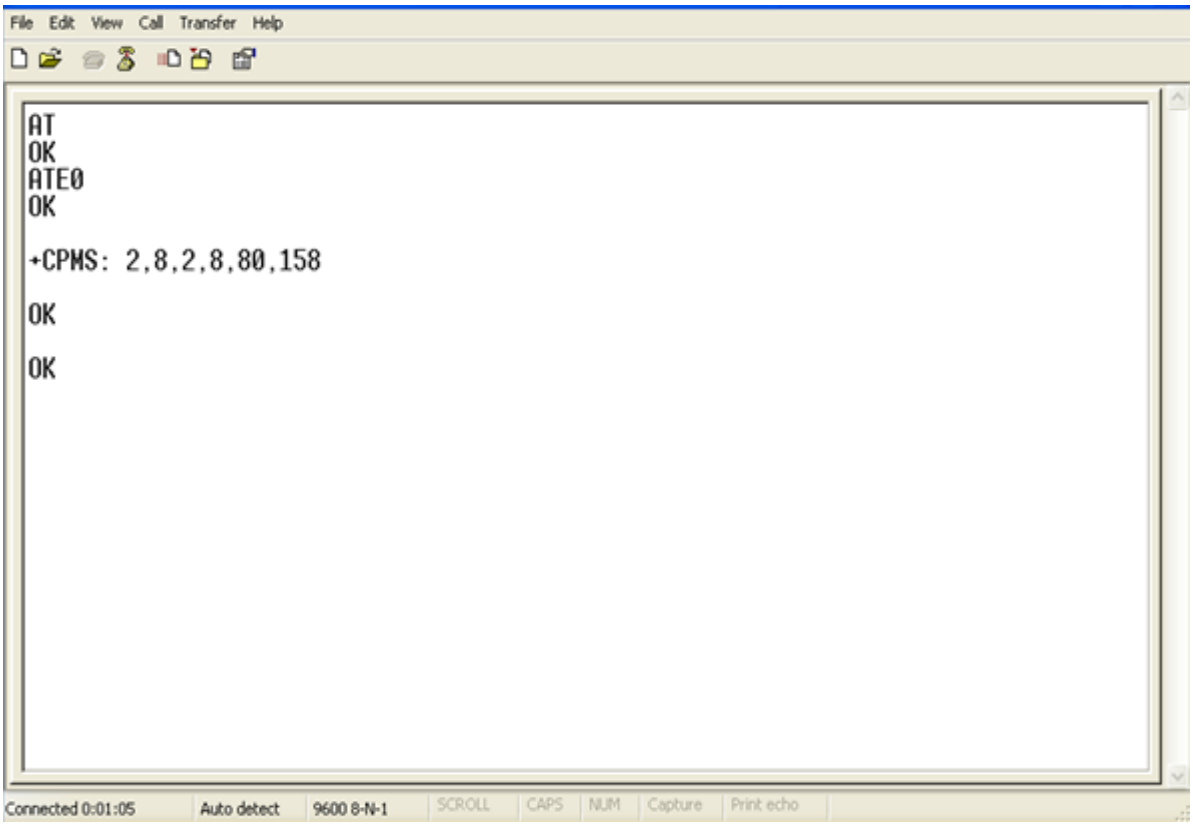
Comando AT & AT+	**Acción**
AT	Prueba de conexión.
ATE0	Texto de recepción sin eco
AT+CPMS="SM","SM","MT"	Selección de memoria de SMS.
AT+CMGF=1	Modem en Modo Texto.

Además es necesario que la velocidad de transmisión de datos tenga los siguientes parámetros y sean configurados en la transmisión serial (ver Figura 3.3a)

- Bits por segundo: 9600
- Bits de datos: 8
- Paridad: Ninguna
- Bits de parada: 1
- Control de flujo: Ninguno



a) Velocidad de Transmisión serial



b) Configuración de parámetros para Modem GSM

Figura. 3.3. Configuración de Modem GSM/GPRS F1103

3.2. SOFTWARE

El desarrollo del software constituye una de las partes esenciales del proyecto, puesto que de la programación dependen tanto el SERVlet gobernada por el Modem GSM/GPRS receptor, la aplicación MIDlet para la interactividad del usuario con el SERVlet, así como también la comunicación entre los 2 elementos. En esta sección se describirá el desarrollo del SERVlet así como también de la MIDlet las cuales están descritas de la siguiente forma:

- Diseño de aplicación SERVlet usando JAVA

- Diseño de aplicación MIDlet usando J2ME

3.2.1.Diseño de aplicación SERVlet usando JAVA

Si bien el sistema recolector de datos puede almacenarlos temporalmente en la memoria RAM de la SIM del modem GSM, estos deben ser recibidos en forma de un mensaje de texto hacia una PC para su procesamiento.

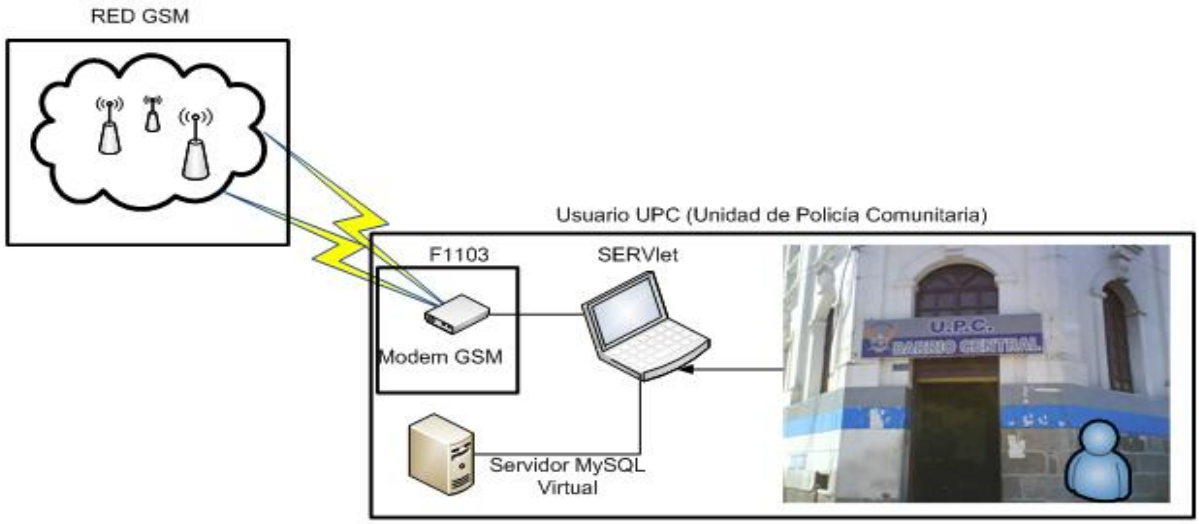


Figura. 3.4. Diagrama de bloques - Recepción de mensajes de texto en UPC

Con este propósito y para la notificación de alarmas comunitarias se decidió desarrollar una aplicación que permita bajar y almacenar los datos de cada mensaje de texto recibido en el modem GSM/GPRS para así poderlos comparar con los datos de los usuarios que han sido previamente ingresados y que el SERVlet los administrará a través del servidor MySQL. El mensaje de texto se almacena en el SERVlet de manera que pueda ser leído por el usuario final, en este caso el jefe de turno de la UPC (Unidad de Policía Comunitaria), por lo cual se implementó una aplicación totalmente interactiva con el usuario para la administración de mensajes del modem GSM/GPRS usando lenguaje JAVA, en la Figura 3.4 se muestra el diagrama de bloques para la recepción de mensajes de texto.

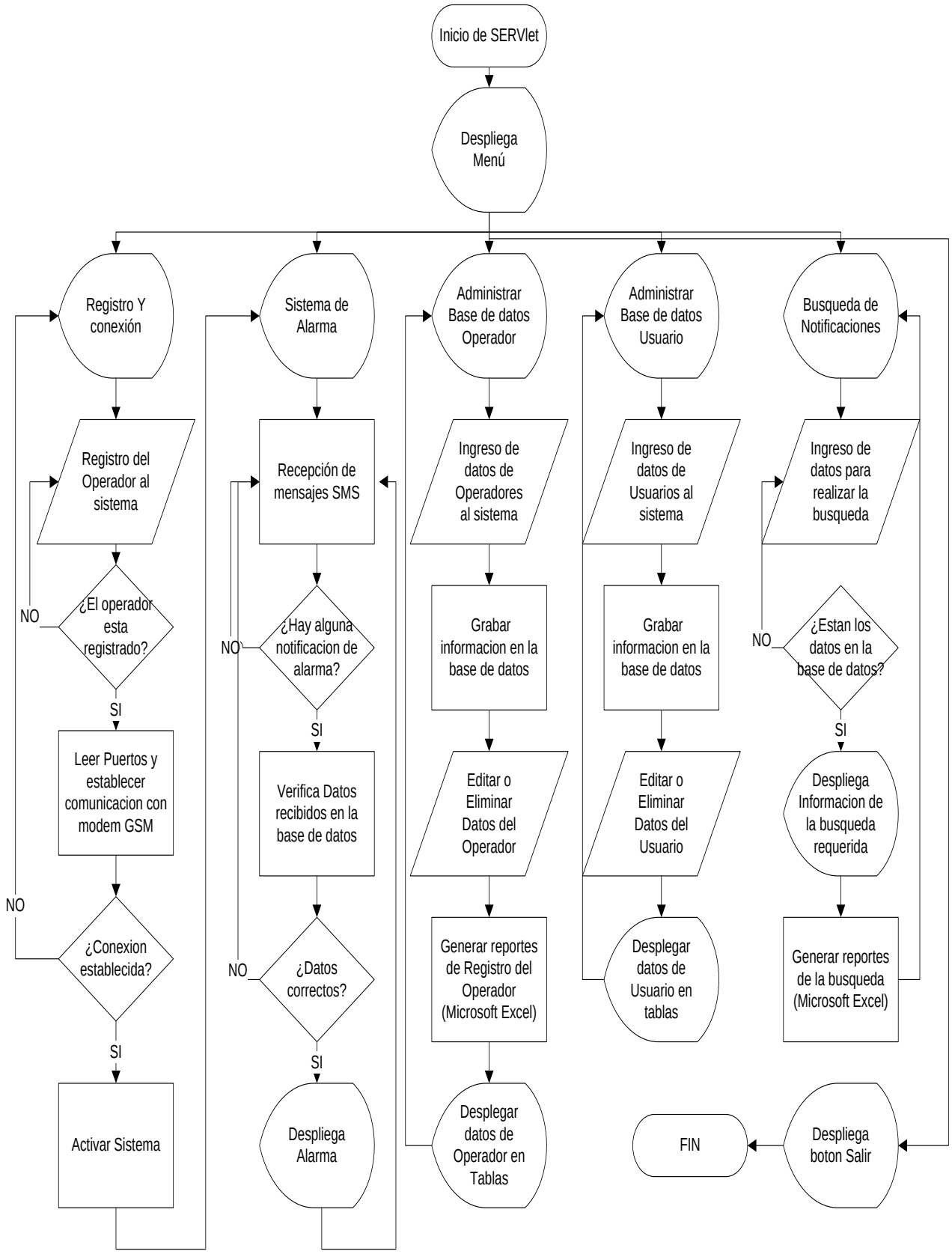


Figura. 3.5. Diagrama de flujo - SERVlet

Para el diseño del SERVlet es necesario también agregar algunas opciones para la operación del mismo, con el fin de obtener un sistema seguro y confiable para una buena administración sobre la base de datos de los usuarios terminales.

Es por eso que se opto incrementar dentro del diseño las siguientes opciones:

Tabla. 3.4. Menú de opciones del SERVlet

Opciones	Detalle
Registro y Comunicación	Es la opción encargada de que el operador del SERVlet se registre para saber quien está usando la aplicación, además tendrá la potestad de establecer la comunicación con el modem GSM y activar el sistema.
Sistema de Alarma	Se mostrarán las notificaciones de alarma enviadas por los usuarios para que el operador pueda tomar acción sobre ellas, las acciones sobre las notificaciones son: Falsa alarma, En progreso y Atendido.
Administrar base de datos del Operador	Establece el registro del operador y su gestión ya sea para editar, cambiar clave o eliminar. También instaura la interfaz para que el número de celular del jefe de turno pueda ser cambiado para que así exista el desvío de alarma cuando no exista presencia del mismo. Además muestra las tablas de registro del operador así como también los datos de los operadores registrados para poderlas exportar hacia un archivo de Microsoft Excel.
Administrar base de datos del Usuario	Establece el registro de los usuarios y su gestión ya sea para editar o eliminar. Muestra

	también tablas de datos de los usuarios registrados.
Búsqueda de notificaciones	Será la encargada de buscar los datos de las notificaciones para realizar informes, también se mostrará la información mediante tablas y comprende la exportación de datos a Microsoft Excel.

El diagrama de flujo del SERVlet está descrito en la Figura 3.5 junto con las opciones del diseño descritas en la Tabla 3.4, con todos los procesos y decisiones que toma para cumplir los detalles de cada una de las opciones.

A continuación se puntualizará la implementación de los procesos más significativos de la aplicación con códigos en lenguaje JAVA usando Netbeans IDE cuya patente está registrada en *Oracle Corporation*©.

3.2.1.1. Implementación de conexión serial

Para la implementación de la conexión es importante conocer dos fundamentos que facilitan y ayudan para que el enlace con el modem GSM no sea un inconveniente.

a) Se utiliza la librería de comunicación serial *RXTXcomm* la cual se debe añadir a los repositorios del proyecto para que esté embebida al mismo. Una vez añadida la librería debe ser importada al proyecto como *gnu.io.** para así hacer uso de la misma y poder continuar con la declaración de variables para el uso de puertos.

b) Se realiza el reconocimiento de puertos del PC que además comprende la utilización de los mismos y la notificación si ya están siendo usados.

Se esquematiza los diagramas de flujo que sirven específicamente para la lectura de los puertos (ver Figura 3.6) y la selección del puerto para que este sea usado (ver Figura 3.7).

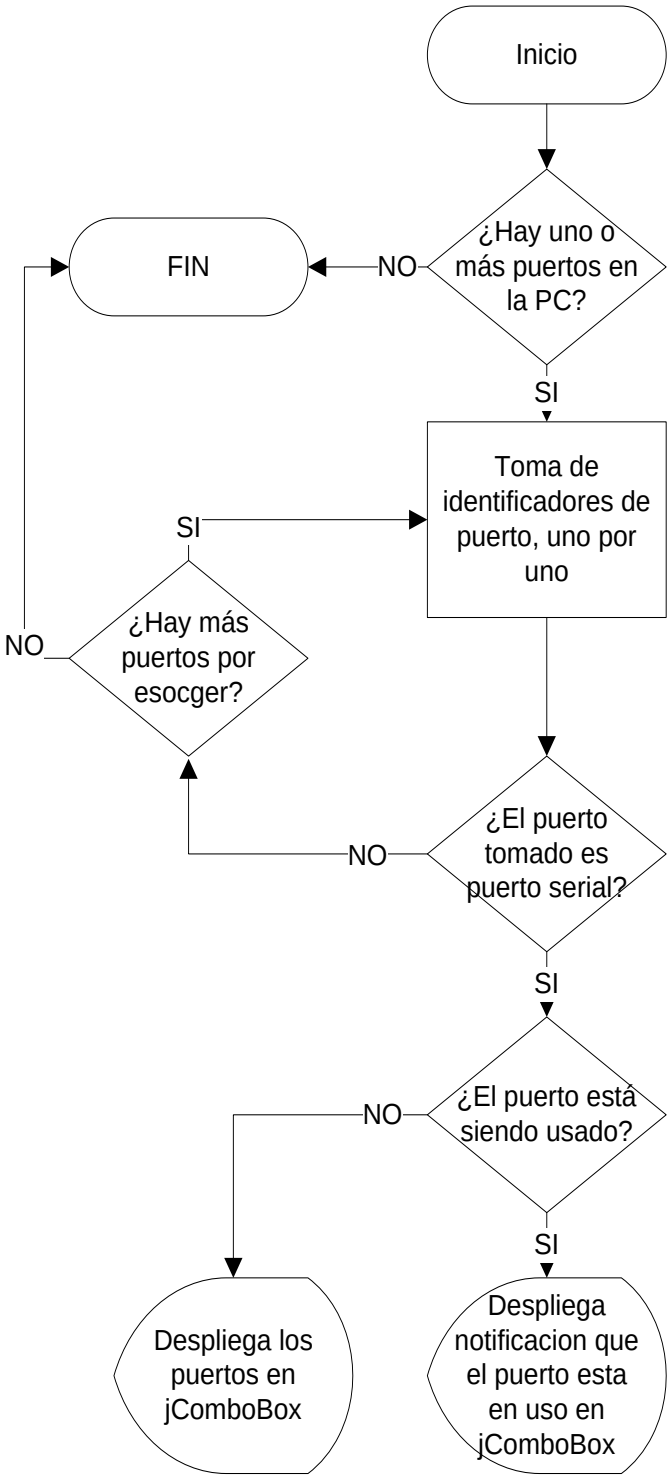


Figura. 3.6. Diagrama de flujo – Lectura de puertos en la PC

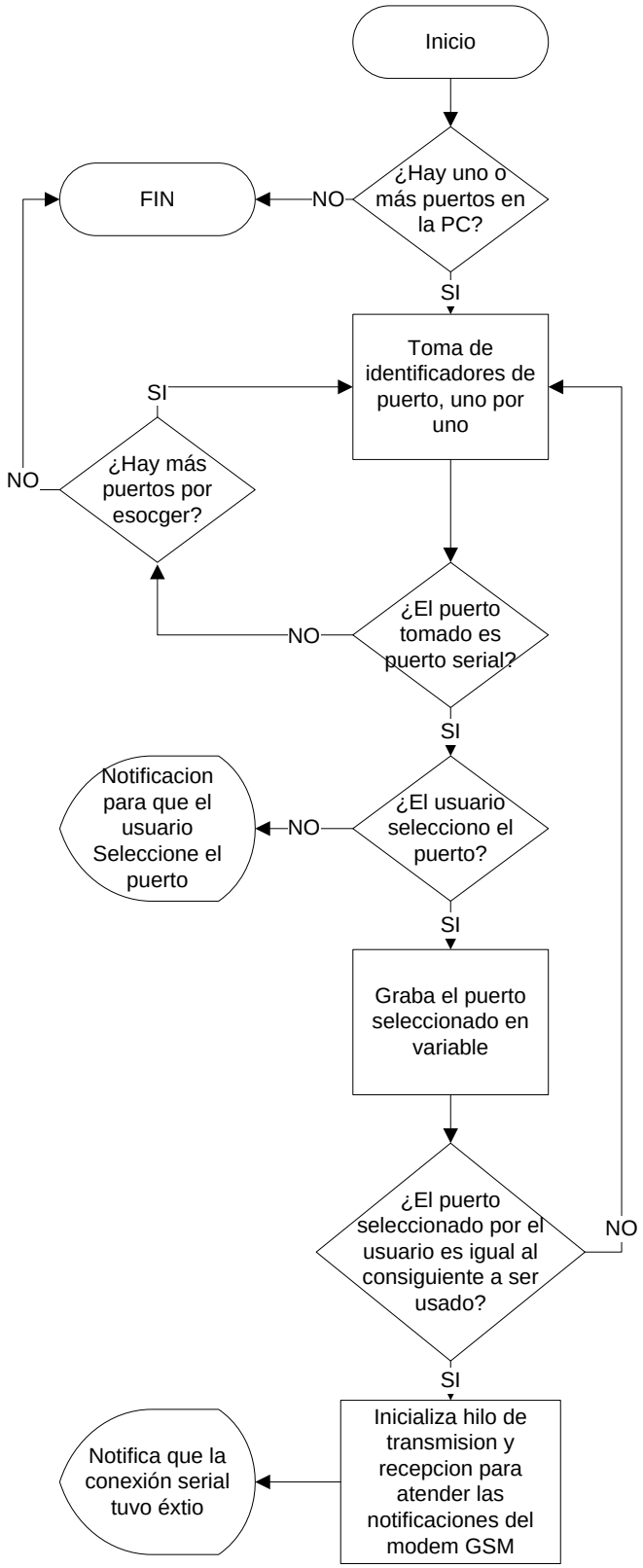


Figura. 3.7. Diagrama de flujo – Selección de puerto a ser usado e inicialización de transmisión y recepción con el modem GSM/GPRS

La implementación del código de programa de la conexión serial se la realiza utilizando identificadores además de que las notificaciones de los puertos para su selección y uso son desplegables según el PC o dispositivo en donde se instale la aplicación.

A continuación se muestra el código de programa el cual especifica las pautas para implementar la conexión serial con el modem GSM.

Código de programa JAVA – Importar librerías RXTXcomm e Inicialización de variables para el puerto serial

```
import gnu.io.*
static CommPortIdentifier C;
static Enumeration E;
static SerialPort S;
static OutputStream O;
static InputStream I;
```

Código de programa JAVA - Lectura de puertos serial

```
E=C.getPortIdentifiers();
while(E.hasMoreElements())
{
   C=(CommPortIdentifier)E.nextElement();
   if(C.getPortType()== C.PORT_SERIAL)
   {
     boolean s = C.isCurrentlyOwned();
     if(s==false)
     {
      jComboBox1.insertItemAt(C.getName(), i);
     }
     else
     {
      jComboBox1.addItem(C.getName()+"El puerto esta ocupado
      por" + C.getCurrentOwner());
     }
   }
   i++;
}
```

Código de programa JAVA - Selección de puerto a ser usado e inicialización de transmisión y recepción con el modem GSM/GPRS

```
E=C.getPortIdentifiers();
```

```
while(E.hasMoreElements())
{
   C=(CommPortIdentifier)E.nextElement();
   if (C.getPortType()==C.PORT_SERIAL)
   {
      if(C.getName().compareTo(puerto)==0)
      {
         Recibir R=new Recibir();
      }
   }
}
```

Además se detalla la función *Recibir* la cual permite la configuración en cuanto a velocidad de transmisión y parámetros de puerto, así como también la inicialización del hilo que se requiere para conocer las notificaciones de recepción y transmisión, como se mencionó y se mostró anteriormente en la Figura 3.3a.

Código de programa JAVA – Configuración de velocidad de transmisión en el puerto, así como también sus parámetros

```
public Recibir()
{
  try{
  S = (SerialPort) C.open("SimpleReadApp", 2000);
  } catch (PortInUseException e) {}
  try {
   O = S.getOutputStream();
   I = S.getInputStream();
  } catch (IOException e) {}
  try {
   S.addEventListener(this);
  } catch (TooManyListenersException e) {}
  S.notifyOnDataAvailable(true);
  try {
   S.setSerialPortParams(9600,
     SerialPort.DATABITS_8,
     SerialPort.STOPBITS_1,
     SerialPort.PARITY_NONE);
   S.setFlowControlMode(S.FLOWCONTROL_NONE);
  } catch (UnsupportedCommOperationException e) {}
  readThread = new Thread(this);
  readThread.start();
}
```

3.2.1.2. Implementación y análisis de comandos AT & AT+

En este apartado se podrá realizar el manejo y control del modem GSM/GPRS F1103, es importante recalcar que la teoría antes mencionada describe la utilización de dichos comandos y la interacción ya sea con una PC o cualquier otro dispositivo electrónico.

Intrínsecamente en la aplicación siempre debe realizarse la apertura de puertos del modem GSM/GPRS de manera automática cada vez que es reiniciada y en conjunto con la actualización debida para saber si hay o no notificaciones de los usuarios. Dentro del capítulo 3.1.3 se puntualiza la configuración del modem GSM teniendo como parámetros de envió las tramas de configuración de puertos como lo describe el diagrama de flujo ilustrado en la Figura 3.8, y el envío de comandos AT & AT+ que facilitan la comunicación y enlace con el modem GSM ilustrado en la Figura 3.9 obteniendo la automatización de configuración del modem y la activación del sistema respectivamente.

Es elemental conocer que la aplicación está implementada para que cuando no exista ningún tipo de respuesta del administrador, dentro de un tiempo establecido (dos minutos) se realice el desvío de alarma, el tiempo que usa el SERVlet para calcular los dos minutos que necesita para saber si existió o no una acción del administrador lo toma usando el reloj del sistema operativo como se muestra en el diagrama de flujo de la Figura 3.10 de esa manera se confirma que todos los datos lleguen al dispositivo móvil del jefe de turno como se puntualiza en el diagrama de flujo ilustrado en la Figura 3.11 haciendo que el modem realice la transmisión y recepción de datos por la red celular del operador contratado.

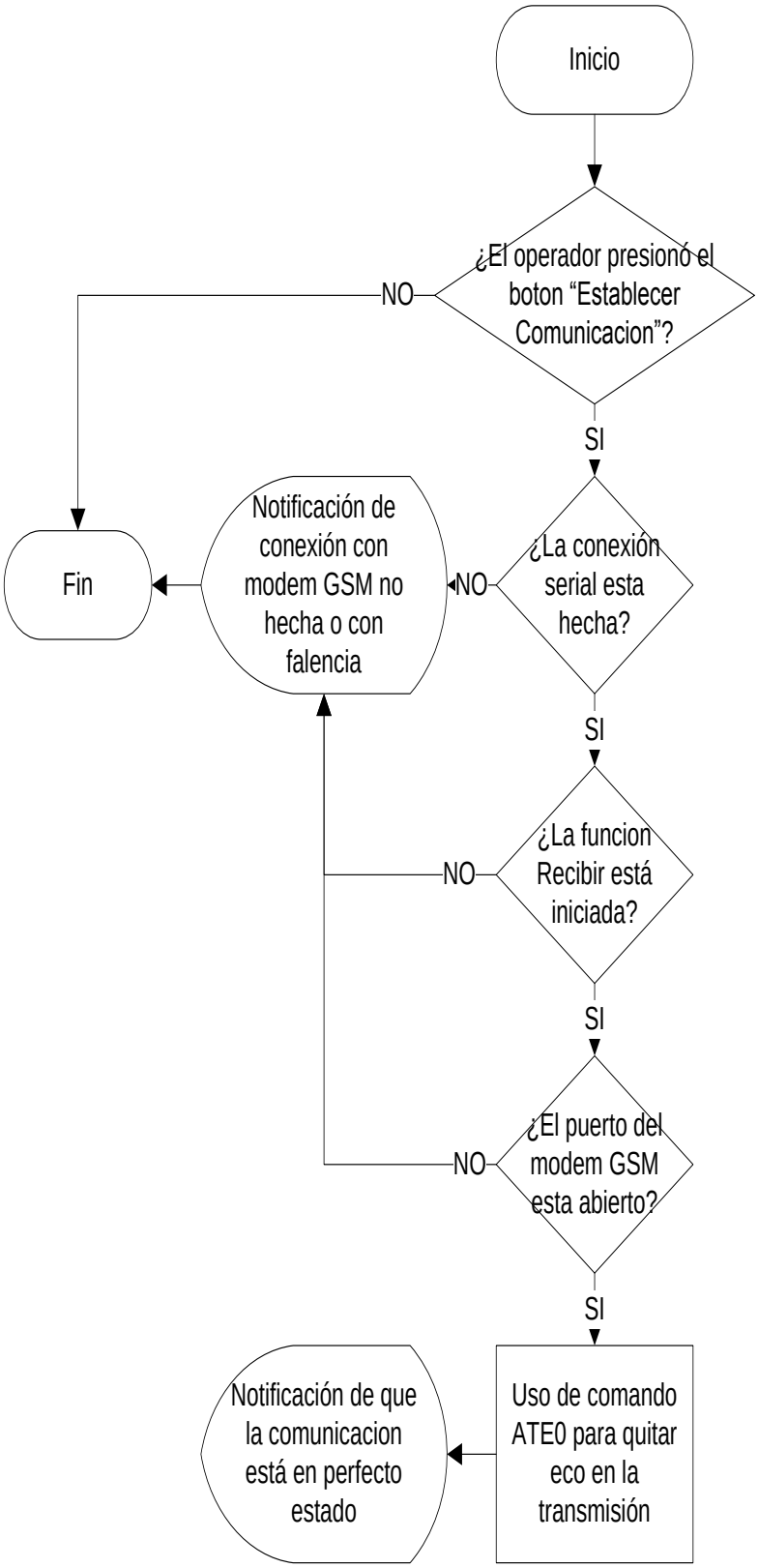


Figura. 3.8. Diagrama de flujo – Inicialización de función "Recibir", envío de tramas y apertura de puerto serial para modem GSM

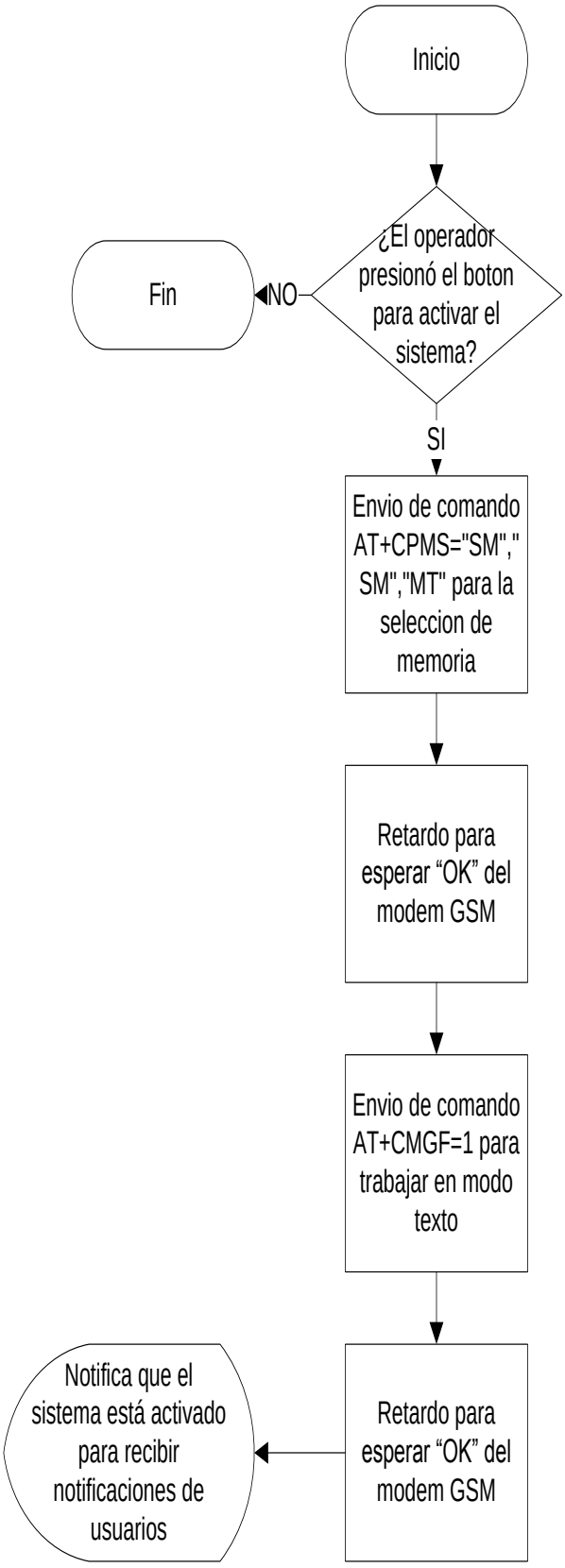


Figura. 3.9. Diagrama de flujo – Activación de sistema para recibir notificaciones de usuarios

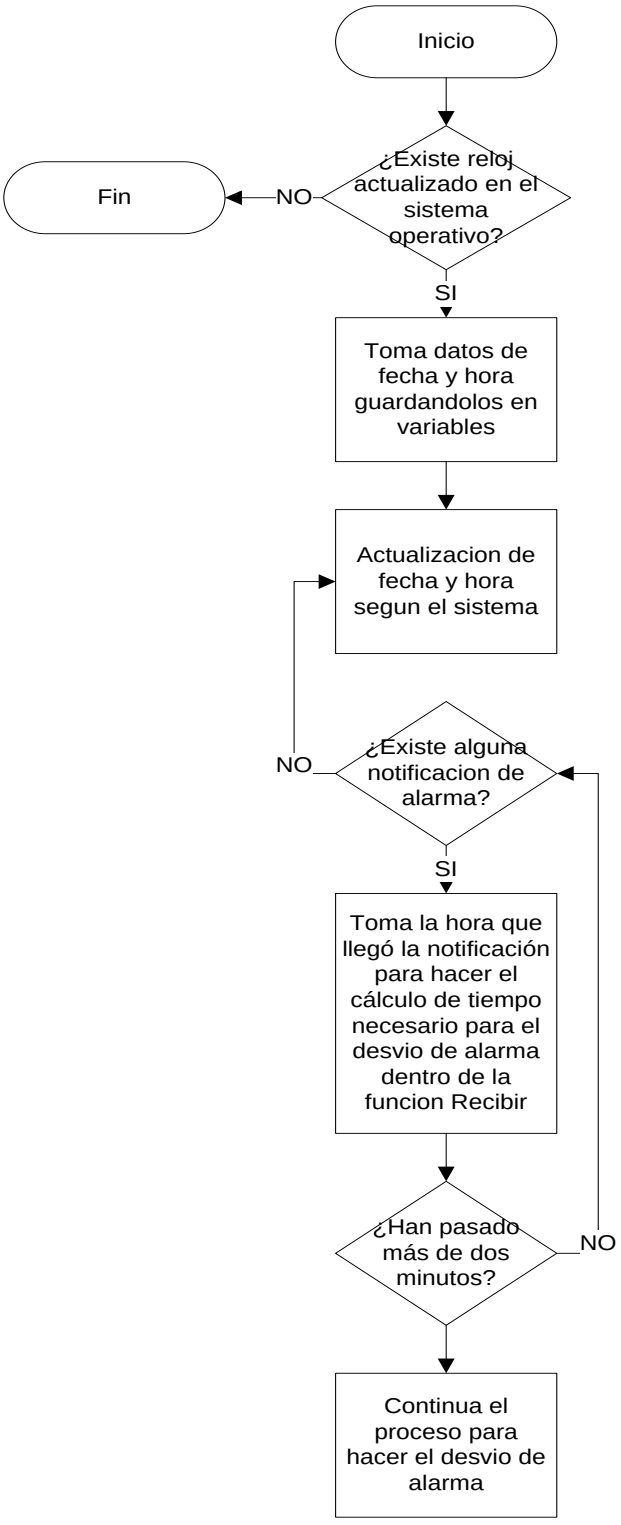


Figura. 3.10. Diagrama de flujo – Tiempo del sistema operativo y cálculo de dos minutos para el desvío de alarma

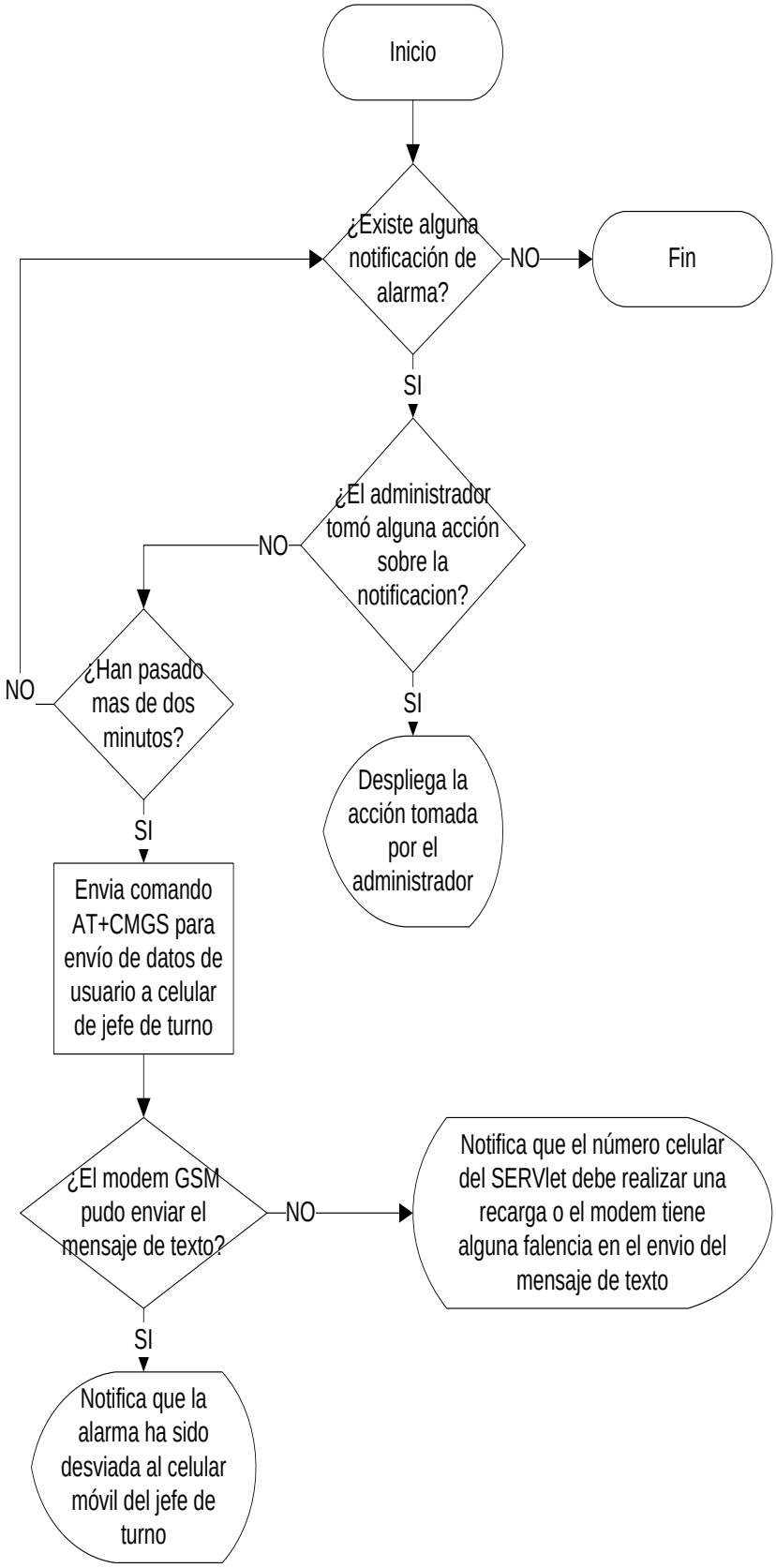


Figura. 3.11. Diagrama de flujo – Desvío de alarma al celular móvil de jefe de turno

Una vez que la función *Recibir* fue inicializada es importante recalcar que cumple también la función de atender al modem GSM con los eventos del puerto serial cada vez que éste tenga datos en su buffer, es por eso que se le ha dado un espacio de 1KB (1024B) para que atienda la recepción de datos, la función es encargada también de reconocer todos los comandos de recepción que el modem GSM entrega para conocer qué operación tomar sobre ellos.

A continuación se detalla en el diagrama de flujo que ilustra la Figura 3.12 de como la función *Recibir* trabaja para operar la recepción de datos una vez que la configuración del modem y el sistema de alarmas estén inicializados.

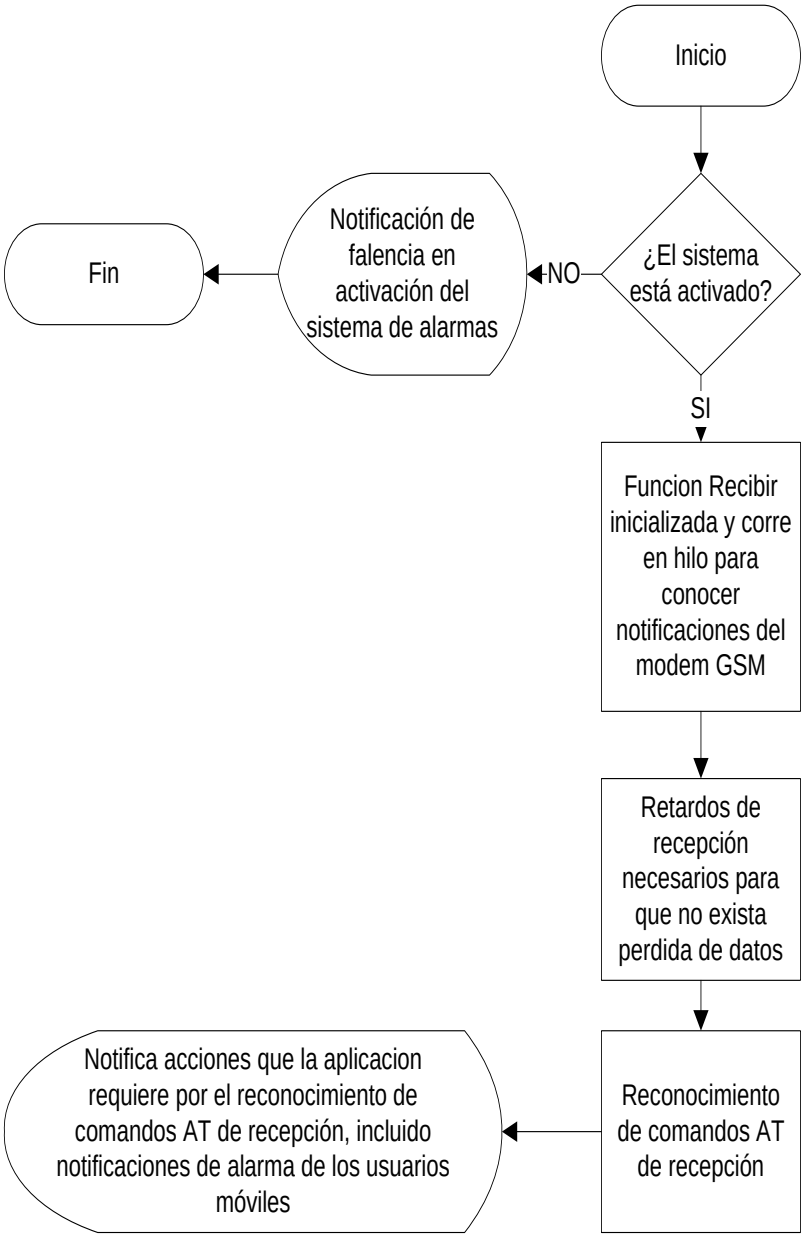


Figura. 3.12. Diagrama de flujo – Recepción de datos (Función Recibir)

Consecutivamente se muestra el código de programa el cual especifica las pautas para implementar los comandos AT & AT+ en el SERVlet, es de suma importancia resaltar que la declaración de variables para manejar flujos ya no deben ser tomadas en cuenta dado que se las usa en la implementación de conexión serial.

Código de programa JAVA – Inicialización y declaración de comandos AT

```
String message0 = "ATE0\r";
String message1 = "AT+CPMS=";
String message2 = "AT+CMGL=";
String r = "REC UNREAD";
String message3 = "AT+CMGF=1\r";
String message4 = "AT+QMGDA=";
String d = "DEL READ";
String message5 = "AT+CMGS=";
```

Código JAVA - Inicialización de función “Recibir”, envío de tramas y apertura de puerto serial para modem GSM

```
E=C.getPortIdentifiers();
while(E.hasMoreElements())
{  C=(CommPortIdentifier)E.nextElement();
   if (C.getPortType()==C.PORT_SERIAL)
   {  if(C.getName().compareTo(puerto)==0)
      { Recibir R=new Recibir();}}
   }
try
{ O.write(message0.getBytes());
} catch(IOException e){}
```

Código de programa JAVA - Activación de sistema para recibir notificaciones de usuarios

```
try
{
   //AT+CPMS="SM","SM","MT"
   O.write(message1.getBytes());
   O.write(0x22);O.write(0x53);O.write(0x4D);O.write(0x22);
   O.write(0x2C);
   O.write(0x22);O.write(0x53);O.write(0x4D);O.write(0x22);
   O.write(0x2C);
   O.write(0x22);O.write(0x4D);O.write(0x54);O.write(0x22);
   O.write(0x0D);
} catch(IOException e){}
delayt();
try
{
   //AT+CMGF=1
   O.write(message3.getBytes());
} catch(IOException e){}
delayt();
delayt();
```

Código de programa JAVA - Recepción de datos (Función Recibir)

```
public class Recibir implements Runnable,
SerialPortEventListener
{
    public Recibir() {//Citada anteriormente en 3.2.1.1.}
    public void run()
    {
     //Compara la hora de llegada de la notificacion con los dos
     //minutos para el desvio de alarma
      minutollegada2 = jTextField6.getText();
     int ml1=Integer.valueOf(minutollegada1);
     int ml2=Integer.valueOf(minutollegada2);
     if (ml1==58){ml1=-2;}
     if (ml1==59){ml1=-1;}
     //Verifica si el administrador tomo accion sobre la alarma
     if(jComboBox2.getSelectedItem().equals("Ninguno")==true && ml1
     + 2 == ml2)
     {
           if (Integer.valueOf(jTextField7.getText())>45 &&
           Integer.valueOf(jTextField7.getText())<59)
          {
          try
          {
           //Compara datos de usuario en la base de datos
          catch (SQLException ex) {}
          try
          {
            //Toma el numero celular del jefe de turno para el
            //desvio de alarma
            String ce = jTextField19.getText();
            char[] H = new char[8];
            ce.getChars(3, 11, H, 0);
            ce = "0" + String.valueOf(H);
            String mensajesms = "ALARMA en la central por
            favor¡ATENDERLA!";
            delayt();
            delayt();
            //AT+CMGS=”Numero celular”, Mensaje.
            O.write(message5.getBytes());
            O.write(0x22); O.write(ce.getBytes());O.write(0x22);
            O.write(0x0D);
            delayt();
            delayt();
            O.write(mensajesms.getBytes());
            O.write(0x1A);
          }
          catch(IOException e){}
     }

           //El SERVlet observa si hay algún evento del puerto serial
           public void serialEvent(SerialPortEvent event) {
```

```
            switch(event.getEventType()) {
            case SerialPortEvent.BI:
            case SerialPortEvent.OE:
            case SerialPortEvent.FE:
            case SerialPortEvent.PE:
            case SerialPortEvent.CD:
            case SerialPortEvent.CTS:
            case SerialPortEvent.DSR:
            case SerialPortEvent.RI:
            case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
                break;
           case SerialPortEvent.DATA_AVAILABLE:
            byte[] readBuffer = new byte[1024];
            try
            {
                while (I.available() > 0)
                {
                   this.run();
                   int numBytes = I.read(readBuffer);
                   String w=new String(readBuffer);
                 //Envío de comando AT para lectura de mensajes SMS
                       //AT+CMGL="REC UNREAD";
                 //Reconocimiento de recepción AT
                       //K, OK, +CPMS, +CMGL, +CMS ERROR
                 //Envió de comandos AT+ para eliminar mensajes SMS
                 //de la bandeja de entrada
                       //AT+QMGDA="DEL READ";
                 }
         }catch (IOException e) {}
        }
      }
}
```

Dentro del último código de programa descrito no se especifica rotundamente la lectura de mensajes dado que implica otros pasos que incumben bases de datos y la comparación de los mismos para validar notificaciones; lo cual se detalla claramente en capítulos siguientes y en la totalidad del proyecto, pero si define el uso de comandos AT & AT+ que la PC recibe de los flujos de datos y eventos del puerto serial quedando de esta manera implementada sobre el SERVlet.

3.2.1.3. Diseño e implementación de base de datos con servidor MySQL

En este apartado se hará énfasis al manejo exclusivo de las tablas con bases de datos usadas en el SERVlet para la facilidad y manejo del administrador que opera el sistema y requiere de todos los datos posibles de los usuarios para que sean atendidos de una buena manera y eficazmente ante una notificación de alarma.

El diseño se hizo en base a los fundamentos y referencias que se necesitan para los datos de los usuarios y su debida manipulación, además de otro tipo de información que el SERVlet requiere para así gestionar todos los procesos que incumben bases de datos ya antes mencionados, cabe recalcar que las principales funciones que el diseño de base de datos implica necesariamente que la definición de los datos no sea redundante, la manipulación de los datos, garantice la seguridad e integridad de los datos y la gestión de las transacciones y el acceso concurrente.

En el punto que es importante realizar un análisis es justamente en el garantizar la seguridad e integridad de los datos dado que la aplicación está dirigida a administradores para la gestión de los datos, a continuación se detalla los parámetros que se tomaron en cuenta para el diseño de este punto valioso

- **Un control sobre los usuarios que acceden a la base de datos y los tipos de instrucciones que están autorizados a efectuar**, a este control se le ha llamado *gestión de autorizaciones,* el cual permite crear o borrar usuarios y conceder o retirar derechos a efectuar determinados tipos de operaciones.

- **La validación de las instrucciones efectuadas con los datos,** este control se lo forja mediante un conjunto de reglas llamadas *restricciones de integridad***,** en donde necesariamente se impone que las modificaciones realizadas sobre algunos datos, obliguen a realizar modificaciones de otros datos con los que están enlazados.

- **Protección de los datos contra los accesos malintencionados y los fallos**. Los accesos malintencionados se suelen impedir con la asignación de claves a los usuarios, la definición de vistas y protección física de los datos, es decir obtener encriptado de los datos tomando como referencia las funciones del servidor MySQL y usando específicamente dentro del diseño del SERVlet la encriptación MD5.

La correcta utilización de todas estas operaciones de seguridad e integridad constituye una tarea esencial del Administrador de la base de datos.

Una vez recogidos todos los requerimientos, el siguiente paso es crear un **diseño conceptual** para la base de datos mediante un modelo de datos conceptual de alto nivel. El esquema conceptual contiene una descripción detallada de los requerimientos de información de los usuarios, y contiene descripciones de los tipos de datos, relaciones entre ellos y restricciones de la base de datos (ver Figura 3.13).

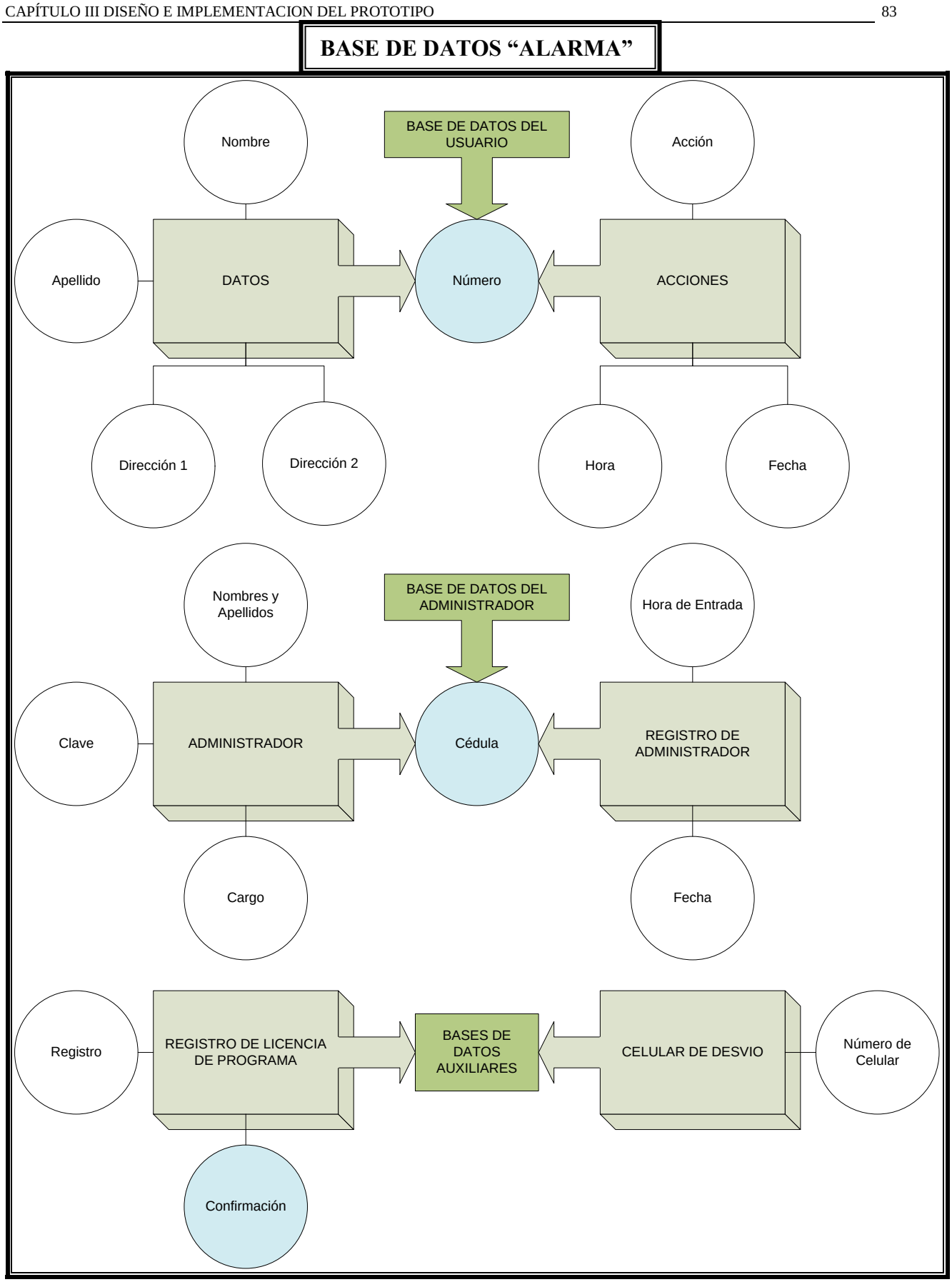


Figura. 3.13. Diseño Conceptual de Alto Nivel de la Base de Datos

A partir de la Figura 3.13 se puede obtener las tuplas para la implementación de la base de datos en el SERVlet cuyo nombre general es "ALARMA". Proporcionados todos los parámetros y opciones que requiere se optó emplazar todos los datos como variables de flujo en cadena de caracteres (VARCHAR), las cuales se detallan a continuación:

Tuplas de la Base de datos "ALARMA"

Tupla de Usuarios "DATOS"

Número	Nombre	Apellido	Dirección1	Dirección2
VARCHAR(11)	VARCHAR(30)	VARCHAR(30)	VARCHAR(60)	VARCHAR(60)
Primary Key	Not Null	Not Null	Not Null	Not Null

Tupla de Usuarios "ACCIONES"

Número	Acción	Fecha	Hora	Responsable
VARCHAR(11)	VARCHAR(13)	VARCHAR(10)	VARCHAR(15)	VARCHAR(60)
Primary Key	Not Null	Not Null	Not Null	Not Null

Tupla del Administrador "ADMINISTRADOR"

Cédula	Administrador	Clave	Cargo
VARCHAR(10)	VARCHAR(60)	VARCHAR(32)	VARCHAR(20)
Primary Key	Not Null	Not Null	Not Null

Tupla del Administrador "REGISTRO DE ADMINISTRADOR"

Cédula	Hora de Entrada	Fecha
VARCHAR(10)	VARCHAR(15)	VARCHAR(10)
Primary Key	Not Null	Not Null

Base de datos Auxiliar "REGISTRO DE LICENCIA DEL PROGRAMA"

Confirmación	Registro
VARCHAR(2)	VARCHAR(32)
Primary Key	Not Null

Base de datos Auxiliar "CELULAR DE DESVIO"

Celular de Desvío
VARCHAR(11)
Not Null

De la misma forma se detalla en la Figura 3.14 el diagrama de flujo que la aplicación realiza para la creación de tuplas de la base de datos "ALARMA"

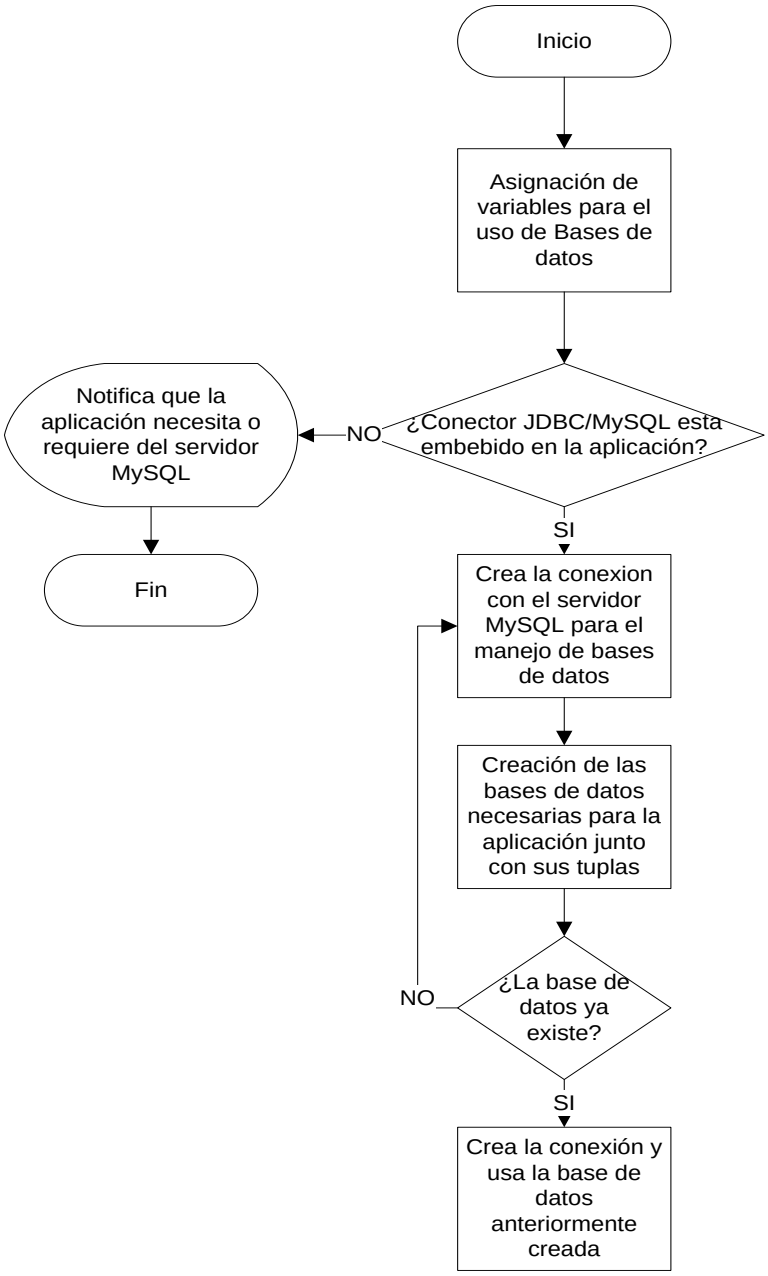


Figura. 3.14. Diagrama de flujo – Creación de base de datos "Alarma"

A continuación se detallará la implementación en el SERVlet de la base de datos en lenguaje de programación JAVA, es necesario que la librería así como también el conector JDBC/MySQL estén embebidos en la aplicación como se mencionó anteriormente en la teoría del proyecto.

Código de programa JAVA – Implementación y creación de tuplas para base de datos "ALARMA"

```
//Importación del servidor SQL con driver JDBC
import java.sql.*;
//Declaración de variables usadas por la base de datos
Connection conexion;
Statement sentenciaSQL;
ResultSet cdrr=null;
try{
  //Verifica la conexión con el driver JDBC embebido
  Class.forName("com.mysql.jdbc.Driver");
  //Realiza la conexión con el servidor MySQL asignando el //puerto
  de salida así como también el usuario y clave.
conexion=java.sql.DriverManager.getConnection("jdbc:mysql://localhos
t:3306","root","admin");
  sentenciaSQL =
conexion.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
  //Crea la base de datos "ALARMA" y sus tuplas
  sentenciaSQL.execute("CREATE DATABASE alarma;");
  sentenciaSQL.execute("USE ALARMA");
  sentenciaSQL.execute("CREATE TABLE registro(REG VARCHAR(32)NOT
  NULL, CONF VARCHAR(2) PRIMARY KEY);");
  sentenciaSQL.execute("CREATE TABLE datos(NOMBRE VARCHAR(30)NOT
  NULL, APELLIDO VARCHAR(30) NOT NULL, DIRECCION1 VARCHAR(60) NOT
  NULL, DIRECCION2 VARCHAR(60) NOT NULL, NUMERO VARCHAR(11) PRIMARY
  KEY);");
  sentenciaSQL.execute("CREATE TABLE acciones(NUMERO VARCHAR(11) NOT
  NULL,ACCION VARCHAR(13) NOT NULL, FECHA VARCHAR(10) NOT NULL, HORA
  VARCHAR(15) NOT NULL, RESPONSIBLE VARCHAR(60) NOT NULL);");
  sentenciaSQL.execute("CREATE TABLE administrador(ADMIN VARCHAR(60)
  NOT NULL, CLAVE VARCHAR(32) NOT NULL, CARGO VARCHAR(20) NOT NULL,
  CEDULA VARCHAR(10) PRIMARY KEY);");
  sentenciaSQL.execute("CREATE TABLE adminreg (HORAIN VARCHAR(15)
  NOT NULL, FECHA VARCHAR(10) NOT NULL, CEDULA VARCHAR(10) NOT
  NULL);");
  sentenciaSQL.execute("CREATE TABLE celdesvio (CELULARDESVIO
  VARCHAR(11) NOT NULL);");
  sentenciaSQL.close();
  conexion.close();
}
catch (SQLException e){
try {
  //Si la base de datos fue antes creada y se reinicia la
  //aplicacion solamente realiza la conexion para el uso de la misma
  Class.forName("com.mysql.jdbc.Driver");

conexion=java.sql.DriverManager.getConnection("jdbc:mysql://localhos
t:3306","root","admin");
  sentenciaSQL =
```

```
  conexion.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
  ResultSet.CONCUR_UPDATABLE);
  sentenciaSQL.execute("USE ALARMA");
  } catch (SQLException ex) {}
catch (ClassNotFoundException ex){}
}
catch (ClassNotFoundException ex){}
```

A partir del código de programa descrito se puede manipular las tuplas de la base de datos cuyo objetivo se basa en realizar y cumplir todas las funciones mencionadas dentro del diseño para que el administrador tenga control de las notificaciones de alarma de los usuarios, así como también del manejo de los datos que se encuentran ingresados dentro de la misma.

Para una mejor visión de las tuplas de la base de datos dentro de la aplicación se muestra a continuación los diagramas de flujo para la manipulación de las mismas de forma general ya sea para información del usuario o información del administrador, empezando por el ingreso de datos (ver Figura 3.15), edición de datos (ver Figura 3.16), borrado de datos (ver Figura 3.17) y búsqueda de datos (ver Figura 3.18).

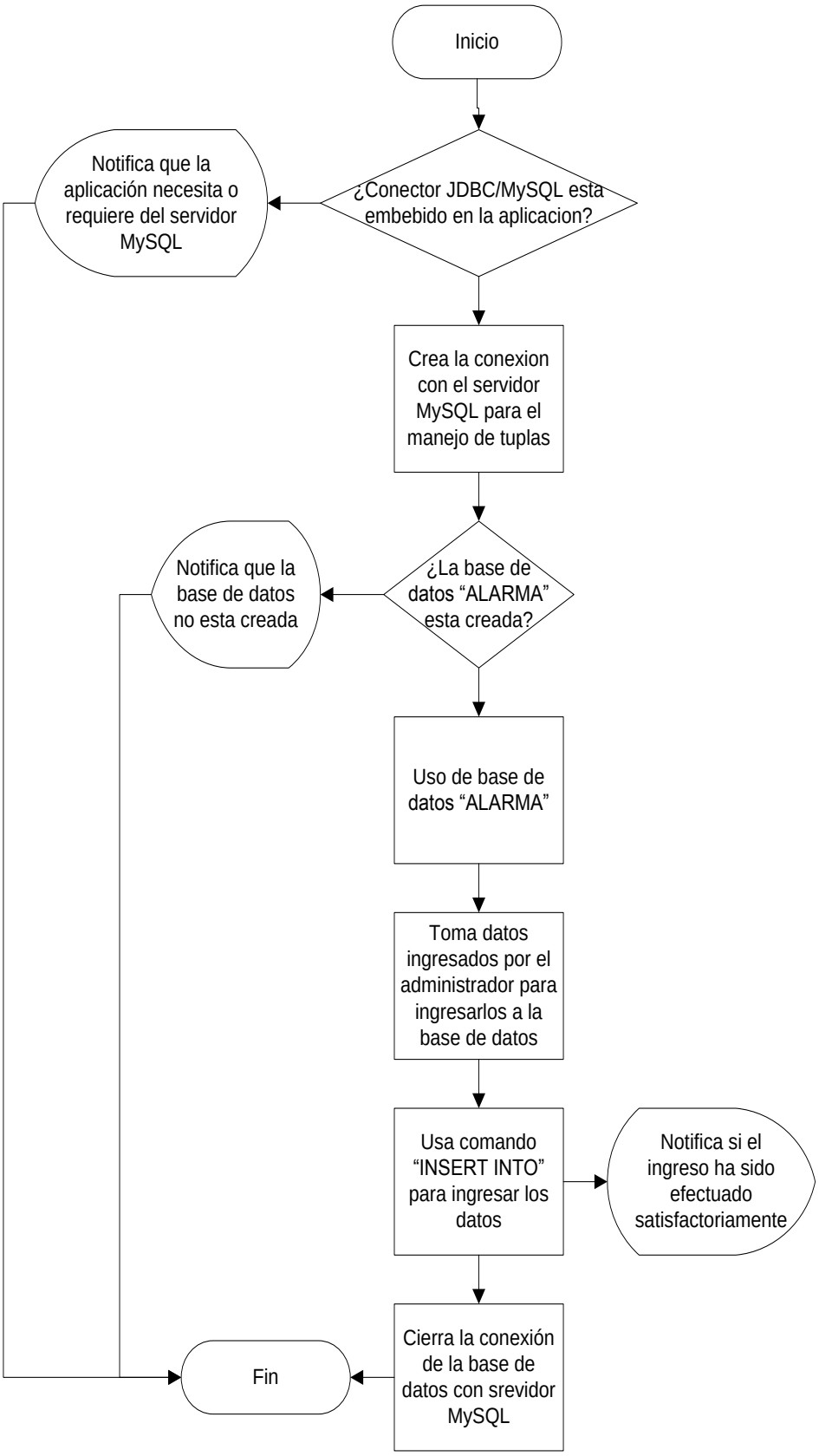


Figura. 3.15. Diagrama de flujo – Inserción de datos

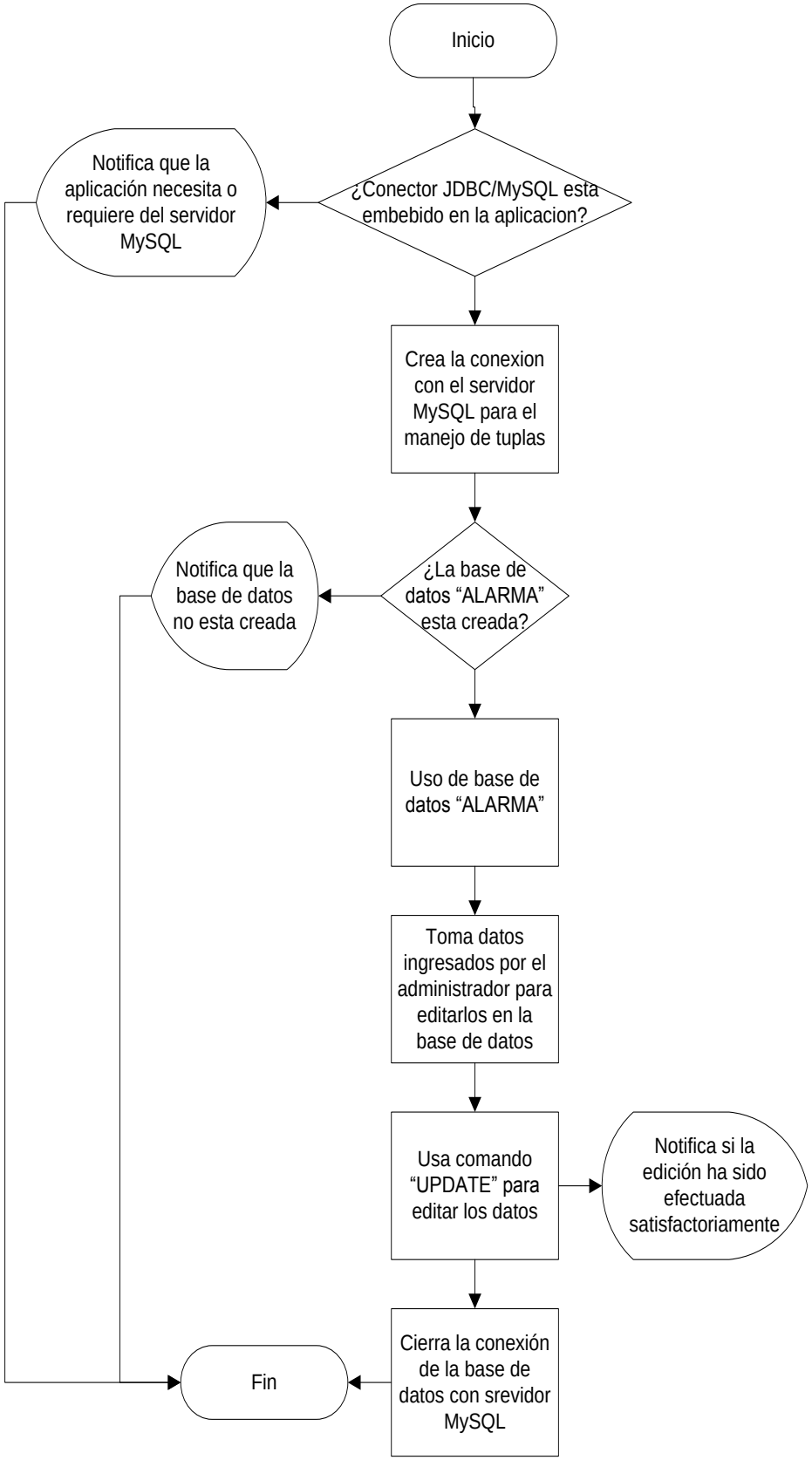


Figura. 3.16. Diagrama de flujo – Edición de datos

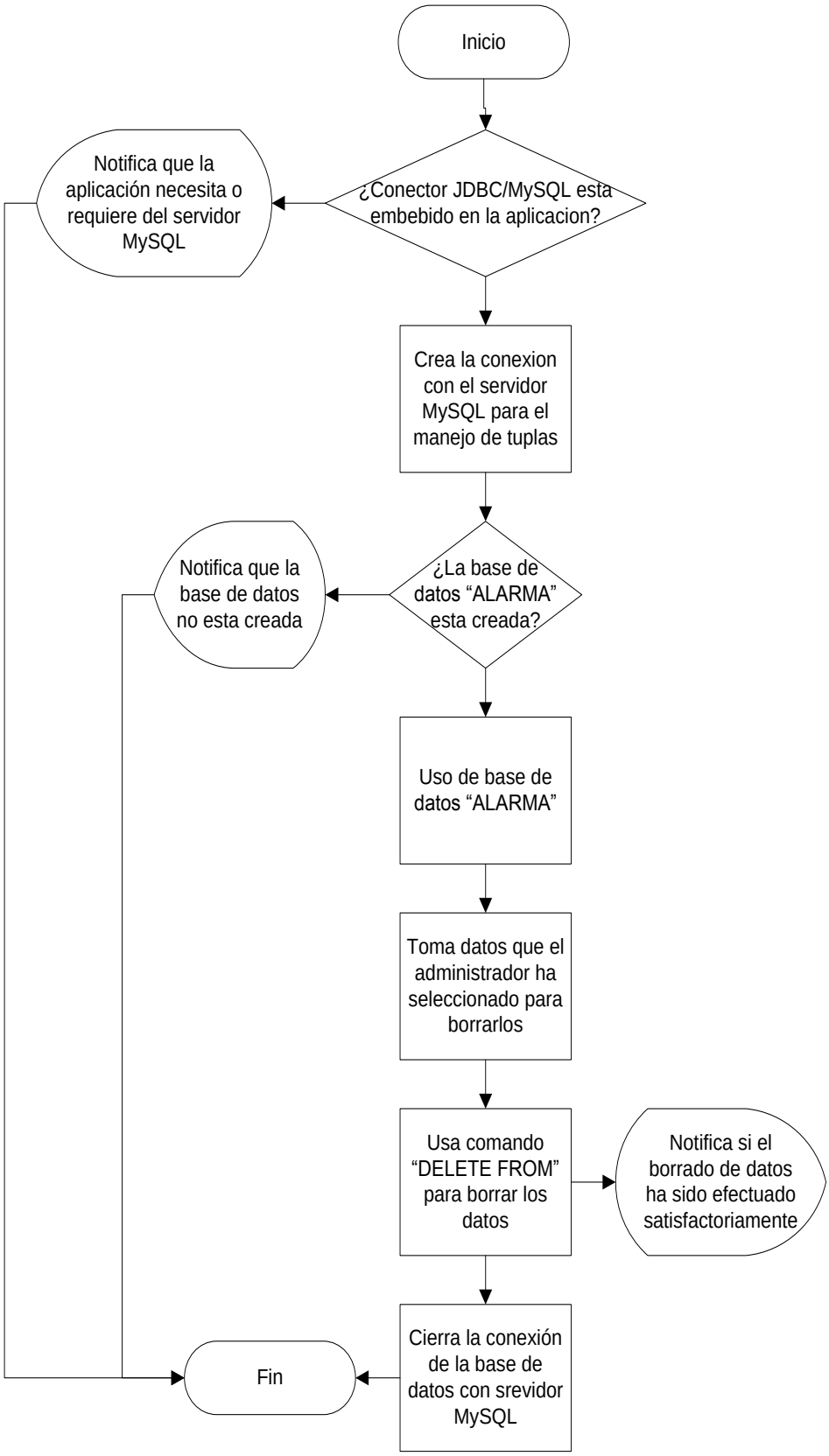


Figura. 3.17. Diagrama de flujo – Borrado de datos

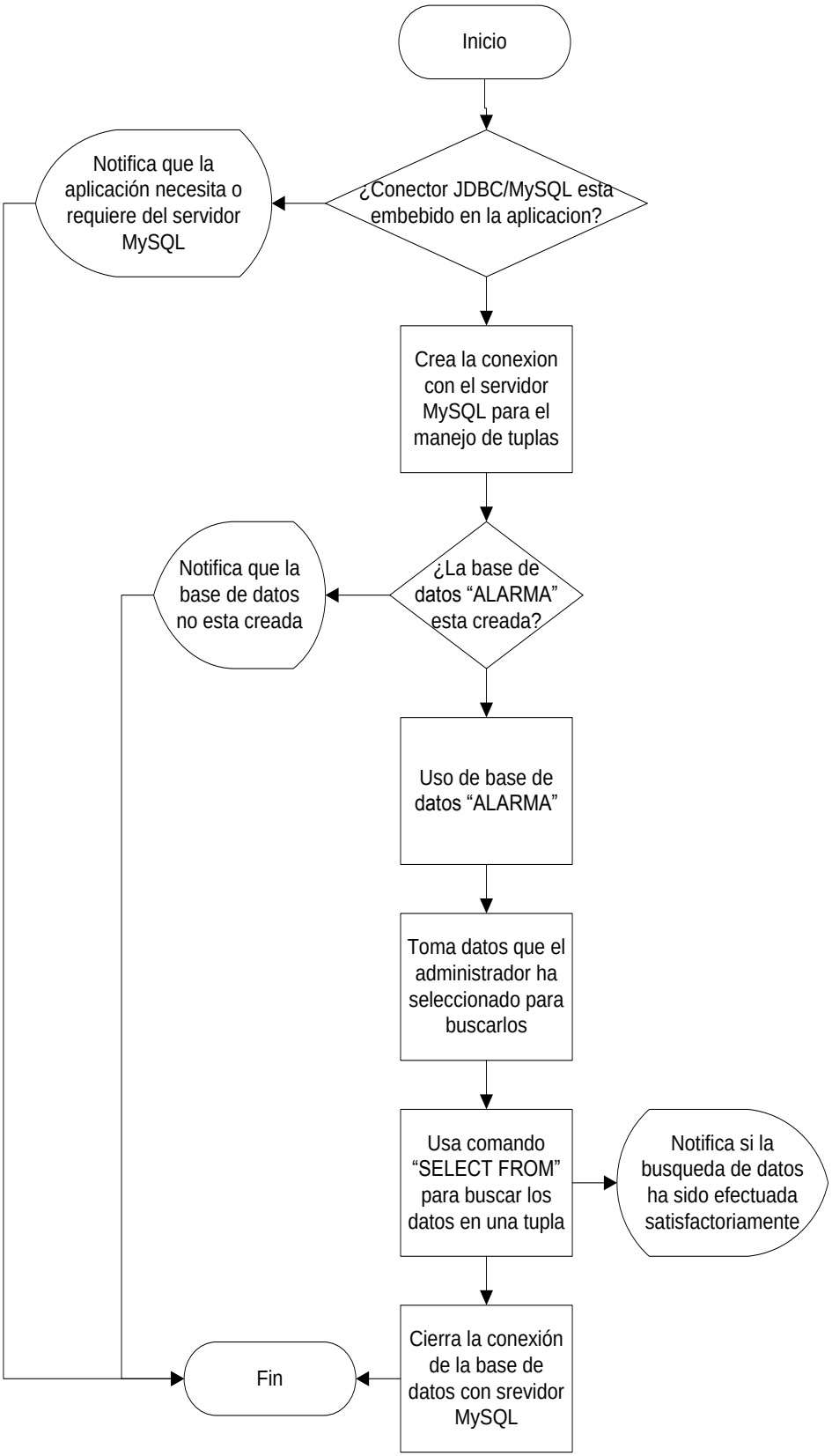


Figura. 3.18. Diagrama de flujo – Búsqueda de datos

Para complementar el subcapítulo 3.2.1.3 junto con el anterior 3.2.1.2 se menciona a continuación el código de programa para la validación y comparación de datos cuando el modem GSM/GPRS hace la lectura de los mensajes de alarma recibidos

Código de programa JAVA – Manipulación de tuplas dentro del SERVlet con recepción y transmisión de comandos AT.

```
public void run()
{
  //Compara los tiempos de llegada con el tiempo de ausencia
  //del jefe de turno en la UPC para hacer el desvio de alarma
  minutollegada2 = jTextField6.getText();
  int ml1=Integer.valueOf(minutollegada1);
  int ml2=Integer.valueOf(minutollegada2);
  if (ml1==58){ml1=-2;}
  if (ml1==59){ml1=-1;}
  if (jComboBox2.getSelectedItem().equals("Ninguno")==true &&
  ml1 + 2 == ml2){
  if (Integer.valueOf(jTextField7.getText())>45 &&
  Integer.valueOf(jTextField7.getText())<59){
  try {
  //Establece la conexion de la Base de datos para validar y
  //hacer el desvio de alarma con los datos de los usuarios
  conexion =
java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306","
root","admin");
  sentenciaSQL =
  conexion.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
  ResultSet.CONCUR_UPDATABLE);
  sentenciaSQL.execute("USE ALARMA");
  cdrr = sentenciaSQL.executeQuery("SELECT * FROM datos WHERE
  NUMERO='" + nsms + "';");
  cdrr.first();
  //Direccion
  cadee1=cdrr.getString(3);
  //Nombres
  cadee2=cdrr.getString(1);
  //Apellidos
  cadee3=cdrr.getString(2);
  //Cierra la conexion con el servidor MySQL
  cdrr.close();
  sentenciaSQL.close();
  conexion.close();
} catch (SQLException ex) {}
try
{
```

```
  String ce = jTextField19.getText();
  char[] H = new char[8];
  ce.getChars(3, 11, H, 0);
  ce = "0" + String.valueOf(H);
  String mensajesms = "ALARMA en la central por
  favor¡ATENDERLA! Datos del Remitente. Celular: " + nsms +"
Nombres: " + cadee2 + " Apellidos: " + cadee3 + " Direccion: " +
cadee1;
  delayt();
  delayt();
  //Envio de mensaje al jefe de turno con la notificacion de
  //alarma del usuario junto con todos sus datos guardados
  //en las variables nsms, cadee2, cadee3 y cadee1
  O.write(message5.getBytes());
  O.write(0x22); O.write(ce.getBytes()); O.write(0x22);
  O.write(0x0D);
  delayt();
  delayt();
  O.write(mensajesms.getBytes());
  O.write(0x1A);
}
catch(IOException e){}
cadee1=""; cadee2=""; cadee3="";

public void serialEvent(SerialPortEvent event) {
switch(event.getEventType()) {
case SerialPortEvent.BI:
case SerialPortEvent.OE:
case SerialPortEvent.FE:
case SerialPortEvent.PE:
case SerialPortEvent.CD:
case SerialPortEvent.CTS:
case SerialPortEvent.DSR:
case SerialPortEvent.RI:
case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
break;
case SerialPortEvent.DATA_AVAILABLE:
byte[] readBuffer = new byte[1024];
try
{
  while (I.available() > 0)
  {
   this.run();
   int numBytes = I.read(readBuffer);
   String w=new String(readBuffer);
   if (w.trim().startsWith("+CMS ERROR")==true)
   {//Verifica si el modem tiene un error en el envio de SMS}
   if (w.equals(null)==true && flag==0)
   {//Si no hay ningun mensaje del modem GSM}
   if (w.trim().endsWith("K")==true && flag==0)
   {//Verifica el buffer cuando recibe "K" y asigna flag=1
    flag=1;
```

```
    }
    if (w.trim().startsWith("+CPMS")==true && flag==1){flag=2;}
    if (w.trim().startsWith("OK")==true && flag==2)
    {
     flag=3;
     delayt();
     //Verifica el buffer cuando recibe "OK" y asigna flag=3
     delayt();
    }
    if (flag==3)
    {
     int arroba=0;
     delayt();
     delayt();
     try
     {
     //Envio de commando AT para lectura de SMS
     O.write(message2.getBytes());
     O.write(0x22); O.write(r.getBytes()); O.write(0x22);
     O.write(0x0D);
     }
     catch(IOException e){}
     delayt();
     delayt();
     if (w.trim().startsWith("+CMGL")==true)
     {
     //Verifica si existe un mensaje SMS en el buffer
     flagnum=0;
     mensajes=w.trim();
     //Guarda todo el buffer de mensajes SMS en "A" para que
     //sea manipulado para comparar si el usuario esta en la BD
     char A[]=new char[mensajes.length()];
     A = mensajes.toCharArray();
     for (int b=0; b<mensajes.length();b++)
     {
     if (A[b]==0x40){arroba++;}
     }
     for (int a=0;a<mensajes.length();a++)
     {if (A[a]=='K')
     {
     try
     {
       //Borra los mensajes de la memoria SIM
       O.write(message4.getBytes());
       O.write(0x22); O.write(d.getBytes()); O.write(0x22);
       O.write(0x0D);
     }
     catch (IOException ex) {}
     delayt();
     delayt();
   }
}
```

```
//Tratado de mensajes para su validación en la base de datos
mensajes="";
aumenta=0;
arroba=0;
for (int d=0;d<9;d++)
{
  if (infmensajes[d].equals(null)==false)
  {
  infmensajes[d].getChars(0, 8, numerosms, 0);
  nsms="593"+String.valueOf(numerosms).trim();
  try
  {
  conexion =
java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306","
root","admin");
  sentenciaSQL =
  conexion.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
  ResultSet.CONCUR_UPDATABLE);
  sentenciaSQL.execute("USE ALARMA");
  cdrr = sentenciaSQL.executeQuery("SELECT * FROM datos WHERE
  NUMERO='" + nsms + "';");
  cdrr.first();
  cadenumsms=cdrr.getString(5);
  if (cadenumsms.equals(nsms)==true){cadenumsms="";flagnum=1;}
  else {cadenumsms=""; flagnum=0;}
  //Cierra la conexion con el servidor MySQL
  cdrr.close();
  sentenciaSQL.close();
  conexion.close();
}
catch (SQLException e){}
//Escoge y particiona los datos de la hora, fecha y contenido
//de los mensajes SMS validados
infmensajes[d].getChars(12, 20, fechasms, 0);
infmensajes[d].getChars(21, 29, horasms, 0);
infmensajes[d].getChars(35, 41, sms, 0);
//Compara si el contenido del SMS es @1@4m@ para mostrar en
//pantalla la notificación que proviene del usuario
  if (flagnum==1 && String.valueOf(sms).equals("@1@4m@"))
  {
  if (String.valueOf(sms).equals("@1@4m@")==true){tipo=1;}
  }
  //Muestra la notificación de alarma en pantalla y produce un
  //sonido de alarma en el SERVlet.
  }catch (IOException e) {}
}
```

3.2.1.4. Creación de archivos *Microsoft Excel (.xls)* como reportes de bases de datos

En el subcapítulo anterior se tomo en cuenta la búsqueda que el administrador puede realizar en base a los datos que tiene el SERVlet ya sean de usuario o de operadores, es por eso que se optó por utilizar una herramienta muy eficaz para que importe los datos que se requieran y puedan ser examinarlos para la realización de informes que la comunidad necesite con respecto a las notificaciones que hayan sido enviadas por algún miembro de la misma, o a su vez para llevar un registro de las personas que están a cargo del SERVlet dentro del UPC (Unidad de Policía Comunitaria).

Para ello es necesario que la librería *jxl.jar* de tablas *Microsoft Excel* esté importada y embebida en la aplicación así como se mencionó antes con las librerías de conexión serial *RXTXcomm.jar* y driver JDBC/MySQL.

Se presenta de forma rápida y precisa la manipulación de tablas de datos que están en el SERVlet para su debida exportación a archivos con extensión *.xls* ilustrado en el diagrama de flujo de la Figura 3.19.

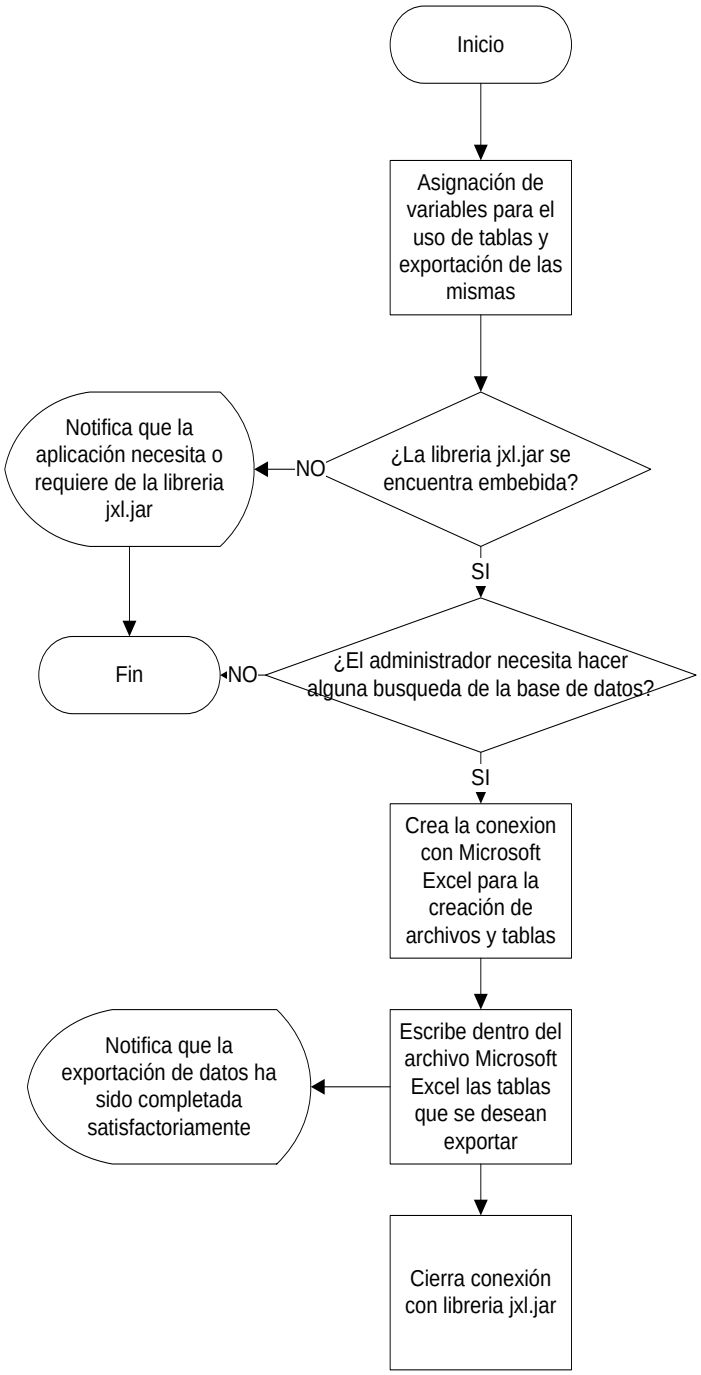


Figura. 3.19. Diagrama de flujo – Exportación de tablas del SERVlet a Microsoft Excel

A continuación se detallará la implementación en el SERVlet la exportación de tablas de datos en lenguaje de programación JAVA existiendo una clase llamada *ExcelTableExporter* dentro del código general del formulario usado y creado para la creación del proyecto.

Código de programa JAVA – Implementación y creación de clase para la exportación de datos a Microsoft Excel

```
//Creacion de la clase necesaria con sus atributos para la
//exportacion de archivos
public class ExcelTableExporter {

  private File archivo;
  private JTable tabla;
  private String nombreTabla;
  private WritableCellFormat cf;

public ExcelTableExporter(JTable tabla, File archivo, String
nombreTabla) {
  this.archivo = archivo;
  this.tabla = tabla;
  this.nombreTabla = nombreTabla;
}

public boolean export() {
  try {
  //Creación de conexion y de archive Microsoft Excel
  DataOutputStream out = new DataOutputStream(new
  FileOutputStream(archivo));
  WritableWorkbook w = Workbook.createWorkbook(out);
  WritableSheet s = w.createSheet(nombreTabla, 0);
  //Crea los titulos de las tablas que estan en busqueda
  if (flagtabla==1)
  {
  s.addCell(new Label(0,0,"Cedula"));
  s.addCell(new Label(1,0,"Hora de Entrada"));
  s.addCell(new Label(2,0,"Fecha"));
  }
  if (flagtabla==2)
  {
  s.addCell(new Label(0,0,"Número"));
  s.addCell(new Label(1,0,"Nombres"));
  s.addCell(new Label(2,0,"Apellidos"));
  s.addCell(new Label(3,0,"Fecha"));
  s.addCell(new Label(4,0,"Hora"));
  s.addCell(new Label(5,0,"Direccion 1"));
  s.addCell(new Label(6,0,"Direccion 2"));
  s.addCell(new Label(7,0,"Accion"));
  s.addCell(new Label(8,0,"Responsable"));
}
//Escribe los datos que el administrador selecciono para la
//exportacion de los mismos
for (int i = 0; i < tabla.getRowCount(); i++) {
for (int j = 0; j < tabla.getColumnCount(); j++) {
  Object objeto = tabla.getValueAt(i, j);
  s.addCell(new Label(j,i+1, String.valueOf(objeto)));
```

```
 }
}
  //Cierra la conexion con la libreria jxl.jar
  w.write();
  w.close();
  out.close();
  return true;
} catch (Exception ex) {
  return false;
  }
 }
}
```

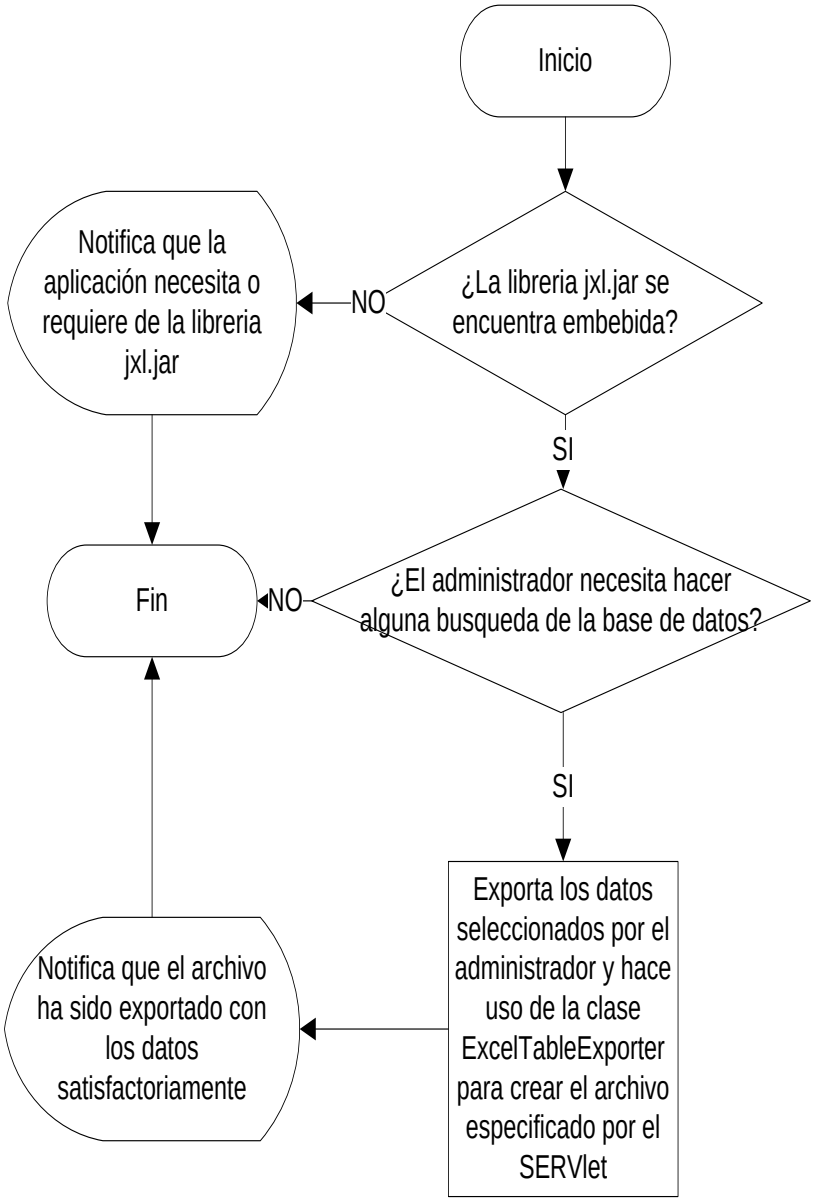


Figura. 3.20. Diagrama de flujo – Uso de clase ExcelTableExporter para la exportación de datos seleccionados por el administrador

Una vez definido el código que se usa para la exportación y creación de archivos Microsoft Excel se hace uso de la clase que tiene esa función junto con los datos que el administrador del SERVlet hace referencia para su exportación ilustrando así su diagrama de flujo en la Figura 3.20 y explicado a continuación de forma más detallada en el código de programa JAVA generalizando las búsquedas y exportaciones que la aplicación puede realizar.

Código de programa JAVA – Uso de clase ExcelTableExporter para la creación de archivos con datos del SERVlet

```
//Llama a la clase para hacer uso de la misma y crea un //archivo en
la dirección que el desarrollador le asigna
ExcelTableExporter exportador=new
ExcelTableExporter(this.tabla,new
File("C:\\SecAlarm\\Reportes\\Nombre del archivo.xls"),"Nombre del
archivo");
boolean exportacionOk=exportador.export();
//Mensaje de confirmación diciendo si los datos han sido
//exportados o no.
if(exportacionOk) {
  JOptionPane.showMessageDialog(this,"Exportacion a Excel
  correcta","Mensaje",JOptionPane.INFORMATION_MESSAGE);}
else {
  JOptionPane.showMessageDialog(this,"Error en la
  exportacion","Mensaje",JOptionPane.ERROR_MESSAGE);
}
```

3.2.1.5. Conversión de archivos java (.JAR) a ejecutables (.EXE)

En este subcapítulo se hace referencia a la facilidad de uso que el SERVlet debe tener para el administrador, para esto es necesario un Software adicional para la conversión de archivos con extensión *.jar* a ejecutables con extensión *.exe* llamado JSmooth en su versión 0.9.9-7 de carácter libre sin licencia el cual trabaja con generadores avanzados para la detección de funciones u opciones con base JRE. Crea archivos estandarizados binarios ejecutables de Windows (.exe) que contienen toda la información necesaria para ejecutar la aplicación JAVA. También brinda la facilidad de poder controlar algunos problemas de desarrollo con archivos JRE que no se encuentren en buen estado instalados en la maquina del

usuario, además de facilitar y solucionar problemas con la PC cuando no se tiene instalado el paquete JRE/JDK para el uso de la aplicaciones JAVA, incluso tiene una detección de "Java Virtual Machine" automática, crea interfaces de usuario amigables, la configuración de la aplicación es sencilla y sobretodo comprende la agrupación de librerías que algunas aplicaciones necesitan para el uso de archivos con extensión .jar externas, sin hacer a un lado el SERVlet que el proyecto contempla.

Se detalla a continuación los pasos que se deben seguir para la conversión de extensiones para que la aplicación quede al fin completa para su uso.

a) Ejecutando JSmooth 0.9.9-7 para su uso

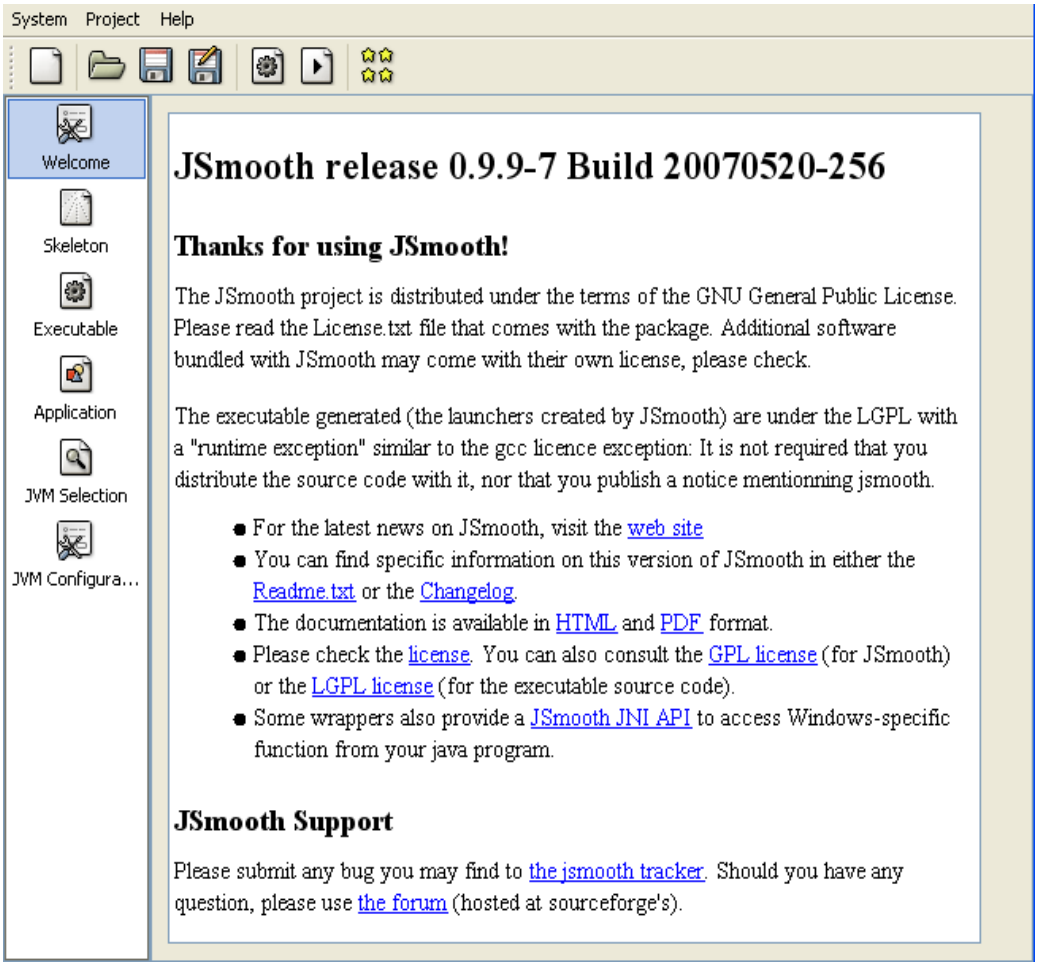


Figura. 3.21. Ejecución de JSmooth 0.9.9-7

En la Figura anterior se observa la primera pantalla después de la ejecución de JSmooth para su uso, además nos entrega catálogos de utilización y foros en páginas WEB para que el usuario pueda usar y hacer la conversión de archivos fácilmente.

b) Configurando el esqueleto de la aplicación para la verificación de paquetes JRE instalados en la máquina.

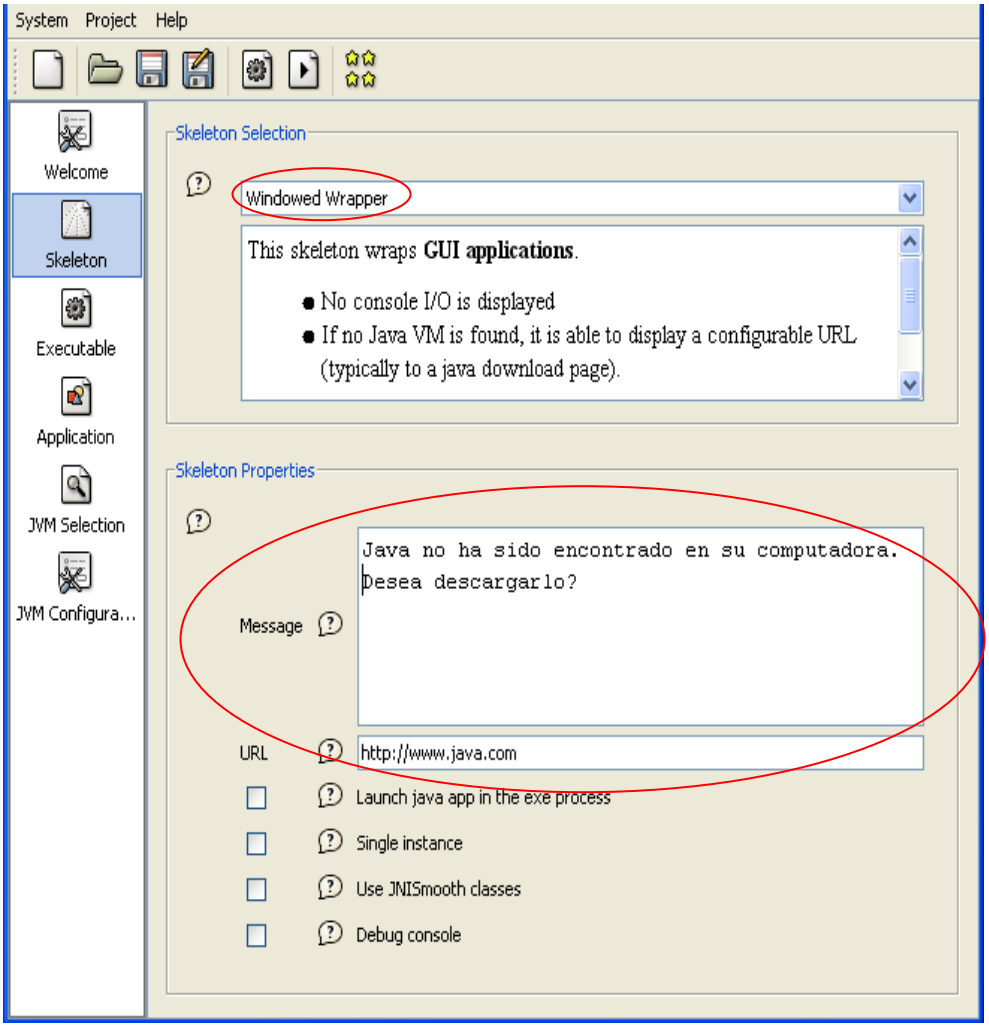


Figura. 3.22. Esqueleto del SERVlet para archivo ejecutable.

El esqueleto de la aplicación descrito en la Figura 3.22 se basa en saber si la aplicación será ejecutada sobre la consola de Windows DOS o simplemente sobre una interfaz gráfica. El

SERVlet está desarrollado sobre un formulario de JAVA totalmente gráfico es por eso que se prefirió escoger la opción "*Windowed Wrapper*" para que la aplicación se ejecute en una interfaz totalmente gráfica y sencilla para el usuario, además el esqueleto de JSmooth busca si la PC del usuario tiene los paquetes necesarios de JAVA instalados para su uso, si no los tiene inmediatamente notifica mostrando un mensaje y el URL de descarga para los paquetes necesarios del SERVlet.

c) Configurar las opciones de la aplicación .exe

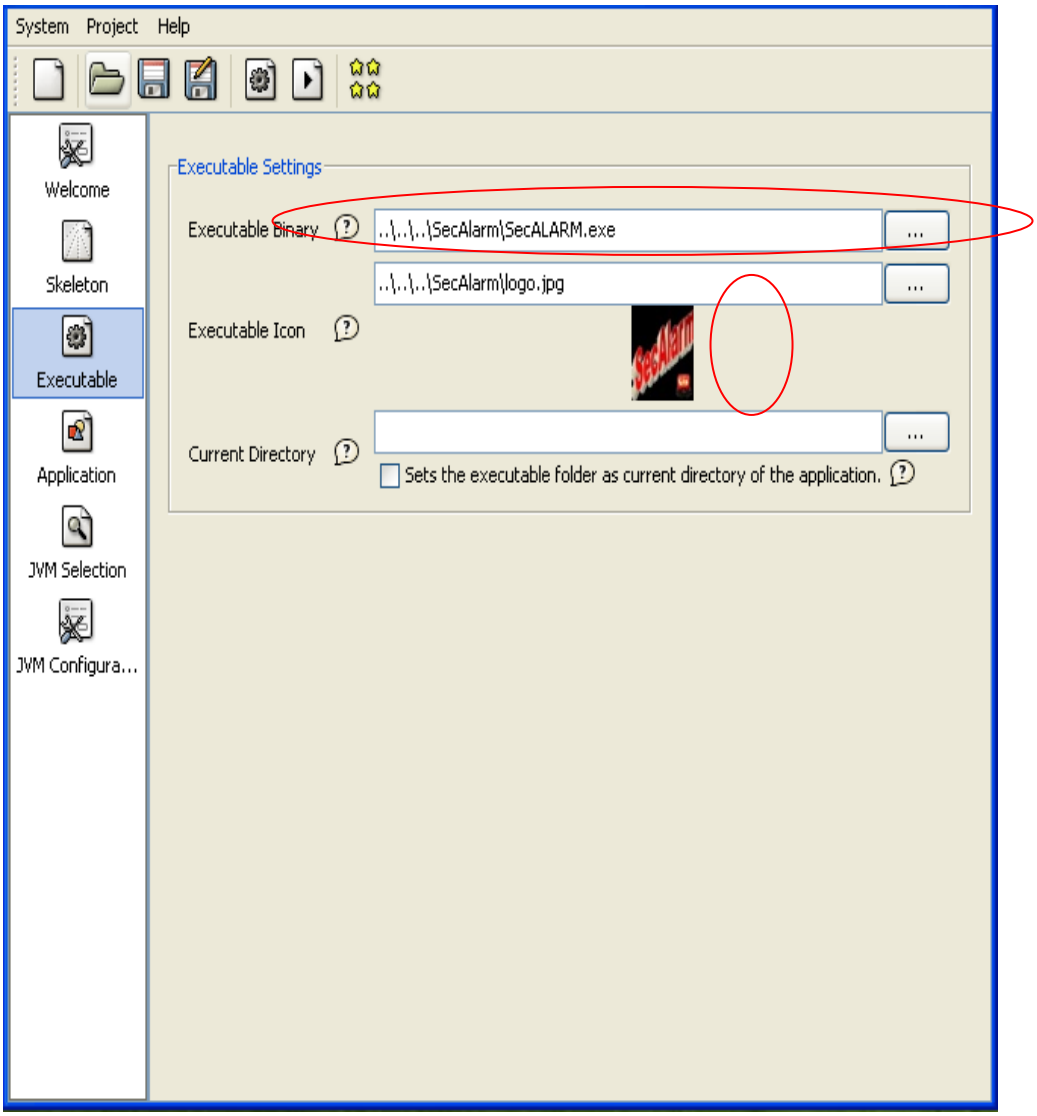


Figura. 3.23. Opciones de archivo ejecutable

Es estrictamente necesario conocer una dirección de alguna carpeta para que la aplicación pueda ser guardada y sobretodo ejecutada de buena forma, es por eso que el desarrollador antes de obrar con las opciones de los archivos ejecutables, debe crear un directorio el cual contendrá todos los archivos del SERVlet. Para la realización del proyecto se optó por crear un directorio en la dirección *C:* y llamarlo *SecAlarm*, junto con su archivo ejecutable *SecALARM.exe* y su logo respectivamente añadido en las opciones como se muestra en la Figura 3.23.

d) Configuración de aplicación JAVA y librerías embebidas.

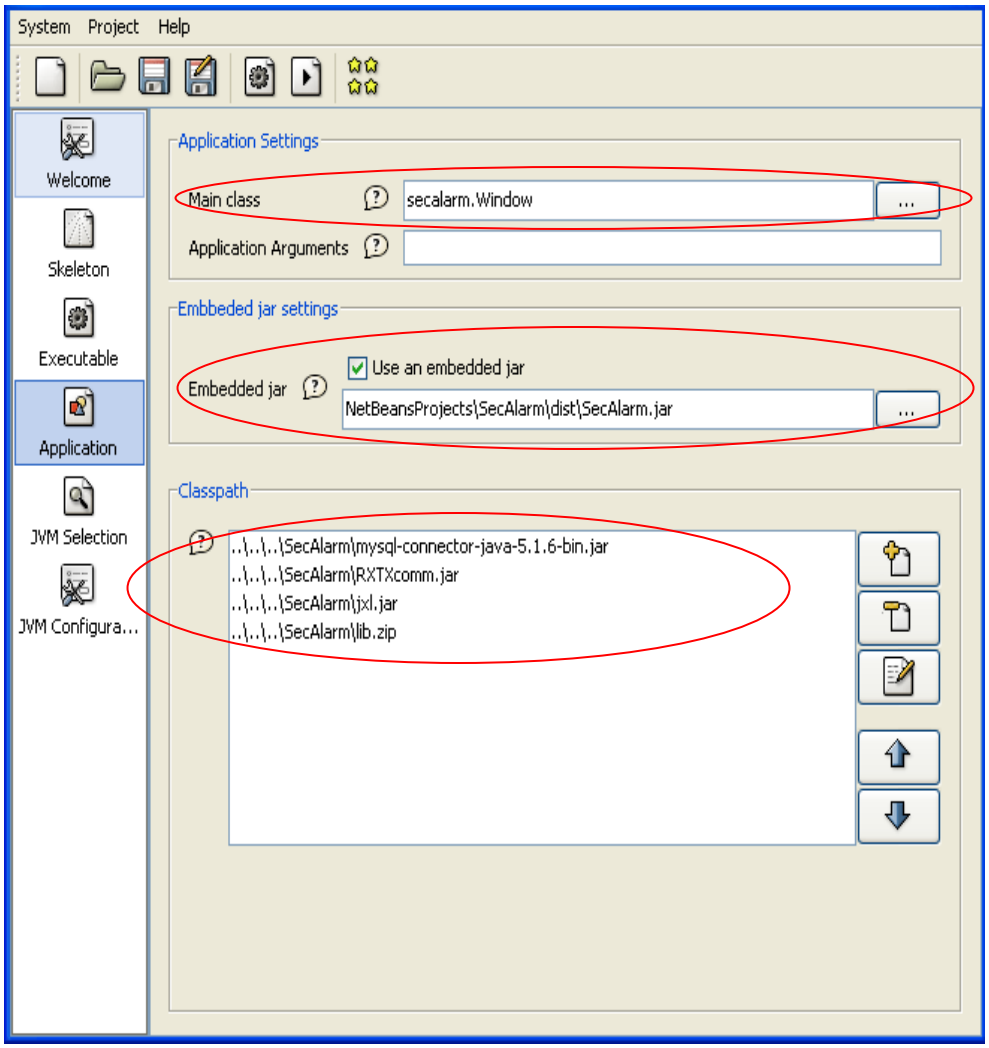


Figura. 3.24. Opciones de aplicación JAVA junto con librerías embebidas externas con extensión .jar

El software JSmooth tiene como característica principal la selección de la aplicación con extensión .jar que el desarrollador selecciona, así como también la añadidura de aplicaciones externas con la misma extensión. En la Figura 3.24 se detalla la selección de la interfaz gráfica del SERVlet, además de sus librerías embebidas que debe usar detalladas y mencionadas en subcapítulos anteriores que necesariamente deberán estar en el directorio

C:\SecAlarm

e) Configuración de maquina virtual de JAVA.

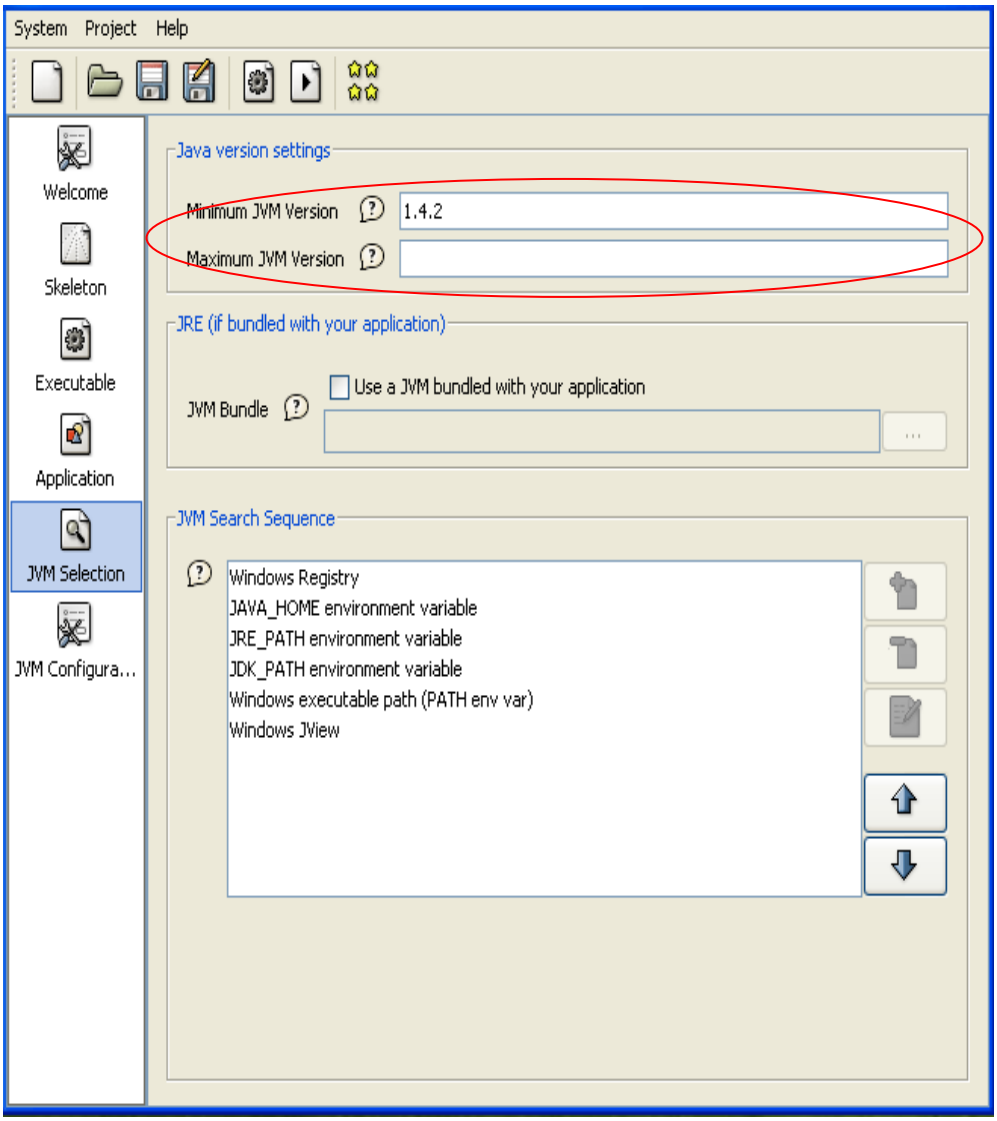


Figura. 3.25. Opciones de configuración de maquina virtual JAVA.

Para el desarrollador es un poco complicado saber desde que versión de maquina virtual JAVA puede ser ejecutado el SERVlet sin ningún tipo de problemas, es por eso que en los manuales de JSmooth se describe que la versión mínima para las aplicaciones gráficas de Windows más estable es la 1.4.2 siendo ésta la seleccionada para la realización del proyecto y la conversión de extensiones como se muestra en la Figura 3.25.

f) Configuración de Memoria de la PC

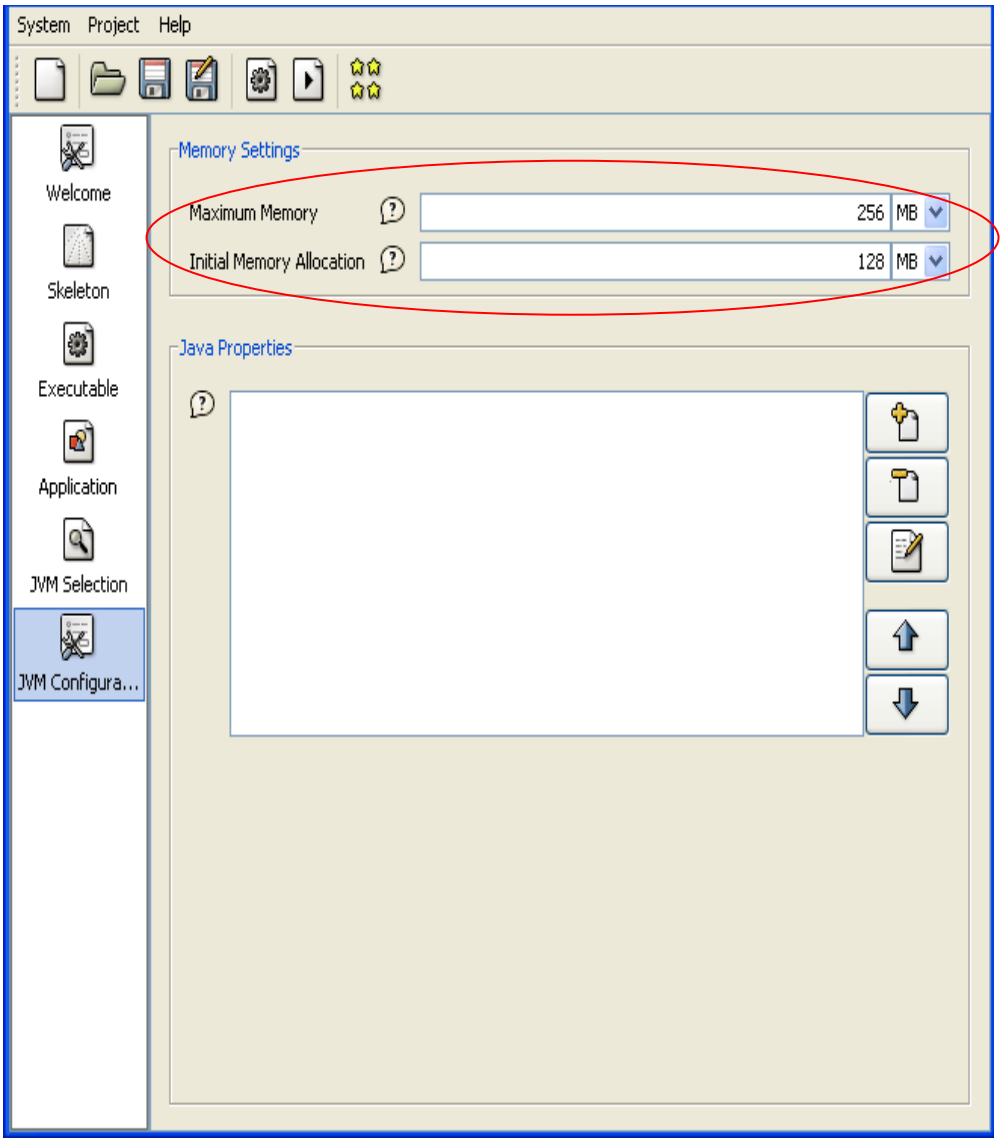


Figura. 3.26. Opciones de configuración de memoria del PC en donde el SERVlet estará ejecutándose

A partir de las pruebas realizadas con el SERVlet detalladas en el capítulo 4 de este documento se sabe de manera explícita como trabaja la memoria del PC en donde el SERVlet ha sido instalado, para optimización de recursos y ocupación de los mismos después de las pruebas realizadas en varios computadores se ha tomado como referencia valores de memoria máximos de 256MB cuando el SERVlet está en pleno trabajo adquiriendo datos del modem GSM y 128MB para cuando sea inicializado, como se ilustra en la Figura 3.26.

g) Compilación y resultado de conversión

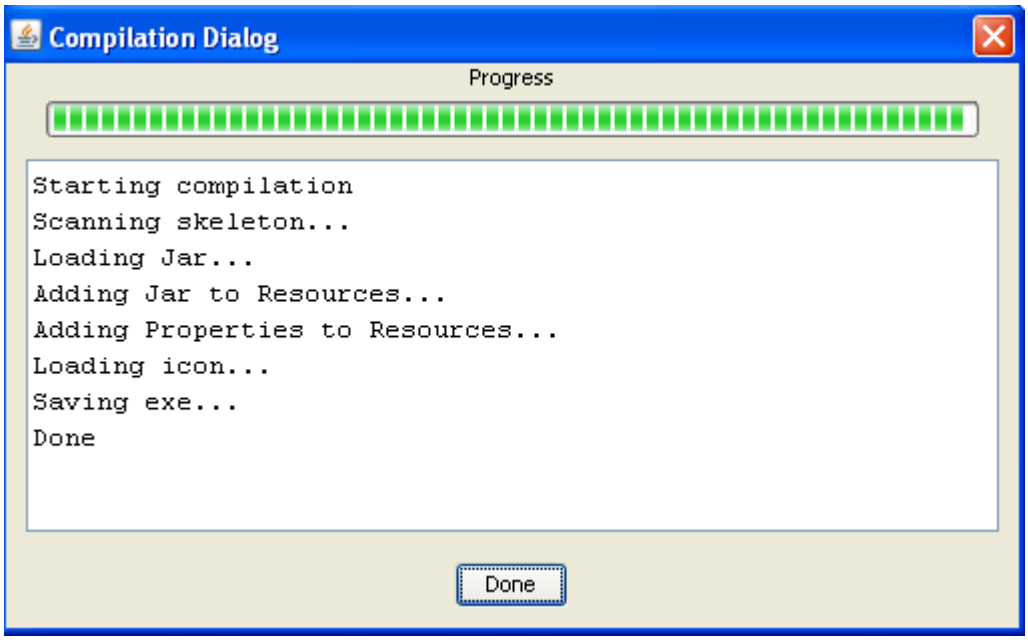


Figura. 3.27. Compilación y conversión de archivo con extensión .jar a ejecutable .exe

Una vez que todas las opciones de configuración para JSmooth estén totalmente concordadas por el desarrollador serán compiladas para tener como resultado un archivo ejecutable del SERVlet como ilustra el recuadro de confirmación en la Figura 3.27 el cual será almacenado en el directorio asignado por el mismo como se muestra en la Figura 3.28.

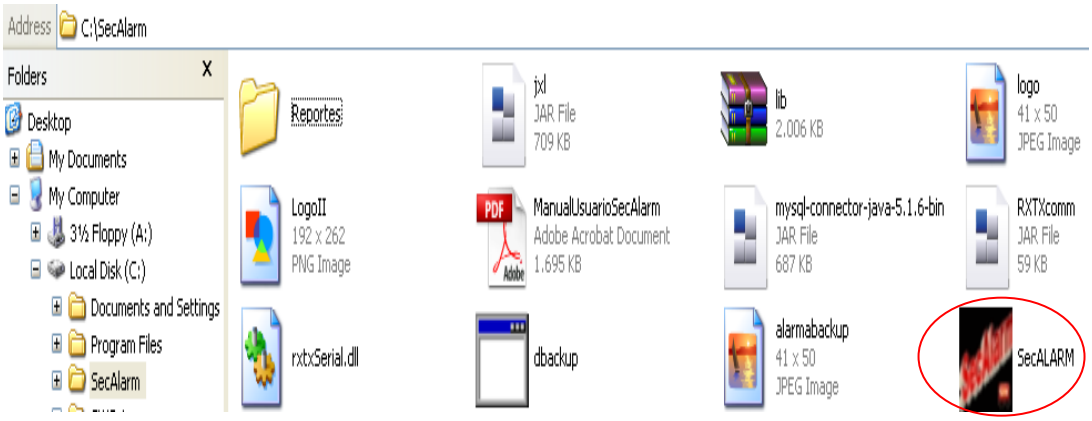


Figura. 3.28. Resultado en el directorio asignado por el desarrollador de la compilación de Software JSmooth para la conversión de archivos .jar a ejecutables .exe

De esta forma queda definido e implementado absolutamente el SERVlet con la base de datos y sus tuplas así como también la transmisión y recepción de información del modem GSM con la validación de datos de los usuarios mediante la conexión del servidor MySQL detallados en los subcapítulos 3.2.1.1, 3.2.1.2 y 3.2.1.3, la exportación de datos a una herramienta poderosa llamada *Microsoft Excel©* para la presentación de informes de notificaciones de alarma y registro de administradores con hora y fecha detallado en 3.2.1.4 y que el administrador pueda abrir y cerrar el SERVlet de manera sencilla y sin complicaciones para el manejo del mismo detallado en 3.2.1.5.

3.2.2.Diseño de aplicación MIDlet usando J2ME (*Java 2 Micro Edition*)

La aplicación J2ME es la interfaz gráfica para el envío de alarmas para el usuario dentro de la comunidad hacia el UPC en donde el sistema recolector de datos esté instalado para así poder procesar.

Con este propósito y para la notificación de alarmas comunitarias provenientes de los usuarios se decidió desarrollar una aplicación que permita realizar el envío de mensajes cortos SMS hacia la UPC de forma rápida y confiable, siendo también muy interactiva con el usuario y fácil de usar.

Una vez que la aplicación pueda ser manipulada por el usuario enseguida podrá realizar el envío de alarmas hacia el SERVlet y que este reciba las notificaciones como se muestra en la Figura 3.4 y se especifica detalladamente en el subcapítulo 3.2.1, en seguida se ilustra en la Figura 3.29 el envío de mensajes cortos SMS desde el usuario hacia la UPC.

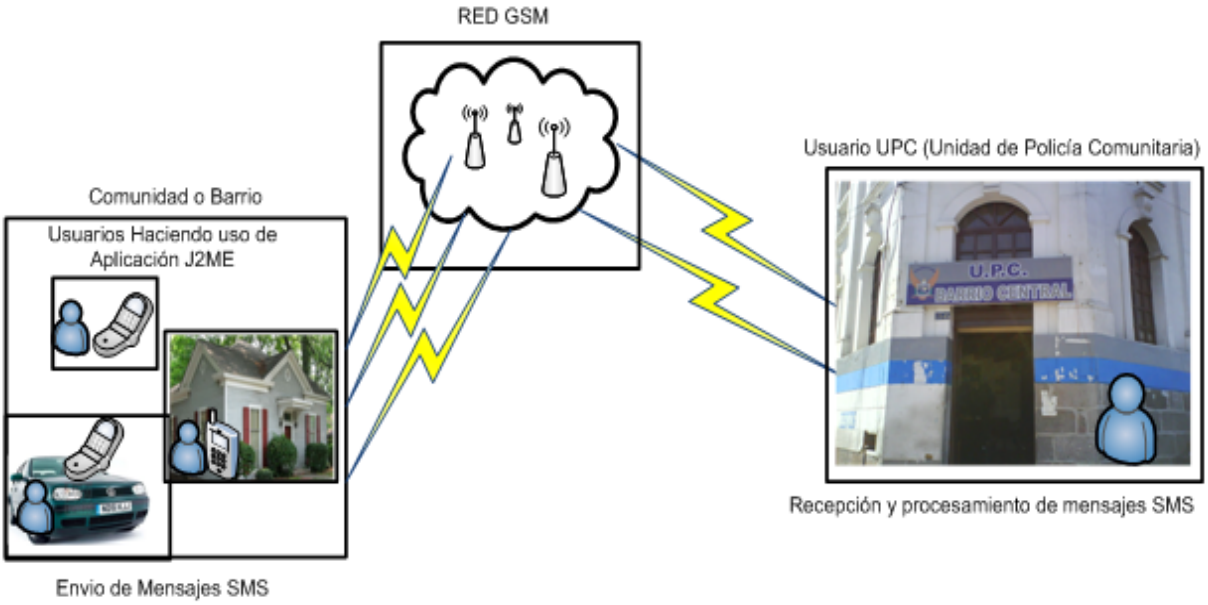


Figura. 3.29. Diagrama de bloques - Envío de mensajes cortos SMS desde la comunidad o barrio hasta el UPC haciendo uso de la red de operador celular

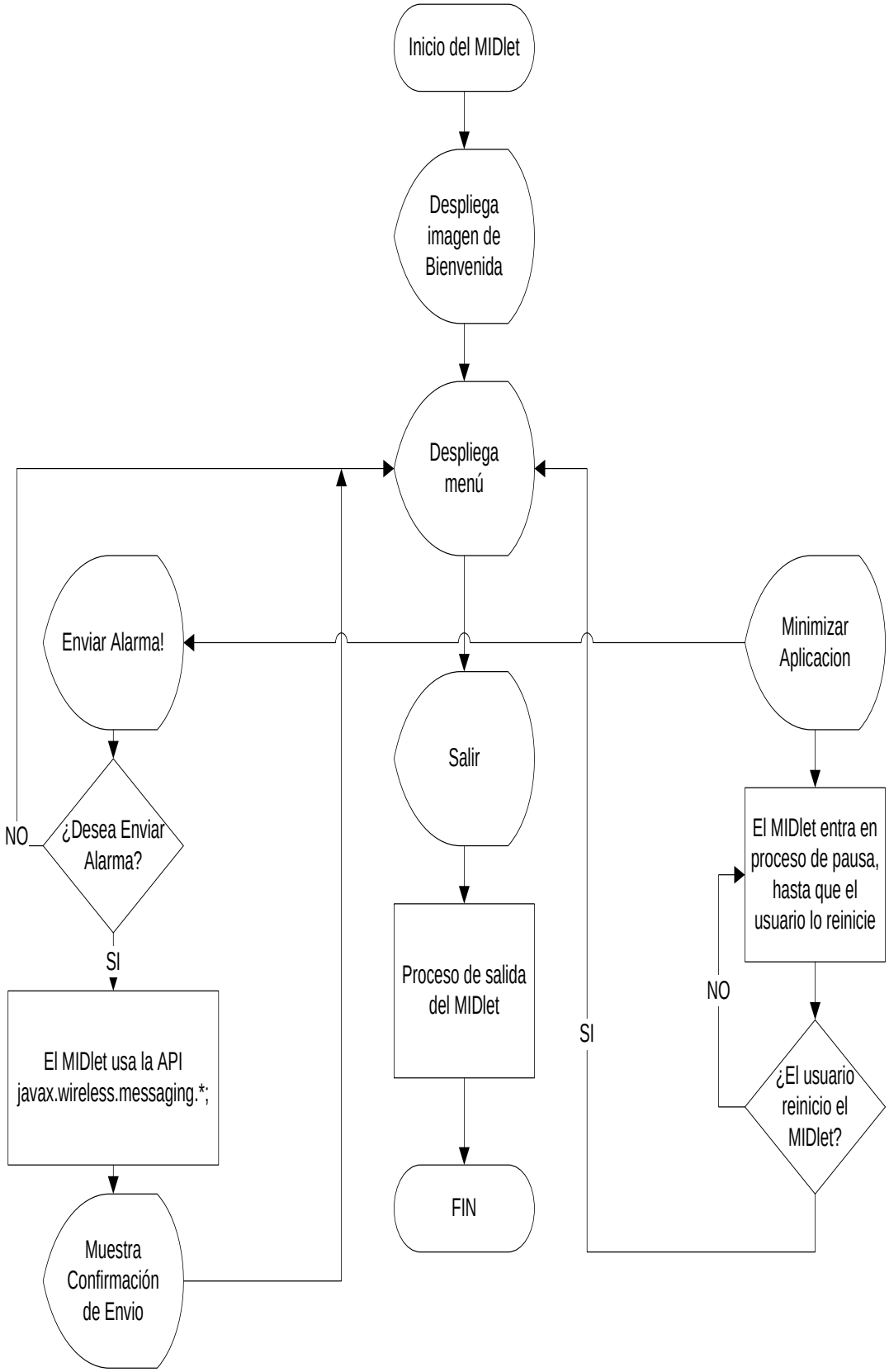


Figura. 3.30. Diagrama de flujo – MIDlet

Para el diseño del MIDlet es necesario también agregar algunas opciones para la operación del mismo, con el fin de obtener una aplicación que no tenga complejidad para el usuario del barrio o comunidad.

Es por eso que se optó incrementar dentro del diseño las siguientes opciones:

Tabla. 3.5. Menú de opciones del MIDlet

Opciones	Detalle
Enviar Alarma	Esta opción es la encargada del manejo de APIs embebidas dentro del MIDlet para el envío de mensajes cortos SMS hacia el SERVlet.
Minimizar MIDlet	Es la autorizada de establecer una comunicación de la aplicación para navegar en otras aplicaciones del celular si cerrar el MIDlet.
Salir del MIDlet	Proceso de cerrado de aplicación java y expiración del MIDlet.

El diagrama de flujo del MIDlet está descrito en la Figura 3.30 junto con las opciones del diseño descritas en la Tabla 3.5, con todos los procesos y decisiones que toma para cumplir los detalles de cada una de las opciones.

A continuación se puntualizará la implementación de los procesos más significativos de la aplicación con códigos en lenguaje JAVA usando Netbeans IDE cuya patente está registrada en *Oracle Corporation*© haciendo uso del paquete JavaME para implementación y simulación de dispositivos celulares.

3.2.2.1. Flujo de aplicación

En el MIDlet es importante establecer un modo de programación para la facilidad del desarrollador, en el capítulo 2 se describe minuciosamente la creación de MIDlets para su uso, exhortando la creación de aplicaciones MIDP/CLDC las cuales conllevan a que el desarrollador programe el MIDlet vía diagramas de flujo lo cual proporciona un control ordenado sobre la aplicación como se detalla en la Figura 3.31, siguiendo como prototipo el diagrama de flujo que se impuso anteriormente en la Figura 3.30.

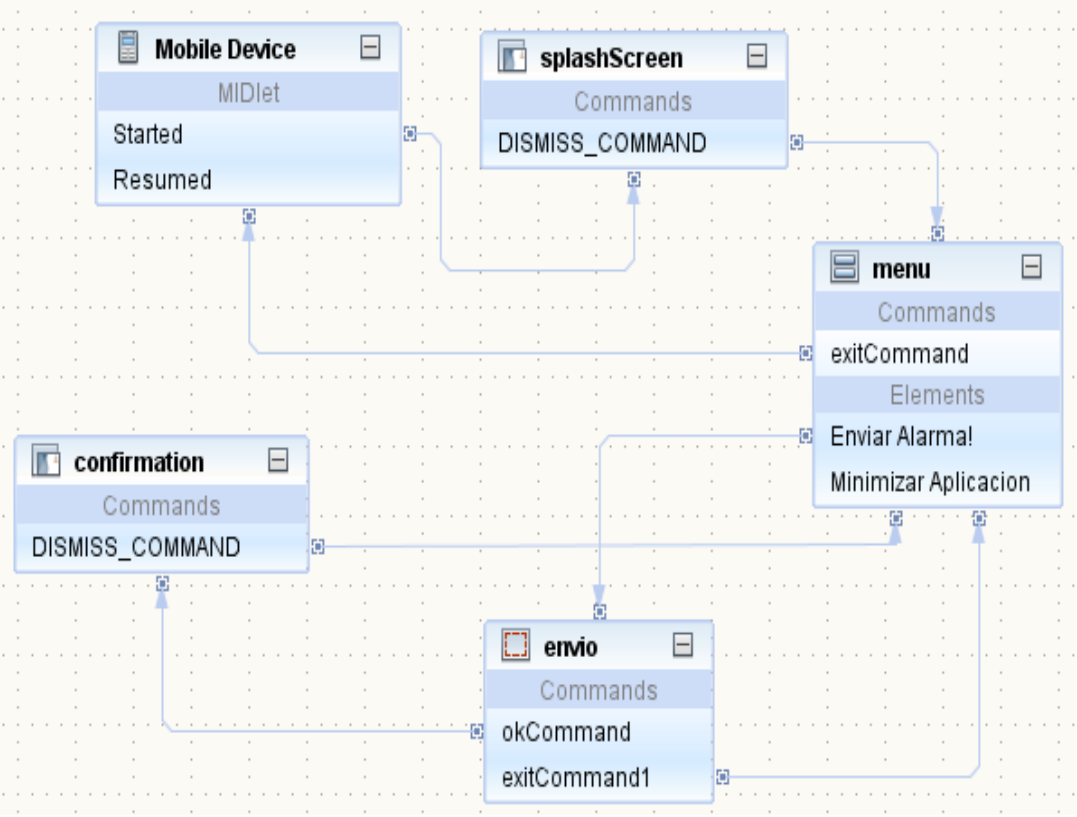


Figura. 3.31. Flujo de Aplicación – Programación MIDP/CLDC del MIDlet (Netbeans IDE)

Para la descripción de la Figura 3.31 se mostrará a continuación lo que hace cada recuadro del flujo de la aplicación y la función que tiene dentro del MIDlet, cabe recalcar que la programación MIDP/CLDC minimiza la escritura de código proveniente del desarrollador, es decir Netbeans IDE será el desarrollador automático de código de programa del MIDlet.

- *Mobile Device:* Es el encargado de que el MIDlet sea inicializado y perfectamente para su uso en el dispositivo, así como también tener la posibilidad de reactivarse cuando el MIDlet es cerrado por causas externas que no sea el usuario.

- *SplashScreen:* Se encarga de mostrar una imagen reducida la cual es presentada cada vez que el MIDlet pasa por el proceso *Mobile Device* durante 1 segundo.

- *menu:* Contiene el menú del MIDlet el cual está a cargo de establecer la comunicación e interfaz del dispositivo con el usuario de manera sencilla, presentando los elementos *Enviar Alarma!* y *Minimizar Aplicación* los cuales son insertados por el desarrollador y los elementos *Seleccionar* y *Salir* insertados por el IDE. El elemento "*Enviar Alarma!"* guía directamente al siguiente recuadro de "*envio"* para la confirmación de que el usuario requiere o solicita enviar una notificación hacia el SERVlet.

- *envio:* Es el encargado de establecer la comunicación enviando los mensajes SMS llamando a la función implementada por el desarrollador llamada WMA (Wireless Message API) la cual será puntualizada en el subcapítulo siguiente.

- *confirmation:* Tras el envío de mensajes cortos SMS resuelve si el usuario contiene un paquete de mensajes contratado en su celular y verifica que puede enviar el mensaje hacia el SERVlet mostrando imágenes de confirmación, es decir un "visto" si el mensaje fue enviado o una "cruz" si el mensaje fue denegado en su envío.

Para una mejor visión de cómo funciona el MIDlet se ilustra en la Figura 3.32 la forma de cómo trabaja el mismo en el celular.

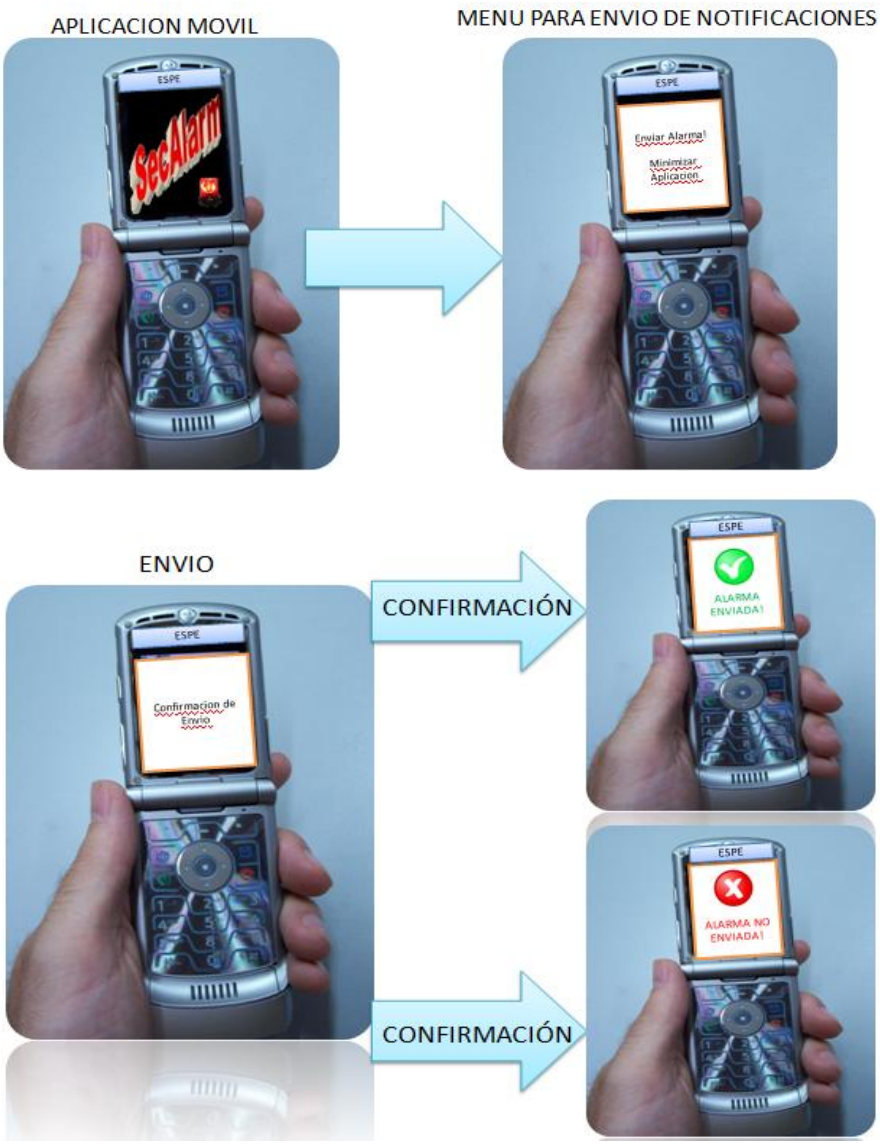


Figura. 3.32. Flujo de Aplicación de manera demostrativa en celulares

3.2.2.2. Implementación de Wireless Message API (WMA)

Debido a que se plantearon objetivos específicos para este proyecto fue necesaria la implementación de código de programa para que cumpla la interacción del MIDlet con el SERVlet vía mensajes de texto SMS, es por eso que se ilustra en la Figura 3.33 el diagrama de flujo que WMA debe cumplir para que la aplicación móvil pueda realizar lo planteado.

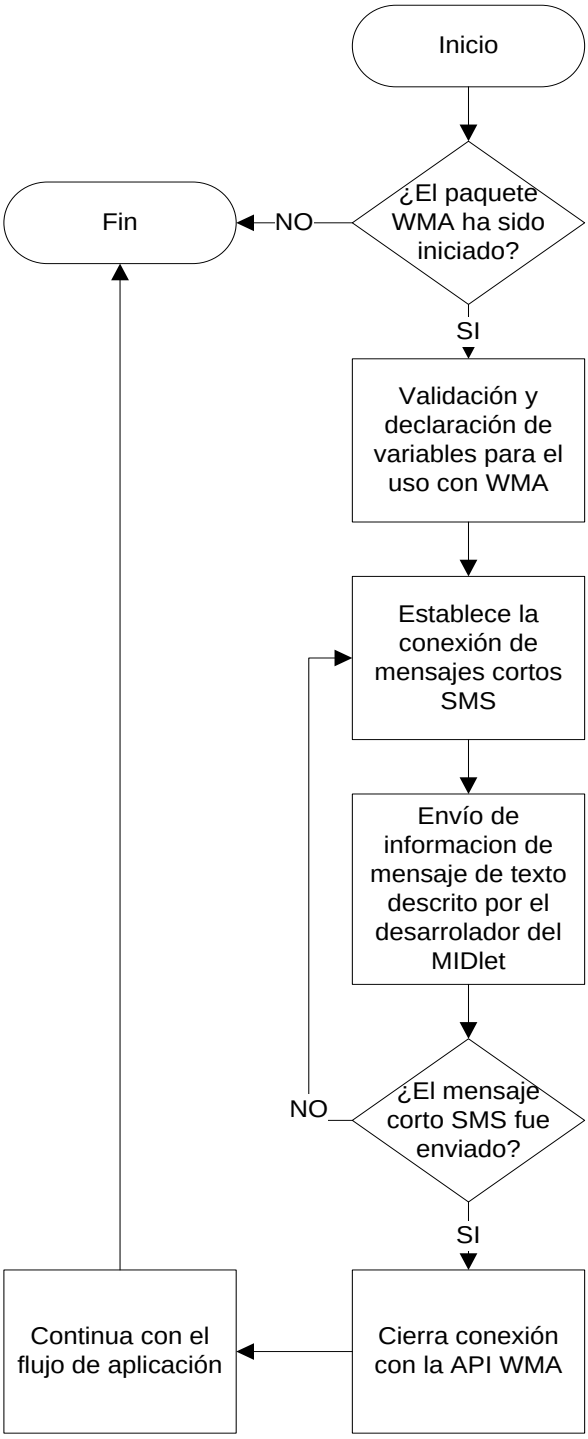


Figura. 3.33. Diagrama de flujo – Uso de WMA

Es de suma importancia insistir que la librería de WMA debe estar inicializada he importada para su uso, contiene también variables que deben ser declaradas para que en el

código de programa no se confundan con las declaradas automáticamente por el IDE como se muestra a continuación

Código de programa JAVA – Implementación de WMA

```
//Importa WMA
import javax.wireless.messaging.*
//Describe numero de celular al cual estará dirigido el envio //de
mensajes de texto asi como también el contenido.
private String slave1cell = "091401623"; //UPC CENTRAL
private String message1 = "@1@4m@";

//Implementacion de funcion para el manejo de WMA, llevándose //como
parametros el número telefonico y el contenido del //mensaje de
texto
public boolean sendSMS(String number, String message) {
boolean result = true;
try {
  //Especifica la direccion para el envio de SMS
  String addr = "sms://"+number;
  //Realiza la conexion con WMA
  MessageConnection conn = (MessageConnection)
  Connector.open(addr);
  //Puesta de informacion del mensaje de texto
   TextMessage msg =
 (TextMessage)conn.newMessage(MessageConnection.TEXT_MESSAGE);
  msg.setPayloadText(message);
  //Envio de mensaje SMS
  conn.send(msg);
  //Cierra la conexion con WMA
  conn.close();
}
catch (SecurityException e) {result = false;}
catch (Exception e) { result = false; }
//Devuelve el resultado de WMA
return result;
}
```

De esta manera queda detallado el diseño del MIDlet para el uso en los dispositivos móviles que soportan aplicaciones JAVA de cada uno de los usuarios de la comunidad que requieran comunicarse de manera rápida y sencilla con el UPC del barrio, es decir el SERVlet.

CAPITULO IV

PRUEBAS Y RESULTADOS EXPERIMENTALES

4.1. PRUEBAS Y ANALISIS DE COMUNICACION GSM

Dentro del capítulo descrito a continuación es importante conocer las pruebas que se deben hacer para que las notificaciones de alarma de los usuarios lleguen a su destino, el cual es el UPC del barrio. Para esto se consiguió hacer las pruebas con el software descrito anteriormente en el capítulo 3 sección 3.1.3, el cual fue necesario para conocer las tramas que usa el modem GSM para ser configurado y poder de esa manera tener implementado dentro del SERVlet la comunicación con el mismo.

Para tener la percepción de cómo funciona la comunicación serial dentro del SERVlet se detalla a continuación las pruebas y el análisis realizado junto con el software *Free Serial Port Monitor.*

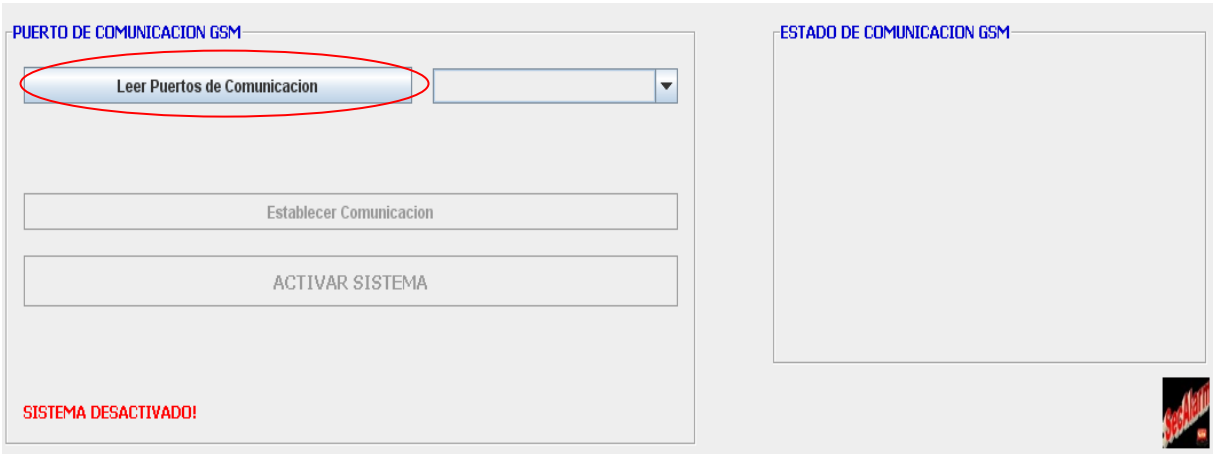


Figura. 4.1. SERVlet – Lectura de puertos de comunicación

Una vez que el operador del SERVlet se haya registrado tendrá acceso a la manipulación de la aplicación, una de ellas es la lectura de puertos como se muestra en la Figura 4.1, la cual es analizada en la Figura 4.2.

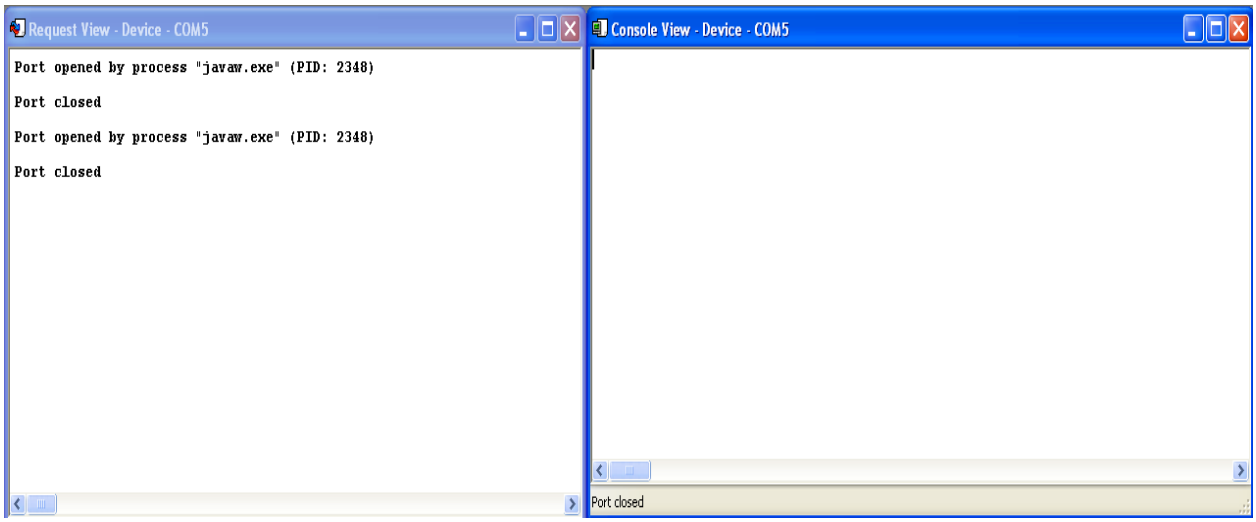


Figura. 4.2. Free Serial Port Monitor – Lectura de puertos del PC

Cuando se obtenga la lectura de puertos del PC todos los puertos del PC serán desplegados en el SERVlet para que el administrador haga uso de los mismos, seleccionando así el puerto en el cual el modem GSM esté conectado y realizando la conexión como se muestra en la Figura 4.3 junto con su análisis en la Figura 4.4.

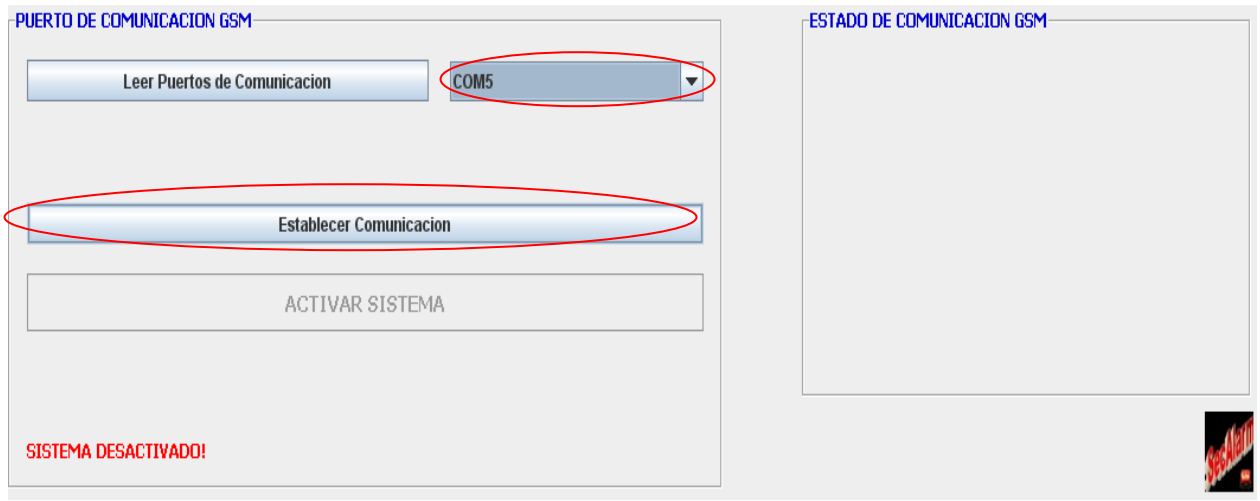


Figura. 4.3. SERVlet – Selección de puerto serial y establecer comunicación con el modem GSM

Cuando el operador realiza estos pasos obtiene notificaciones del estado de comunicación GSM dentro del SERVlet como se muestra en la Figura 4.5.

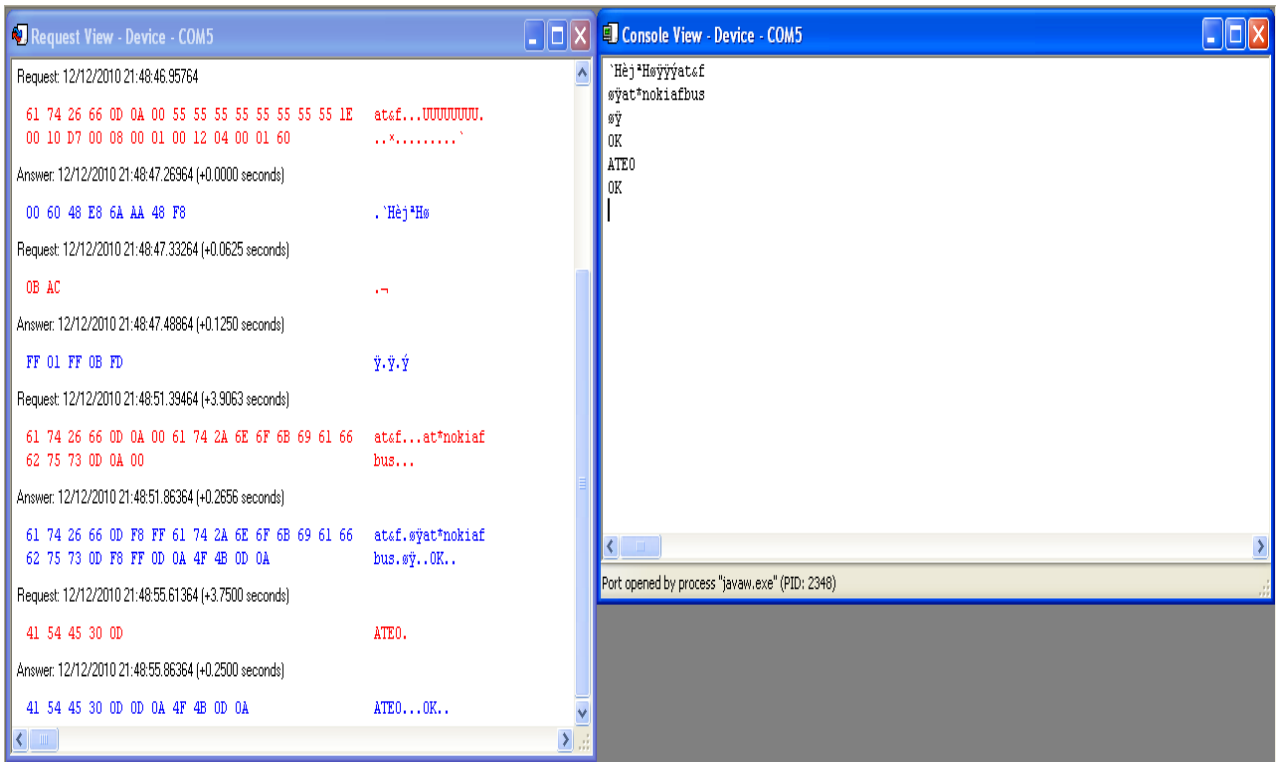


Figura. 4.4. Free Serial Port Monitor – Establecer comunicación con el modem GSM

En la Figura 4.4 se percibe el envío de la trama de comunicación, además de la apertura de puertos del modem GSM junto con la configuración debida enviada por el operador para así recibir respuesta del mismo.

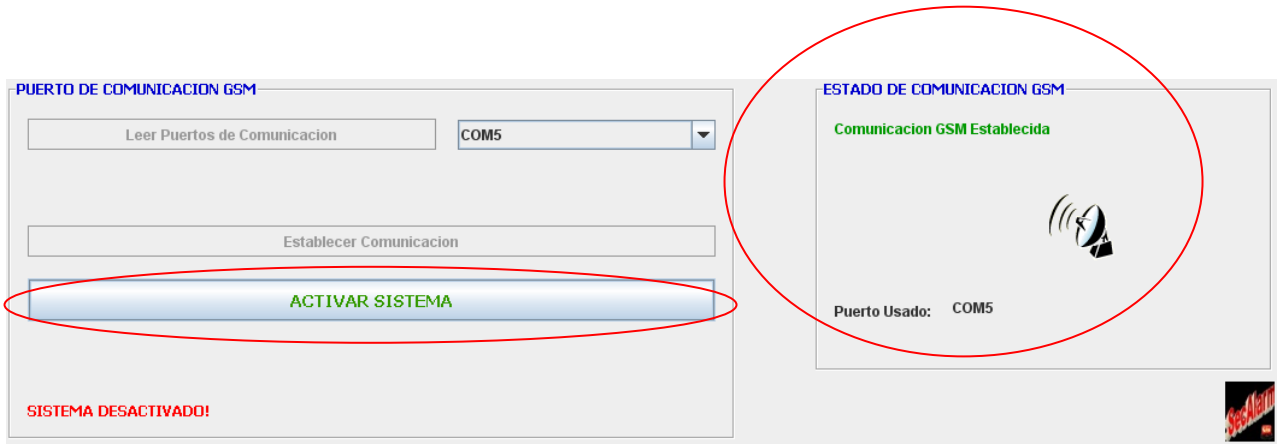


Figura. 4.5. SERVlet – Notificación del modem GSM, Estado de comunicación.

Cuando el operador recibe la notificación tendrá acceso a la activación del sistema, para así poder recibir satisfactoriamente las notificaciones provenientes de los usuarios, en la Figura 4.6 se muestra el análisis cuando el botón "ACTIVAR SISTEMA" es presionado obteniendo la notificación respectiva dentro del SERVlet ilustrada en la Figura 4.7.

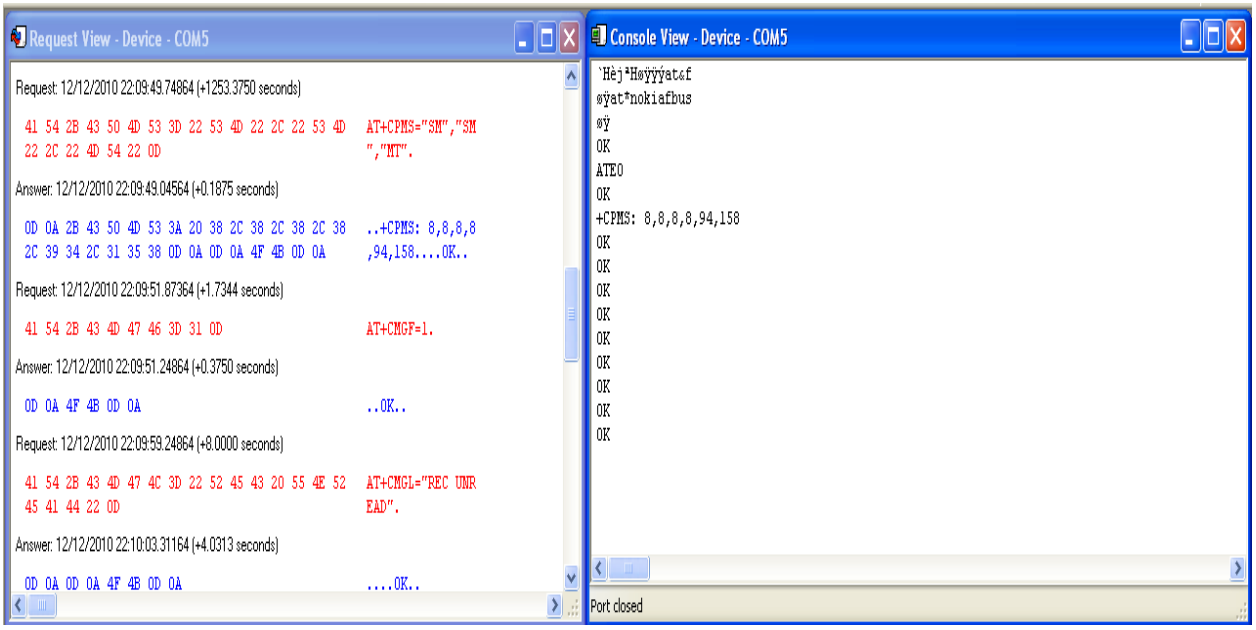


Figura. 4.6. Free Serial Port Monitor – Activación del sistema

En la Figura 4.6 se muestra también el envío periódico de los comandos AT&AT+ para que el modem GSM esté en constante lectura de mensajes provenientes de los usuarios.

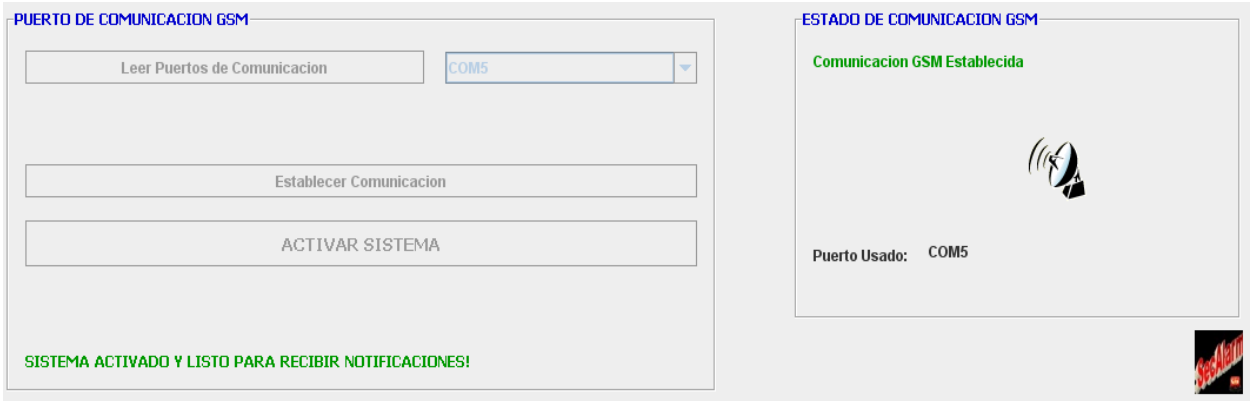


Figura. 4.7. SERVlet – Notifica cuando el Sistema está activado y listo para recibir notificaciones de alarma.

4.2. PRUEBAS Y RESULTADOS DEL SISTEMA

En este capítulo se muestran los resultados obtenidos después de haber ejecutado las pruebas para evaluar el funcionamiento de la conexión del prototipo el cual se compone de lo siguiente:

- MIDlet
- SERVlet
- Integración

4.2.1.MIDlet

Para las pruebas del MIDlet hay que tener en cuenta que la instalación del mismo debe realizarse en cualquier dispositivo que acepte aplicaciones JAVA, para el envió de alarma se debe seguir los siguientes pasos ilustrados en la Figura 4.8.

Figura. 4.8. MIDlet – Pasos para el envío de alarma

Obteniendo como se ilustra en la Figura 4.8 el envío de alarma hacia el UPC del barrio de manera satisfactoria. Para la mejor utilización de la aplicación J2ME hay dispositivos que crean accesos directos para la apertura de aplicaciones de forma rápida.

4.2.2.SERVlet

Para las pruebas y resultados del SERVlet se ha tomado en cuenta el uso de las bases de datos que usa junto con el servidor MySQL, al igual que la elaboración de reportes para la exportación de datos en archivo *Microsoft Excel.*

- **Registro de administradores en la base de datos**

El operador estará encargado de registrar las personas que van a hacer uso del SERVlet, dado que la base de datos no contiene ningún administrador así como también ninguna persona que este viable al uso de la aplicación, para realizar el registro de Administradores dentro de la base de datos se detallan los pasos a continuación.

Primeramente el operador debe dirigirse directamente a la pestaña *"Administrar Base de Datos de Operador"* en donde podrá registrar los operadores o administradores quienes estarán a cargo de la central y también el número de celular al cual la alarma será desviada para la notificación móvil.

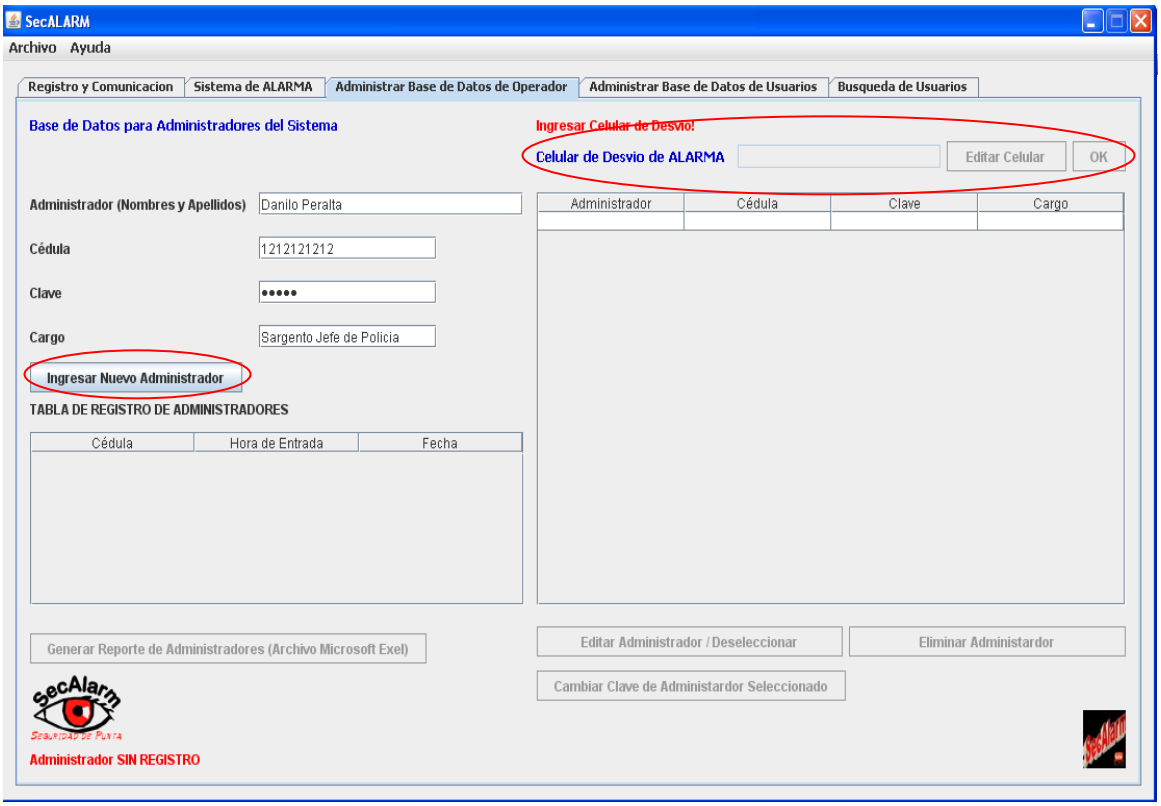


Figura. 4.9. SERVlet – Ingreso de operador y celular de desvío

Tal y como se muestra en la Figura 4.9 el operador podrá registrar y llenar los datos de las personas encargadas de la central, teniendo en cuenta que la cédula tiene 10 dígitos y la clave de Administrador debe tener mínimo 5 caracteres y máximo 10. Luego de verificar que los datos estén correctos ingresamos el nuevo Administrador el cual estará registrado en la base de datos.

Una vez ingresado, la tabla de administradores se actualizará y se verá llena con el número de operadores que se ingresen los cuales estarán a cargo de la central.

Cuando se requiera hacer algún tipo de cambio, lo único que se hace es dar click sobre la tabla seleccionando el operador que se desea modificar, dando tres posibilidades que son: **Editar Administrador/Deseleccionar, Cambiar clave de Administrador Seleccionado y Eliminar Administrador Seleccionado**, como se muestra en la Figura 4.10.

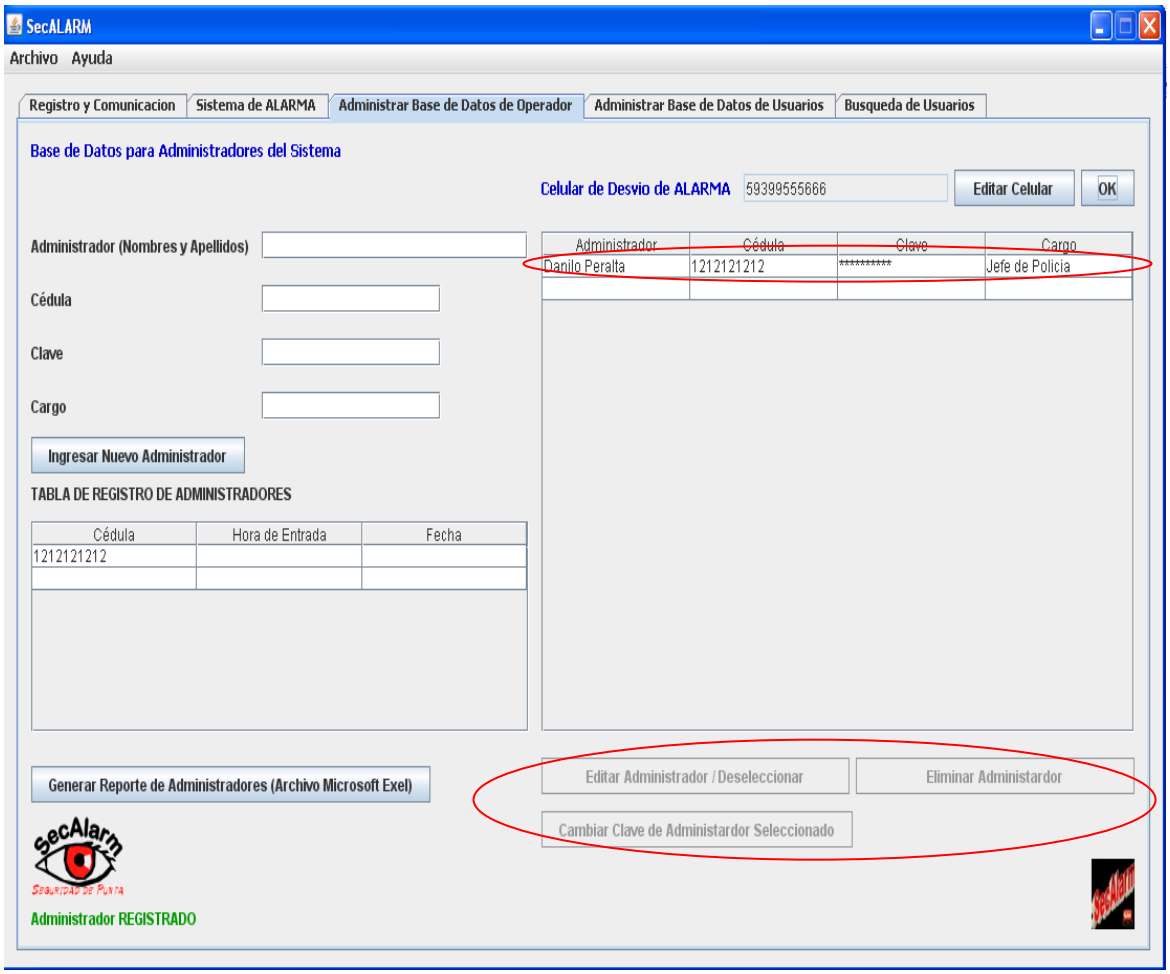


Figura. 4.10. SERVlet – Modificación de datos de administrador

Cuando se tengan todos los operadores dentro de la base de datos automáticamente estarán listos para administrar la aplicación, tomando en cuenta que deben estar registrados los usuarios del barrio y realizar el registro de Fecha y Hora de los operadores cada vez que requieran usar la misma.

- **Registro de usuarios en la base de datos**

El registro de usuarios se lo puede hacer libremente, es decir sin ser operador fijo de la central de notificaciones tomando en cuenta que el usuario debe contar con la aplicación de notificaciones en su celular o dispositivo electrónico.

El operador debe dirigirse directamente a *"Administrar Base de Datos de Usuarios"* en donde podrá registrar los usuarios quienes harán uso de la central para que lleguen sus notificaciones, como se muestra en la Figura 4.11.

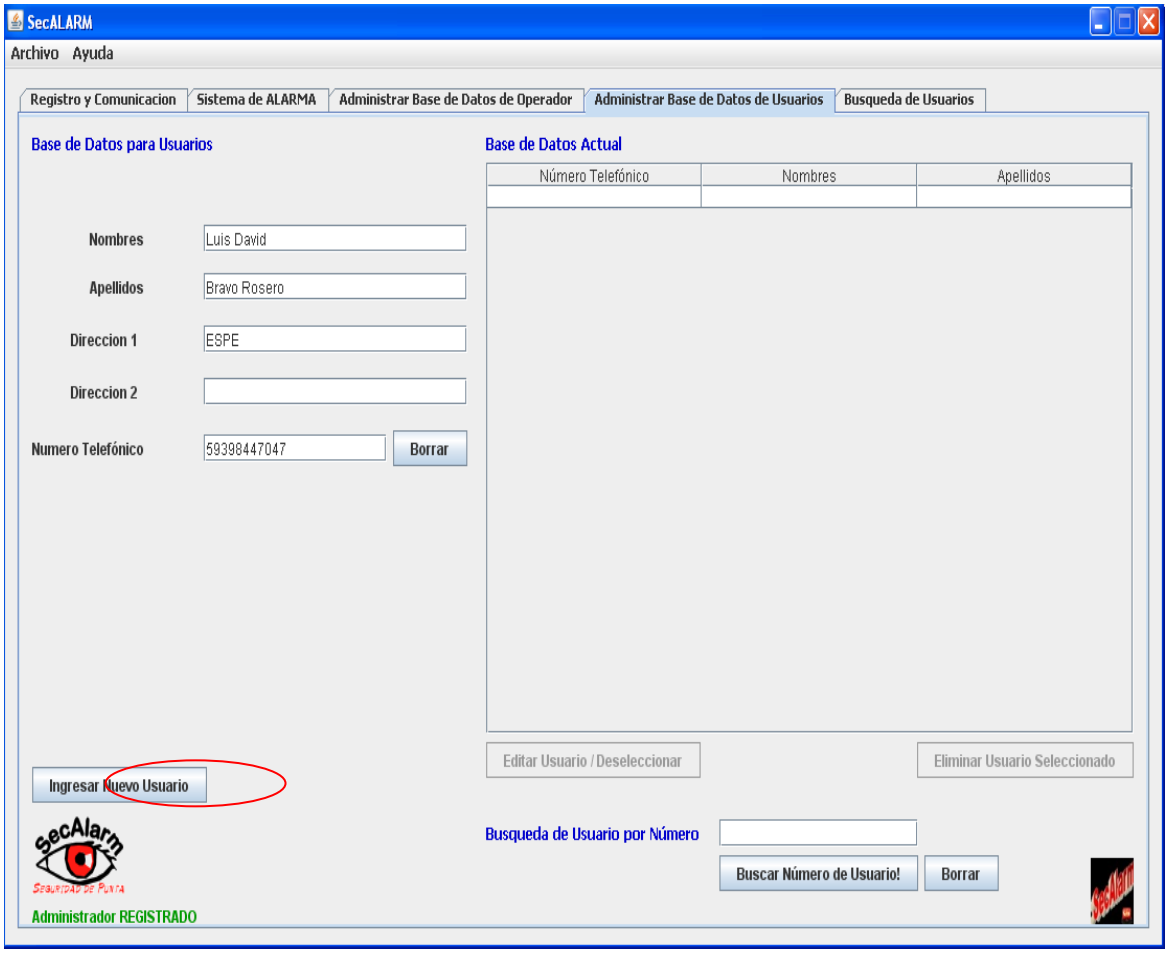


Figura. 4.11. SERVlet – Ingreso de usuarios

La Figura 4.11 especifica también que el operador podrá registrar y llenar los datos de los usuarios, teniendo en cuenta que la *Dirección 1* es la principal y necesariamente la de su domicilio la cual debe ser específica y la *Dirección 2* si requiere el envío de alarma en otro domicilio. Una vez ingresado, la tabla de usuarios se actualizará y se verá llena con el número de usuarios que vayamos a ingresar los cuales harán uso de la central para que lleguen sus notificaciones.

Cuando se requiera hacer algún tipo de cambio, lo único que se debe hacer es dar click sobre la tabla seleccionando el usuario que se desea modificar, dando dos posibilidades que son: **Editar Usuario/Deseleccionar y Eliminar Usuario Seleccionado**, como se muestra en la Figura 4.12.

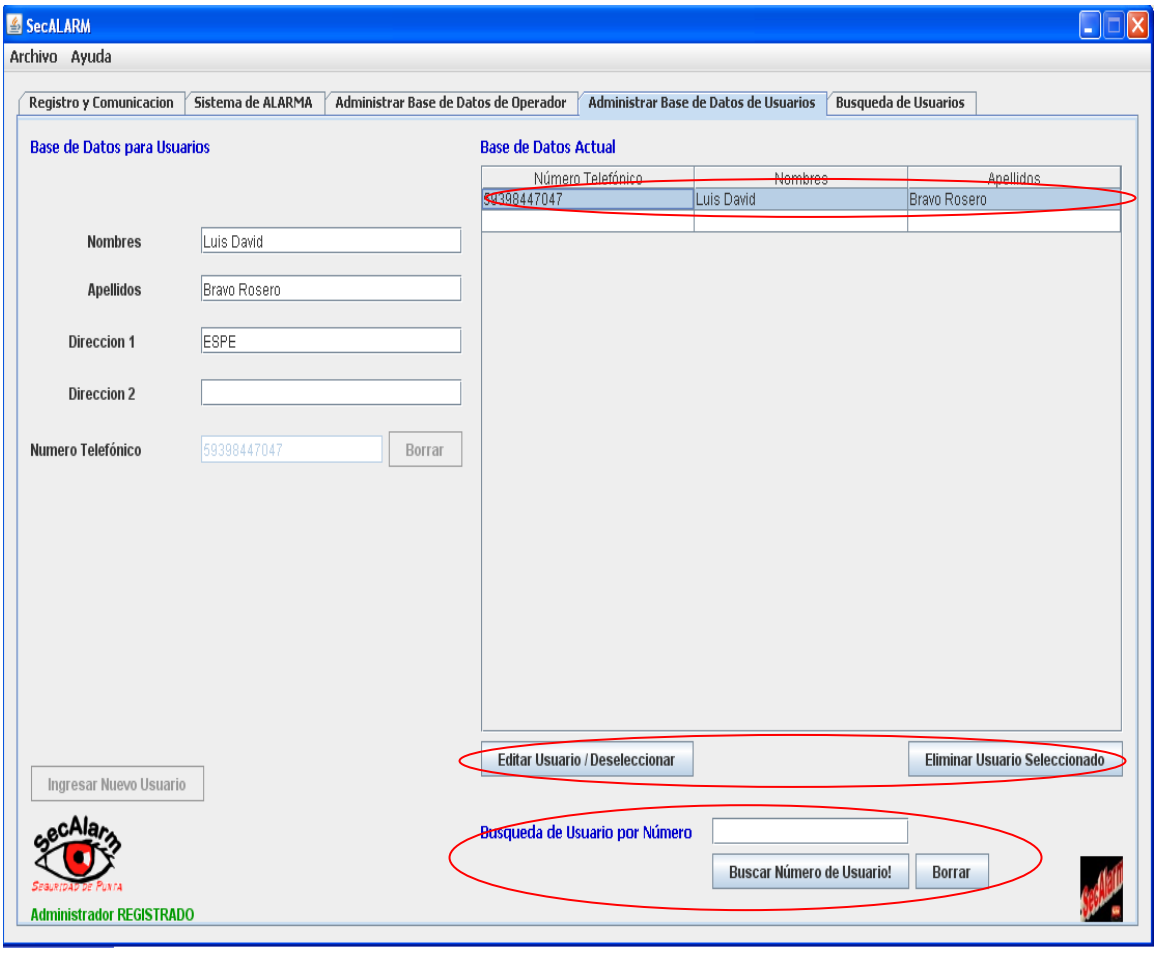


Figura. 4.12. SERVlet – Modificación de datos de usuario

La Figura 4.12 especifica también la búsqueda de usuario por número, lo cual nos permite verificar realizando una búsqueda rápida y saber si el usuario está ingresado en la base de datos. A partir de esto se obtiene los datos necesarios para que la aplicación con su o sus operadores y usuarios pueda funcionar y recibir notificaciones de los mismos, lo cual se especifica de mejor manera en la integración del MIDlet y el SERVlet.

- **Búsqueda de usuarios**

En esta pestaña el operador puede acceder a la base de datos dependiendo el número telefónico del usuario, nombre del usuario o acción tomada por el operador que atendió la alarma, teniendo así toda la información cuando hubo alguna notificación de ayuda. El operador podrá trasladar esta información a un archivo de *Microsoft Excel* generando así un reporte como muestra la Figura 4.13.

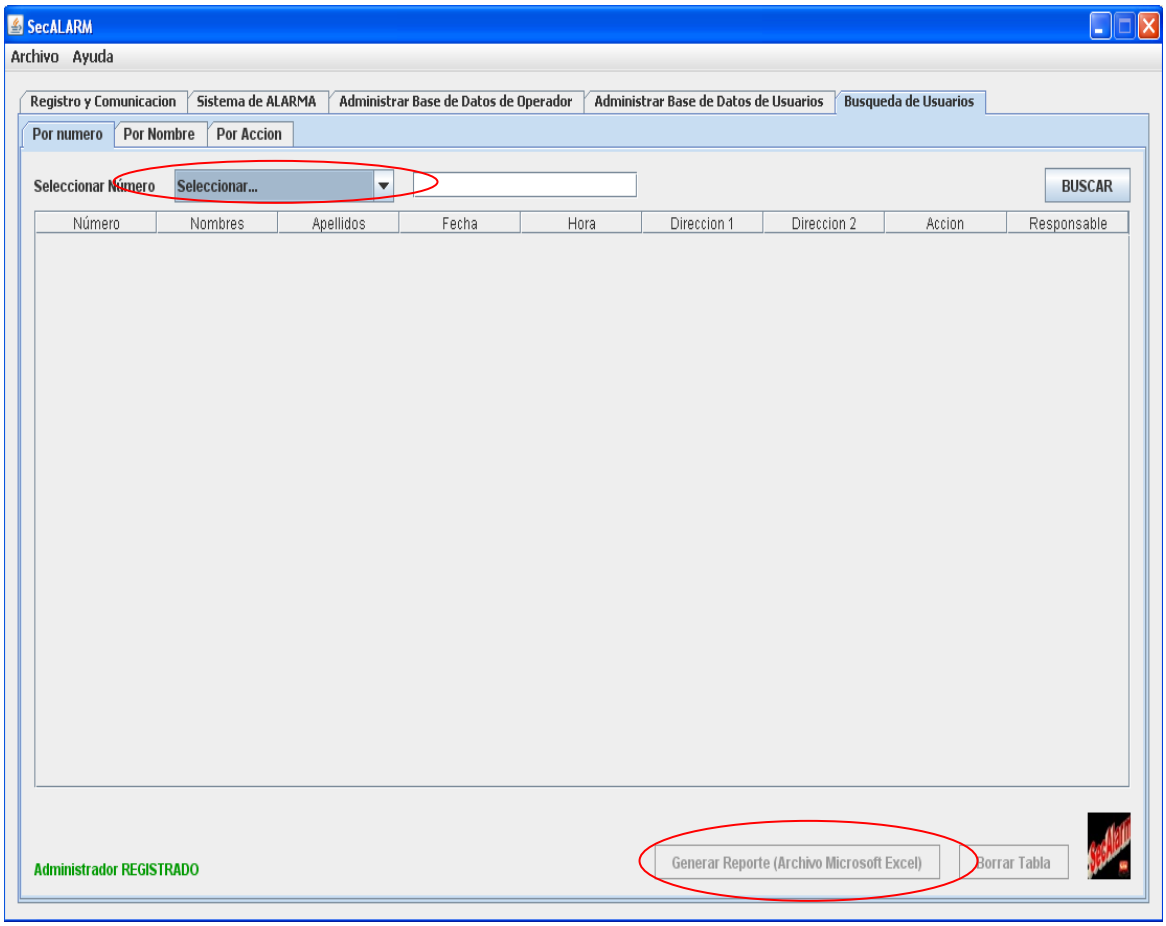


Figura. 4.13. SERVlet – Búsqueda de acciones de usuarios y generación de reportes Microsoft Excel

Cabe recalcar que el reporte exportador desde la aplicación tendrá un lugar en el disco duro de la central el cual será guardado en la siguiente dirección

C:\SecAlarm\Reportes\Reporte.xls

Si el administrador escoge la opción *"GENERAR REPORTE (ARCHIVO MICROSOFT EXCEL)"* aparecerá un aviso diciendo que el reporte ha sido generado como se muestra en la Figura 4.14

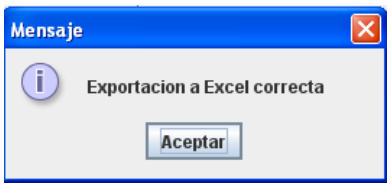


Figura. 4.14 SERVlet - Generación de reporte *Microsoft Excel*

A partir de este mensaje, el operador deberá trasladarse a la dirección antes mencionada para ubicar el archivo de Microsoft Excel como ilustra la Figura 4.15.

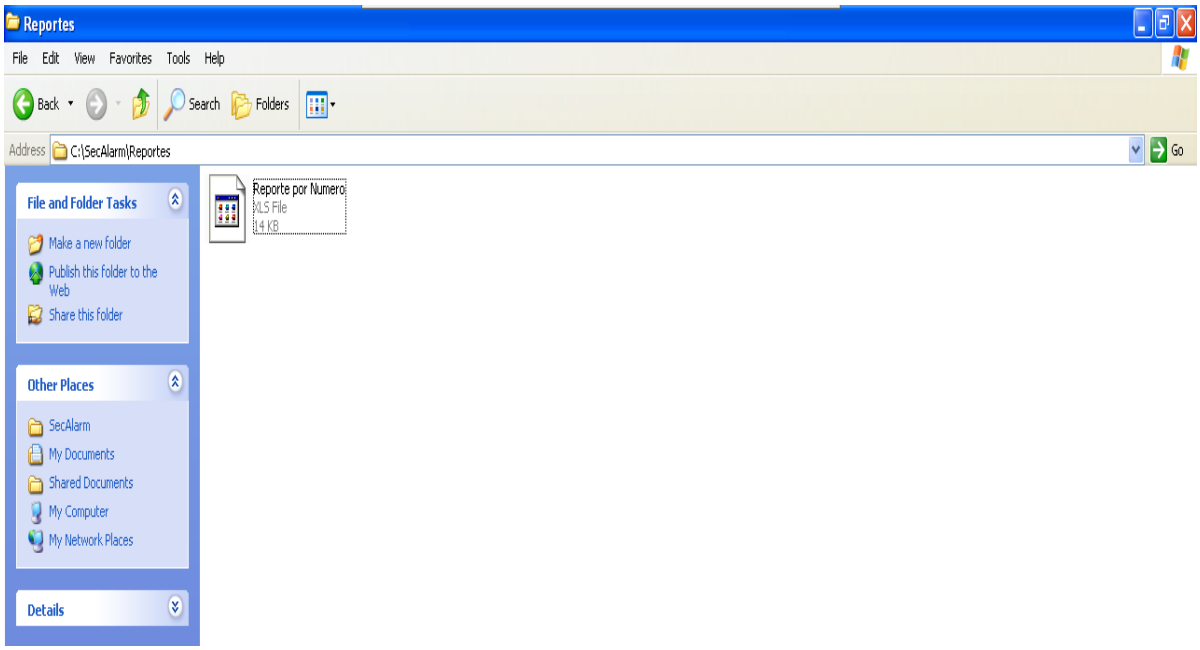


Figura. 4.15 SERVlet - Búsqueda de reporte exportado

4.2.3.Integración

La integración del prototipo se la realizó cuando se obtuvo la implementación del mismo en la comunidad del barrio Central en Sangolquí. Se detalla a continuación el envío de una notificación de alarma recibida en el SERVlet instalado en el UPC Central.

Para el sistema de alarma es de suma importancia que la comunicación con el dispositivo GSM se haya realizado satisfactoriamente, caso contrario la central no estará apta para recibir notificaciones de ayuda de los usuarios, es por eso que el *"ESTADO DEL SISTEMA"* se refleja en un panel al lado derecho anunciando si la comunicación esta o no está hecha. Cuando la comunicación NO está establecida la pantalla de Sistema de ALARMA se ve como se muestra en la Figura 4.16.

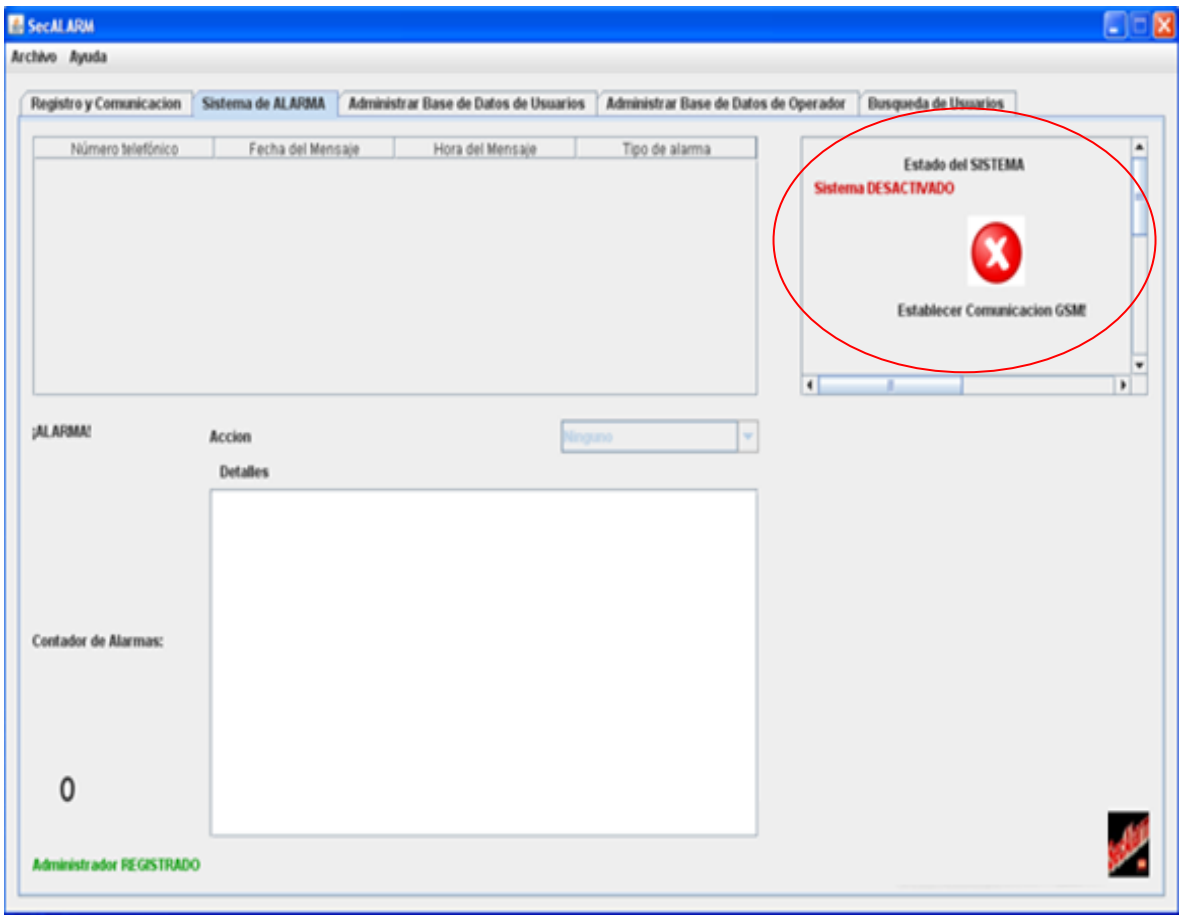


Figura. 4.16. SERVlet - Sistema de Alarma SIN comunicación GSM

Para cuando se haya realizado de manera correcta la comunicación y activado el sistema satisfactoriamente para recibir las notificaciones la pantalla se verá como se muestra en la Figura 4.17.

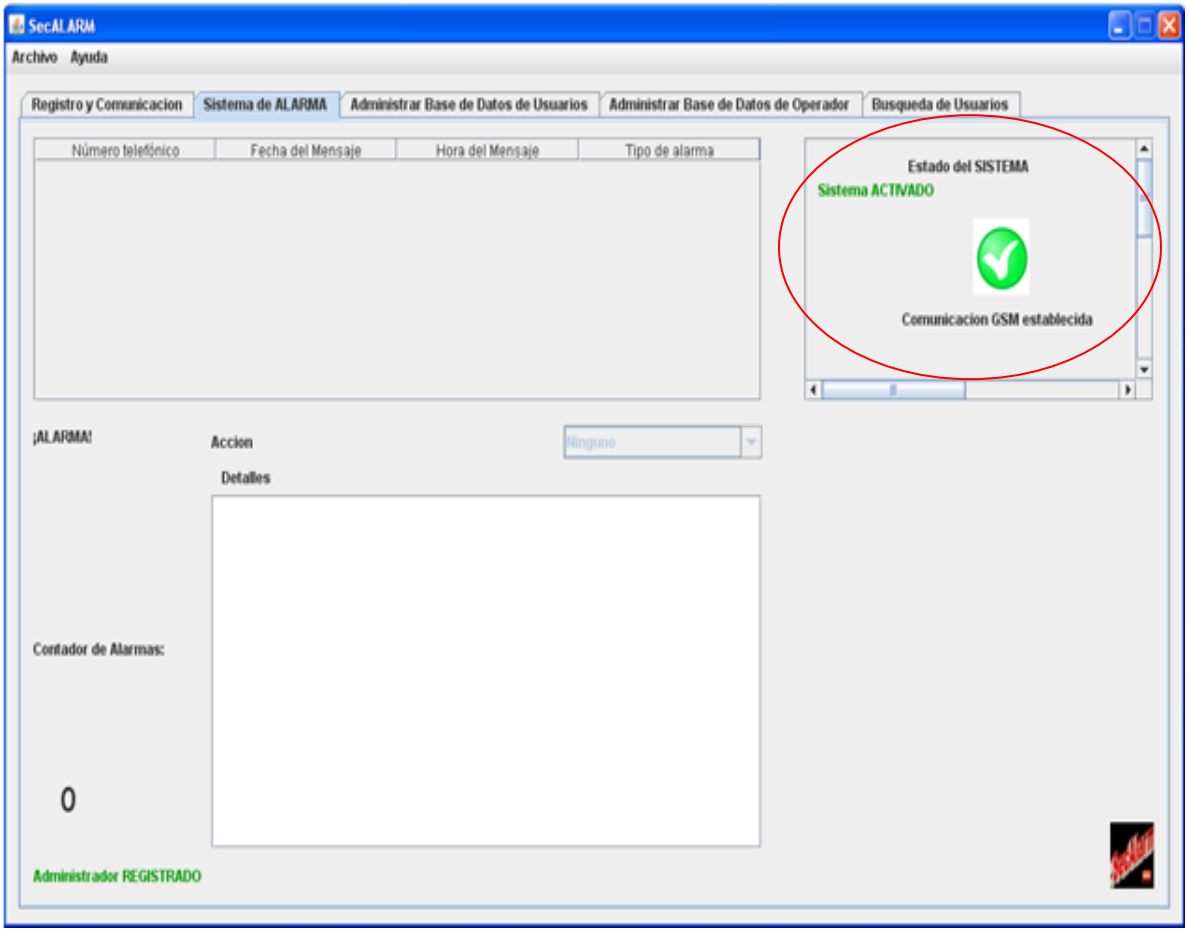


Figura. 4.17. SERVlet - Sistema de Alarma CON comunicación GSM

Cuando todos los pasos anteriores hayan sido correctamente hechos y sin algún contratiempo el operador estará listo para recibir las notificaciones de los usuarios de ***SecAlarm*** sin tener que modificar ni tocar ningún botón, cuando existiese alguna notificación enseguida la tabla se llenará con los datos de la persona que pide ayuda, así como también sonará una alarma.

El operador en su potestad sabrá como desenvolverse al rato de recibir una notificación, para esto dará click sobre la tabla escogiendo así la notificación de la persona que pide ayuda, enseguida en *"DETALLES"* se le desplegará todos los datos de la persona que previamente tuvo que haber sido registrada. Para esto el operador moverá sus unidades y tomara diferentes acciones sobre la notificación teniendo tres: **FALSA ALARMA, EN PROGRESO y ATENDIDO**, como se muestra en la Figura 4.18.

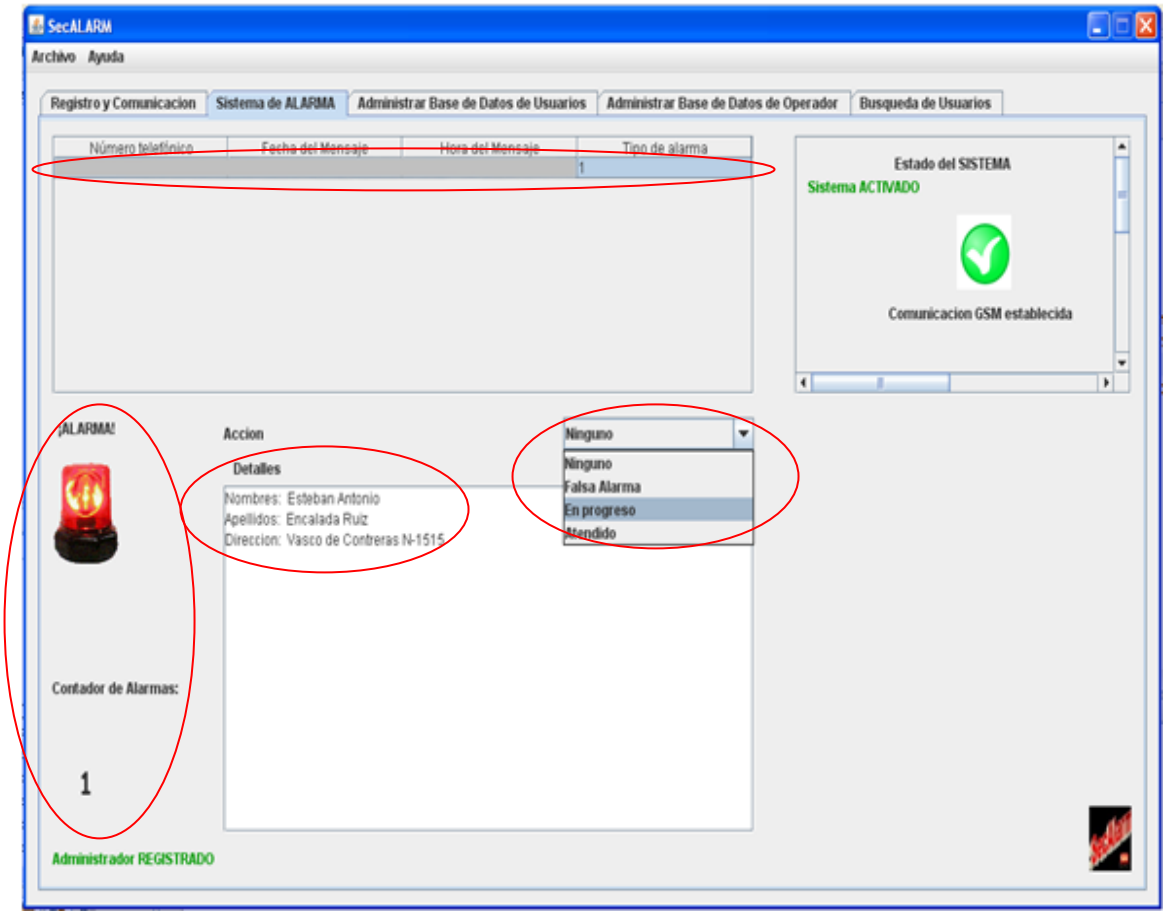
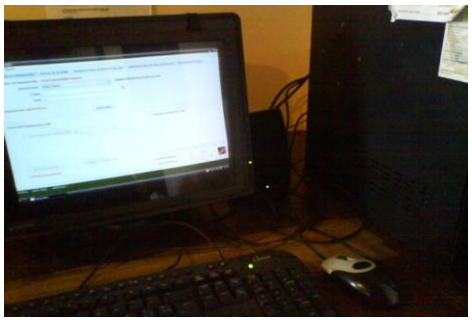


Figura. 4.18. SERVlet - Sistema de Alarma cuando recibe una notificación y muestra detalles de la persona que requiere de ayuda, listo para tomar alguna acción sobre la notificación

Con esto la persona que envía la notificación estará recibiendo las unidades que el operador envía, o aceptando la potestad del operador frente a un llamado de ayuda, siendo de gran beneficio y rapidez para algún tipo de emergencia domiciliaria, obteniendo resultados extraordinarios al hacer uso del sistema en conjunto con la comunidad, como se muestra en secuencia en la Figura 4.19.



Instalación del Sistema

Capacitación al personal

Uso de la aplicación

Simulacro de notificación de alarma

Figura. 4.19. Pruebas de la Implementación del Sistema de Alarmas

CAPITULO V

ANALISIS FINACIERO

5.1. ANALISIS DE VARIABLES

El análisis financiero del proyecto constituye la técnica matemático-financiera y analítica, a través de la cual se determinan los beneficios o pérdidas, en donde uno de sus objetivos es obtener resultados que apoyen la toma de decisiones referente a actividades de inversión.

Asimismo, el poder de analizar el proyecto como una forma de inversión en donde se determinan los costos de oportunidad en que se incurre al invertir al momento para obtener beneficios al instante, mientras se sacrifican las posibilidades de beneficios futuros, o si es posible privar el beneficio actual para trasladarlo al futuro, al tener como base especifica a las inversiones.

Una de las evaluaciones que deben de realizarse para apoyar la toma de decisiones en lo que respecta a la inversión de un proyecto, es la que se refiere a la evaluación financiera, que se apoya en el cálculo de los aspectos financieros del proyecto.

Sus fines son, entre otros:

a. Establecer razones e índices financieros derivados del balance general.
b. Calcular las utilidades y pérdidas que se estiman obtener en el futuro, a valores actualizados.

c. Determinar la tasa de rentabilidad financiera que ha de generar el proyecto, a partir del cálculo e igualación de los ingresos con los egresos, a valores actualizados.
d. Establecer una serie de igualdades numéricas que den resultados positivos o negativos respecto a la inversión de que se trate.

En base a estos fines se engloba el análisis financiero del proyecto a variables que ayudan a saber si el mismo es de utilidad o no, las variables que se analizan son:

- **Tasa interna de retorno (TIR).-** Se utiliza para decidir sobre la aceptación o rechazo de un proyecto.
- **Valor actual neto (VAN).-** Es un procedimiento que permite calcular el valor presente de un determinado número de flujos de caja futuros, originados por una inversión dentro de un proyecto.
- **Tasa de interés de oportunidad (TIO).-** Es la tasa más alta que un inversionista sacrifica con el objeto de realizar un proyecto.
- **Costo de oportunidad.-** Es la máxima tasa que se podría obtener si los recursos se invirtieran en la alternativa más rentable.
- **Tasa de descuento.-** Tasa de interés a la cual son puestas en valor presente las sumas futuras.

5.1.1.Tasa interna de retorno (TIR)

Como se mencionó anteriormente el TIR es una de las variables más importantes dado que es aquella que se relaciona con la apertura o realización del proyecto, es decir si se obtiene mayor TIR mayor será la rentabilidad que se logra del proyecto. Es de suma importancia recalcar que para el proyecto no es necesaria una inversión tan alta con respecto al funcionamiento y provecho del mismo, al igual que la ganancia e ingresos que se obtienen.

Dentro del proyecto se calculó el TIR con referencia al total de ingresos y egresos de cada año respectivamente durante 5 años, tomando en cuenta la tasa de descuento para actualización que es una constante de 12%. Dentro del análisis se toma en cuenta que el TIR

debe ser mayor o igual a un 20% en proyectos de telecomunicaciones para que tengan aceptación de inversión, obteniendo de todas estas variables un valor de TIR favorable para la realización del proyecto de alarmas comunitarias a través de un operador móvil de un 92.3%, como se muestra en la Tabla 5.2.

5.1.2.Valor actual neto (VAN)

El valor actual neto es muy importante para la valoración de inversiones en activos fijos, a pesar de sus limitaciones en considerar circunstancias imprevistas o excepcionales de mercado. Si su valor es mayor a cero, el proyecto es rentable, considerándose el valor mínimo de rendimiento para la inversión. Una empresa suele comparar diferentes alternativas para comprobar si un proyecto le conviene o no. Normalmente la alternativa con el VAN más alto suele ser la mejor para la entidad; pero no siempre tiene que ser así. Puede considerarse también la interpretación del VAN, en función de la Creación de Valor para la Empresa como se muestra en la tabla 5.1.

Tabla. 5.1. Creación de Valor para la Empresa - Decisiones con respecto al Valor Actual Neto (VAN)

VALOR	Significado	Decisión a tomar
VAN > 0	La inversión produciría ganancias por encima de la rentabilidad exigida.	El proyecto puede aceptarse
VAN = 0	La inversión no produciría ni ganancias ni pérdidas	Dado que el proyecto no agrega valor monetario por encima de la rentabilidad exigida, la decisión debería basarse en otros criterios, como la obtención de un mejor posicionamiento en el mercado u otros factores.
VAN < 0	La inversión produciría	El proyecto debería rechazarse

	ganancias por debajo de la rentabilidad exigida.	

Para la toma de decisión dentro del proyecto, el valor actual neto VAN en dólares obtuvo 207.642 (ver Tabla 5.2) siendo este un valor viable para la realización del proyecto.

5.1.3.Tasa de interés de oportunidad (TIO)

El valor de TIO va relacionado con cuan oportuno es el proyecto frente a otros y depende de cada inversionista, de acuerdo con la congruencia de utilización de recursos monetarios y generación de riqueza. Entre mejor sea su "otra" alternativa de inversión, mayor será la tasa de oportunidad TIO, esta tasa puede variar en la medida y en el momento en que cambien las oportunidades de inversión.

Para el proyecto en curso el valor TIO se lo puede distinguir a partir de los valores de las variables del TIR y VAN calculadas las cuales son muy favorables, en la Tabla 5.2 se hace referencia a un valor de cien si el valor TIO es bueno o un valor de cero si el valor de TIO es malo para la visión del inversionista.

5.1.4.Costo de oportunidad y Tasa de descuento

El costo de oportunidad de una disposición financiera que tiene varias alternativas, es el valor de la mejor opción no realizada, es decir que hace referencia a lo que un individuo deja de ganar cuando elije una alternativa entre varias disponibles, para el proyecto en curso el inversionista puede observar el costo de oportunidad haciendo referencia los valores de recuperación de inversión en años, siendo de un año y seis meses aproximadamente como se muestra en la Tabla 5.2.

El valor de tasa de descuento hace referencia al costo de oportunidad agregando el costo riesgo país siendo menor a un 16%.

Para la realización del análisis de variables dentro del proyecto se tomó un valor constante del 12% como se muestra en la Tabla 5.2.

5.2. ANALISIS DE SENSIBILIDAD DEL PROYECTO

Para el análisis financiero general del proyecto se realizó una tabla la cual contiene el análisis de variables respectivo dentro de 5 años, tomando en cuenta la inversión, costos, gastos y beneficios del proyecto.

Contempla también el detalle de inversión para que la empresa sea constituida de manera formal, a continuación se muestra el análisis de sensibilidad del proyecto

Tabla. 5.2. Análisis de Sensibilidad del Proyecto

ANALISIS DE SENSIBILIDAD DEL PROYECTO

PROYECTO ALARMAS COMUNITARIAS A TRAVES DE UN OPERADOR MÓVIL

VARIABLES	ESTIMACION DEL FLUJO DE CAJA LIBRE					
	0 2010	1 2011	2 2012	3 2013	4 2014	5 2015
1 Ingresos		**50,400**	**76,800**	**96,000**	**115,200**	**134,400**
Instalación		36,000	57,600	72,000	86,400	100,800
Mantenimiento		14,400	19,200	24,000	28,800	33,600
2 Costos		**-30,720**	**-32,040**	**-33,000**	**-33,960**	**-34,920**
Costos de Explotación						
Servicios Básicos		-1,800	-1,800	-1,800	-1,800	-1,800
Nómina		-26,400	-26,400	-26,400	-26,400	-26,400
2.50% Actualización de Activos		-1,260	-1,920	-2,400	-2,880	-3,360
2.50% Mantenimiento		-1,260	-1,920	-2,400	-2,880	-3,360
MARGEN OPERACIONAL BRUTO		**19,680**	**44,760**	**63,000**	**81,240**	**99,480**
Otros Gastos		**-504**	**-768**	**-960**	**-1,152**	**-1,344**
1% ADMINISTRACIÓN		-504	-768	-960	-1,152	-1,344
3 Gastos no desembolsables		**9,794**	**9,794**	**9,794**	**9,794**	**9,794**
Depreciación equipos		9,794.00	9,794	9,794	9,794	9,794
MARGEN OPER. ANTES DE IMPUESTOS		**28,970**	**53,786**	**71,834**	**89,882**	**107,930**

VARIABLES	ESTIMACION DEL FLUJO DE CAJA LIBRE					
	0 2010	1 2011	2 2012	3 2013	4 2014	5 2015
4 Cálculo de Tasas e Impuestos		**7,887**	**14,215**	**18,818**	**23,420**	**28,022**
0% Licencia		500	500	500	500	500
25% Impuesto a la Renta		7,243	13,447	17,959	22,471	26,983
0.5% SuperIntendencia de Compañías		145	269	359	449	540
MARGEN OPER. DESPUES DE IMPUESTOS		**36,857**	**68,001**	**90,652**	**113,302**	**135,952**
5 Ajuste por Gastos no desembolsables		**-9,794**	**-9,794**	**-9,794**	**-9,794**	**-9,794**
Depreciación por equipamiento		-9,794	-9,794	-9,794	-9,794	-9,794
6 Costos y Beneficios no afectos a Impuestos	**-51,970**	**-1,400**	**-1,455**	**-1,746**	**-1,812**	**-1,885**
Inversiones	**51,970**	**-1,400**	**-1,455**	**-1,746**	**-1,812**	**-1,885**
Equipos	48,970	-	-	-	-	-
Equipos de desarrollo para aplicaciones (5)	10,000	-	-	-	-	-
Modems GSM (300)	36,000	-	-	-	-	-
Cables de transmision de datos (300)	1,200	-	-	-	-	-
Impresora Láser	350	-	-	-	-	-
Reguladores (10)	120	-	-	-	-	-
UPS	800	-	-	-	-	-
Otros	500	550	605	666	732	805

VARIABLES	ESTIMACION DEL FLUJO DE CAJA LIBRE					
	0 2010	1 2011	2 2012	3 2013	4 2014	5 2015
Servicios						
Renta de oficina	800	800	800	1,000	1,000	1,000
Muebles de oficina	1,200	-	-	-	-	-
Pagina WEB de la empresa	1,000	50	50	80	80	80
Valor Residual Inversión	-	-	-	-	-	-
Total **Valores en USD**	**-51,970**	**25,663**	**56,752**	**79,112**	**101,696**	**124,273**

Tasa de descuento para actualización **12.00%**

Tasa Interna de Retorno (TIR)	**92.3%**
Valor Actual Neto (VAN) (US$)	**207,642**
Periodo de Recuperación (Años)	**1.46**
Periodo Recuperación Descontado (Años)	**1.71**
Tasa de interés de Oportunidad (TIO)	**100.00**

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

- Se comprobó que el diseño y desarrollo tanto del SERVlet y MIDlet se adaptan de forma amigable, confiable, robusta y totalmente fiable al medio para implementar un sistema de alarmas comunitarias a través de un operador móvil.

- Con la investigación y adquisición de información se puede comprobar que las tecnologías usadas para este proyecto tanto en hardware como software facilitan la recepción y envió de mensajes cortos SMS.

- Con la adquisición del dispositivo modem GPRS/GSM que trabaja con cualquier operador móvil en el ecuador se pudo conocer, procesar y analizar los datos que entrega para así poder manipularlos y familiarizarlos con el protocolo de comunicación que maneja.

- Se desarrolló el SERVlet de tal manera que pueda controlar y manipular la información proveniente de los usuarios para que automatice la comunicación entre las estaciones móviles y un computador central.

- Java es un lenguaje de programación multiplataforma lo que conlleva a que el desarrollo del SERVlet sea independiente del sistema operativo en donde requiera ser instalado.

- El diseño y desarrollo de base de datos con MySQL define implícitamente la iteración con la MIDlet y el SERVlet para su sencillo uso dentro del sistema.

- Se concluye que el uso de bases de datos con MySQL define un sistema robusto y fiable para que la información no sea fácilmente manipulada por usuarios externos al sistema.

- El uso de la tecnología de desarrollo celular MIDP/CLDC facilitó en un 80% el diseño del MIDlet mediante diagramas de flujo y analizadores de dispositivos celulares el cual que permitió la interacción de un equipo móvil con el usuario para así poder enviar las notificaciones de alarmas usando mensajes cortos SMS.

- Para la selección del modem GSM se hicieron algunas pruebas con diferentes dispositivos móviles. El modem GPRS/GSM F1103 ha sido elegido gracias a que cumple con los requisitos para la interacción con el sistema diseñado.

- Con la adquisición del modem GPRS/GSM F1103 se evitó cualquier tipo de inconveniente para la instalación y manipulación del mismo en cualquiera de los sistemas operativos.

- El uso de las instrucciones para comandos AT son palabras codificadas que conforman un lenguaje de comunicación entre el hombre y un Modem. La implementación de los comandos AT corre a cuenta del modem GPRS/GSM usado en el proyecto más no de la red de comunicaciones GSM.

- La transmisión de un mensaje de texto SMS se hace en un solo sentido, es decir solamente desde el MIDlet hacia la estación de base de datos SERVlet, mas no en sentido contrario.

- Se comprobó que el SERVlet después de no haber atendido una notificación de alarma realiza el desvió de información al teléfono celular del jefe de UPC para que así pueda la alarma ser atendida en el menor tiempo posible, siempre y cuando la línea del modem GPRS/GSM este activa y con fondos para realizar dicho desvío.

- Los resultados contribuyeron para que los usuarios de la comunidad usen de manera sencilla la aplicación desarrollada tanto como para el UPC y los dispositivos móviles.

- El proyecto contempló también la implementación de todos los equipos necesarios en la UPC del Barrio central de Sangolquí para el uso del sistema, permitiendo de tal manera las pruebas de evaluación necesarias de las cuales se obtuvieron excelentes resultados.

- Se realizaron simulacros dentro de la comunidad en donde se pudo estimar un tiempo de llegada de alarma al SERVlet de aproximadamente quince segundos y un tiempo de respuesta del personal policial aproximado de dos a tres minutos, siendo estos tiempos cortos de respuesta para evitar la inseguridad en el barrio.

- Se concluye que la utilización del modem GPRS/GSM es de gran utilidad dado que se obtuvieron resultados excelentes cuando las notificaciones de alarma llegaban de manera simultánea.

6.2. RECOMENDACIONES

- La manipulación y manejo del SERVlet debe ser realizado por el personal que fue certeramente capacitado para que así no se tenga problemas de funcionamiento y pérdidas de notificaciones de alarmas.

- Para la instalación del SERVlet se recomienda hacerlo en algún lugar en donde no existan variaciones de señal de la red GSM, de esta manera se evita pérdidas de

notificaciones de alarma provenientes de los usuarios haciendo así que el sistema sea fiable.

- Para la instalación del MIDlet en los dispositivos móviles se recomienda que estos soporten aplicaciones Java *.jar*

- Para establecer una comunicación entre el SERVlet y el modem GPRS/GSM por medio de comandos AT se debe instalar el controlador del cable USB-Serial GUC-232A, de lo contrario se requiere de una tarjeta madre para el computador donde se requiera instalar que posea puerto de comunicación serial con conector DB-9.

- Para el registro de los usuarios de la comunidad en el gestor de base de datos del SERVlet se recomienda que tengan todos los datos necesarios y requeridos para evitar pérdidas de tiempo en la ubicación del domicilio de donde proviene la notificación de alarma.

- Para el buen funcionamiento del SERVlet se recomienda tener una alimentación constante hacia el computador en donde se encuentre instalado, ya que por cada corte de energía el sistema se reinicia y requiere de un operador para su inicialización.

- El modem GPRS/GSM F1103 requiere de una instalación correcta y segura para que no pueda ser manipulado por cualquier persona dentro de la UPC.

- Se recomienda tener una línea activa con saldo permanente en el modem GPRS/GSM para que el desvío de notificaciones de alarma hacia el jefe de la UPC sea confiable.

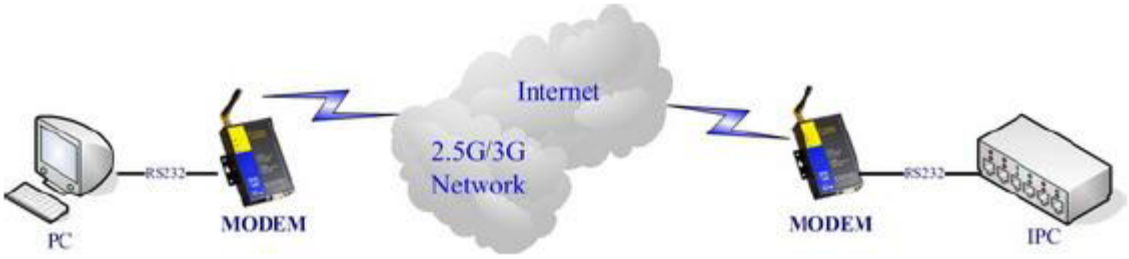
ANEXO 1

HOJA TÉCNICA MODEM GPRS/GSM F1103

Chapter 1 Brief Introduction of Product

1.1 General

EF1103 GPRS MODEM is based on GPRS network and adopts industrial wireless GPRS module. It supports SMS, CSD and data function. It is designed with industrial standard and has been widely used on finance, water supply, environment protection, electric power system, post, weather fields and so on.



1.2 Product Features

- ✧ Designed with industrial standard
- ✧ Support SMS, CSD and data function
- ✧ Support RS232
- ✧ Power range: DC 5~35V

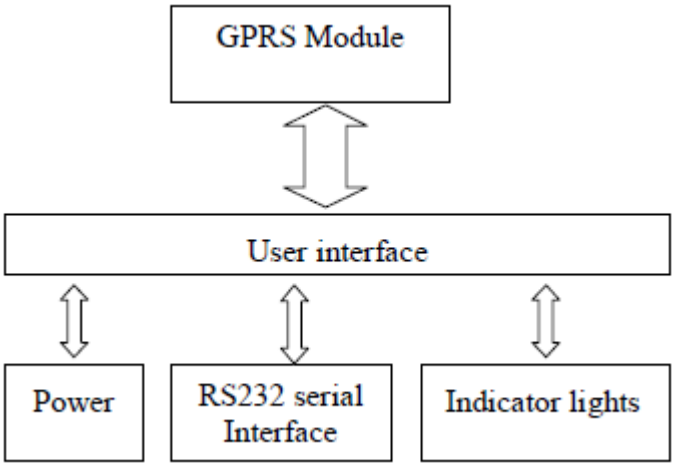
1.3 System Component

The MODEM is composed of the following parts mainly:

- ✧ Industrial GPRS module
- ✧ DC/DC module
- ✧ RS232 module

1.4 Working Principle

The principle chart of the MODEM is as following:



Chapter 2 Installation Introduction

2.1 General

The MODEM must be installed correctly to make it work properly.

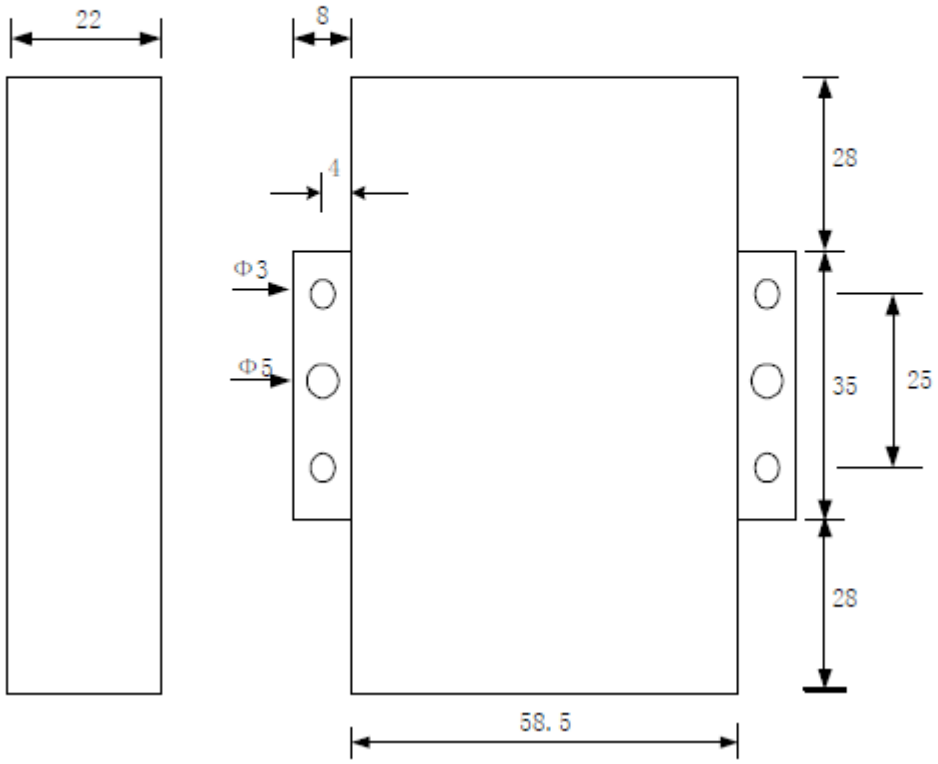
Warning: Forbid to install the MODEM when powered!

2.2 Encasement List

Name	Quantity	Remark
MODEM host	1	
Antenna	1	
Power adapter	1	
RS232 data cable	1	
Manual CD	1	
Certification card	1	
Maintenance card	1	

2.3 Installation and Cable Connection

Dimension: (unit: mm)



Installation of SIM/UIM card:

Firstly power off the MODEM, and press the out button of the SIM/UIM card outlet with a needle object. Then the SIM/UIM card sheath will flick out at once. Put SIM/UIM card into the card sheath (Pay attention to put the side which has metal point outside), and insert card sheath back to the SIM/UIM card outlet.

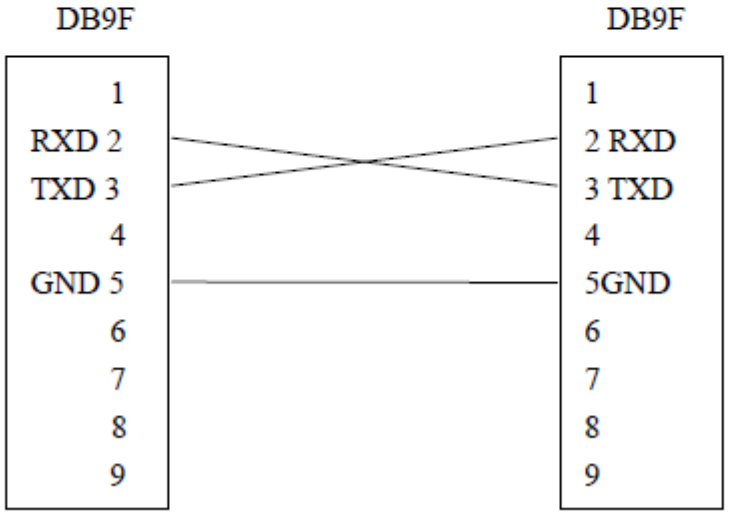
Installation of antenna:

Screw the SMA male pin of the antenna to the female SMA outlet of the MODEM tightly. Warning: The antenna must be screwed tightly, or the signal quality of antenna will be influenced!

Installation of cable:

Insert DB9F end of the RS232 data cable into the DB9M interface of MODEM, and connect the other end with user's device.

The signal connection of the RS232 data cable is as follows:



RS232 data cable

2.4 Power

The power range of the MODEM is 5~35V.

Warning: When we use other power, we should make sure that the power can supply power above 4W.

We recommend user to use the standard DC 12V/0.5A power adaptor.

2.5 Indicator Lights Introduction

The MODEM provides three indicator lights: "Power", "ACT", "Online".

Indicator Light	State	Introduction
Power	ON	MODEM is powered on
	OFF	MODEM is powered off
ACT	BLINK	Data is communicating
	OFF	No data
Online	ON/ BLINK	MODEM has logged on network
	OFF	MODEM hasn't logged on network

Four-Faith 四信

厦门四信通信科技有限公司
Xiamen Four-Faith Communication Technology Co., Ltd.

F1103 GPRS MODEM SPECIFICATION

F1103 GPRS modem is based on 2.5G GPRS network and adopts industrial wireless GPRS module. It supports SMS, CSD and data function. F1103 GPRS modem is designed with industrial standard and has been widely used on finance, water supply, environment protection, electric power system, post and weather fields.

A. Wireless Specification

1. EGSM900/GSM1800
2. GPRS multi-slot class 10
3. Compliant to GSM phase 2/2+
4. Support DTMF
5. Support SMS, CSD and data function
6. Standard AT command interface

B. Hardware System

1. Interface:

   RS232 serial port, rates: 110~230400bits/s

   Indicator lights: "Power", "ACT", "Online"

   Antenna interface: standard SMA female interface, 50 ohm

   SIM/UIM interface: standard user card interface

   Power interface: standard 3-PIN power jack

2. Power supply:

   Standard power supply: DC 12V/0.5A

   Power range: DC 5~35V

3. Wireless module:

   High-powered industrial GPRS module

4. Size:

5. Outline dimension: 91x58.5x22 mm

6. Weight: 630 g

7. Others:

   Operation temperature:-25~+65℃

   Storage temperature: -40~+85℃

   Humidity: 95% (unfreezing)

ANEXO 2

HOJA TÉCNICA DE COMANDOS A&AT+ USADOS EN EL PROYECTO

QUECTEL

M10

Quectel Cellular Engine

AT Commands Set

M10_ATC_V1.01

4 AT Commands According to GSM07.05

The GSM 07.05 commands are for performing SMS and CBS related operations. M10 supports both Text and PDU modes.

4.1 Overview of AT Commands According to GSM07.05

Command	Description
AT+CMGD	DELETE SMS MESSAGE
AT+CMGF	SELECT SMS MESSAGE FORMAT
AT+CMGL	LIST SMS MESSAGES FROM PREFERRED STORE
AT+CMGR	READ SMS MESSAGE
AT+CMGS	SEND SMS MESSAGE
AT+CPMS	PREFERRED SMS MESSAGE STORAGE
AT+QMGDA	DELETE ALL SMS

4.2 Detailed Descriptions of AT Commands According to GSM07.05

4.2.1 AT+CMGD Delete SMS Message

AT+CMGD Delete SMS Message	
Read Command **AT+CMGD=?**	Response **+CMGD:** (Range of SMS on SIM card can be deleted) **OK**
Write Command **AT+CMGD=<index>**	Response TA deletes message from preferred message storage <mem1> location <index>. **OK** **ERROR** If error is related to ME functionality: **+CMS ERROR:<err>**
	Parameter **<index>** integer type; value in the range of location numbers supported by the associated memory
Reference GSM 07.05	Note

4.2.2 AT+CMGF Select SMS Message Format

AT+CMGF Select SMS Message Format	
Read Command **AT+CMGF?**	Response **+CMGF: <mode>** **OK**
	Parameter see Write Command
Test Command **AT+CMGF=?**	Response **+CMGF:** (list of supported **<mode>**s) **OK**
Write Command **AT+CMGF=[<mode>]**	Response TA sets parameter to deNote which input and output format of messages to use. **OK**
	Parameter **<mode>** 0 PDU mode 1 text mode
Reference GSM 07.05	Note

4.2.3 AT+CMGL List SMS Messages From Preferred Store

AT+CMGL List SMS Messages From Preferred Store	
Test Command **AT+CMGL=?**	Response **+CMGL:** (list of supported **<stat>**s) **OK**
	Parameters see Write Command
Write Command **AT+CMGL=<stat>[,<mode>]**	Parameters 1) If text mode: **<stat>** "REC UNREAD" Received unread messages (default) "REC READ" Received read messages "STO UNSENT" Stored unsent messages "STO SENT" Stored sent messages "ALL" All messages **<mode>** 0 normal 1 not change status of the specified SMS record 2) If PDU mode: **<stat>** 0 Received unread messages (default) 1 Received read messages 2 Stored unsent messages

4.2.4 AT+CMGR Read SMS Message

AT+CMGR Read SMS Message	
Test Command **AT+CMGR=?**	Response **OK**
Write Command **AT+CMGR=<index>[,<mode>]**	Parameters **<index>** integer type; value in the range of location numbers supported by the associated memory **<mode>** 0 normal 1 not change status of the specified SMS record
	Response TA returns SMS message with location value <index> from message storage <mem1> to the TE. If status of the message is 'received unread', status in the storage changes to 'received read'. 1) If text mode (+CMGF=1) and Command successful: for SMS-DELIVER: **+CMGR:** **<stat>,<oa>,[<alpha>],<scts>*[,<tooa>,<fo>,<pid>,<dcs>,<sca>,<tosca>,<length>]*<CR><LF><data>** for SMS-SUBMIT: **+CMGR:** **<stat>,<da>,[<alpha>]*[,<toda>,<fo>,<pid>,<dcs>,[<vp>],<sca>,<tosca>,<length>]*<CR><LF><data>**

	for SMS-STATUS-REPORTs: **+CMGR: <stat>,<fo>,<mr>,[<ra>],[<tora>],<scts>,<dt>,<st>** for SMS-COMMANDs: **+CMGR:** ***<stat>,<fo>,<ct>[,<pid>,[<mn>],[<da>],[<toda>],<length><CR><LF><cdata>]*** for CBM storage: **+CMGR: <stat>,<sn>,<mid>,<dcs>,<page>,<pages><CR><LF><data>** 2) If PDU mode (+CMGF=0) and Command successful: **+CMGR: <stat>,[<alpha>],<length><CR><LF><pdu>** **OK** 3) If error is related to ME functionality: **+CMS ERROR: <err>** Parameters **<alpha>** string type alphanumeric representation of <da> or <oa> corresponding to the entry found in MT phonebook; implementation of this feature is manufacturer specific **<da>** GSM 03.40 TP-Destination-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted to characters of the currently selected TE character set (specified by +CSCS in TS 07.07); type of address given by <toda>

	<data> In the case of SMS: GSM 03.40 TP-User-Data in text mode responses; format: - if <dcs> indicates that GSM 03.38 default alphabet is used and <fo> indicates that GSM 03.40 TPUser-Data-Header-Indication is not set: - if TE character set other than "HEX" (refer Command Select TE Character Set +CSCS in TS 07.07):ME/TA converts GSM alphabet into current TE character set according to rules of Annex A - if TE character set is "HEX": ME/TA converts each 7-bit character of GSM alphabet into two IRA character long hexadecimal number (e.g. character P (GSM 23) is presented as 17 (IRA 49 and 55)) - if <dcs> indicates that 8-bit or UCS2 data coding scheme is used, or <fo> indicates that GSM 03.40 TP-User-Data-Header-Indication is set: ME/TA converts each 8-bit octet into two IRA character long hexadecimal number (e.g. octet with integer value 42 is presented to TE as two characters 2A (IRA 50 and 65)) In the case of CBS: GSM 03.41 CBM Content of Message in text mode responses; format: - if <dcs> indicates that GSM 03.38 default alphabet is used:

	- if TE character set other than "HEX" (refer Command +CSCS in GSM 07.07): ME/TA converts GSM alphabet into current TE character set according to rules of Annex A - if TE character set is "HEX": ME/TA converts each 7-bit character of GSM alphabet into two IRA character long hexadecimal number - if <dcs> indicates that 8-bit or UCS2 data coding scheme is used: ME/TA converts each 8-bit octet into two IRA character long hexadecimal number
	<dcs> depending on the Command or result code: GSM 03.38 SMS Data Coding Scheme (default 0), or Cell Broadcast Data Coding Scheme in integer format
	<fo> depending on the Command or result code: first octet of GSM 03.40 SMS-DELIVER, SMS-SUBMIT (default 17), SMS-STATUS-REPORT, or SMS-COMMAND (default 2) in integer format
	<length> integer type value indicating in the text mode (+CMGF=1) the length of the message body <data> (or <cdata>) in characters; or in PDU mode (+CMGF=0), the length of the actual TP data unit in octets (i.e. the RP layer SMSC address octets are not counted in the length)
	<mid> GSM 03.41 CBM Message Identifier in integer format

	<oa>	GSM 03.40 TP-Originating-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted characters of the currently selected TE character set (specified by +CSCS in TS 07.07); type of address given by <tooa>
	<pdu>	In the case of SMS: GSM 04.11 SC address followed by GSM 03.40 TPDU in hexadecimal format: ME/TA converts each octet of TP data unit into two IRA character long hexadecimal number (e.g. octet with integer value 42 is presented to TE as two characters 2A (IRA 50 and 65)). In the case of CBS: GSM 03.41 TPDU in hexadecimal format.
	<pid> 0)	GSM 03.40 TP-Protocol-Identifier in integer format (default
	<sca>	GSM 04.11 RP SC address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are are converted to characters of the currently selected TE character set (specified by +CSCS in TS 07.07);; type of address given by <tosca>
	<scts>	GSM 03.40 TP-Service-Centre-Time-Stamp in time-string format (refer <dt>)

	<stat>	0	"REC UNREAD" Received unread messages
		1	"REC READ" Received read messages
		2	"STO UNSENT" Stored unsent messages
		3	"STO SENT" Stored sent messages
		4	"ALL" All messages
	<toda>	GSM 04.11 TP-Destination-Address Type-of-Address octet in integer format (when first character of <da> is + (IRA 43) default is 145, otherwise default is 129)	
	<tooa>	GSM 04.11 TP-Originating-Address Type-of-Address octet in integer format (default refer<toda>)	
	<tosca>	GSM 04.11 RP SC address Type-of-Address octet in integer format (default refer <toda>)	
	<vp>	depending on SMS-SUBMIT <fo> setting: GSM 03.40 TP-Validity-Period either in integer format (default 167) or in time-string format (refer <dt>)	
Reference GSM 07.05	Note		

4.2.5 AT+CMGS Send SMS Message

AT+CMGS Send SMS Message	
Test Command **AT+CMGS=?**	Response **OK**
Write Command 1) If text mode (+CMGF=1): +CMGS=<da>[,<toda>]<CR> **text is entered** <ctrl-Z/ESC> ESC quits without sending 2) If PDU mode (+CMGF=0): +CMGS=<length><CR>	Parameters **<da>** GSM 03.40 TP-Destination-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted to characters of the currently selected TE character set (specified by +CSCS in TS 07.07); type of address given by <toda> **<toda>** GSM 04.11 TP-Destination-Address Type-of-Address octet in integer format (when first character of <da> is + (IRA 43) default is 145, otherwise default is 129) **<length>** integer type value indicating in the text mode (+CMGF=1) the length of the message body <data> (or <cdata>) in characters; or in PDU mode (+CMGF=0), the length of the actual TP data unit in octets (i.e. the RP layer SMSC address octets are not counted in the length)
PDU is given <ctrl-Z/ESC>	Response TA sends message from a TE to the network (SMS-SUBMIT). Message reference value <mr> is returned to the TE on successful message delivery. Optionally (when +CSMS <service> value is 1 and network supports) <scts> is returned. Values can be used to identify message upon unsolicited delivery status report result code. 1) If text mode(+CMGF=1) and sending successful: **+CMGS: <mr>**
	OK 2) If PDU mode(+CMGF=0) and sending successful: **+CMGS: <mr>** **OK** 3)If error is related to ME functionality: **+CMS ERROR: <err>** Parameter **<mr>** GSM 03.40 TP-Message-Reference in integer format

6.2.25 AT+QMGDA Delete All SMS

AT+QMGDA Delete All SMS	
Test Command **AT+QMGDA=?**	Response **+QMGDA:** (listed of supported <**type**>s) **OK** **+CMS ERROR: <err>**
	Parameter See Write Command
Write Command **AT+QMGDA=<type>**	Response **OK** **ERROR** **+CMS ERROR: <err>**
	Parameter 1) If text mode: **"DEL READ"** delete all read messages **"DEL UNREAD"** delete all unread messages **"DEL SENT"** delete all sent SMS **"DEL UNSENT"** delete all unsent SMS **"DEL INBOX"** delete all received SMS **"DEL ALL"** delete all SMS 3) if PDU mode : **1** delete all read messages **2** delete all unread messages **3** delete all sent SMS **4** delete all unsent SMS **5** delete all received SMS **6** delete all SMS

REFERENCIAS BIBLIOGRÁFICAS

[1] GOLDING, Paul, *Next generation wireless applications*, Tercera Edición, Magic E Company, UK, 2004.

[2] Wikimedia Foundation, Inc., "Global System for Mobile Communications GSM", http://en.wikipedia.org/wiki/GSM, 2010, Fecha de consulta 02-09-2010.

[3] Cellular.co.za, "GSM Technical Data", http://www.cellular.co.za/gsmtechdata.htm, 1996-2004, Fecha de consulta: 06-09-2010.

[4] Wikimedia Foundation, Inc., "Sistema Global para las Comunicaciones móviles", http://es.wikipedia.org/wiki/Sistema_global_para_las_comunicaciones_m%C3%B3viles, 2011, Fecha de consulta 23-12-2010.

[5] "Tecnología GSM", http://www.todo-cel.com.ar/info/gsm.html, 2005, Fecha de consulta: 14-01-2011.

[3] HarmoniousTech Limited, "Short Message Service / SMS tutorial", http://www.developershome.com/sms/, 2004-2010, Fecha de consulta: 09-09-2010.

[4] Syeda Anila Nusrat, "Send and read SMS through a GSM Modem using AT Commands", http://www.codeproject.com/KB/IP/Send_and_Read_SMS.aspx, Agosto 2010, Fecha de consulta: 09-09-2010.

[8] Wikimedia Foundation, Inc., "Hayes commands set", http://en.wikipedia.org/wiki/Hayes_command_set, Hayes command set, 2010, Fecha de consulta: 18-12-2010.

[5] Oracle USA, Inc., "Starting the MySQL Server Instance Configuration Wizard", http://dev.mysql.com/doc/refman/5.0/en/mysql-config-wizard-starting.html, 1997-2010, Fecha de consulta: 08-09-2010.

[6] Oracle USA, Inc., "Connecting to a MySQL Database", http://netbeans.org/kb/docs/ide/mysql.html, 1997-2010, Fecha de consulta: 08-09-2010.

[7] Sun Microsystems, Inc., "Tutorial-Mobility Modules for NetBeans", http://www.scribd.com/doc/7205809/NetBeans-J2ME-Tutorial, 2004, Fecha de consulta: 18-08-2010.

[8] Oracle USA, Inc., "Java ME MIDP Development Quick Start Guide", http://netbeans.org/kb/docs/javame/quickstart.html, 2010, Fecha de consulta: 18-08-2010.

[9] Oracle USA, Inc., "Java ME Technical Documentation" http://download.oracle.com/javame/index.html#api, 2010, Fecha de consulta: 19-08-2010.

[11] García, Carlos, "J2ME, Java Wireless Message API (WMA)", http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=wma, 2007, Fecha de consulta: 20-08-1010.

[15] "Export Table data to an Excel Spreadsheet", http://book.javanb.com/swing-hacks/swinghacks-chp-3-sect-6.html, 2006-2007, Fecha de consulta: 04-08-2010.

[16] Reyes, Rodrigo, "Jsmooth 0.9.9-7 User Manual", http://jsmooth.sourceforge.net/docs/jsmooth-doc.html#N100B0, 2007, Fecha de consulta: 03-02-2011

[17] Oracle USA, Inc., "A Database Backup Program.", http://dev.mysql.com/doc/refman/5.5/en/mysqldump.html, 2010, Fecha de consulta: 06-02-2011.

FECHA DE ENTREGA

El presente proyecto de grado fue entregado en la fecha.

Sangolquí, _____ 2011

Realizado por:

_____

Luis David Bravo Rosero

_____

Ing. Gonzalo Olmedo

COORDINADOR INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES