UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ANÁLISIS Y DISEÑO DE UN SISTEMA DE CONTROL PARA LAS ANTENAS LECTORAS DE RFID MARCA WAVETREND MODELO L-RX201 UTILIZANDO METODOLOGÍA ORIENTADA A OBJETOS.

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO ELÉCTRICO– ELECTRÓNICO

P R E S E N T A:

ROBERTO JESÚS CALVA GARCÍA

DIRECTOR DE TESIS:
ING. CARLOS ALBERTO ROMÁN ZAMITIZ

CIUDAD UNIVERSITARIA 2009

AGRADECIMIENTOS.

- ✔ A **Jesucristo**, por ser siempre mi esperanza y guiar mi camino en esta vida. Gracias por mi destino.

- ✔ A mi abuelito **Jesús✝** por que donde quiera que se encuentre, es nuestro ángel de la guarda. A mi abuelita **Juana**, por sus consejos y enseñanzas.

- ✔ A mis padres: **Nicolás** y **María Elena**:

  - ➔ A mi papá, por ser un ejemplo de tenacidad, fortaleza y lealtad hacia todo. Por enseñarme a que todo en la vida se puede conseguir. Por que siempre ha estado ahí para corregir el camino. Gracias por todo el apoyo y amistad.
  - ➔ A mi mamá, por tener siempre paciencia y amor cuando pequeños. Por enseñarme a sonreír. Por que también me enseñó a que la sensibilidad es parte de la melancolía, pero además es parte de la vida. Gracias por todo.

- ✔ A mi esposa **Nancy**, por ser mi todo. Por enseñarme a sonreír de nuevo y enseñarme a creer en el amor y en la esperanza. Por que cambió mi vida desde que la conocí y ahora es distinta. Por su apoyo y consejos en todo lo que realizo y por ser una mujer fantástica y única, ¡Gracias nanny! ¡Eres el amor de mi vida! Hasta el infinito y más allá!!

- ✔ A mis hermanos: **Haydeé**, **Ricardo** y **Marlene**, por su apoyo y consejos. Por haber disfrutado de una niñez divertida y alegre, llena de sueños, a pesar de todo. Gracias por sus palabras en el momento indicado. A Ricardo le agradezco el estar conmigo en las buenas y en las malas desde pequeños y haber superado diferencias y momentos difíciles que ahora nos han unido mucho más. Gracias gordinflas!

- ✔ A mis sobrinos **Alex** y **Nicole**, por su cariño y sus sonrisas. Que esta tesis algún día les sirva de apoyo. A mi cuñado **Salvador** por su apoyo y consejos.

- ✔ A la familia **Benavides**: Sra. **Lety**, Sr. **Juan Carlos**, **Alfredo** y **Daniel**, por brindarme siempre su apoyo, su cariño y amistad desde el primer día. Por hacerme sentir como de la familia y ofrecerme siempre un segundo hogar.

- ✔ A **Don Pascual** y **Doña Esther**, por todo su apoyo, cariño y consejos. Por que no es aquel que por sangre es, sino aquel que en verdad sabe querer, gracias por ser los abuelitos que siempre soñé. Gracias por todo.

- ✔ A la familia **Solares**: Sr. **David**, Sra. **Eva**, **Erika**, **Lalo** e **Israel**, por su apoyo, cariño y amistad desde el primer día. Gracias por todo.

- ✔ A mis tíos: **Juventino** y **Silvia** y a mi primo **Omar**, por su cariño y consejos desde pequeños. Gracias por todo. A Omar, espero esta tesis lo anime a seguir adelante.

- ✔ A mis tíos: **Anita** y **Arturo** y mis primos: **Juan** y **Anahí**. Gracias por todo su cariño, apoyo y amistad. A Juan y Anahi, espero esta tesis los anime a seguir por el camino correcto.

- ✔ A mis amigos: **Luis Carlos (Charlie)**, **Toño**, **Manuel**, **Jose Luis**, **Irving** y **Soulé**, por haber disfrutado juntos varios momentos importantes de nuestras vidas. Gracias por todo su apoyo y amistad.

- ✔ A **Goofy**, a **Macaria**, a **Ponchito**, al **Tonto**, al **Gordo**, al **Batman**, al **Guero**, al **Pudhi**, a **Lashi**, a **Fenrricia**, al **Ben** y a **Simba**, por que siempre han estado ahí para brindarnos cariño y amistad. Por que siempre tienen una sonrisa para nosotros. Por ser siempre fieles. Gracias a todos!!!

- ✔ A William **Hanna**, Joseph **Barbera**, Walter **Lantz** y Friz **Freleng**, por hacer de mi niñez una vida de sueños y alegrías, a través de sus dibujos.

- ✔ A la **música**, por liberarme. por hacerme llegar a aquellos lugares que alguna vez soñé y nunca creí poder alcanzar. Por ser mi espacio. Por dejar expresarme sin barreras ni ideales falsos y absurdos. A **Zyanya** y **DJ AZULMAN**, por dejarme hacer y sentir la música. A **The Beatles**, **Radiohead**, **Depeche Mode**, **Daft Punk**, **Supergrass**, **Chopan**, **Vivaldi**, **Lennon**, **Yorke**, **Gore,** etc., etc. por ser parte de mi influencia musical. Gracias al **"In Rainbows"** por servir de inspiración al escribir esto.

- ✔ Al **software libre** y al proyecto **GNU/Linux**. A Richard Matthew **Stallman**, Linus **Torvalds**, Ian **Murdock**. Al proyecto Debian y en general a los sistemas Linux por brindarme su conocimiento a través de la transparencia y libertad, además de la participación de todos como comunidad.

- ✔ A mi **Universidad Nacional Autónoma de México**. A mi Facultad de **Ingeniería**, por brindarme todo lo necesario para llegar al éxito moral, humano y profesional. Gracias a los profesores por todo el conocimiento y apoyo brindado.

- ✔ A mi asesor **Carlos Román Zamitiz**, por todo el apoyo y dedicación para el término de esta tesis.

- ✔ A todos y cada uno de los que han estado y han sido parte de mi vida, mil gracias.

Roberto Jesús Calva García.

ÍNDICE GENERAL

CAPÍTULO I: SITUACIÓN ACTUAL Y PROPUESTA DE SOLUCIÓN. **1**
1.1 INTRODUCCIÓN. **1**
1.2 SITUACIÓN ACTUAL DEL PROBLEMA. **2**
1.3 OBJETIVO GENERAL. **3**
1.3.1 OBJETIVOS ESPECÍFICOS. **3**
1.4 DESARROLLO DE LA PROPUESTA DE SOLUCIÓN. **3**

CAPÍTULO II: ANTECEDENTES. **5**
2.1 PANORAMA GENERAL DE LA TECNOLOGÍA RFID (Radio Frequency Identification). **5**
2.1.1 AYER Y HOY. **5**
2.1.2 SU IMPULSO EN EL RETÁIL (VENTAS AL MENUDEO). **7**
2.1.3 LA DESILUSIÓN, ¿POR VENIR?. **8**
2.1.4 RFID VS CODIGO DE BARRAS. **8**
2.2 RADIOFRECUENCIA (RF). **8**
2.2.1 ESPECTRO ELECTROMAGNÉTICO. **9**
2.2.2 ONDAS DE RADIO. **11**
2.3 MODULACION Y DEMODULACION ASK. **11**
2.4 CONCEPTOS DE MICROCONTROLADOR. **13**
2.4.1 DIFERENCIA ENTRE MICROPROCESADOR Y MICROCONTROLADOR. **14**
2.4.2 ARQUITECTURA INTERNA. **16**
2.4.2.1 EL PROCESADOR. **16**
2.4.2.2 MEMORIA DE PROGRAMA. **17**
2.4.2.3 MEMORIA DE DATOS. **18**
2.4.2.4 LINEAS DE E/S PARA LOS CONTROLADORES PERIFÉRICOS. **18**
2.4.2.5 RECURSOS AUXILIARES. **19**
2.5 COMUNICACIONES SERIE. **19**
2.5.1 COMUNICACIONES SERIE ASINCRONAS. **19**
2.5.1.1 NORMA RS-232. **20**
2.5.1.2 CONEXIÓN DE UN MICROCONTROLADOR A EL PUERTO SERIE DE UNA PC. **21**
2.5.1.3 EL CONECTOR DB9 DE LA PC. **21**
2.5.1.4 EL CHIP MAX 232. **23**
2.5.1.5 NORMA RS485. **24**

CAPÍTULO III: ANÁLISIS DE LOS LECTORES WAVETREND L-RX201. **26**
3.1 LECTORES RFID DE LA MARCA WAVETREND. **26**
3.1.1 UN VISTAZO A LOS LECTORES L-RX201. **26**
3.1.2 DIAGRAMA FUNCIONAL. **28**

3.2 RED DE LECTORES RFID L-RX201. **29**
3.2.1 ESTRUCTURA DE UNA SOLA RED DE LECTORES. **29**
3.2.2 OPERACIÓN BÁSICA DE UNA RED DE LECTORES. **30**
3.2.3 ESTABLECIENDO LA DIRECCIÓN DE IDENTIFICACIÓN DE CADA NODO AUTOMÁTICAMENTE (NODE ID). **31**
3.2.4 CONTROL DE LOS PAQUETES DE DATOS. **33**
3.2.5 AUTODETECCIÓN. **34**
3.3 PROTOCOLOS Y DIRECCIONES. **35**
3.3.1 PAQUETES DE COMANDOS. **36**
3.3.2 PAQUETES DE RESPUESTA. **36**
3.3.3 TÉCNICAS DE DIRECCIONAMIENTO DE LOS LECTORES RFID L-RX201. **37**
3.4 COMANDOS DE CONTROL. **38**
3.4.1 LISTA DE COMANDOS. **38**

CAPÍTULO IV: HERRAMIENTAS TECNOLÓGICAS DE DISEÑO. **40**
4.1 PROGRAMACIÓN ORIENTADA A OBJETOS. **40**
4.2 CICLOS DE DESARROLLO DEL SOFTEARE. **42**
4.3 PROCESO UNIFICADO DE RATIONAL (RUP). **49**
4.3.1 CICLO DE VIDA DEL SOFTWARE. **50**
4.4 EL LENGUAJE DE MODELADO UNIFICADO (UML). **52**
4.4.1 DIAGRAMA DE CASOS DE USO. **54**
4.4.1.1 Extensión. **55**
4.4.1.2 Inclusión. **56**
4.4.1.3 Generalización. **56**
4.4.1.4 Documentación. **57**
4.4.2 DIAGRAMA DE CLASES. **59**
4.4.2.1 Clase. **59**
4.4.2.1.1 Atributos y Métodos. **59**
4.4.2.2 Relaciones entre Clases. **60**
4.4.2.3 Casos particulares. **63**
4.5 JAVA, EL LENGUAJE DE PROGRAMACION ELEGIDO. **63**
4.5.1 ELECCIÓN DE JAVA. **64**
4.6 HERRAMIENTA DE DESARROLLO: NETBEANS. **65**
4.6.1 NETBEANS IDE. **66**
4.8 SISTEMAS DE GESTIÓN DE BASE DE DATOS. **67**
4.8.1 SQL. **67**
4.8.2 MYSQL, EL MANEJADOR DE BASE DE DATOS ELEGIDO. **68**
4.8.3 MODELO ENTIDAD-RELACIÓN. **69**
4.9 SISTEMAS OPERATIVOS. **69**
4.9.1 FUNCIONES DE LOS SISTEMAS OPERATIVOS. **70**
4.9.2 CATEGORÍA DE LOS SISTEMAS OPERATIVOS. **70**
4.9.3 DEBIAN LINUX, EL SISTEMA OPERATIVO ELEGIDO. **70**

CAPÍTULO V: ANÁLISIS Y DISEÑO DEL SISTEMA. **73**
5.1 MODELO DE REQUISITOS. **73**
5.1.1 DESCRIPCIÓN DEL PROBLEMA. **75**
5.1.2 FUNCIONES DEL SISTEMA. **77**
5.1.3 ATRIBUTOS DEL SISTEMA. **79**
5.1.4 MODELO DE CASOS DE USO. **80**
5.1.5 MODELO DE INTERFACES PARA LOS CASOS DE USO. **89**
5.2 CONSTRUCCIÓN DEL SISTEMA. **89**
5.2.1 ANÁLISIS DEL SISTEMA. **89**
5.2.1.1 Expansión de los casos de uso seleccionados. **89**
5.2.2 DISEÑO DEL SISTEMA. **95**
5.2.2.1 Modelo del dominio del problema. **96**
5.2.2.2 Identificación de Clases. **96**
5.2.2.3 Selección de Clases. **98**
5.2.2.4 Diagrama de clases. **99**
5.2.2.5 Diccionario de Clases. **99**
5.2.2.6 Diseño de la base de datos. **103**
5.2.2.7 Prototipo del sistema. **106**

CONCLUSIONES. **111**

BIBLIOGRAFÍA. **113**
REFERENCIAS. **114**

ANEXO A: Casos de uso expandidos. **115**
ANEXO B: Prototipo del sistema. **132**
ANEXO C: Código del sistema. **139**
ANEXO D: Comandos de control de Lectores RFID Wavetrend. **146**
ANEXO E: Diseño de la base de datos. **152**

ÍNDICE DE FIGURAS

Figura 2.1 Kit de iniciación RFID: Adaptador de corriente, cables de conexión y de interfaz para la PC, Lector RFID y Tags de la marca Wavetrend. **5**
Figura 2.2 Distintos tipos de Tags y Lector RFID. **6**
Figura 2.3 Tag RFID tipo brazalete de la marca Wavetrend. **7**
Figura 2.4 Red RFID para detección de Tags en un rango especificado. **7**
Figura 2.5 Corrimiento en Amplitud. **12**
Figura 2.6 Modulación por corrimiento en la amplitud (Amplitude shift keying). **12**
Figura 2.7 Análisis de la modulación por corrimiento en la amplitud. **13**
Figura 2.8 El microcontrolador PIC 16F870 de Motorola al interior de un Lector Wavetrend. **13**
Figura 2.9 Estructura de un sistema abierto basado en un microprocesador. **14**
Figura 2.10 El microcontrolador en un sistema cerrado. **15**
Figura 2.11 Circuitería para el PIC16F870 en un Lector Wavetrend. **15**
Figura 2.12 Arquitectura de von Neumann. **16**
Figura 2.13 Arquitectura Harvard. **17**
Figura 2.14 Ejemplo de transmisión de un dato binario. **20**
Figura 2.15 Transmisión y recepción de datos hacia y de una PC con la norma RS-232. **20**
Figura 2.16 Conexión de un microcontrolador a una PC con la norma RS-232. **21**
Figura 2.17 Conectores DB9 tipo macho y tipo hembra. **22**
Figura 2.18 Tabla de los pines en un conector DB9. **22**
Figura 2.19 Adaptación de niveles de voltaje para la norma RS-232. **23**
Figura 2.20 Chip MAX232 para la interfaz con la PC al interior de un Lector Wavetrend. **23**
Figura 2.21 Distribución de pines en un conector tipo DB9. **24**
Figura 3.1 Lector L-RX201. Wavetrend Technologies Ltd. **27**
Figura 3.2 Partes que componen un Lector Wavetrend. **27**
Figura 3.3 Diagrama funcional de un Lector L-RX201. **28**
Figura 3.4 Tabletas al interior de un Lector Wavetrend. **28**
Figura 3.5 Red de Lectores L-RX201. **29**
Figura 3.6 Los puertos RS485 en cada Lector Wavetrend. **30**
Figura 3.7 Red completa de Lectores. **30**
Figura 3.8 Flujo de información Comando/Respuesta . **31**
Figura 3.9 Detección de los lectores de forma automática . **32**
Figura 3.10 Establecimiento automático de cada NODE ID. **32**
Figura 3.11 Transmisión de la información en una red de Lectores. **33**
Figura 3.12 Transmisión de la información en una red de Lectores. **34**
Figura 3.13 Estado de reposo u “ocioso” de una red de Lectores. **34**
Figura 3.14 Paquetes de comandos. **36**
Figura 3.15 Paquetes de respuesta. **36**

Figura 3.16 Diagrama de flujo para la asignación y reconocimiento de Lectores en una red. **37**
Figura 3.17 Ejemplos de direccionamiento para los Lectores L-RX201. **38**
Figura 3.18 Lista de comandos para los Lectores L-RX201. **39**
Figura 4.1 Gráfica de fallos vs tiempo para el desarrollo de software. **43**
Figura 4.2 Esquema de la aproximación convencional. **44**
Figura 4.3 Desarrollo orientado a prototipos. **45**
Figura 4.4 Desarrollo evolutivo. **47**
Figura 4.5 Desarrollo en espiral. **48**
Figura 4.6 Ciclo de vida RUP. **50**
Figura 4.7 Evolución de UML. **53**
Figura 4.8 Ejemplo de casos de uso mostrando la relación con los actores. **54**
Figura 4.9 Casos de uso Hacer Reservación con extensión de Pagar Reservación. **55**
Figura 4.10 Casos de uso Consultar Información con inclusión de Validar Usuario. **56**
Figura 4.11 Casos de uso Pagar Reservación con generalización de pagos. **57**
Figura 4.12 Clase. **59**
Figura 4.13 Clase con atributos y métodos. **60**
Figura 4.14 Herencia o Generalización. **61**
Figura 4.15 Agregación y composición. **62**
Figura 4.16 Asociación. **62**
Figura 4.17 Dependencia. **62**
Figura 4.18 Clase abstracta. **63**
Figura 4.19 Clase parametrizada. **63**
Figura 4.20 Logotipo oficial del proyecto NetBeans. **65**
Figura 4.21 NetBeans IDE 5.5.1 **66**
Figura 4.22 Logo Debian Linux. **71**
Figura 5.1 Dependencia de los distintos modelos del proceso de software del modelo de casos de uso. **74**
Figura 5.2 Desarrollo iterativo en la fase de Construcción. **75**
Figura 5.3 Delimitación del sistema de control para los Lectores RFID LRX201. **80**
Figura 5.4 Diagrama de casos de uso para nuestro sistema en estudio. **88**
Figura 5.5 Diagrama de Clases identificadas para nuestro Sistema de Control. **99**
Figura 5.6 Diagrama de Clases con Atributos para nuestro Sistema de Control. **111**
Figura 5.7 Diagrama de Clases con Métodos para nuestro Sistema de Control. **102**
Figura 5.8 Modelo Entidad – Relación de la Base de Datos MURCIELAGOBD con atributos. **104**
Figura 5.9 Pantalla de Acceso al Sistema (P-AS). **106**
Figura 5.10 Pantalla de Carga del Sistema (FrameCargandoRFID). **106**
Figura 5.11 Pantalla Principal del Sistema (P-P). **107**
Figura 5.12 Pantalla Principal del Sistema para Operador (P-PO). **108**
Figura 5.13 Comandos de Control y Configuración de Lectores L-RX201. **109**
Figura 5.14 Software comercial de la marca Wavetrend. **111**

ÍNDICE DE TABLAS

Tabla 2.1 Radiofrecuencias. **8**
Tabla 2.2 Espectro electromagnético. **10**
Tabla 5.1 Funciones de validación de usuarios (R1). **77**
Tabla 5.2 Funciones de Control y Configuración (R2). **77**
Tabla 5.3 Funciones de Monitoreo (R3). **78**
Tabla 5.4 Funciones extras para control de acceso. (R4). **79**
Tabla 5.5 Atributos del sistema. **79**
Tabla 5.6 Actor Operador. **81**
Tabla 5.7 Actor Usuario RFID. **81**
Tabla 5.8 Actor Administrador. **81**
Tabla 5.9 Actor Lector RFID. **81**
Tabla 5.10 Caso de uso Validar Usuario Sistema de Control niveles 0 y 1. **82**
Tabla 5.11 Caso de uso Consultar bitácora de Accesos RFID niveles 0 y 1. **82**
Tabla 5.12 Caso de uso Consultar bitácora de Accesos al Sistema de Control niveles 0 y 1. **83**
Tabla 5.13 Caso de uso Administración de Usuarios niveles 0 y 1. **83**
Tabla 5.14 Caso de uso Consultar Bitácora de Errores Sistema de Control niveles 0 y 1. **83**
Tabla 5.15 Caso de uso Mantenimiento del Sistema de Control niveles 0 y 1. **84**
Tabla 5.16 Caso de uso Mantenimiento Catálogos en el sistema de control niveles 0 y 1. **84**
Tabla 5.17 Caso de uso Configurar Puerto Control niveles 0 y 1. **85**
Tabla 5.18 Caso de uso Configurar Lectores niveles 0 y 1. **85**
Tabla 5.19 Caso de uso Controlar Lectores niveles 0 y 1. **85**
Tabla 5.20 Caso de uso Detener auto-detección niveles 0 y 1. **86**
Tabla 5.21 Caso de uso Consultar Total de Lectores Red en niveles 0 y 1. **86**
Tabla 5.22 Caso de uso Validar Información Recibida niveles 0 y 1. **87**
Tabla 5.23 Caso de uso Validar Información Recibida Dispositivo Electrńico niveles 0 y 1. **87**
Tabla 5.24 Documentación Casos de Uso. **90**
Tabla 5.25 Clases candidatas para nuestro sistema de control identificadas de la descripción del problema. **98**
Tabla 5.26 Clases identificadas para nuestro Sistema de Control de Lectores RFID. **99**
Tabla 5.27 Descripción de las Tablas de la Base de Datos. **105**

CAPÍTULO I: SITUACIÓN ACTUAL Y PROPUESTA DE SOLUCIÓN.

1.1 INTRODUCCIÓN.

El desarrollo de mi Servicio Social en el Centro de Diseño y Manufactura (CDM) tuvo el objetivo de lograr la comunicación entre una PC y algunos Lectores (Readers en inglés) de Identificación por Radiofrecuencia (RFID) de la marca Wavetrend modelo L-RX201 para el reconocimiento de los datos en tarjetas electrónicas o etiquetas (Tags) de la misma marca y así tener el control de estos Lectores para ser usados en diversas aplicaciones. Los usos más comunes de los Lectores RFID son referentes a seguridad, tanto para tener un control de acceso a puertas y plumas de estacionamiento, como para monitorear equipo costoso, tal es el caso de algún equipo de cómputo, aparatos de medición entre otros. En la actualidad, además de su uso en seguridad, la tecnología RFID está tomando mucho auge en grandes centros comerciales y bodegas en países de primer mundo para llevar a cabo el control de inventarios de todos sus productos, realizar pagos instantáneos en las cajas de los supermercados con sólo pasar el carrito de compras por un Lector RFID sin necesidad de registrar uno por uno los productos a comprar y en un futuro no muy lejano, desplazará a la obsoleta tecnología del código de barras, todo esto mientras las etiquetas electrónicas o Tags se hagan mucho más baratas.

Al término de mi servicio social logré descifrar los datos recibidos por los Lectores obtenidos de los Tags y gracias a esto decidí continuar estudiando y mejorando los resultados obtenidos en el sistema de control, además de comprender el funcionamiento interno de los Lectores.

El motivo por el cual decidí realizar esta tarea de Servicio Social es que la licencia de compra de un software en la actualidad es demasiado costosa, sobre todo para las instituciones. La licencia para adquirir el software original de control para las antenas de la marca Wavetrend oscila en los miles de dólares, algo que se puede evitar gracias a la filosofía del software libre. La idea del análisis y diseño de un sistema de control para estas Lectores, permitiría a quien lo quisiera, adquirir únicamente los Lectores e implementarlos con el sistema de control que con la presente tesis se puede llevar a cabo. La Torre de Ingeniería situada en nuestra Ciudad Universitaria presenta este grave problema, el cual se podría evitar a través de un sistema de control de libre distribución y multiplataforma, es decir, que pueda funcionar en cualquier sistema operativo.

1.2 SITUACIÓN ACTUAL DEL PROBLEMA.

Existe un gran problema en todo el mundo referente al desarrollo de software e implementación, sobre todo en países en vías de desarrollo y de tercer mundo: el pago por una Licencia de un producto o software a una empresa multinacional. Las licencias generalmente oscilan en precio entre los miles o incluso los millones de dólares. De ahí que surgieran organizaciones como la fundación GNU/Linux que realizan software de libre distribución para aquellas personas que deseen utilizarlo y no tener que, necesariamente, pagar por una licencia costosa. Un gran problema económico es el que la UNAM o estrictamente la Facultad de Ingeniería a través de su Centro de Diseño y Manufactura (CDM) o a través de la Torre de Ingeniería tengan que pagar una licencia de miles de dólares por utilizar el software que controla a los Lectores Wavetrend, además de que esto limita al mismo comprador a tener que instalar forzosamente este software en una PC con Sistema Operativo Windows, el cual es un problema aparte de una costosa licencia más por pagar.

Debido a que al momento al CDM y a la Torre de Ingeniería les es imposible facilitarme alguno de los Lectores RFID (por intereses propios de los investigadores, en especial por estar realizando el presente trabajo para libre distribución) para poder concluir las pruebas del sistema de control, he decidido trabajar en el presente tema de tesis “Análisis y diseño de un sistema de control para los Lectores de RFID marca Wavetrend modelo L-RX201 utilizando metodología orientada a objetos” el cual se dedicará, como el título lo indica, únicamente a realizar un análisis de los Lectores RFID y a su funcionamiento, además de diseñar a través del modelado orientado a objetos UML (Unified Modeling Language, o Lenguaje de Modelado Unificado) un sistema de control para las mismas, dejando algunas bases en Java necesarias para llevar a cabo el proyecto y permitir a algún estudiante interesado en el tema proponer la siguiente parte como un tema de tesis para desarrollar e implementar el sistema de control.
Por todo lo anterior, el presente trabajo tiene la obligación moral y ética de realizar el análisis y diseño del sistema de control de los Lectores Wavetrend a través de la filosofía del software libre, así como brindar las bases para un próximo desarrollo e implementación (los cuales quedarán de propuesta de interés al lector) para que sea programado y posteriormente probado en un sistema operativo de libre distribución, (como lo puede ser cualquier plataforma basada en Linux), pero sin perder la posibilidad de trabajar bajo Windows a sabiendas de que la mayoría de los usuarios trabaja bajo el régimen de Microsoft, impidiendo así lucrar con un diseño que lo que pretende es evitar el pago por el mismo, basándome en la visión de software como servicio, no como producto. Todo esto puede lograrse gracias a las ventajas

multiplataforma de las herramientas tecnológicas a utilizar, las cuales se analizarán en el capítulo correspondiente.

Se pretende dejar todas las bases necesarias para el posterior desarrollo e implementación del sistema de control, alcances que no se verán en la presente tesis.

1.3 OBJETIVO GENERAL.

El objetivo general de la presente tesis es: analizar y diseñar un sistema de control para los Lectores de RFID (Radio Frequency Identification) marca Wavetrend modelo L-RX201 utilizando metodología orientada a objetos, partiendo de la filosofía del desarrollo de software libre y utilizando el lenguaje de modelado unificado (UML).

1.3.1 OBJETIVOS ESPECÍFICOS.

Los objetivos específicos de este trabajo pretenden dejar las bases necesarias para el posterior desarrollo e implementación del sistema de control y un uso del software libre (para lograr en un futuro un desarrollo de libre distribución y multiplataforma que no dependa de ningún sistema operativo ni de programas con licencia como una base de datos, la cual por supuesto debe de seguir la filosofía de fuente abierta u open source, o al menos ser software libre), así como el uso de los Lectores de RFID en México. Con el presente libro y todo el estudio detrás se pretende que este problema no se quede en el simple análisis y diseño de este sistema de control.

1.4 DESARROLLO DE LA PROPUESTA DE SOLUCIÓN.

Para el desarrollo de este sistema de control se tienen varios caminos en cuanto al modelado orientado a objetos, todos basados en el lenguaje UML antes nombrado. Existen varios lenguajes orientados a objetos, tales como Visual.NET, C++, Java entre otros. Si se analiza que lo que se busca es tener el diseño de un sistema de control que siga la filosofía del software libre y que además sea multiplataforma, sin olvidar la interacción gráfica y la portabilidad. Por tanto, una muy buena elección por ser un lenguaje propiamente de filosofía “escribe una vez y ejecuta donde sea” nos encamina al lenguaje Java.

Se tendrían entonces, ya entrando en materia y dando una visión general del sistema de control, varias clases en Java, tales como una clase que proporcione el control de escritura a través de la norma RS232 (comunicación serial tipo asíncrona, la cual se verá más adelante) para enviar datos a los Lectores; una clase de control de lectura para recibir datos de los Lectores a través de la misma norma; una clase para dar de alta y de baja a usuarios o

dispositivos la cual estará ligada a alguna base de datos de libre distribución, (como sería el caso de MySQL o PostgreSQL), utilizando los números de identificación contenidos en los Tags que se deberán reconocer y aceptar para el acceso seguro o para el monitoreo de aparatos costosos, según se requiera en la aplicación; una clase principal o menú principal para la interacción entre el usuario y el software y una clase que será de lectura y escritura a la vez, la última para algunos comandos. Esto es a grandes rasgos la descripción de las partes que componen el sistema de control a diseñar.

Para comprender el sistema y así desarrollar en los siguientes capítulos la propuesta de solución dispondré de la siguiente metodología:

- Estudiar la composición física de los Lectores RFID de la marca Wavetrend, para comprender su funcionamiento.
- Estudiar y comprender los comandos y funciones de estos Lectores para lograr el acoplamiento con el modelado orientado a objetos.
- Aplicar los temas de programación orientada a objetos (POO) y del lenguaje UML para realizar el diseño del sistema de control.
- Estudiar las interfaces y normas de comunicación utilizadas según el manual para los Lectores de RFID Wavetrend.
- Analizar y Diseñar el sistema de control con todo lo anterior.

Con todo lo antes dicho, el siguiente paso en la presente tesis es conocer los antecedentes o teoría básica para comprender terminologías, los cuales se verán en el capítulo II; conocer el funcionamiento de los Lectores Wavetrend en el capítulo III y tener las bases iniciales de Análisis y Diseño, elegir el lenguaje de programación, plataforma de sistema operativo y base de datos, esto en el capítulo IV, para así analizar y diseñar nuestro sistema de control en el capítulo V.

Así entonces se abre paso a los siguientes capítulos que hablarán sobre estos antecedentes básicos.

CAPÍTULO II: ANTECEDENTES.

2.1 PANORAMA GENERAL DE LA TECNOLOGÍA RFID (Radio Frequency Identification).

RFID, acrónimo del inglés Radio Frequency Identification, (identificación por radiofrecuencia), es una tecnología que consiste en una antena, un receptor, un procesador de almacenamiento y coprocesador de datos y, en algunos casos, una batería. Su alcance puede ser desde unos centímetros hasta decenas de metros, con una capacidad de almacenamiento de algunos bytes hasta Megabytes (Mb) y con diversas velocidades de transmisión de información.

Las etiquetas de identificación, disponibles como dispositivos pasivos, semi-pasivos y activos, son el componente que contiene una antena y microchips codificados con datos, mismos que serán capturados por un Lector y enviados a los sistemas de cómputo para su interpretación, almacenamiento y acción correspondiente, dependiendo de lo que se requiera.

Los dispositivos pasivos pueden ser leídos a una distancia de poco más de tres metros y no utilizan batería, mientras que las etiquetas semi-pasivas sí la requieren, para proveer de energía a sus circuitos y transmisión. La ventaja de éstas últimas es su capacidad de comunicarse en un rango más amplio y en ambientes hostiles.

En el caso de las activas, también utilizan energía de una batería, con lo que superan las limitantes de las etiquetas pasivas, en cuanto al mayor alcance de lectura, además de permitir la detección de temperatura. Su costo oscila entre $4 y $5 dólares, que puede incrementarse hasta los cientos de dólares, a diferencia de los entre $40 centavos y $10 dólares que cuestan las pasivas, lo que limita su uso. Los tres tipos de etiquetas pueden ser de sólo lectura o lectura-escritura.

Figura 2.1 Kit de iniciación RFID: Adaptador de corriente, cables de conexión y de interfaz para la PC, Lector RFID y Tags de la marca Wavetrend.

2.1.1 AYER Y HOY.

En 1949 en Estados Unidos, N. J. Woodland presentó la primera patente de un código de barras y su uso se inició en la década de los 60 en los ferrocarriles. Mientras que los antecedentes de RFID datan de la Segunda Guerra Mundial, cuando los ingleses requerían identificar a los aviones enemigos en la costa con Francia. Por lo que se desarrolló un sistema denominado IFF (Identify Friend or Foe, Identifica a un Amigo o Enemigo), que daba la respuesta correcta a una señal de radio. En los años 60 y 70, su uso se enfocó en la seguridad de materiales nucleares.

En la actualidad RFID se utiliza principalmente en el rubro de seguridad, como es el caso de los chips de identificación que llevan algunas mascotas bajo la piel desde hace años, credenciales de identidad, sistemas de acceso a zonas restringidas para empleados, peajes en las carreteras que no requieren que nos detengamos, cruces fronterizos, facturación de equipajes más eficaz, para evitar la falsificación de moneda así como en el control vehicular, identificación de ganado, envío de paquetes, control de equipaje en los aeropuertos y de artículos para renta o préstamo –películas y libros– en videoclubes y bibliotecas, para acceder a grandes eventos deportivos o de ocio (esta tecnología tuvo aplicación en los juegos olímpicos de Beijing 2008 para el registro de los tiempos de los atletas en el maratón y caminata, entre otros), etc.

Figura 2.2 Distintos tipos de Tags y Lector RFID.

Incluso, existen empresas como Applied Digital Systems que defienden la implantación de estos chips bajo la piel de todos los ciudadanos como un método de identificación personal infalible, imposible de robar o de perder. En la figura 2.2 se observan los distintos tipos de Tags RFID, de la empresa Wavetrend Technologies y en la figura 2.3 un tag tipo brazalete.

Figura 2.3 Tag RFID tipo brazalete de la marca Wavetrend.

2.1.2 SU IMPULSO EN EL RETAIL (VENTAS AL MENUDEO).

A partir de ventajas como la identificación simultánea de la información de varias etiquetas, velocidad y capacidad de almacenamiento, así como el no requerir una línea de visión para leer o escribir datos en la etiqueta, ni proximidad entre ésta y el Lector, se vieron oportunidades para el sector retail (ventas al por menor) tales como: unicidad del producto, control de inventarios en tiempo real, control de caducidad, disponibilidad, rastreo en la cadena de suministro y para evitar robos y piratería. Cabe señalar que en 2003, el Ministerio de Defensa y el Servicio de Aduanas de Estados Unidos comenzaron a requerir a sus proveedores la adopción de esta tecnología.

En el caso de Wal-Mart sucedió lo mismo, pues el corporativo solicitó a sus 100 principales proveedores migraran al etiquetado RFID para el 1 de enero de 2005. Por su parte, Metro Group, empresa de supermercados alemana, convocó a proveedores de tecnología a desarrollar aplicaciones de RFID en programas piloto. Intel, SAP e IBM participaron en el proyecto, junto con otras empresas, para poner en marcha lo que podría ser el inicio de “la Tienda del Futuro”. En la figura 2.4 se muestra la detección de tags en un rango especificado.

En dicho establecimiento, ubicado en Rheinberg, Alemania, el cliente vive una experiencia automatizada desde el saludo, registro de la lista de artículos adquiridos en su última visita, asistencia en la localización de artículos, hasta el cobro sin operación manual, aunque en un futuro se planea el cálculo total de compra al pasar el carrito contenedor a un costado de la caja, para concluir en la elección de pago para el cliente entre tarjeta de crédito, débito, cheque o efectivo.
Varios son los proyectos piloto que se trabajan en todo el mundo, no obstante no hay ninguna empresa en ese giro que haya adoptado RFID por completo.

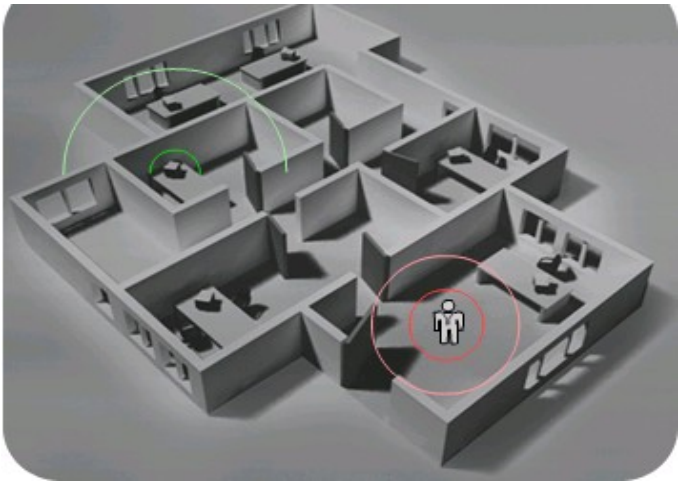


Figura 2.4 Red RFID para detección de Tags en un rango especificado.

2.1.3 LA DESILUSIÓN, ¿POR VENIR?

Según el estudio de Gartner “Prepare for RFID Disillusionment”, los usuarios de esta tecnología experimentarán una desilusión, pues prevé para 2007 un alto índice de probabilidad de falla en por lo menos 50% de los proyectos adoptados durante este año. También refiere que en el caso de Wal-Mart, la compañía entiende los riesgos que implica la adopción del sistema, por lo que sugiere a todo interesado tener en cuenta ese perfil de riesgo y no basar su experiencia de negocio en el éxito de otros. La firma analista menciona que actualmente RFID está a disposición de “afortunados y ricos”, no obstante se mantiene optimista a largo plazo, pues señala que será una de las tecnologías más estratégicas para las empresas en 2018. De acuerdo con Miguel Gerberoff, director general de Hand Held Products (HHP) en México, las dificultades tecnológicas, falta de estándares, costos, pruebas fallidas, entre otras razones, no permiten consolidar la implantación de RFID y, por ende, el código de barras sigue vigente.

2.1.4 RFID VS CODIGO DE BARRAS.

A diferencia de un código de barras, el contacto visual no es necesario para obtener una lectura, por lo que las velocidades de lectura son mucho mayores y en muchas ocasiones el factor humano no es necesario.

RFID es una tecnología reciente y muy efectiva para la captura de datos en forma automática. Esta tecnología utiliza la radio frecuencia para la identificación de objetos a distancia de proximidad.

Las etiquetas identificadoras (Tags en inglés) poco a poco sustituirán en muchos casos a las típicas etiquetas de código de barras, pues funcionan eficientemente para los procesos de identificación, más comunes ya descritos con anterioridad.

2.2 RADIOFRECUENCIA (RF).

El término **Radiofrecuencia**, o **RF**, se aplica a la porción del espectro electromagnético en el que se pueden generar ondas electromagnéticas aplicando corriente alterna a una antena.

Dichas frecuencias cubren las siguientes bandas del espectro (Tabla 2.1):

Nombre	Abreviatura inglesa	Banda ITU	Frecuencias	Longitud de onda
			Inferior a 3 Hz	> 100.000 km
Extra baja frecuencia Extremely low frequency	ELF	1	3-30 Hz	100.000 km – 10.000 km
Super baja frecuencia Super low Frequency	SLF	2	30-300 Hz	10.000 km – 1000 km
Ultra baja frecuencia Ultra low frequency	ULF	3	300–3000 Hz	1000 km – 100 km
Muy baja frecuencia Very low frequency	VLF	4	3–30 kHz	100 km – 10 km
Baja frecuencia Low frequency	LF	5	30–300 kHz	10 km – 1 km
Media frecuencia Medium	MF	6	300–3000 kHz	1 km – 100 m

Frequency				
Alta frecuencia High frequency	HF	7	3–30 MHz	100 m – 10 m
Muy alta frecuencia Very high frequency	VHF	8	30–300 MHz	10 m – 1 m
Ultra alta frecuencia Ultra high frequency	UHF	9	300–3000 MHz	1 m – 100 mm
Super alta frecuencia Super high frequency	SHF	10	3–30 GHz	100 mm – 10 mm
Extra alta frecuencia Extremely high frequency	EHF	11	30–300 GHz	10 mm – 1 mm
			Sobre 300 GHz	< 1 mm

Tabla 2.1 Radiofrecuencias.

*P*or encima de 300 Ghz, la absorción de la radiación electromagnética por la atmósfera terrestre es tan alta que la atmósfera se vuelve opaca a frecuencias más altas de radiación electromagnética, hasta que vuelve de nuevo a ser transparente en los denominados rangos de frecuencia infrarrojos y ópticos.

Las bandas ELF, SLF, ULF y VLF se superponen al espectro de AF (audio frecuencia), que se encuentra entre 20 y 20000 Hz aproximadamente. De todos modos, los sonidos se mueven a la velocidad del sonido, en vez de a la velocidad de la luz.

Los conectores eléctricos diseñados para trabajar con frecuencias de radio se conocen como conectores RF. RF también es el nombre del conector estándar de audio/video, mejor conocido como BNC (BayoNet Connector).

2.2.1 ESPECTRO ELECTROMAGNÉTICO.

Se denomina espectro electromagnético al conjunto de ondas electromagnéticas. Van desde las de menor longitud de onda, como son los rayos cósmicos, los rayos gamma y los rayos X, pasando por la luz ultravioleta, la luz visible y los rayos infrarrojos, hasta las ondas electromagnéticas de mayor longitud de onda, como son las ondas de radio. En cualquier caso, cada una de las categorías son de ondas de variación de campo electromagnético.

La tabla 2.2 muestra el espectro electromagnético, con sus longitudes de onda, frecuencias y energías de fotón:

ESPECTRO E.M.	Longitud de onda (m)	Frecuencia (Hz)	Energía (J)
Rayos gamma	< 10 pm	>30.0 EHz	>19.9E-15 J
Rayos X	< 10 nm	>30.0 PHz	>19.9E-18 J
Ultravioleta Extremo	< 200 nm	>1.5 PHz	>993E-21 J
Ultravioleta Cercano	< 380 nm	>789 THz	>523E-21 J
Luz Visible	< 780 nm	>384 THz	>255E-21 J
Infrarrojo Cercano	< 2.5 um	>120 THz	>79.5E-21 J
Infrarrojo Medio	< 50 um	>6.00 THz	>3.98E-21 J
Infrarrojo Lejano/submilimétrico	< 1 mm	>300 GHz	>199E-24 J
Microonda	< 30 cm	>1.0 GHz	>1.99e-24 J
Ultra Alta Frecuencia Radio	<1 m	>300 MHz	>1.99e-25 J
Muy Alta Frecuencia Radio	<10 m	>30 MHz	>2.05e-26 J
Onda corta Radio	<180 m	>1.7 MHz	>1.13e-27 J
Onda Media (AM) Radio	<650 m	>650 kHz	>4.31e-28 J

Onda Larga Radio	<10 km	>30 kHz	>1.98e-29 J
Muy Baja Frecuencia Radio	>10 km	<30 kHz	<1.99e-29 J

Tabla 2.2 Espectro electromagnético.

2.2.2 ONDAS DE RADIO.

La radio es una tecnología que posibilita la transmisión de señales mediante la modulación de ondas electromagnéticas. Éstas son ondas que pueden propagarse tanto a través del aire como del espacio vacío y no requieren un medio de transporte.

Una onda de radio se origina cuando una partícula cargada (por ejemplo, un electrón) se excita a una frecuencia situada en la zona de radiofrecuencia (RF) del espectro electromagnético. Otros tipos de emisiones que caen fuera de la gama de RF son los rayos gamma, los rayos X, los rayos cósmicos, los rayos infrarrojos, los rayos ultravioleta y la luz visible.

Cuando la onda de radio actúa sobre un conductor eléctrico (la antena), induce en él un movimiento de la carga eléctrica (corriente eléctrica) que puede ser transformado en señales de audio u otro tipo de señales portadoras de información.

Aunque empleamos la palabra *radio*, las transmisiones de televisión, radio, radar y telefonía móvil están todas ellas incluidas en esta clase de emisiones de radiofrecuencia.

2.3 MODULACIÓN Y DEMODULACIÓN ASK.

Para la transmisión de datos digitales, existen principalmente tres métodos de modulación que permiten alterar el ancho de banda sobre el cual será enviada la información. Estos tres métodos son muy empleados debido a su relativa sencillez y a que son ideales para la transmisión de datos digitales, ellos son, el ASK (Amplitude Shift Keying), FSK (Frequency Shift Keying) y PSK (Phase Shift Keying).

La modulación en ASK no es otra cosa que una variante de la modulación en AM que se adapta perfectamente a las condiciones de los sistemas digitales, además de que les permite trabajar sobre una sola frecuencia de transmisión en ves de tener que lidiar con pulsos cuadrados que contienen componentes en todas las frecuencias del espectro.

Su recuperación también resulta ser más sencilla, dado que sólo depende de sincronizar la frecuencia de las señales sinusoidales que sirven de portadoras y regeneradoras dependiendo si se hallan en el modulador o el demodulador.

El ASK por sí sólo, a pesar de todas estas consideraciones, no es uno de los métodos más utilizados debido a que para cada frecuencia es necesario realizar un circuito independiente, además de que sólo puede transmitirse un solo bit al mismo tiempo en una determinada frecuencia. Otro de los inconvenientes es que los múltiplos de una frecuencia fundamental son inutilizables y que este tipo de sistemas son susceptibles al ruido.

Sin embargo, conocer su funcionamiento es esencial para poder comprender el diseño de otro tipo de modulaciones como el FSK, el PSK, el QPSK y demás derivados, dado que en buena parte este tipo de diseños están basados en variaciones o combinaciones de dos o más señales moduladas en ASK.

El ASK que es el método que nos atañe en especial, pues es utilizada por los Lectores RFID. Es una forma de modulación mediante la cual la amplitud de la señal está dada por la ecuación en la figura 2.5.

$$\phi = \left\langle \begin{array}{lll} A\,\mathrm{sen}(\omega_o t) & ; & 0 \le t \le T \\ 0 & ; & en\ otro\ caso \end{array} \right\rangle$$

Figura 2.5 Corrimiento en Amplitud.

ASK entonces, puede ser descrito como la multiplicación de la señal de entrada f(t)=A (válido en sistemas digitales) por la señal de la portadora. Además, esta técnica es muy similar a la modulación en amplitud AM, con la única diferencia que para este caso m=0.

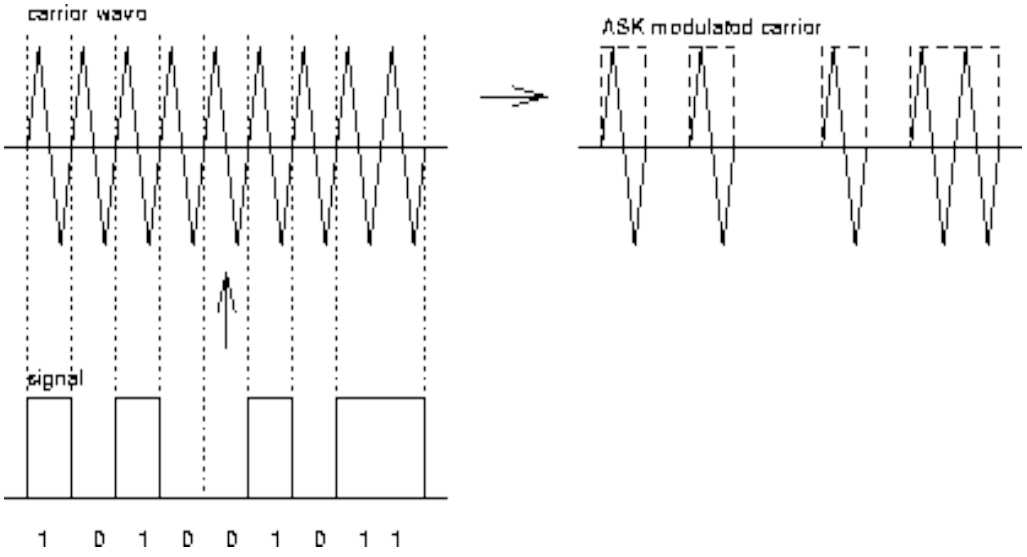


Figura 2.6 Modulación por corrimiento en la amplitud (Amplitude shift keying).

En el dominio de la frecuencia, tal y como ya lo habíamos mencionado, el efecto de la modulación por ASK permite que cualquier señal digital sea adecuada para ser transmitida en un canal de ancho de banda restringida sin ningún problema, además al estar en función de una sola frecuencia, es posible controlar e incluso evitar los efectos del ruido sobre la señal con tan sólo utilizar un filtro paso-bandas, o bien, transmitir más de una señal independientes entre sí sobre un mismo canal, con tan sólo modularlas en frecuencias diferentes. Esto queda demostrado gráficamente si observamos la representación de la figura 2.7.

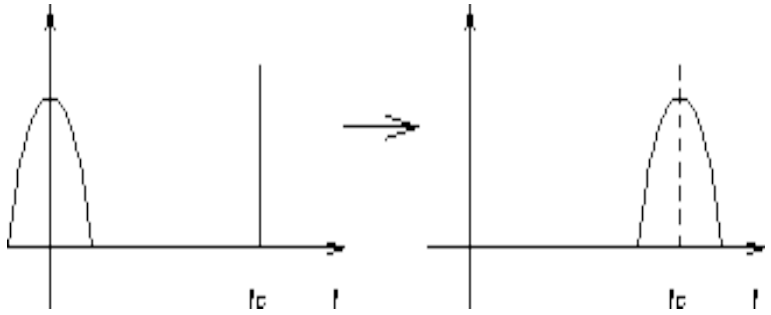


Figura 2.7 Análisis de la modulación por corrimiento en la amplitud.

2.4 CONCEPTOS DE MICROCONTROLADOR.

Es un circuito integrado programable que contiene todos los componentes de una computadora. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que gobierna. Esta última característica es la que contiene la denominación de <<controlador incrustado>> (embedded controller).

El microcontrolador es una computadora dedicada. En su memoria sólo reside un programa destinado a gobernar una aplicación determinada; sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar, y todos los recursos complementarios disponibles tienen como única finalidad atender sus requerimientos. Una vez programado y configurado el microcontrolador solamente sirve para gobernar la tarea asignada. En la figura 2.8 podemos observar un PIC al interior de un Lector RFID de la marca Wavetrend.

Un microcontrolador es una computadora completa, aunque de limitadas prestaciones, que está contenido en el chip de un circuito integrado y se destina a gobernar una sola tarea.

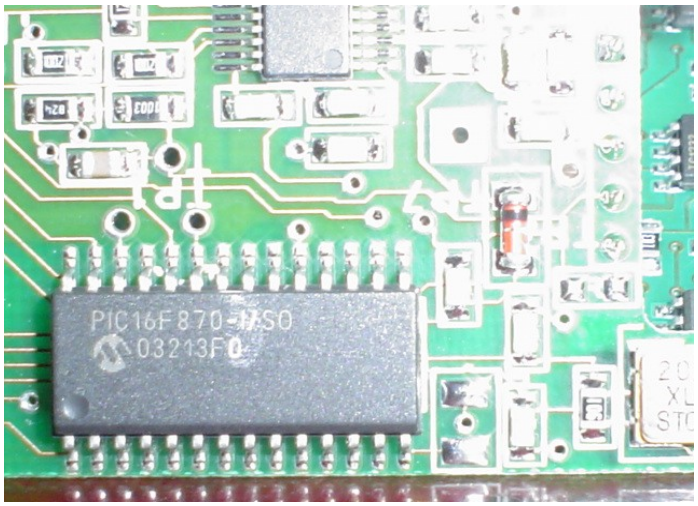


Figura 2.8 El microcontrolador PIC 16F870 de Microchip al interior de un Lector Wavetrend.

El número de productos que funcionan en base a uno o varios microcontroladores aumenta de forma exponencial. No es aventurado pronosticar que en el siglo XXI habrá pocos elementos que carezcan de un microcontrolador. En esta línea de prospección al futuro, la empresa Dataquest calcula que en cada hogar americano existirán varios centenares de microcontroladores en los comienzos del tercer milenio.
La industria Informática acapara gran parte de los microcontroladores que se fabrican. Casi todos los periféricos del computador, desde el ratón o el teclado hasta la impresora, son regulados por el programa de un microcontrolador.
Los electrodomésticos de línea blanca (lavadoras, hornos, lavavajillas, etc.) y de línea marrón (televisores, vídeos, aparatos musicales, etc.) incorporan numerosos microcontroladores. Igualmente, los sistemas de supervisión, vigilancia y alarma en los edificios utilizan estos chips. También se emplean para optimizar el rendimiento de ascensores, calefacción, aire acondicionado, alarmas de incendio, robo, etc.
Como podremos observar más adelante, un microcontrolador es parte importante en el funcionamiento de nuestras antenas lectoras RFID (Radio Frequency Identification) en el procesamiento de los datos recibidos y por supuesto de los comandos mismos en la antena lectora. Las comunicaciones y sus sistemas de transferencia de información utilizan demasiado estos pequeños computadores incorporándolos en los grandes automatismos y en los modernos teléfonos.
La instrumentación y la electromedicina son dos campos idóneos para la implantación de estos circuitos integrados. Una importante industria consumidora de microcontroladores es la automovilística, que aplica el control de aspectos tan populares como la climatización, la seguridad y los frenos ABS.

Las comunicaciones y los productos de consumo general absorben más de la mitad de la producción de microcontroladores. El resto se distribuye entre el sector de la automoción, las computadoras y la industria.

2.4.1 DIFERENCIA ENTRE MICROPROCESADOR Y MICROCONTROLADOR.

El microprocesador es un circuito integrado que contiene la Unidad Central de Proceso (UCP), también llamada procesador, de una computadora. La UCP está formada por la Unidad de Control, que interpreta las instrucciones y el Bus de Datos que las transmite para ser ejecutadas.

Las terminales (patillas) de un microprocesador sacan al exterior las líneas de sus buses de direcciones, datos y control, para permitir conectarle con la Memoria y los Módulos de Entrada/Salida y configurar un computador implementado por varios circuitos integrados. Se dice que un microprocesador es un sistema abierto por que su configuración es variable de acuerdo con la aplicación a la que se destine (Figura 2.9).

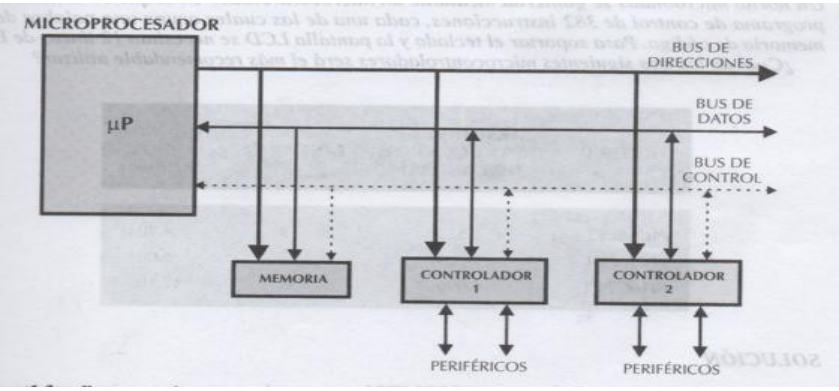


Figura 2.9 Estructura de un sistema abierto basado en un microprocesador. La disponibilidad de los buses en el exterior permite que se configure a la medida de la aplicación.

Un **microprocesador** es un **sistema abierto** con el que puede construirse una computadora con las características que se desee, acoplándole los módulos necesarios.
Un **microcontrolador** es un **sistema cerrado** que contiene una computadora completa y de prestaciones limitadas que no se pueden modificar. (Fig. 2.10)

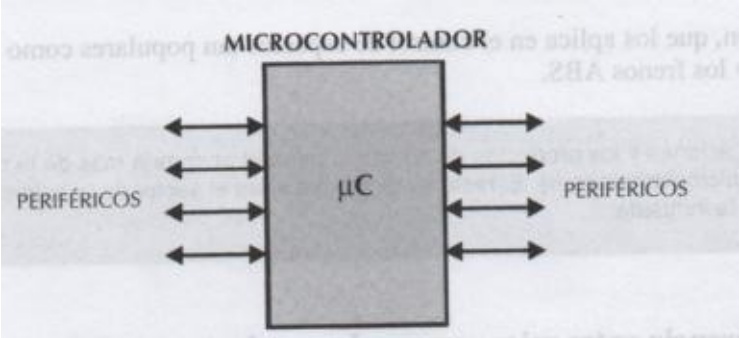


Figura 2.10 El microcontrolador en un sistema cerrado. Todas las partes de la computadora están conectadas en su interior y sólo salen al exterior las líneas que gobiernan los periféricos.

Si solo se dispusiese de un modelo de microcontrolador, éste debería tener muy potenciados todos sus recursos para poderse adaptar a las exigencias de las diferentes aplicaciones. Esta potenciación supondría en muchos casos un despilfarro. En la práctica cada fabricante de microcontroladores oferta un elevado número de modelos diferentes, desde los más sencillos hasta los más poderosos. Es posible seleccionar la capacidad de las memorias, el número de líneas de Entrada/Salida, la cantidad y potencia de los elementos auxiliares, la velocidad de funcionamiento, etc. Por todo ello, un aspecto muy destacado del diseño es la selección del microcontrolador a utilizar.

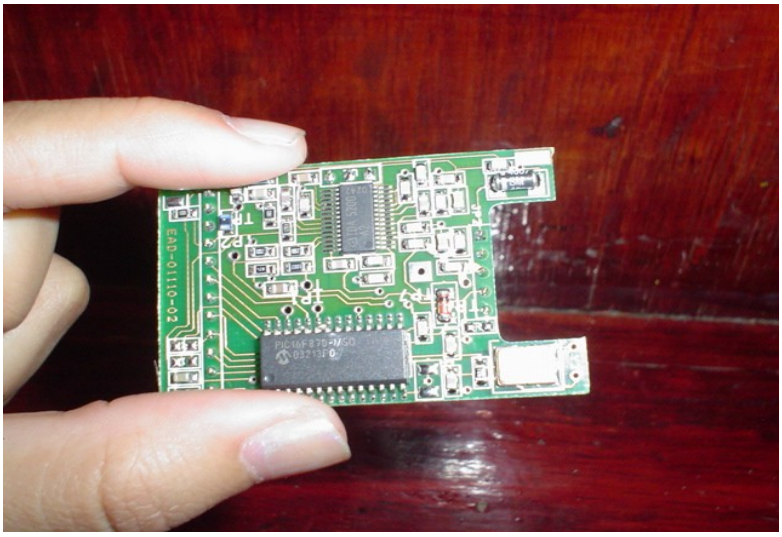


Figura 2.11 Circuitería para el PIC16F870 en un Lector Wavetrend.

2.4.2 ARQUITECTURA INTERNA.

Un microcontrolador posee todos los componentes de una computadora, pero con unas características fijas que no pueden alterarse.

Las partes principales de un microcontrolador son:

1. Procesador
2. Memoria no volátil para contener el programa
3. Memoria de lectura y escritura para guardar los datos
4. Líneas de E/S para los controladores de periféricos:
   a) Comunicación paralelo
   b) Comunicaciones serie
   c) Diversas puertas de comunicación (bus, USB, etc.)

5. Recursos auxiliares:
   a) Circuito de reloj
   b) Temporizadores
   c) Perro guardián (<<watchdog>>)
   d) Conversores AD y DA
   e) Comparadores analógicos
   f) Protección ante fallos de la alimentación
   g) Estado de reposo o de bajo consumo.

2.4.2.1 EL PROCESADOR.

La necesidad de conseguir elevados rendimientos en el procesamiento de las instrucciones ha desembocado en el empleo generalizado de procesadores de arquitectura **Harvard** frente a los tradicionales que seguían la arquitectura de **Von Neumann.** Esta última se caracterizaba porque la UCP Unidad Central de Proceso) se conectaba con una memoria única, donde coexistían datos e instrucciones, a través de un sistema de buses (véase Figura 2.12).

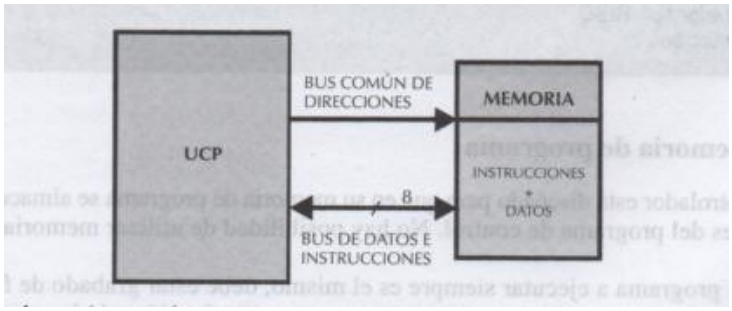


Figura 2.12 En la arquitectura de von Neumann la UCP se comunicaba a través de un sistema de buses con la Memoria, donde se guardaban las instrucciones y los datos.

En la arquitectura Harvard son independientes la memoria de instrucciones y la memoria de datos y cada una dispone de su propio sistema de buses para el acceso. Esta dualidad, además de propiciar el paralelismo, permite la adecuación del tamaño de las palabras y los buses a los requerimientos específicos de las instrucciones y de los datos. También la capacidad de cada memoria es diferente (Figura 2.13).

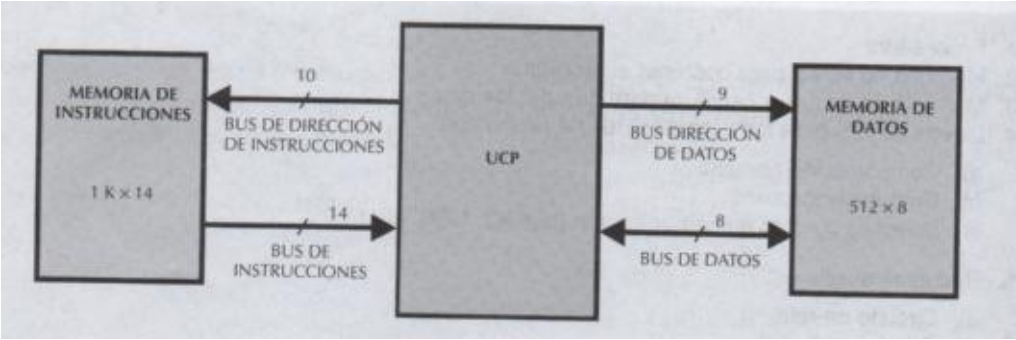


Figura 2.13 En la arquitectura Harvard, la memoria de instrucciones tiene 1K posiciones de 14 bits cada una, mientras que la de datos sólo dispone de 512 posiciones de 1 byte.

El procesador de los modernos microcontroladores responde a la arquitectura **RISC** (Computadoras de Juego de Instrucciones Reducido), que se identifica por poseer un repertorio de instrucciones máquina pequeño y simple, de forma que la mayor parte de las instrucciones se ejecuta en un ciclo de instrucción.
Otra aportación frecuente que aumenta el rendimiento de la computadora es el fomento del paralelismo implícito, que consiste en la segmentación del procesador (pipe-line), descomponiéndolo en etapas para poder procesar una instrucción diferente en cada una de ellas y trabajar con varias a la vez.

El alto rendimiento y elevada velocidad que alcanzan los modernos procesadores, como el que poseen los microcontroladores PIC, se debe a la conjunción de tres técnicas:

- Arquitectura Harvard
- Computador tipo RISC
- Segmentación

2.4.2.2 MEMORIA DE PROGRAMA.

El microcontrolador está diseñado para que en su memoria de programa se almacenen todas las instrucciones del programa de control. No hay posibilidad de utilizar memorias externas de aplicación.
Como el programa a ejecutar siempre es el mismo, debe estar grabado de forma permanente. Los tipos de memoria adecuados para soportar esta función admiten cinco versiones diferentes:

a) ROM con máscara.

En este tipo de memoria el programa se graba en el chip durante el proceso de su fabricación mediante el uso de máscaras. Los altos costos de diseño e instrumentación sólo aconsejan usar este tipo de memoria cuando se precisan series muy grandes.

b) EPROM.

La grabación de esta memoria se realiza mediante un dispositivo físico gobernado desde un computador personal, que recibe el nombre de grabador. En la superficie de la cápsula del microcontrolador existe una ventana de cristal por la que se puede someter el chip de la memoria a rayos ultravioletas para producir su borrado y emplearla nuevamente. Es interesante la memoria EPROM en la fase de diseño y depuración de los programas, pero su costo unitario es elevado.

c) OTP (Programable una vez).

Este modelo de memoria sólo se puede grabar una vez por parte del usuario, utilizando el mismo procedimiento que con la memoria EPROM. Posteriormente no se puede borrar. Su bajo precio y la sencillez de la grabación aconsejan este tipo de memoria para prototipos finales y series de producción cortas.

d) EEPROM.

La grabación es similar a las memorias OTP y EPROM, pero el borrado es mucho más sencillo al poderse efectuar de la misma forma que el grabado, o sea, eléctricamente. Sobre el mismo zócalo del grabador puede ser programada y borrada tantas veces como se quiera, lo cual la hace ideal en la enseñanza y en la creación de nuevos proyectos. El fabuloso PIC16C84 dispone de 1 k palabras de memoria EEPROM para contener instrucciones y también tiene algunos bytes de memoria de datos de este tipo para evitar que cuando se retira la alimentación se pierda la información.
Aunque se garantiza 1’ 000, 000 de ciclos de escritura/borrado en una EEPROM, todavía su tecnología de fabricación tiene obstáculos para alcanzar capacidades importantes y el tiempo de escritura de las mismas es relativamente grande y con elevado consumo de energía.

e) FLASH.

Se trata de una memoria no volátil, de bajo consumo, que puede escribir y borrar en circuito al igual que las EEPROM, pero suelen disponer de mayor capacidad que estas últimas. El borrado sólo es posible con bloques completos y no se puede realizar sobre posiciones concretas. En las FLASH se garantizan 1000 ciclos de escritura/borrado.
Son muy recomendables en aplicaciones en las que sea necesario modificar el programa a lo largo de la vida del producto, como consecuencia del desgaste o cambios de piezas, como sucede con los vehículos.

2.4.2.3 MEMORIA DE DATOS.

Los datos que manejan los programas varían continuamente, y esto exige que la memoria que les contiene debe ser de lectura y escritura, por lo que la memoria RAM estática (SRAM) es la más adecuada, aunque sea volátil.
Hay microcontroladores que también disponen como memoria de datos una de lectura y escritura no volátil, del tipo EEPROM. De esta forma, un corte en el suministro de la alimentación no ocasiona la pérdida de la información, que está disponible al reiniciarse el programa.

La memoria tipo EEPROM y la tipo Flash pueden escribirse y borrarse eléctricamente. Sin necesidad de quitar el circuito integrado del zócalo del grabador pueden ser escritas y borradas numerosas veces.

2.4.2.4 LÍNEAS DE E/S PARA LOS CONTROLADORES PERIFÉRICOS.

A excepción de dos terminales destinadas a recibir la alimentación, otras dos para el cristal de cuarzo, que regula la frecuencia de trabajo, y una más para provocar el Reset, las restantes terminales de un microcontrolador sirven para soportar su comunicación con los periféricos externos que controla.

Las líneas de E/S que se adaptan con los periféricos manejan información en paralelo y se agrupan en conjuntos de ocho, que reciben el nombre de Puertas. Hay modelos con líneas que soportan la comunicación en serie.
Otros disponen de conjuntos de líneas que implementan puertas de comunicación para diversos protocolos como el USB, RS-232, etc.

2.4.2.5 RECURSOS AUXILIARES.

Según las aplicaciones a las que orienta el fabricante cada modelo de microcontrolador, incorpora una diversidad de complementos que refuerzan la potencia y la flexibilidad del dispositivo. Entre los recursos más comunes se citan los siguientes:

a) **Circuito de reloj**, encargado de generar los impulsos que sincronizan el funcionamiento de todo el sistema.
b) **Temporizadores**, orientados a controlar tiempos.
c) **Perro Guardián** o watchdog, destinado a provocar una reinicialización cuando el programa queda bloqueado.
d) **Conversores AD y DA**, para poder recibir y enviar señales analógicas.
e) **Comparadores analógicos**, para verificar el valor de una señal analógica.
f) Sistema de protección ante fallos de alimentación
g) **Estado de reposo**, en el que el sistema queda congelado y el consumo de energía se reduce al mínimo.

2.5 COMUNICACIONES SERIE.

Una manera de conectar a dos dispositivos es mediante comunicaciones serie asíncronas. En ellas los bits de datos se transmiten "en serie" (uno de tras de otro) y cada dispositivo tiene su propio reloj. Previamente se ha acordado que ambos dispositivos transmitirán datos a la misma velocidad.

Las siguientes secciones nos servirán como fundamento para comprender la interacción de las antenas-lectoras (Readers) de la marca Wavetrend con nuestro sistema de control en el siguiente capítulo.

2.5.1 COMUNICACIONES SERIE ASÍNCRONAS.

Los datos serie se encuentran encapsulados en tramas de la forma:

Primero se envía un bit de inicio (start), a continuación los bits de datos (primero el bit de mayor peso) y finalmente los bits de paro (stop).

El número de bits de datos y de bits de paro es uno de los parámetros configurables, así como el criterio de paridad par o impar para la detección de errores. Normalmente, las comunicaciones serie tienen los siguientes parámetros: 1 bit de inicio (start), 8 bits de datos, 1 bit de paro (stop) y sin paridad. En la figura 2.14 se puede ver un ejemplo de la transmisión del dato binario 10011010. La línea en reposo está a nivel alto:

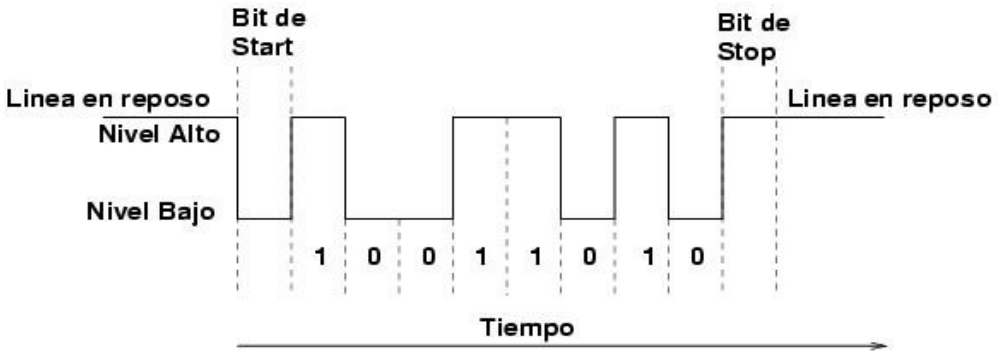


Figura 2.14 Ejemplo de transmisión de un dato binario.

2.5.1.1 NORMA RS-232.

La Norma RS-232 fue definida para conectar una computadora a un MODEM. Este estándar fue diseñado en los años 60's para comunicar un equipo terminal de datos o DTE (Data Terminal Equipment, la computadora en este caso) y un equipo de comunicación de datos o DCE (Data Communication Equipment, habitualmente un módem). Además de transmitirse los datos de forma serie asíncrona, son necesarias una serie de señales adicionales, que se definen en la norma. Las tensiones empleadas están comprendidas entre +15/-15 volts.

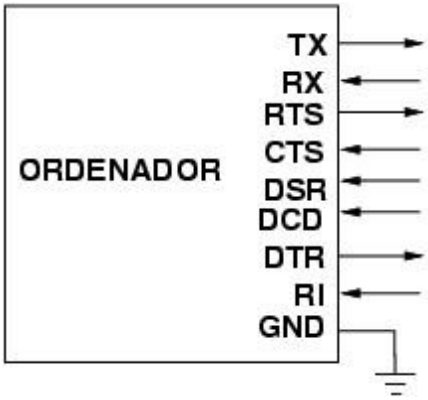


Figura 2.15 Transmisión y recepción de datos hacia y de una PC con la norma RS-232.

Características:

- 9 (DB9) o 25 (DB25) pines de señal.
- Conector de DTE debe ser macho y el conector de DCE hembra.
- Los voltajes para un nivel lógico alto están entre -3V y -15V, y un nivel bajo +3V y +15V.
- Los voltajes más usados son +12V/-12V, +9V/–9V
- Dependiendo de la velocidad de transmisión empleada, es posible tener cables de hasta 15 metros.
- Velocidades: 300, 600, 1200, 2400, 4800 y 9600 bps

Las señales más utilizadas en un puerto serie, además de las de transmisión y recepción de datos, son las siguientes:

- DTR (Data-Terminal-Ready)
- DSR (Data-Set-Ready)
- RTS (Request-To-Send)
- CD (Carrier-Detect)
- CTS (Clear-To-Send)

Estas señales las analizaremos más adelante en la sección dedicada al conector tipo DB9 serie, en este mismo capítulo.

2.5.1.2 CONEXIÓN DE UN MICROCONTROLADOR AL PUERTO SERIE DE UNA PC.

Para conectar el PC a un microcontrolador por el puerto serie se utilizan las señales Tx, Rx y GND. El PC utiliza la norma RS232, por lo que los niveles de tensión de los pines están comprendidos entre +15 y -15 voltios. Los microcontroladores normalmente trabajan con niveles TTL (0-5v). Es necesario por tanto intercalar un circuito que adapte los niveles:

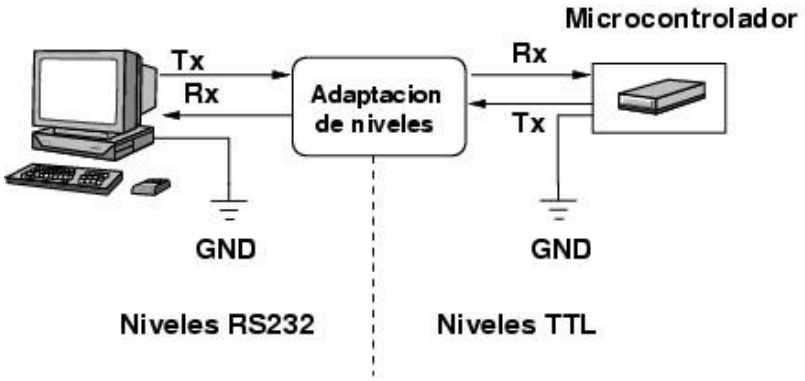


Figura 2.16 Conexión de un microcontrolador a una PC con la norma RS-232.

Uno de estos circuitos muy utilizado para adaptar los niveles de tensión, es el **MAX232**.

2.5.1.3 EL CONECTOR SERIE DB9 DE LA PC.

En las PCs hay conectores DB9 macho de 9 pines, a través de los cuales se conectan los dispositivos al puerto serie. Los conectores hembra que se conectan tienen una colocación de pines diferente, de manera que se conectan el pin 1 del macho con el pin 1 del hembra, el pin2 con el 2, etc. (Figura 2.17).

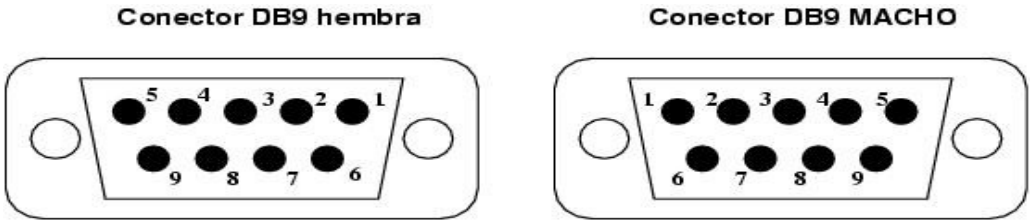


Figura 2.17 Conectores DB9 tipo macho y tipo hembra.

La información asociada a cada uno de los pines es la siguiente:

Número de pin DB9	Señal
1	**DCD** (Data Carrier Detect)
2	**RX**
3	**TX**
4	**DTR** (Data Terminal Ready)
5	**GND**
6	**DSR** (Data Sheet Ready)
7	**RTS** (Request To Send)
8	**CTS** (Clear To Send)
9	**RI** (Ring Indicator)

Figura 2.18 Tabla de los pines en un conector DB9.

Las señales utilizadas en el DB9 son:

- 4 DTR (Data-Terminal-Ready): El PC indica al modem que esta encendido y listo para enviar datos.
- 6 DSR (Data-Set-Ready): El modem indica al PC que esta encendido y listo para transmitir o recibir datos.
- 7 RTS (Request-To-Send): El PC indica al modem activando esta señal a 1 cuando tiene un caracter listo para enviar.
- 1 DCD (Carrier-Detect): El modem pone esta señal a 1 cuando el ordenador remoto esta listo, esta señal indica que hay portadora en la linea anlógica que hay entre ambos modems.
- 8 CTS (Clear-To-Send): El modem esta preparado para transmitir datos. El ordenador empezara a enviar datos al modem.
- 3 Tx: El modem recibe datos desde el PC.
- 2 RxD: El modem transmite datos al PC.
- 9 RI: Indicador de llamada. Señal que manda el modem hacia el PC cuando recibe una llamada.
- 5 GND: Tierra

Recordar que la PC es el DTE (Data Terminal Equipment) y el modem o equipo equivalente es el DCE (Data Communication Equipment).

Las señales TXD, DTR y RTS son de salida, mientras que RXD, DSR, CTS y DCD son de entrada. La referencia para todas las señales es GND (GrouND, o tierra). Finalmente, existen otras señales como RI (Indicador de Llamada).

2.5.1.4 El CHIP MAX 232.

Este chip permite adaptar los niveles RS232 y TTL, permitiendo conectar una PC con un microcontrolador. Sólo es necesario este chip y 4 capacitores electrolíticos de 22 micro-faradios. El esquema se puede ver en la figura 2.19:

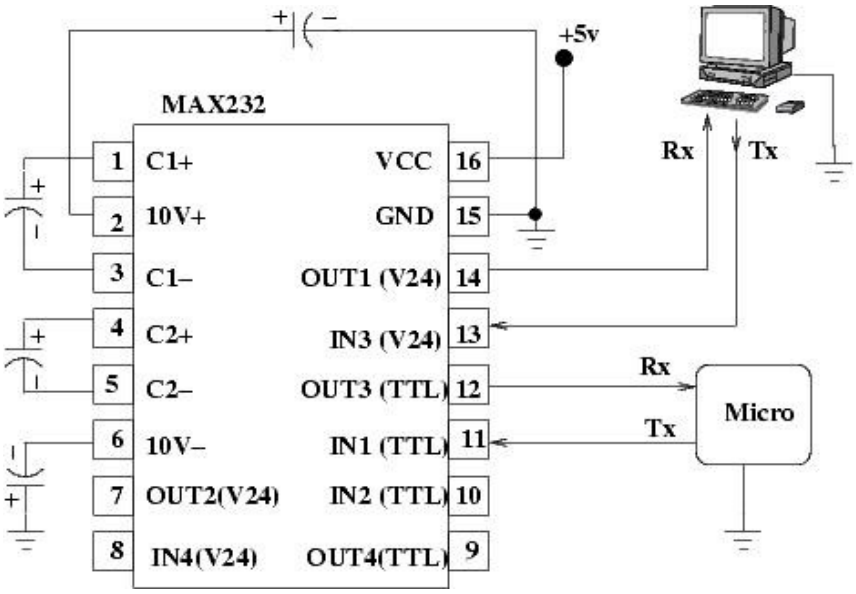


Figura 2.19 Adaptación de niveles de voltaje para la norma RS-232.

En la Figura 2.20 podemos apreciar que el chip MAX232 se encuentra presente en el interior de un Lector Wavetrend, esto para la parte de interfaz con alguna computadora, a través del puerto serie (RS-232).

Figura 2.20 Chip MAX232 para la interfaz con la PC al interior de un Lector Wavetrend.

2.5.1.5 NORMA RS-485.

(RS485 o EIA-485). Estándar de comunicaciones multipunto de la EIA (Electronic Industries Association). Es una especificación eléctrica (de la capa física en el modelo OSI) de las conexiones half-duplex y two-wire. Éste estándar es ideal para transmitir a altas velocidades sobre largas distancias y a través de canales ruidosos, ya que reduce los ruidos que aparecen en los voltajes producidos en la línea de transmisión. Gracias a estas capacidades, es posible crear redes de dispositivos conectados a un solo puerto RS-485. Esta capacidad, y la gran inmunidad al ruido, hacen que este tipo de transmisión serial sea la elección de muchas aplicaciones industriales que necesitan dispositivos distribuidos en red conectados a una PC u otro controlador para la colección de datos. El medio físico de transmisión es un par entrelazado y la transmisión diferencial permite múltiples dispositivos, dando la posibilidad de una configuración multipunto.

Al tratarse de un estándar bastante abierto permite muchas y muy diferentes configuraciones y utilizaciones. La norma RS-485 usa señales eléctricas diferenciales, en comparación con señales referenciadas a tierra como en RS-232.

Características:

- ➔ Velocidad máxima de 100kbps hasta 1.2m y de 35Mbps hasta 12m.
- ➔ Señales de cómo máximo 6V y de como mínimo 200mV.
- ➔ Conexión multipunto
- ➔ Alimentación única de +5V
- ➔ Rango de bus de -7V a +12V

La relación de los pines de un conector DB9 serial y las funciones de estos en la norma RS-485 se muestran a continuación:

```
 -------------
\ 1 2 3 4 5 /
  \ 6 7 8 9 /
    ---------
```

Figura 2.21 Distribución de pines en un conector tipo DB9.

Funciones de los pines en RS-485 y RS-422:

Datos: TXD+ (pin 8), TXD- (pin 9), RXD+ (pin 4), RXD- (pin 5)
Handshake: RTS+ (pin 3), RTS- (pin 7), CTS+ (pin 2), CTS- (pin 6)
Tierra: GND (pin 1)

El método de comunicación usado por RS-232 requiere de una conexión muy simple, utilizando sólo tres líneas: Tx, Rx, y GND. Sin embargo, para que los datos puedan ser transmitidos correctamente ambos extremos deben estar sincronizados a la misma velocidad. Aun y cuando este método es más que suficiente para la mayoría de las aplicaciones, es limitado en su respuesta a posibles problemas que puedan surgir durante la comunicación; por ejemplo, si el receptor se comienza a sobrecargar de información. Es en estos casos cuando el intercambio de pulsos de sincronización, o handshaking, es útil. En esta sección se describirán brevemente las dos formas más populares de handshaking con RS-232: handshaking for software y handshaking por hardware.

1. Handshaking por software: Ésta será la primera forma de handshaking que discutiremos. Esta forma de sincronización utiliza bytes de datos como caracteres de control. Las líneas necesarias para la comunicación siguen siendo Tx, Rx, y GND, ya que los caracteres de control se envían a través de las líneas de transmisión como si fueran datos.

2. Handshaking por hardware: El segundo método de handshaking utiliza líneas de hardware. De manera similar a las líneas Tx y Rx, las líneas RTS/CTS y DTR/DSR trabajan de manera conjunta siendo un par la entrada y el otro par la salida. El primer par de líneas es RTS (por sus siglas en inglés, Request to Send) y CTS (Clear to Send). Cuando el receptor está listo para recibir datos, cambia la línea RTS a estado alto; este valor será leído por el transmisor en la línea CTS, indicando que está libre para enviar datos. El siguiente par de líneas es DTR (por sus siglas en inglés, Data Terminal Ready) y DSR (Data Set Ready). Estas líneas se utilizan principalmente para comunicación por modem, permiten al puerto serial y modem indicarse mutuamente su estado. Por ejemplo, cuando el modem se encuentra preparado para que la PC envíe datos, cambia la línea DTR a estado alto indicando que se ha realizado una conexión por la línea de teléfono. Este valor se lee a través de la línea DSR y la PC comienza a enviar datos. Como regla general, las líneas DTR/DSR se utilizan para indicar que el sistema está listo para la comunicación, mientras que las líneas RTS/CTS se utilizan para paquetes individuales de datos.

Con toda la información escrita en el presente capítulo y después de haber analizado las normas de comunicación serie asíncronas, nos encontramos en condiciones de hacer un análisis de los Lectores RFID Wavetrend en el siguiente capítulo III.

CAPÍTULO III: ANÁLISIS DE LOS LECTORES WAVETREND L-RX201.

3.1 LECTORES DE LA MARCA WAVETREND.

Una gran opción para ingresar al nuevo mundo de la tecnología RFID son los Lectores de la marca Wavetrend, los cuales (a manera de exponer en esta tesis uno de tantos ejemplos) son utilizados actualmente en la Torre de Ingeniería de la UNAM, teniendo un Lector por cada piso, conectados todos en red con protocolo tipo RS-485 y toda esta red es controlada vía una PC (Personal Computer) a través de un software especial fabricado por la misma marca Wavetrend.

Estos Lectores son utilizados como medida de seguridad para tener controlado y detectado el equipo con el cual se labora en esta Torre, tales como pueden ser las PC's de escritorio, laptops (PC's Portátiles), scanners, impresoras, osciloscopios digitales y equipos de medición entre otros, los cuales tienen un alto valor monetario y es necesario mantenerlos vigilados.

Los Lectores (Readers) L-RX201 fueron desarrollados por Wavetrend® Technologies Ltd. para el uso conjuntamente con Link-it® Tagging System. Los Lectores son utilizados en el sistema para realizar las funciones siguientes:

- Recibir, descifrar y validar datos de Tags (etiquetas electrónicas).
- Salida de datos relevantes de los Tags fuera de la red de Lectores hacia la PC, por ejemplo.

La utilización de la tecnología RFID tiene múltiples ventajas como se vio en el capítulo anterior.

Con estos Lectores en cada piso de la Torre es posible, por ejemplo, contar el número de equipos que se tienen en cada uno de los mismos pisos o activar alguna alarma si algún equipo se aleja del radio de frecuencia en cada Lector. Todo esto es posible debido a que cada equipo en la Torre de Ingeniería tiene en su interior una etiqueta identificadora o Tag la cual permite tener toda la información del aparato o equipo en cuestión si es que así se desea. Los Lectores RFID permiten entonces la detección de Tags en un radio de circunferencia específico, el cual por supuesto se puede programar.

Cuando alguno de los Tags, (o en concreto algún aparato costoso con un Tag en su interior) sale de este radio de circunferencia, los Lectores envían información a la PC que controla la Red de estos Lectores y esa señal puede ser a su vez utilizada para activar alarmas en el edificio, hacer funcionar algunas cámaras con tecnología Wireless IP que se encuentren conectadas a la red de la PC de control y así poder saber quién intenta extraer algún equipo del piso en donde se detectó el faltante, activar o desactivar puertas, etcétera.

3.1.1 UN VISTAZO A LOS LECTORES L-RX201.

El Lector L-RX201 abarca la funcionalidad y las características siguientes:

- Módulo de RF (receptor y demodulador de RF).
- Microcontrolador.

- LED’s Indicadores en los conectadores de salida.

Un diagrama básico de un Lector se ilustra en la figura 3.1.

El Lector tipo RS485 se puede utilizar con las antenas siguientes de Wavetrend®:

- LAN -100/200/300/400
- Cualquier otra antena 433MHz que tenga una resistencia de terminación de 50 ohms para la unión antena-lector. (Ver Fig. 3.2)

Figura 3.1. Lector L-RX201. Cortesía Wavetrend Technologies Ltd.

Figura 3.2 Partes que componen un Lector Wavetrend de izquierda a derecha: antena, circuitería y carcasa.

3.1.2 DIAGRAMA FUNCIONAL.

Un Lector L-RX201 tiene el siguiente diagrama funcional:

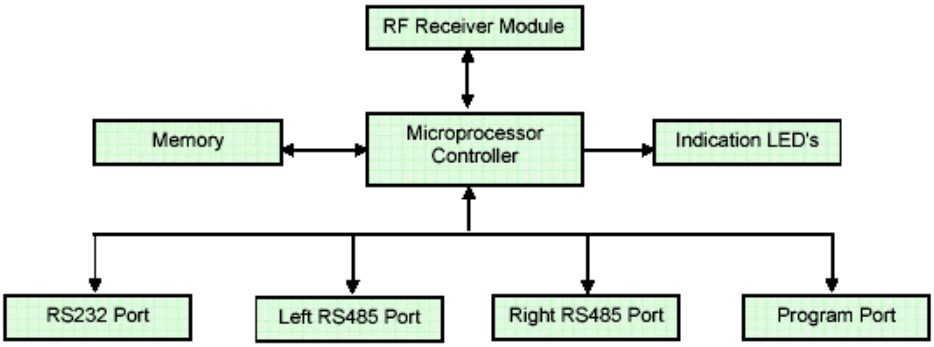


Figura 3.3 Diagrama funcional de un Lector L-RX201. Cortesía Wavetrend Technologies Ltd.

Este receptor consiste en un microcontrolador que se comunica directamente con el módulo de recepción de RF y conecta con el mundo exterior a través de 3 interfaces serie. El análisis y diseño del sistema para el control de los Lectores RFID es el objetivo de esta tesis para permitir la interacción con el usuario, así como el ajuste de parámetros directamente a través del puerto serie vía la red de Lectores. Los datos se pueden enviar independientemente a y desde los 2 puertos RS485 y estos serán conectados simultáneamente en el puerto RS232 de nuestra PC.

Todas las conexiones entre Lectores en una red se hacen a través de 2 conectores RJ45, los cuales se encuentran protegidos contra ruido de la **EMI** (Electro-magnetic Interference) y de **ESD** (Electrostatic Discharge).

En la figura 3.4 podemos observar el interior de un Lector Wavetrend dividido en dos tabletas: tableta que contiene la interfaz con la PC a través del chip MAX232 además del circuito modulador/demodulador de radiofrecuencia y el conector tipo BNC para la antena y tableta para el microcontrolador PIC16F870

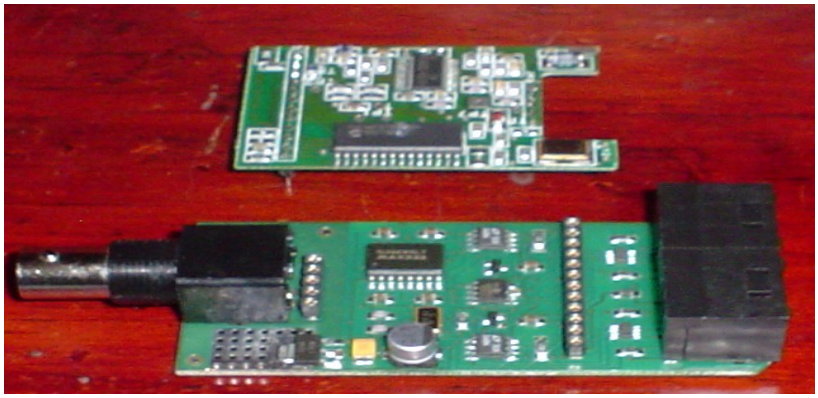


Figura 3.4 Tabletas al interior de un Lector Wavetrend.

3.2 RED DE LECTORES RFID L-RX201.

3.2.1 ESTRUCTURA DE UNA SOLA RED DE LECTORES.

Los Lectores L-RX201 en red están conectados en serie o cascada. Es decir, el Lector 1 está conectado con el Lector 2; el Lector 2 está conectado con el Lector 3 y así sucesivamente. La red puede manejar un máximo de 255 lectores debido a la limitación de direcciones. Las comunicaciones entre los Lectores se hacen a través de 2 cables en conexión RS485. La conexión del Lector 1 a la PC se puede realizar a través de la norma RS232 con el puerto izquierdo RS485 del mismo Lector (Figura 3.5).

Cabe señalar que una red de Lectores no necesita conectarse con una PC, sino que puede ser otro dispositivo tal como una computadora de bolsillo, un microcontrolador, una Laptop, etc.

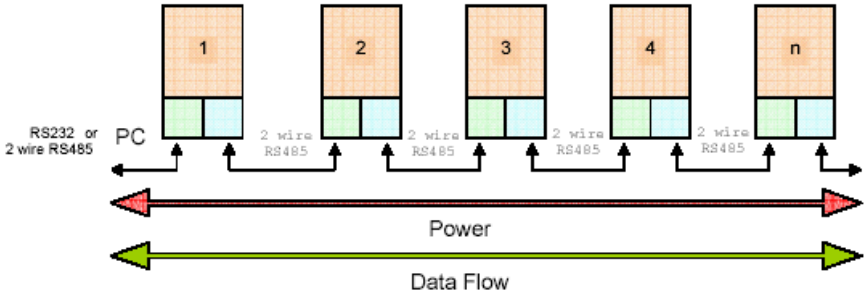


Figura 3.5 Red de Lectores L-RX201. Cortesía Wavetrend Technologies Ltd.

La energía también se lleva a cada Lector en el cable que se conecta entre ellos. Cada conexión entre los Lectores entonces consistiría de 4 cables.

Sólo se deben conectar 4 cables entre Lectores.

Los 4 cables serían entonces:

- Voltaje.
- GND (Tierra).
- RS485 +
- RS485 -

Se han arreglado estas conexiones de manera tal que el cable forme una conexión recta y no requiera torcer o cambiar algún cable entre los Lectores. Esto hace la instalación de esta red muy simple. Puesto que cada conexión RS485 entre Lectores es teóricamente una red separada, las distancias entre ellos pueden ser de hasta 1.2 kilómetros según el estándar RS485, dependiendo de la calidad del cable usado. Las resistencias de terminación RS485 se incluyen en cada Lector y por lo tanto no necesitan ser agregadas. Los paquetes de información fluyen bidireccionalmente en esta red, permitiendo a cada Lector satisfacer sus funciones.

Esta red funciona a velocidades de hasta 115200 baudios y tan bajo como 9600 baudios. Estas velocidades se pueden alterar, usando el software comercial o, por supuesto, se puede utilizar el diseño de esta tesis para realizar un software propio, por medio de los comandos

predefinidos y mencionados en la sección 3.4 de este capítulo. La comunicación de datos a y desde la PC debe estar a la misma velocidad. La red de los Lectores se puede proveer de energía de puntos múltiples dentro de la misma red.
Cuando se utiliza más de una fuente de alimentación, las fuentes se deben conectar vía la misma línea de tierra. En caso de que las fuentes de alimentación no estén conectadas comúnmente, el voltaje diferenciado máximo entre las líneas de fuente negativas no debe exceder los 7 volts. Si se utiliza más de una fuente de alimentación es importante que el primer comando enviado después de que la energía se haya estabilizado sea el comando para reinicializar la red (Reset Network Command).

3.2.2 OPERACIÓN BÁSICA DE UNA RED DE LECTORES.

Los datos que pasan a través de la red se arreglan en formato de paquete (explicado más adelante). Cada Lector puede ser individualmente direccionado dentro de una red vía 2 técnicas de direccionamiento, o a una red entera se le puede asignar la dirección correspondiente como valor máximo 255. Estas técnicas de dirección también serán explicadas más adelante en este capítulo.

Figura 3.6 Los puertos RS485 en cada Lector están definidos como puerto izquierdo y puerto derecho. Cortesía Wavetrend Technologies Ltd.

Los puertos RS485 en cada lector se definen como un puerto izquierdo y un puerto derecho. Los datos transmitidos del puerto derecho de un lector serán recibidos en el puerto izquierdo del siguiente Lector a la derecha y viceversa. Con esta clase de configuración, es posible que el Lector controle la dirección en que están fluyendo los datos.

La red completa se vería como en la figura 3.7:

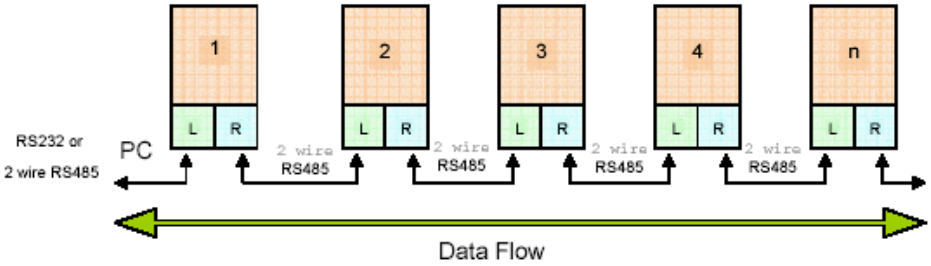


Figura 3.7 Red completa de Lectores. Cortesía Wavetrend Technologies Ltd.

Esta red trabaja en una operación del tipo Command/Response (Comando/Respuesta). Es decir, un comando se envía hacia un Lector específico en una dirección de izquierda a derecha, mientras que la respuesta del Lector se envía entonces de derecha a la izquierda. Simplemente, los comandos funcionan de izquierda a la derecha, mientras que las respuestas funcionan de derecha a la izquierda (Fig. 3.8).

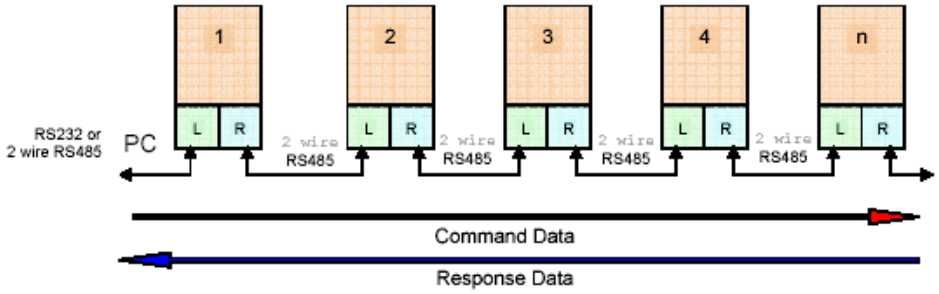


Figura 3.8 Flujo de información Comando/Respuesta. Cortesía Wavetrend Technologies Ltd.

Cada Comando y Respuesta tienen un formato característico en un paquete específico de datos con su respectivo error incluido en el formato del protocolo. Los datos pueden fluir solamente en una dirección a la vez, puesto que el hardware está utilizando un solo puerto serie para controlar todos estos datos seriales.
Debido a que este sistema puede determinar la dirección de los datos, es posible establecer qué Lector es el número 1 y de allí en adelante las direcciones consecutivas pueden ser establecidas. Según lo mencionado anteriormente, hay 2 métodos distintos de dirigirse a un lector al enviar un comando. La primera dirección se llama la IDENTIFICACION DEL NODO (NODE ID). Ésta es la dirección eléctrica del Lector y es definida automáticamente por su posición en la red. Se le asignaría entonces al primer Lector (muy a la izquierda) la identificación ID = 1 e incrementa el NODE ID consecutivamente a la derecha hasta un máximo de 255. La segunda dirección se llama la IDENTIFICACION DEL LECTOR (READER ID). Esta dirección es definida por el usuario y escrita a la memoria permanente de cada Lector y puede ser a partir de la 1 y hasta 255.
Si un Lector es removido de la red, los NODE ID's naturalmente cambiarán. La función de la READER ID es por lo tanto permitir una asignación de dirección constante y permanente a cada uno de éstos Lectores. Lógicamente, es necesario utilizar direcciones de NODE ID en el paquete del comando para asignar una READER ID a un lector en específico. Las NODE ID'S proporcionan un método seguro para tener siempre acceso al lector correcto si se sabe la estructura de la red.

3.2.3 ESTABLECIENDO LA DIRECCIÓN DE IDENTIFICACIÓN DE CADA NODO AUTOMÁTICAMENTE (NODE ID).

En dirección de izquierda a derecha, al momento de energizar la red, las identificaciones de NODO (NODE ID) se establecen automáticamente en la posición de la conexión de los Lectores respecto a la misma red. Básicamente, al ser conectados los Lectores en red, el Lector 1 es automáticamente establecido, al ser el más cercano a la PC de control. De allí, cada Lector subsecuente tiene asignada automáticamente su identificación de NODO. Este proceso entero toma cerca de 3,5 segundos en dirección hacia la derecha y consiste en las secuencias siguientes:

1. Cada Lector reconoce su posición en la red y fija su flujo de datos en dirección hacia la derecha.

2. Cada Lector entonces envía continuamente asteriscos ' * ' s desde su puerto RS485 del lado derecho y se prepara para recibir este carácter en su puerto RS485 del lado izquierdo. Este carácter se envía en un solo valor cada 25ms. Esto evita cualquier error, con respecto a que algún

Lector pudiera bloquearse sobre la secuencia de datos en la posición correcta de izquierda a derecha.

3. Si un ' * ' se recibe en el puerto izquierdo RS485, el TAG LED (indicador auxiliar) estará iluminado para indicar esto. Cada Lector, a excepción del Lector 1, recibirá por lo tanto este carácter en su puerto izquierdo RS485.

4. El Lector 1 se encontrará ahora estabilizado.

5. Este proceso de enviar asteriscos a través del puerto derecho RS485 dura alrededor de 1 segundo.

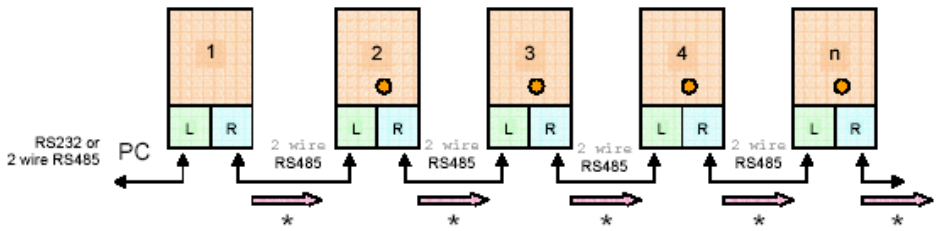


Figura 3.9 Detección de los lectores de forma automática. Cortesía Wavetrend Technologies Ltd.

6. Una vez que se haya estabilizado el Lector 1, éste comenzará a enviar continuamente su identificación de NODO a través del puerto derecho RS485. Éste será un ' 1 continuo ' y dura cerca de 500ms.

7. Cada Lector siguiente en la red, al recibir este valor, lo asume para ser la identificación de NODO del Lector a su izquierda, agrega 1 a este valor y lo asigna como su propia identificación de NODO. Esta nueva identificación de NODO se pasa otra vez a la derecha. El envío continuo de las identificaciones de NODO a la derecha hace que se tenga un efecto subsecuente en la red, hasta que a cada Lector se le ha asignado su identificación de NODO y la red se haya estabilizado.

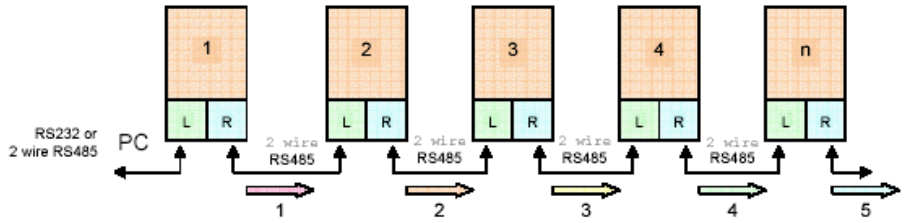


Figura 3.10 Establecimiento automático de cada NODE ID cortesía Wavetrend Technologies Ltd.

8. Aunque el Lector 1 envía solamente la identificación 1 de NODO hacia la red durante 500ms, sigue habiendo en el resto de la misma la asignación de cada Lector dependiendo de la identificación de NODO anterior. Esto previene cualquier dato erróneo del Lector 1 que corrompa la asignación entera de la red.

3.2.4 CONTROL DE LOS PAQUETES DE DATOS.

Según lo explicado anteriormente, los datos se pasan hacia la izquierda o hacia la derecha de esta red en un formato de paquete específico. Sería demasiado lento para un Lector recibir un paquete completo antes de transmitirlo al Lector siguiente. Debido a esto, el paquete es transmitido byte por byte como éste va siendo recibido. Esto logrará entonces una relación entre la velocidad de transmisión y del número de Lectores en la red. Después de iniciar la conexión de red, todos los Lectores estarán fijados en una "condición ociosa", esperando un comando o el paquete de la respuesta. Aquí esperarán hasta que reciban un comando de la PC o de un paquete de respuesta de otro Lector. El estado del LED de IDLE/OK estará encendido en esta etapa. Los paquetes del comando (de izquierda a derecha) se transmiten a través de toda la longitud de la red.

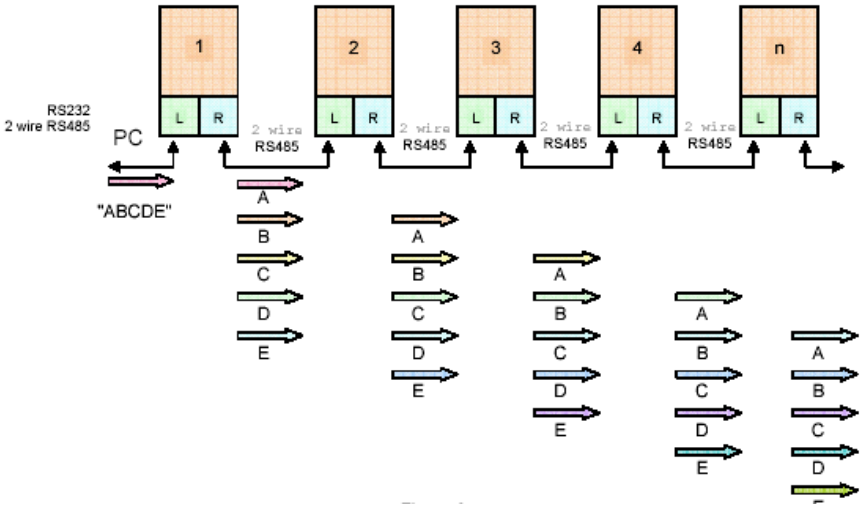


Figura 3.11 Transmisión de la información en una red de Lectores. Cortesía Wavetrend Technologies Ltd.

En este "modo ocioso", los datos se pueden recibir de cualquier dirección. Puesto que hay solamente un puerto serial en el microprocesador, el Byte Cabecero (Header Byte) recibido determinará de qué dirección se está recibiendo el paquete. Una vez que se haya recibido este byte, los drivers de la comunicación harán inmediatamente fluir los datos en la dirección correcta. Permanecerán en esta dirección hasta que se haya recibido el paquete entero, o las comunicaciones hayan terminado en tiempo, por lo que el Lector caerá nuevamente dentro del modo "ocioso". Mientras que estos datos se están transmitiendo bajo la red, cada Lector monta el paquete en su memoria para el análisis cuando esté terminado. Una vez que se reciba un paquete completo de comando, el contenido se analiza para reconocer los errores a tratar. Si esto es un paquete del comando, el(los) Lector(es) tratado(s) entonces responderá(n). Después de recibir un paquete completo, cada Lector entrará inmediatamente dentro del modo "ocioso" (Recibiendo de ambas direcciones). En este modo, el paquete siguiente (podría haber cualquiera) será pasado automáticamente en la dirección correcta.

Si hay una pausa de más de 25 [ms] entre cualquier byte del paquete de datos, cada Lector cambiará automáticamente dentro del modo "ocioso" y no hará caso de ese paquete. De esta manera, ninguna falta de comunicación dará lugar a que la red se pasme.

Si un paquete espontáneo de respuesta es iniciado, por ejemplo, por un reinicio de red como resultado de una intermitencia de energía, el paquete será pasado de nuevo a la PC.

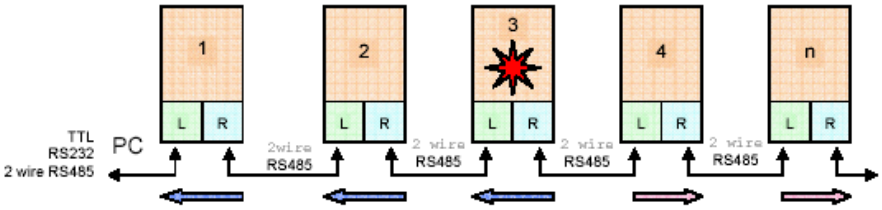


Figura 3.12 Transmisión de la información en una red de Lectores. Cortesía Wavetrend Technologies Ltd.

Cada vez que se recibe un paquete válido, el LED de recepción RX LED destellará. Esto se puede utilizar para determinar si están ocurriendo las comunicaciones válidas. Los paquetes de la respuesta ahora se envían detrás exactamente de la misma manera que el paquete de comando, pero en la dirección opuesta. Cada Lector montaría el paquete como si éste pasara a través de él para determinar cuando la respuesta esté completa. Una vez que haya recibido una respuesta completa, el Lector cambiará al modo "ocioso", listo para el paquete siguiente de algún comando.

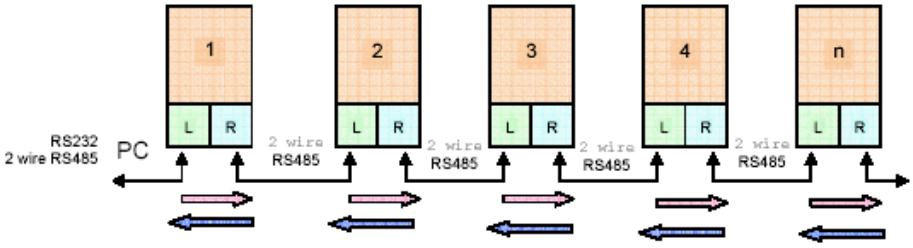


Figura 3.13 Estado de reposo u "ocioso" de una red de Lectores. Cortesía Wavetrend Technologies Ltd.

3.2.5 AUTODETECCIÓN.

Los dispositivos lógico-inalterables internos llevan rutinas para permitir que este sistema sea auto-detector y se encuentre funcionando libremente. Cuando el Lector 1 se encuentra con la auto-detección habilitada, determinará automáticamente el número de Lectores conectados en su red y enviará secuencialmente comandos hacia la misma para recibir los datos de la etiqueta o Tag detectados por cada Lector usando el comando correspondiente para obtener el paquete de bytes del Tag (0x06), el cual veremos más adelante. Los paquetes del Tag pasan a través del Lector 1 y hacia la PC. En este modo, la PC recibirá el flujo de los bytes de información sin la necesidad de utilizar el sistema de control para los Lectores. Esto se puede utilizar en los sistemas o aplicaciones que no tienen ninguna capacidad de enviar datos hacia la red, o la necesidad de supervisar solamente los datos de la etiqueta o Tag, haciéndole un sistema muy simple sin rutinas complicadas.

Cuando no hay datos disponibles provenientes de alguna etiqueta o Tag, será enviado después un paquete vacío de respuesta. Esto permite al software de supervisión determinar si hay algunos Lectores en la red que no estén respondiendo, y sugerir la acción apropiada.

Puesto que estos datos se están recibiendo continuamente, existe un mecanismo que le podrá permitir a la PC parar este proceso de lectura contínua para poder enviar comandos a la red de Lectores. Este mecanismo puede ser detenido enviando desde la PC un carácter de rotura (en

este caso, ' * ') al Lector 1. Si el lector 1 recibe este carácter, parará el envío de comandos para la auto-detección del paquete de información de algún Tag detectado hasta que se le envíe al Lector el comando para reiniciar la auto-detección o el comando correspondiente para el reinicio de la red entera.

Puesto que la velocidad de transmisión es en términos generales rápida, podría suceder que al enviar el carácter ‘ * ’ éste no sea detectado por el Lector y por ende el servicio de auto-detección no se detendría. Esto puede ser un problema, así que el método sugerido es enviar simplemente una gran cantidad de caracteres ' * ' desde la PC hacia el Lector (entre de 400 y 500) para lograr a como de lugar con nuestro cometido. De este modo, no podrá haber fallo en la detención del auto-poleo o auto-detección. Para garantizar un envío claro y evitar que el proceso se bloquee, es necesario enviar los caracteres (ASCII 255 + ' * ') x 400. El carácter ASCII 255 (espacio) nos asegurará que no haya posibilidad de un error en nuestro intento por parar la auto-detección. En términos hexadecimales tendríamos 400 x (0xFF + 0x2A).

Romper con la secuencia de la auto-detección no altera la bandera (flag) de la misma auto-detección en los datos EEPROM, simplemente suspende la auto-búsqueda de información. Después de un reinicio en la red o un fallo de energía, éste sistema comenzará la auto-búsqueda otra vez a menos que ésta se inhabilite con el comando apropiado, el cual veremos más adelante en este capítulo a forma de resumen y de manera detallada en el Anexo B al final de esta tesis.

Las secuencias de auto-detección comienzan con un reajuste completo de las comunicaciones del sistema para asegurarse de que la red está conectada y configurada correctamente. Esto sucede cada vez que se energiza al Lector o red de Lectores.

3.3 PROTOCOLOS Y DIRECCIONES.

Como mencionamos anteriormente en este capítulo, el flujo de datos en esta Red de Lectores tiene un formato de paquete específico. Este formato está definido de la siguiente manera:

Los paquetes de comando y de respuesta son esencialmente idénticos a excepción del carácter en el cabecero del paquete. Este carácter distinto en el cabecero permite al equipo que recibe todos los datos distinguir entre el comando y los paquetes de la respuesta. Puesto que el puerto RS232 envía y recibe estos datos siempre, será necesario aquí poder distinguir entre estos paquetes.

El formato de paquete esta compuesto de la siguiente manera:

1. **HEADER:** byte del cabecero que indica si el paquete es un comando (0xAA) o una respuesta (0x55).
2. **LENGTH:** byte que indica el número de bytes contenidos en la sección de datos en el paquete.
3. **NETWORK ID:** byte que indica la identificación de red (0x00-0xFF).
4. **RECEIVER ID o READER ID:** byte que indica la identificación de Lector (0x00-0xFF).
5. **NODE ID:** byte que indica la identificación de nodo en la red (0x00-0xFF).
6. **COMMAND:** byte que indica el comando que se está enviando. Para este modelo de Lectores existen 21 comandos con sus respectivas respuestas y el valor de este byte así como su correspondiente comando se presenta en la sección 3.4 del presente capítulo a manera de resumen y en forma detallada en el Anexo B de esta tesis.
7. **DATA:** byte o conjunto de bytes de datos enviados o recibidos hacia o desde algún Lector, respectivamente. Si no se reciben datos, el valor definido para este byte es 0x00. Si se reciben datos, éstos pueden llegar a los 64 bytes de datos recibidos.
8. **CHECKSUM:** byte para la comprobación de la integridad de los datos y se obtiene a través de la operación lógica XOR desde el byte LENGTH hasta el último byte de datos en DATA. esto es:

CHECKSUM= [length] XOR [network ID] XOR [reader ID] XOR [node ID] XOR [command] XOR [data]... XOR [data]

A continuación los formatos correspondientes:

3.3.1 PAQUETES DE COMANDOS.

0xAA	Data Length	Network ID	Receiver ID	Node ID	Command Byte	Data	Checksum

All bytes are HEX Values

Packet	Name	Bytes
1.	Header	1 Byte [0xAA]
2.	Length	1 Byte (Number of Bytes in data section)
3.	Network ID	1 Byte
4.	Receiver ID	1 Byte
5.	Node ID	1 Byte
6.	Command	1 Byte
7.	Data	Up to 64 Bytes of Data
8.	Checksum	1 Byte (XOR from Length to Last Data Byte), CRC

Figura 3.14 Paquetes de comandos. (Todos los bytes se manejan en valores hexadecimales)

3.3.2 PAQUETES DE RESPUESTA.

0x55	Data Length	Network ID	Receiver ID	Node ID	Command Byte	Data	Checksum

All bytes are HEX values.

Packet	Name	Bytes
9.	Header	1 Byte [0x55]
10.	Length	1 Byte (Number of Bytes in data section)
11.	Network ID	1 Byte
12.	Receiver ID	1 Byte
13.	Node ID	1 Byte
14.	Command	1 Byte
15.	Data	Up to 64 Bytes of Data
16.	Checksum	1 Byte (XOR from Length to Last Data Byte), CRC

Figura 3.15 Paquetes de respuesta. (Todos los bytes se manejan en valores hexadecimales)

3.3.3 TÉCNICAS DE DIRECCIONAMIENTO DE LOS LECTORES RFID L-RX201.

El sistema de direccionamiento para estas redes de Lectores se hizo tan flexible como fue posible. Esto permitirá varias configuraciones y mantendrá el sistema abierto para una expansión posterior si se desea. Al dirigirse a un Lector, hay 3 direcciones a considerar:

1. NETWORK ID (Identificación de la Red) - Identifica la red (usada en configuraciones de multi- red).

2. READER ID (Identificación de Lector) - El usuario define la dirección para un Lector específico. Es una dirección permanente definida por el usuario a través de un comando.

3. NODE ID (Identificación de Nodo) - Dirección de hardware. Esta dirección es definida por la posición de los Lectores respecto a la red.

Cada uno de estos 3 bytes son configurables en los bytes correspondientes al paquete del comando específico para esta acción. Si se configura un valor cero en cualquiera de estas posiciones, es una indicación hacia la memoria del Lector de no hacer caso de este parámetro, y utiliza solamente las direcciones restantes para determinar a qué Lector es al que se está refiriendo. Si un valor de 255 se pone en cualesquiera de estas direcciones, es una indicación al Lector de que este comando está refiriéndose a todos los elementos de la red o redes interconectadas.
Un valor 255 en el byte de la Identificación de la red (NETWORK ID) significa que el comando está refiriéndose a todas las redes. Mientras que, un valor de 255 en el byte de la Identificación del Lector (READER ID) o un byte en la Identificación del nodo (NODE ID) significa que el comando está haciendo referencia a todos los Lectores en esa red específica.

El direccionamiento de Identificación de Nodo (NODE ID) tiene mayor prioridad sobre un direccionamiento de Identificación de Lector (READER ID). Esto es, un direccionamiento de Identificación de Nodo válido será aceptado antes que una Identificación de Lector válida.

Un diagrama de flujo simple para esta lógica sería como el de la figura 3.16:

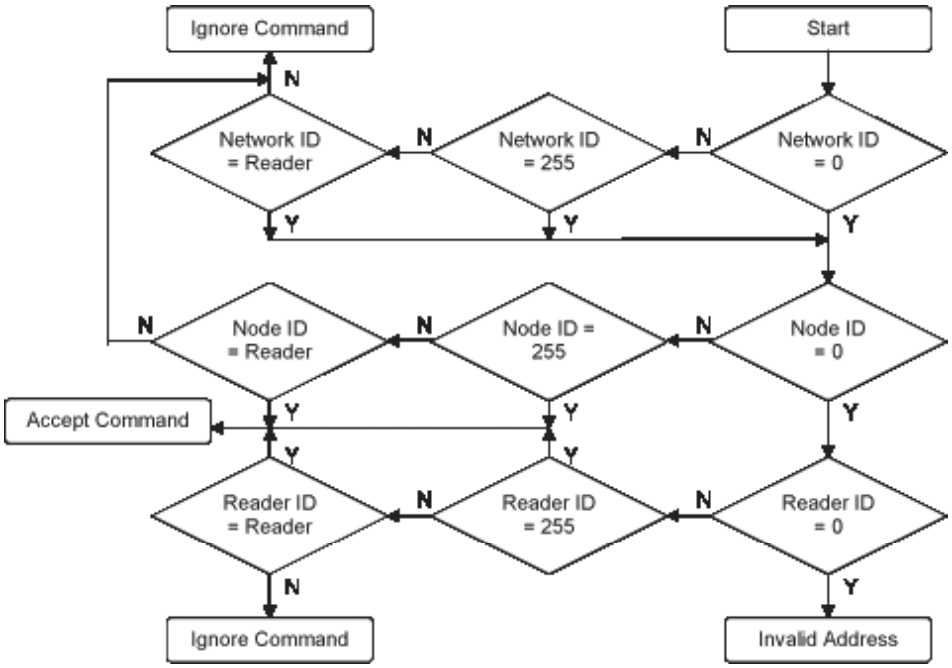


Figura 3.16 Diagrama de flujo para la asignación y reconocimiento de Lectores en una red.

Ejemplos de direccionamiento son los siguientes:

Network ID	Receiver ID	Node ID	Effect
0	0	0	Invalid - will have no response
0	12	0	Access Reader with Reader ID = 12
0	0	5	Access Reader 5 on the network
0	12	5	Access Reader 5 on the network. Reader ID address is ignored
1	0	0	Invalid - will have no response
1	4	0	Access Reader with Reader ID = 4 on Network 1
1	0	123	Access Reader 123 on the Network 1
255	12	0	Access all Readers with Reader ID = 12 on all the networks
255	0	45	Access Reader 45 on all the Networks
255	255	0	Access all possible readers
255	0	255	Access all possible readers
0	255	255	Access all possible readers
			Etc ...

Figura 3.17 Ejemplos de direccionamiento para los Lectores L-RX201.|

3.4 COMANDOS DE CONTROL.

3.4.1 LISTA DE COMANDOS.

A continuación se muestra una pequeña lista en forma de resumen de los comandos con sus respectivas respuestas (se profundiza sobre cada comando en el **Anexo D** de la presente tesis):

Value	Function	Expect Response
0x00	Reset Network	Reply Packet
0x01	Start/Enable Auto Polling	Continuous
0x02	Disable Auto Polling	Reply Packet
0x03	Ping Reader	Reply Packet + Error Number
0x04	Set Network ID	Reply Packet
0x05	Set Reader ID	Reply Packet
0x06	Get Tag Packet	Tag Packet
0x07	Get RSSI Value	Reply Packet + RSSI
0x08	Set RSSI Value	Reply Packet
0x09	Set Site Code	Reply Packet
0x0A	Get Site Code	Reply Packet + Site Code
0x0B	Set Receiver Gain	Reply Packet
0x0C	Get Receiver Gain	Reply Packet + Gain
0x0D	Set Alarm Filter	Reply Packet
0x0E	Get Alarm Filter	Reply Packet + Status
0x0F	Get Number of invalid Tags	Reply Packet + Counter
0x10	Get Supply Voltage	Reply Packet + Voltage
0x11	Start RF white noise calculation	Reply Packet
0x12	Get RF white noise result	Reply Packet + Result
0xFE	Set Baud Rate	No Reply – Broadcast only
0xFF	Get Version Information	Reply Packet + Version Data

Figura 3.18 Lista de comandos para los Lectores L-RX201.

Gracias a los capítulos I, II y III, ya nos encontramos con la información necesaria para, en los siguientes capítulos IV y V, se den las bases tecnológicas de diseño y se llegue por fin al objetivo de análisis y diseño de nuestro sistema de control, respectivamente.

CAPÍTULO IV: HERRAMIENTAS TECNÓLOGICAS DE DISEÑO.

Una vez finalizada la etapa de análisis de los Lectores RFID y de conocer nuestros antecedentes generales, es importante enfocarnos en las herramientas tecnológicas de diseño para así determinar los requerimientos de software para el desarrollo del sistema. Además de determinar con qué sistema operativo, manejador de base de datos y lenguaje de programación se va a trabajar, es necesario explicar un poco la teoría orientada a objetos y las bases del Lenguaje de Modelado Unificado (UML) a través del cual se realizará en el siguiente capítulo el análisis y diseño de nuestro sistema.

4.1 PROGRAMACIÓN ORIENTADA A OBJETOS.

La Programación Orientada a Objetos (POO u OOP según sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo, y encapsulamiento. Su uso se popularizó recién a principios de la década de 1990. Actualmente varios lenguajes de programación soportan la orientación a objetos.

Los objetos son entidades que combinan estado, comportamiento e identidad. El estado está compuesto de datos, y el comportamiento por procedimientos o métodos. La identidad es una propiedad de un objeto que lo diferencia del resto. La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar. De esta forma, un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan ni deben separarse el estado y el comportamiento.

Los métodos y atributos están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta. El programador debe pensar indistintamente en ambos conceptos, sin separar ni darle mayor importancia a ninguno de ellos. Hacerlo podría resultar en seguir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que la manejen por el otro. De esta manera se estaría llegando a una programación estructurada camuflada en un lenguaje de programación orientado a objetos. Esto difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada sólo se escriben funciones que procesan datos. Los programadores que emplean éste nuevo paradigma, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

La programación orientada a objetos es una nueva forma de programar que trata de encontrar una solución a estos problemas. Introduce nuevos conceptos, que superan y amplían conceptos antiguos ya conocidos. Entre ellos destacan los siguientes:

- **Objeto:** entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos). Corresponden a los objetos reales del

mundo que nos rodea, o a objetos internos del sistema (del programa). Es una instancia a una clase.

- **Clase:** definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.

- **Método:** algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

- **Evento:** un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento, a la reacción que puede desencadenar un objeto, es decir la acción que genera.
- **Mensaje:** una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

- **Propiedad o atributo:** contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

- **Estado interno:** es una propiedad invisible de los objetos, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos).
- **Componentes de un objeto:** atributos, identidad, relaciones y métodos.

- **Representación de un objeto:** un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.

Hay un cierto desacuerdo sobre exactamente qué características de un método de programación o lenguaje le definen como "orientado a objetos", pero hay un consenso general en que las características siguientes son las más importantes:

- **Abstracción:** Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción.

- **Encapsulamiento:** Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.

- **Principio de ocultación:** Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que especifica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas. Algunos

lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos.

- **Polimorfismo:** comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++.

- **Herencia:** las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que reimplementar su comportamiento. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay herencia múltiple; esta característica no está soportada por algunos lenguajes (como Java).

4.2 CICLOS DE DESARROLLO DEL SOFTWARE.

El software es un elemento lógico, por lo que tiene unas características muy diferentes a las del hardware:

El software se desarrolla, no se fabrica en el sentido clásico de la palabra. Ambas actividades se dirigen a la construcción de un "producto", pero los métodos son diferentes. Los costes del software se encuentran en la ingeniería, esto implica que los proyectos no se pueden gestionar como si lo fueran de fabricación. A mediados de la década de 1980, se introdujo el concepto de "fábrica de software", que recomienda el uso de herramientas para el desarrollo automático del software.

Si se representa gráficamente la proporción de fallos en función del tiempo, para el hardware se tiene la figura conocida como "curva de bañera". Al principio de su vida hay bastantes fallos (normalmente por defectos de diseño y/o fabricación), una vez corregidos se llega a un nivel estacionario (bastante bajo). Sin embargo conforme pasa el tiempo, aparecen de nuevo, por efecto de suciedad, malos tratos, temperaturas extremas y otras causas. El hardware empieza a estropearse. El software no se estropea. La gráfica de fallos en función del tiempo, tendría forma de caída desde el principio, hasta mantenerse estable por tiempo casi indefinido (véase Fig. 4.1). El software no es susceptible a los males del entorno que provocan el deterioro del hardware. Los efectos no detectados harán que falle el programa durante las primeras etapas de su vida, sin embargo una vez corregidas, no se producen nuevos errores. Aunque no se estropea, si puede deteriorarse. Esto sucede debido a los cambios que se efectúan durante su vida.

Cuando un componente hardware se estropea, se cambia por otro que actúa como una "pieza de repuesto", mientras que para el software, no es habitual este proceso, lo cual significa que el mantenimiento de los programas es muy complejo.

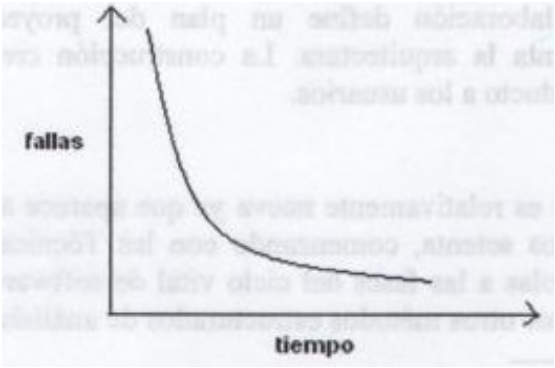


Figura 4.1 Gráfica de fallos vs tiempo para el desarrollo de software.

La mayoría del software se construye a medida, en vez de ensamblar componentes previamente creados. Por contra en el hardware se dispone de todo tipo de circuitos integrados, para fabricar de manera rápida un equipo completo. Los ingenieros de software no disponen de esta comodidad, aunque ya se están dando los primeros pasos en esta dirección, que facilitaría tanto el desarrollo de aplicaciones informáticas.

La formalización del proceso de desarrollo se define como un marco de referencia denominado ciclo de desarrollo del software o ciclo de vida del desarrollo del software o ciclo de vida del desarrollo.

Se puede describir como, "el período de tiempo que comienza con la decisión de desarrollar un producto software y finaliza cuando se ha entregado éste". Este ciclo, por lo general incluye:

- Fase de requisitos
- Fase de diseño
- Fase de implementación
- Fase de prueba
- Fase de instalación
- Fase de aceptación

El ciclo de desarrollo software se utiliza para estructurar las actividades que se llevan a cabo en el desarrollo de un producto software. A pesar de que no hay acuerdo acerca del uso y la forma del modelo, este sigue siendo útil para la comprensión y el control del proceso.

Seguidamente se exponen las distintas aproximaciones de desarrollo de software, en función del tipo de ciclo de vida:

A) Aproximación convencional.

Se introdujo como una técnica rígida para mejorar la calidad y reducir los costos del desarrollo de software. Tradicionalmente es conocido como "modelo en cascada", porque su filosofía es completar un paso con un alto grado de exactitud, antes de empezar el siguiente.

Esquemáticamente se puede representar de la siguiente forma, como se aprecia en la figura 4.2:

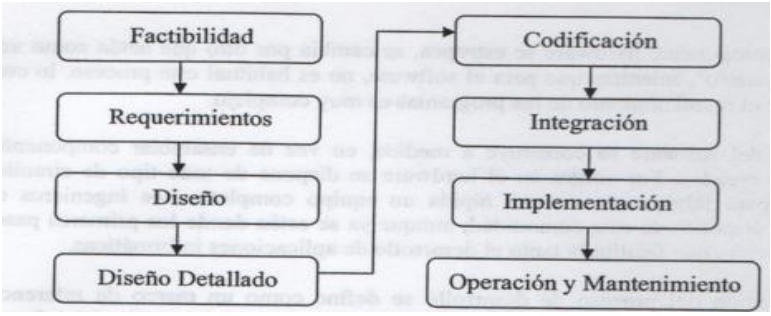


Figura 4.2 Esquema de la aproximación convencional.

Donde:

FACTIBILIDAD: Definir un concepto preferente para el producto de software y determinar su factibilidad de ciclo de vida y superioridad frente a otros conceptos.

REQUERIMIENTOS: Elaborar una especificación completa y validada de las funciones requeridas, sus interfaces y el rendimiento del producto de software.

DISEÑO: Elaborar una especificación completa y validada de la arquitectura global hardware-software, de la estructura de control y de la estructura de datos del producto, así como un esquema de los manuales de usuarios y planes de test.

DISEÑO DETALLADO: Elaborar una especificación completa y verificada de la estructura de control, de la estructura de datos, de las interfaces de relación, dimensionamiento y algoritmos claves de cada componente de programa (rutina con un máximo de 100 instrucciones fuentes).

CODIFICACION: Construir un conjunto completo y verificado de componentes de programas.

INTEGRACION: Hacer funcionar el producto de software compuesto de componentes de programa.

IMPLEMENTACION: Hacer funcionar el sistema global hardware-software incluyendo conversión de programas y datos, instalación y capacitación.

OPERACION Y MANTENIMIENTO: Hacer funcionar una nueva versión del sistema global.

En cada caso, "verificación" tienen la siguiente acepción: establecer la verdad de la correspondencia entre un producto de software y su especificación. Es decir: ¿ESTAMOS CONSTRUYENDO CORRECTAMENTE EL PRODUCTO?

Los principales problemas que se han detectado en esta aproximación son debidos a que se comienza estableciendo todos los requisitos del sistema:

- En muchas ocasiones no es posible disponer de unas especificaciones correctas desde el primer momento, porque puede ser difícil para el usuario establecer al inicio todos los requisitos.

- En otras hay cambio de parecer de los usuarios sobre las necesidades reales cuando ya se ha comenzado el proyecto, siendo probables los verdaderos requisitos no se reflejen en el producto final.

- Otro de los problemas de esta aproximación es que los resultados no se ven hasta muy avanzado el proyecto, por lo tanto la realización de modificaciones, si ha habido un error, es muy costosa.

Esta aproximación es la más empleada por los ingenieros informáticos, aunque ha sido muy criticada, y de hecho se ha puesto en duda su eficacia. Entre los problemas que se pueden encontrar con este modelo, se tienen:

- Los proyectos raras veces siguen el modelo secuencial que se supone. Los cambios pueden causar confusión.
- Es difícil disponer en principio de todos los requisitos. Este modelo presenta dificultades en el momento de acomodar estas incertidumbres.
- La versión operativa de los programas no está disponible hasta que el proyecto está muy avanzado. Un error importante puede ser desastroso, si se descubre al final del proceso.
- Los responsables del desarrollo siempre se retrasan innecesariamente. Algunos integrantes del equipo de desarrollo han de esperar a otros para completar tareas pendientes.

B) Aproximación prototipo.

Es habitual que en un proyecto software no se identifiquen los requisitos detallados de entrada, procesamiento o salida. En otros casos no se está seguro de la eficiencia de un algoritmo, o de la forma en que se ha de implantar la interfaz hombre-máquina.

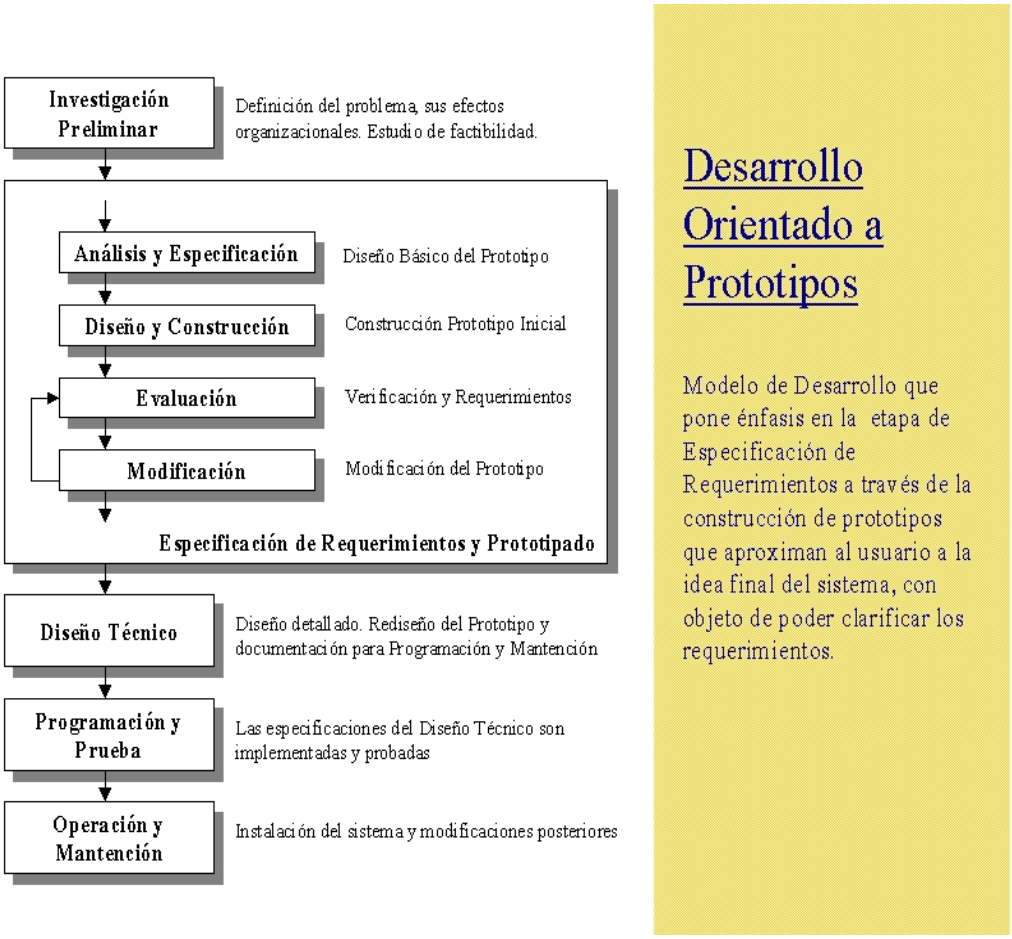


Figura 4.3 Desarrollo orientado a prototipos.

En casos así, lo habitual es construir un prototipo, que idealmente sirviera como mecanismo para identificar los requisitos del software. Esta aproximación consiste en realizar la fase de definición de requisitos del sistema en base a estos tres factores:

- Un alto grado de iteración
- Un muy alto grado de participación del usuario
- Un uso extensivo de prototipos

Las premisas clave de esta aproximación son:

- Que los prototipos constituyen un medio mejor de comunicación que los modelos en papel

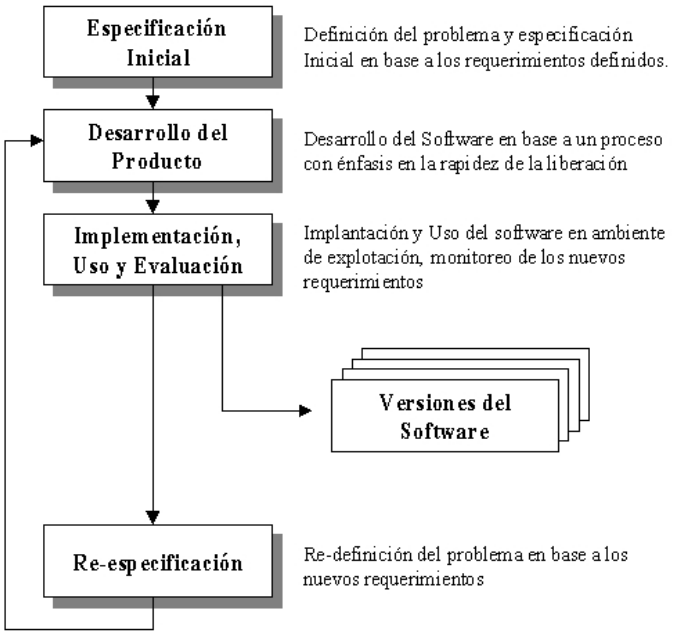
- Que la iteración es necesaria para canalizar, en la dirección correcta, el proceso de aprendizaje. Esta aproximación se enfoca a mejorar la efectividad del proceso de desarrollo y no a mejorar la eficacia de ese proceso.

- El problema, es que los usuarios finales, ven lo que parece ser una versión de trabajo del software, sin considerar que no es la versión definitiva y por lo tanto no se han considerado aspectos de calidad o facilidad de mantenimiento. Cuando se les dice, que el producto es a partir de entonces cuando se debe de empezar a "fabricar", no lo entiende y empieza de nuevo con ajustes, lo cual hace este proceso muy lento.

C) Aproximación evolutiva.

En esta aproximación el énfasis está en lograr un sistema flexible y que se pueda expandir de forma que se pueda realizar muy rápidamente una versión modificada del sistema cuando los requisitos cambien.

Se diferencia de la aproximación anterior, en que en esta los requisitos cambian continuamente, lo cual implicaría en el caso previo que las iteraciones no tendrían fin.



Desarrollo Evolutivo

Modelo de desarrollo que, a diferencia del de Prototipos, busca reemplazar el viejo sistema con uno nuevo que tendría la propiedad de satisfacer los nuevos requerimientos lo más rápido posible. El desarrollo evolutivo asume que los requerimientos están sujetos a cambios continuos y que la estrategia para enfrentar aquello pasa por un reflejo, también continuo, de aquellos cambios.

Figura 4.4 Desarrollo evolutivo.

D) Aproximación incremental.

Es un concepto muy parecido al de desarrollo evolutivo, y frecuentemente comprendido en la aproximación del desarrollo evolutivo. Se comienza el desarrollo del sistema para satisfacer un subconjunto de requisitos especificados. Las últimas versiones prevén los requisitos que faltan. De esta forma se logra una rápida disponibilidad del sistema, que aunque incompleto, es utilizable y satisface algunas de las necesidades básicas de información.

La diferencia con la aproximación anterior es que en este caso cada versión parte de una previa sin cambios pero con nuevas funciones, mientras que la aproximación evolutiva cada vez se desarrolla una nueva versión de todo el sistema.

Un ejemplo de este paradigma se tiene en el desarrollo de una aplicación sencilla, como es un editor de textos. En el primer incremento se podría desarrollar con un reducido conjunto de funciones, como las funciones básicas de gestión de archivos. En un segundo incremento, se puede incluir la gestión avanzada de textos. Y en un tercer incremento se pondría la corrección ortográfica.

E) Aproximación espiral.

Nace con el objetivo de captar lo mejor de la aproximación convencional y de la de prototipo, añadiendo un nuevo componente, el análisis de riesgos. Esquemáticamente se puede ilustrar mediante una espiral, con cuatro cuadrantes que definen actividades (Fig. 4.5).

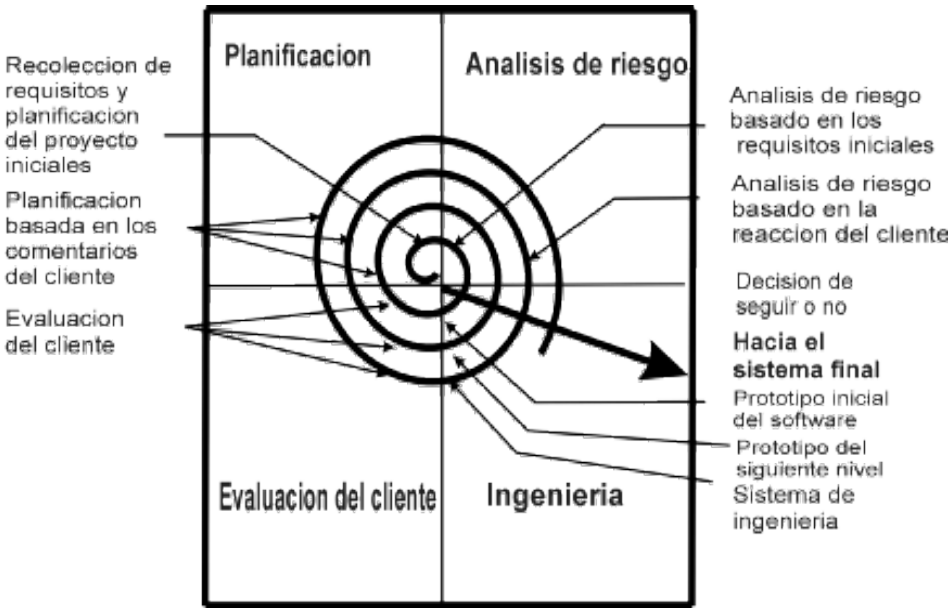


Figura 4.5 Desarrollo en espiral.

En la primera vuelta de la espiral se definen los objetivos, las alternativas y las restricciones y se analizan y se identifican los riesgos. Si como consecuencia del análisis de riesgo se observa que hay incertidumbre sobre el problema entonces en la actividad correspondiente a la ingeniería se aplicará la aproximación prototipo cuyo beneficio principal es el de reducir la incertidumbre de la naturaleza del problema de información y los requerimientos que los usuarios establecen para la solución a ese problema.

Al final de esta primera vuelta alrededor de la espiral el usuario evalúa los productos obtenidos y puede sugerir modificaciones. Se comenzaría avanzando alrededor del camino de la espiral realizando las cuatro actividades indicadas a continuación. En cada vuelta de la espiral, la actividad de ingeniería se desarrolla mediante la aproximación convencional o ciclo de desarrollo en cascada o mediante la aproximación de prototipos.

- Recolección de requisitos y planificación del proyecto final.
- Análisis de riesgo basado en los requisitos iniciales.
- Prototipo inicial de software.
- Evaluación del cliente.
- Planificación basada en los comentarios del cliente.
- Prototipo del siguiente nivel.
- Segunda evaluación del cliente.
- Segunda planificación basada en los comentarios del cliente.
- Análisis de riesgo basado en la reacción del cliente.
- Sistema de Ingeniería.
- Tercera evaluación del cliente.
- Tercera planificación basada en los comentarios del cliente.

F) Aproximación basada en transformaciones.

Con la aparición de las herramientas CASE junto con los generadores de código, el ciclo de desarrollo software en cascada ha cambiado a un ciclo de vida basado en transformaciones.

CASE (Computer Aided Software Engineering), en castellano "Ingeniería de software Asistida por Computadora", es un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información.

La utilización de herramientas CASE afecta a todas las fases del ciclo de vida del software. Este ciclo de vida se puede considerar como una serie de transformaciones. Primero se definen los requisitos del sistema, seguidamente existe un proceso de transformación que hace que la especificación se convierta en un diseño lógico del sistema. Posteriormente, este sufre otro proceso de transformación para lograr un diseño físico, es decir que responda a la tecnología destino.

La tecnología CASE propone que estos procesos de transformación sean lo más automatizado posible. Sus ventajas son:

- Posibilidad de comprobación de errores en etapas iniciales de desarrollo.
- Posibilidad de realizar el mantenimiento en el ámbito de especificación.
- Soporte de seguimiento de los requisitos
- Soporte de reusabilidad.
- Potencia la especificación orientada al problema.

4.3 PROCESO UNIFICADO DE RATIONAL (RUP).

Proceso Unificado de Rational (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

El RUP no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada organización o de cada proyecto.

También se conoce por este nombre al software desarrollado por Rational, hoy propiedad de IBM, el cual incluye información entrelazada de diversos artefactos y descripciones de las diversas actividades. Está incluido en el Rational Method Composer (RMC), que permite la personalización de acuerdo a necesidades.

El RUP (Proceso Racional Unificado) está basado en 5 principios clave que son:

1) Adaptar el proceso.

El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.

2) Balancear prioridades.

Los requerimientos de los diversos participantes pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos.

3) Demostrar valor iterativamente.

Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados.

4) Elevar el nivel de abstracción.

Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes 4GL o marcos de referencia (frameworks) por nombrar algunos. Esto evita que los ingenieros de software vayan directamente de los requisitos a la codificación de software a la medida del cliente, sin saber con certeza qué codificar para satisfacer de la mejor manera los requerimientos y sin comenzar desde un principio pensando en la reutilización del código. Un alto nivel de abstracción también permite discusiones sobre diversos niveles y soluciones arquitectónicas. Éstas se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con el lenguaje UML.

5) Enfocarse en la calidad.

El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

4.2.1 CICLO DE VIDA DEL SOFWARE.

El ciclo de vida RUP es una implementación del Desarrollo en espiral. Fue creado ensamblando los elementos en secuencias semi-ordenadas. El ciclo de vida organiza las tareas en fases e iteraciones.

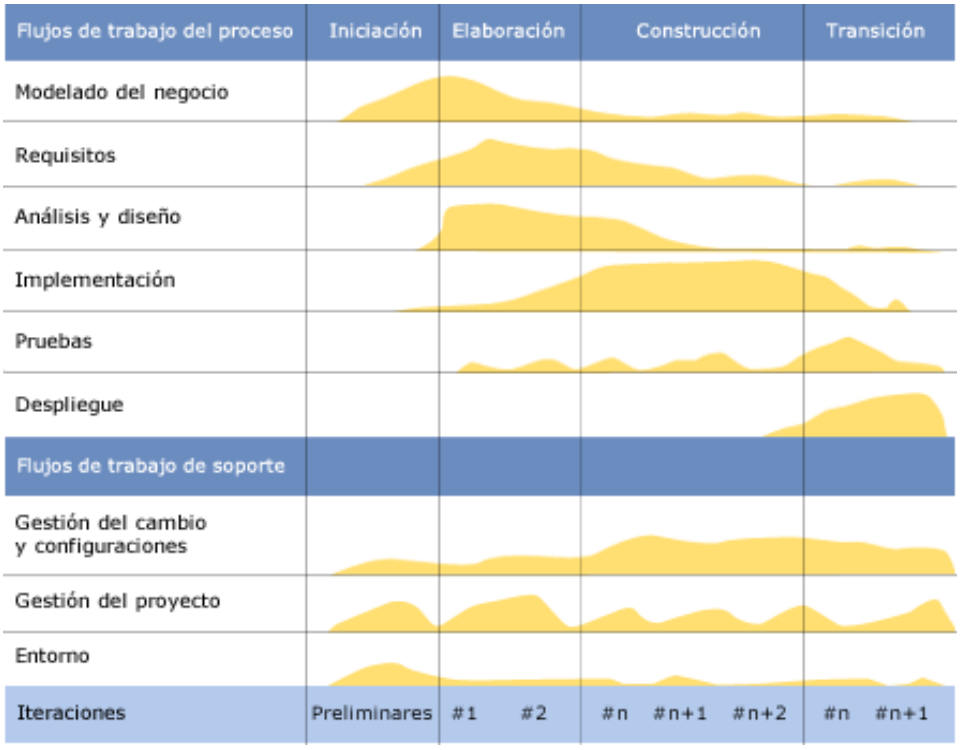


Figura 4.6 Ciclo de vida RUP.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades. En la Figura muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

Las primeras iteraciones (en las fases de Inicio y Elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos.

Durante la fase de inicio las iteraciones hacen mayor énfasis en actividades de modelado del negocio y de requerimientos.

En la fase de elaboración, las iteraciones se orientan al desarrollo de la baseline de la arquitectura, abarcan más los flujos de trabajo de requerimientos, modelo de negocios (refinamiento), análisis, diseño y una parte de implementación orientado a la baseline de la arquitectura.

En la fase de construcción, se lleva a cabo la elaboración del producto por medio de una serie de iteraciones.

Para cada iteración se selecciona algunos Casos de Uso, se refina su análisis y diseño y se procede a su implementación y pruebas. Se realiza una pequeña cascada para cada ciclo. Se realizan tantas iteraciones hasta que se termine la implementación de la nueva versión del producto.

En la fase de transición se pretende garantizar que se tiene un producto preparado para su entrega a la comunidad de usuarios.

Como se puede observar en cada fase participan todas las disciplinas, pero que dependiendo de la fase el esfuerzo dedicado a una disciplina varía.

Las principales características del ciclo de vida RUP son las siguientes:

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo)
- Pretende implementar las mejores prácticas en Ingeniería de Software
- Desarrollo iterativo
- Administración de requisitos
- Uso de arquitectura basada en componentes
- Control de cambios
- Modelado visual del software
- Verificación de la calidad del software

El RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

Las fases para el ciclo de vida RUP son las siguientes:

- Establece oportunidad y alcance
- Identifica las entidades externas o actores con las que se trata
- Identifica los casos de uso

RUP, a nivel de fases, contiene una estructura estática y una estructura dinámica. La estructura estática de RUP comprende 2 aspectos importantes por los cuales se establecen las disciplinas:

Proceso: Las etapas de esta sección son: (Revise nuevamente la gráfica)

- Modelado de negocio
- Requisitos
- Análisis y Diseño
- Implementación
- Pruebas
- Despliegue

Soporte: En esta parte nos conseguimos con las siguientes etapas:

- Gestión del cambio y configuraciones
- Gestión del proyecto
- Entorno

La estructura dinámica de RUP es la que permite que este sea un proceso de desarrollo fundamentalmente iterativo, y en esta parte se ven inmersas las 4 fases descritas anteriormente:

- Inicio (También llamado Concepción)
- Elaboración
- Desarrollo(También llamado Implementación,Construcción)
- Cierre (También llamado Transición)

En lo que se refiere a la metodología, ésta comprende tres frases claves:
Dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

En lo referente a dirigido por los casos de uso, está enfocado hacia el cliente y se utilizan con algunas modificaciones tal vez, hasta la fase de pruebas.

4.4 EL LENGUAJE DE MODELADO UNIFICADO (UML).

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en un estándar de la industria del software, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la cual basar la construcción de sus herramientas. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores. Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa (principalmente Booch, OMT y OOSE). UML ha puesto fin a las llamadas “guerras de métodos” que se mantuvieron a lo largo de los 90's, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

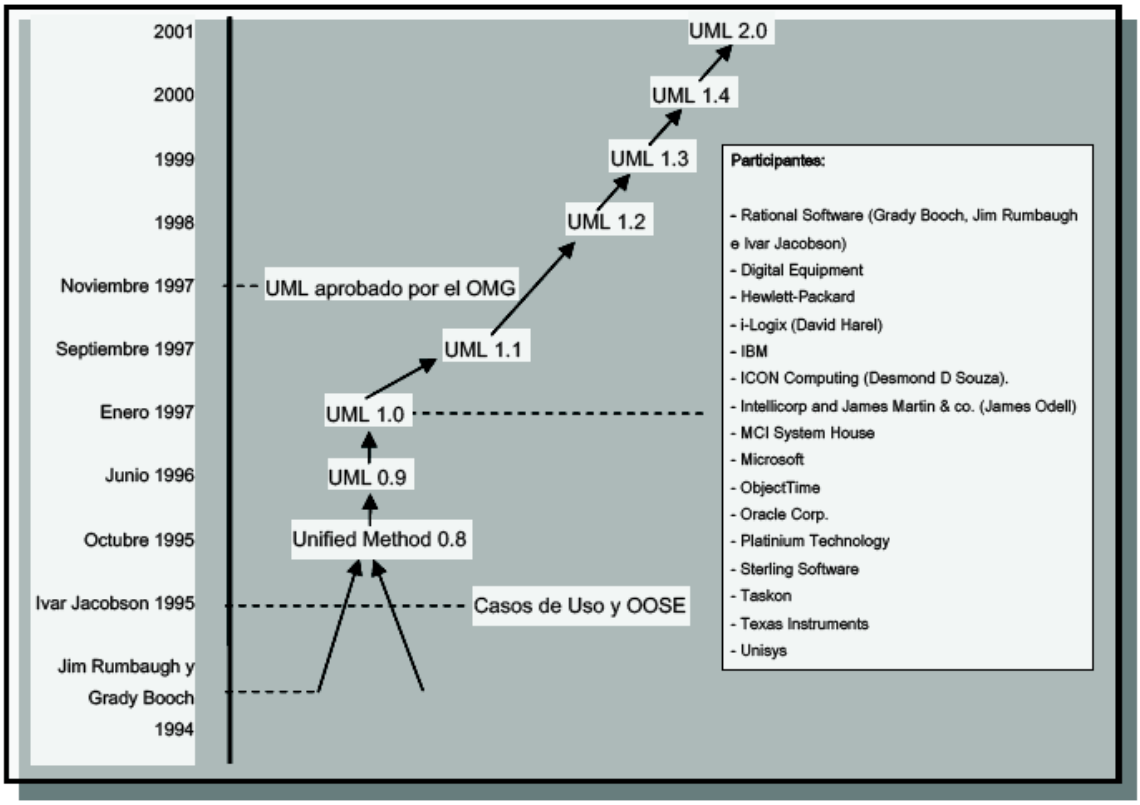


Figura 4.7 Evolución de UML.

Uno de los objetivos principales de la creación de UML era posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado. Para ello era necesario definir una notación y semántica común. En la Figura 4.7 se puede ver cuál ha sido la evolución de UML hasta la creación de UML 2.0, en el que nos basamos para el estudio de esta sección del capítulo. Hay que tener en cuenta que el estándar UML no define un proceso de desarrollo específico, tan solo se trata de una notación de modelado. Para el análisis y diseño en UML se sigue el método propuesto por Craig Larman que se ajusta a un ciclo de vida iterativo e incremental dirigido por casos de uso.

Un modelo representa a un sistema software desde una perspectiva específica. Al igual que la planta y el alzado de una figura en dibujo técnico nos muestran la misma figura vista desde distintos ángulos, cada modelo nos permite fijarnos en un aspecto distinto del sistema.

Los modelos de UML comúnmente utilizados para el diseño de software son los siguientes:

- Diagrama de Casos de Uso.
- Diagrama de Estructura Estática.
- Diagrama de Secuencia.
- Diagrama de Colaboración.
- Diagrama de Estados.
- Diagrama de Clases.

4.4.1 DIAGRAMA DE CASOS DE USO.

Utilizando terminología orientada a objetos, cada caso de uso define una clase o forma particular de usar el sistema mientras que cada ejecución del caso de uso se puede ver como una instancia del caso de uso, o sea, un objeto, con estado y comportamiento. Cada caso de uso constituye un flujo completo de eventos especificando la interacción que toma lugar entre el actor y el sistema. El actor primario es encargado de dar inicio a esta interacción, mientras que los casos de uso son instanciados como respuesta al evento anterior. Una instancia de un actor puede ejecutar varias de estas secuencias, consistiendo de diferentes acciones que a su vez deben llevarse a cabo. La instancia del caso de uso existe mientras el caso de uso se siga ejecutando. La ejecución del caso de uso termina cuando el actor genere un evento que requiera un caso de uso nuevo. Las diferentes instancias de los casos de uso se conocen como *escenarios*. Como varios casos de uso pueden comenzar de una misma forma, no es siempre posible decidir que caso de uso se ha instanciado hasta que éste se haya completado.

La descripción de los casos de uso es mediante diagramas similares a los de transición de estados. Se puede ver a cada caso de uso como representando un estado en el sistema, donde un estímulo enviado entre un actor y el sistema ocasiona una transición entre estados.

En la Figura 4.8 se muestra un ejemplo de casos de uso, donde un programador escribe y depura un programa, mientras que el usuario lo ejecuta.

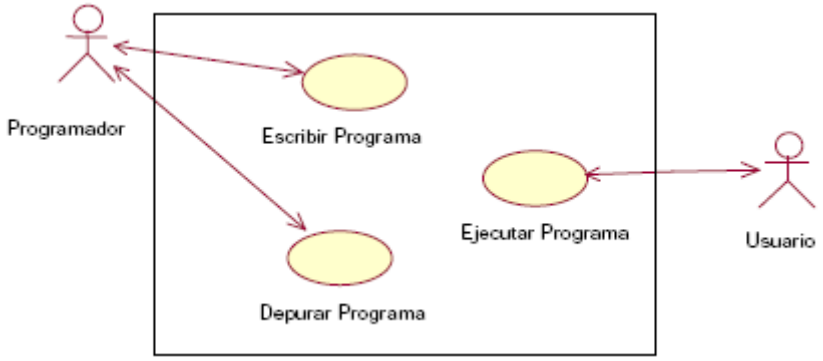


Figura 4.8 Ejemplo de casos de uso mostrando la relación con los actores.

Para identificar los casos de uso, se puede leer la descripción del problema y discutirlo con aquellos que actuarán como actores, haciendo preguntas como:

- ¿Cuales son las tareas principales de cada actor?
- ¿Tendrá el actor que consultar y modificar información del sistema?
- ¿Tendrá el actor que informar al sistema sobre cambios externos?
- ¿Desea el actor ser informado sobre cambios inesperados?

Por ejemplo, en un sistema de conmutación telefónica, un actor podría ser un subscriptor, y un caso de uso típico sería hacer una llamada local. El caso de uso comienza cuando el subscriptor levanta el teléfono. Otro caso de uso es ordenar una llamada de despertar. Ambos casos de uso comienzan cuando el subscriptor levanta el teléfono. Pero cuando el subscriptor levanta el teléfono no sabemos cual caso de uso desea ejecutar. Por lo tanto, los casos de uso pueden comenzar de forma similar y no podemos saber cual se está instanciando hasta que se termine. En otras palabras, el actor no requiere que un caso de uso se ejecute, sólo que inicie una secuencia de eventos que finalmente resulte en la terminación de algún caso de uso.

Aunque la idea del modelo de casos de uso es mantener lo más sencillo posible la complejidad en los diagramas, existen ciertas extensiones al concepto básico de caso de uso que permiten subdividirlos de manera limitada. Para ello se permite la asignación de secciones del flujo completo de un caso de uso en casos de uso separados. Las razones para esto son aprovechar distintas variantes en los casos de uso que se repiten en la lógica del sistema de manera análoga al concepto de herencia. Lamentablemente, no es siempre obvio la funcionalidad que debe ser puesta en casos de uso separados, y qué situación es sólo una pequeña variante de un mismo caso de uso que no amerita tal división. La complejidad de los casos de uso es un factor importante para tal decisión. Existen dos enfoques distintos para expresar variantes:

1. Si las diferencias entre los casos de uso son pequeñas, estos se pueden describir como variantes de un mismo caso de uso, y se pueden definir subflujos separados dentro de un mismo caso de uso, dando lugar al concepto de flujos *básicos* y subflujos *alternos*. Por ejemplo, en el caso de uso registrar un usuario, crear por primera vez el registro y actualizarlo se pueden considerar como dos secuencias de eventos lógicamente similares por lo cual se tratan como subflujos alternos. En general, primero se describe el *flujo básico*, el cual es el flujo más importante dentro del caso de uso. Este flujo corresponde a la secuencia de eventos más común o típica de casos de uso. Posteriormente se describen las variantes o *flujos alternos* que incluyen flujos para el manejo de variantes en la lógica. Normalmente un caso de uso tiene sólo un curso básico, pero varios cursos alternos.
2. Si las diferencias entre los casos de uso son grandes, se deben describir como casos de uso separados. Cuando los casos de uso se dividen en casos de uso separados se utiliza principalmente las relaciones de *extensión*, *inclusión* y generalización como se describe a continuación.

La identificación de casos de uso y de los aspectos anteriores, es a menudo un proceso muy iterativo donde varios intentos se hacen hasta llegar a una solución final estable. Cuando esto ocurre, cada caso de uso debe ser descrito con más detalle. El modelo de casos de uso es la base principal para los demás modelos de diseño.

4.4.1.1 Extensión.

Un concepto importante que se utiliza para estructurar y relacionar casos de uso es la *extensión*. La extensión especifica cómo un caso de uso puede *insertarse* en otro para extender la funcionalidad del anterior. El caso de uso donde se va a insertar la nueva funcionalidad debe ser un flujo completo, por lo cual éste es independiente del caso de uso a ser insertado. De esta manera, el caso de uso inicial no requiere consideraciones adicionales al caso de uso a ser insertado, únicamente especificando su punto de inserción.
Tomando como ejemplo un *Sistema de Reservaciones de Vuelos*, se muestra en la Figura 4.8 la notación para extensión, utilizando la etiqueta “extiende” (“extend”), donde el caso de uso de *Hacer Reservación* es extendido mediante el caso de uso *Pagar Reservación*.

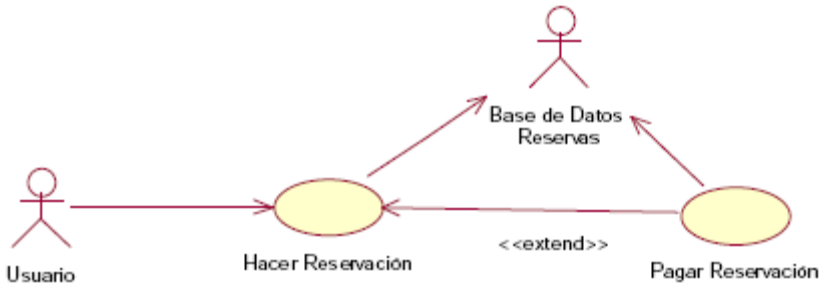


Figura 4.9 Casos de uso Hacer Reservación con extensión de Pagar Reservación.

En general la extensión se utiliza para modelar secuencias de eventos opcionales de casos de uso que al manejarse de manera independiente pueden ser agregados o eliminados del sistema de manera modular. Se puede considerar a la asociación de extensión como una interrupción en el caso de uso original que ocurre donde el nuevo caso de uso se va a insertar. Para cada caso de uso que vaya a insertarse en otro caso de uso, se especifica la posición en el caso de uso original donde el caso de uso de extensión debe insertarse. El caso de uso original se ejecuta de forma normal hasta el punto donde el caso de uso nuevo se inserta. En este punto, se continúa con la ejecución del nuevo curso.

Después que la extensión se ha terminado, el curso original continúa como si nada hubiera ocurrido. Por regla general, se describe primeros los casos de uso básicos totalmente independientes de cualquier extensión de funcionalidad y luego aquellos de extensión.

4.4.1.2 Inclusión.

Una relación adicional entre casos de uso es la *inclusión*. A diferencia de una extensión, la inclusión se define como una sección de un caso de uso que es parte obligatoria del caso de uso básico. El caso de uso donde se va a insertar la funcionalidad depende del caso de uso a ser insertado. Se etiqueta la relación con “incluye” (“include”). Para nuestro ejemplo del *Sistema de Reservaciones de Vuelos*, el caso de uso de *Consultar Información* incluye el caso de uso *Validar Usuario* como se muestra en la Figura 4.10.

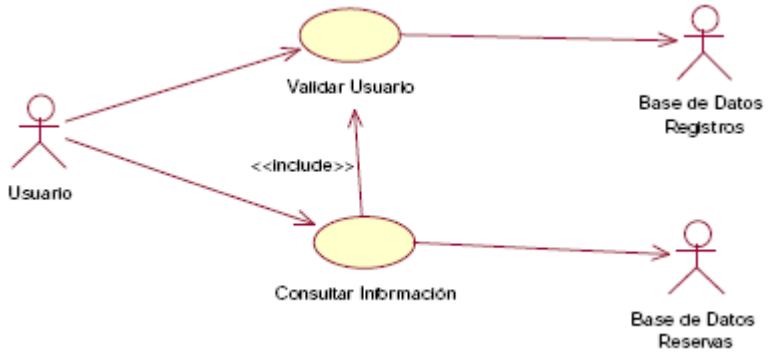


Figura 4.10 Casos de uso Consultar Información con inclusión de Validar Usuario.

4.4.1.3 Generalización.

Una relación adicional entre casos de uso es la generalización la cual apoya la reutilización de los casos de uso.
Mediante la relación de generalización es necesario describir las partes similares una sola vez en lugar de repetirlas para todos los casos de uso con comportamiento común. Se conoce a los casos de uso que se extraen como casos de uso *abstractos***,** ya que no serán instanciados independientemente, sirviendo sólo para describir partes que son comunes a otros casos de uso. Se conoce a los casos de uso que realmente serán instanciados como casos de uso *concretos*. Las descripciones de los casos de uso abstractos se incluyen en las descripciones de los casos de uso concretos. Esto significa que, cuando una instancia de un caso de uso sigue la descripción de un caso de uso concreto, en cierto punto continúa la descripción del caso de uso abstracto en su lugar. Los casos de uso abstractos también pueden ser usados por otros casos de uso abstractos. Es difícil saber cuando ya no tiene sentido seguir extrayendo más casos de uso abstractos. Una técnica para extraer casos de uso abstractos es identificar actores abstractos. Normalmente, no hay razón para crear casos de uso abstractos que se

usan en un solo caso de uso. Por lo general, comportamientos similares entre casos de uso se identifica después que se han descrito los casos de uso. Sin embargo, en algunos casos es posible identificarlos antes. De forma intuitiva, esto es un 'herencia de secuencias' (en lugar de operaciones en el caso normal de herencia).

Para nuestro ejemplo del *Sistema de Reservaciones de Vuelos*, el caso de uso de *Consultar Información* incluye el caso de uso *Validar Usuario* como se muestra en la Figura 4.11.

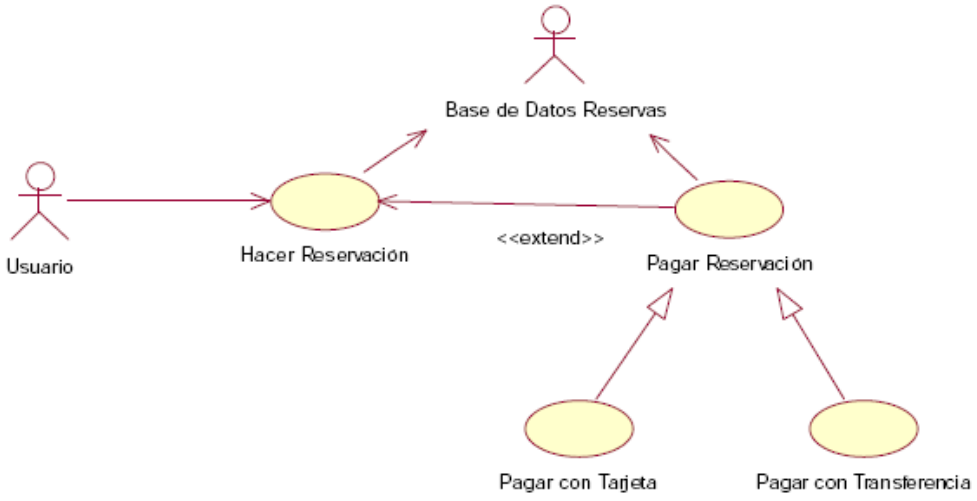


Figura 4.11 Casos de uso Pagar Reservación con generalización de pagos: Pagar con Tarjeta y Pagar con Transferencia.

Las relaciones de *extensión* e *inclusión* entre casos de uso son ambas asociaciones de clases, a diferencia de la *generalización*. En la mayoría de los casos, la selección es bastante obvia, sin embargo el criterio importante es ver qué tanto se acoplan las funcionalidades de los casos de uso. Si el caso de uso a ser extendido es útil por si mismo, la relación debe ser descrita utilizando extensión. Si los casos de uso son fuertemente acoplados, y la inserción debe tomar lugar para obtener un curso completo, la relación debe ser descrita utilizando inclusión. Por otro lado, la generalización es utilizada cuando dos o más casos de uso comparten funcionalidad común la cual es extendida por cada uno al estilo de la generalización entre clases.

También hay una diferencia en cómo se encuentran. Las extensiones se encuentran en un caso de uso existente que el usuario desea ramificar en secuencias adicionales, mientras que las inclusiones se encuentran de la extracción de secuencias comunes entre varios casos de uso diferentes. Las generalizaciones se encuentran al tratar de especializar de diferente manera un mismo caso de uso.
Finalmente, se desean diagramas con baja complejidad que no sean "telarañas" pero que muestren de manera esquemática las posibles secuencias de interacciones entre los actores y el sistema.

4.4.1.4 Documentación.

Parte fundamental del modelo de casos de uso es una descripción textual detallada de cada uno de los actores y casos de uso identificados. Estos documentos son sumamente críticos ya que a partir de ellos se desarrollará el sistema completo. El formato de documentación es el siguiente:

Actor:	Nombre del actor
Casos de Uso:	Nombre de los casos de uso en los cuales participa
Tipo:	*Primario* o *Secundario*
Descripción:	Breve descripción del actor

Las descripciones de los casos de uso representan todas las posibles interacciones de los actores con el sistema para los eventos enviados o recibidos por los actores. En esta etapa no se incluyen eventos internos al propio sistema ya que esto será tratado durante el análisis y únicamente agregaría complejidad innecesaria en esta etapa. El formato de documentación es el siguiente:

Caso de Uso:	Nombre del caso de uso
Actores:	Actores primarios y secundarios que interaccionan con el caso de uso.
Tipo:	Tipo de caso de uso: Primario, Secundario, Opcional. Tipo de flujo: Básico, Inclusión, Extensión, Generalización.
Propósito:	Razón de ser del caso de uso.
Descripción:	Resumen del caso de uso.
Precondiciones:	Condiciones que deben satisfacerse para poder ejecutar el caso de uso.
Flujo Principal:	El flujo de eventos más importante del caso de uso, donde dependiendo de las acciones de los actores se continuará con alguno de los subflujos.
Subflujos:	Los flujos secundarios del caso de uso, numerados como (S-1), (S-2), etc.
Excepciones:	Excepciones que pueden ocurrir durante el caso de uso, numerados como (E-1), (E-2), etc.

Dado que el modelo de casos de uso está motivado y enfocado principalmente hacia los sistemas de información donde los usuarios juegan un papel primordial, es importante ya relacionarse con las interfaces a ser diseñadas en el sistema. Estas interfaces sirven para apoyar de mejor manera la descripción de los casos de uso además de servir de base para prototipos iniciales. Estos prototipos se verán en el siguiente capítulo en la fase de diseño del sistema de control.

Un comentario importante sobre la especificación de estos documentos es que se sigue un proceso iterativo para definir cada uno de ellos, pudiéndose modificar o refinar posteriormente. Obviamente, cuanto más tarde ocurran estos cambios más costosos será implementarlos.

4.4.2 DIAGRAMA DE CLASES.

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contención.

Un diagrama de clases esta compuesto por los siguientes elementos:

- **Clase:** atributos, métodos y visibilidad.
- **Relaciones entre Clases:** Herencia, Composición, Agregación, Asociación y Uso.

4.4.2.1 Clase.

Es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.).

En UML, una clase es representada por un rectángulo que posee tres divisiones:

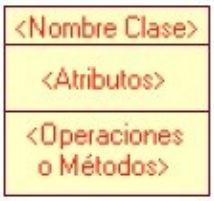


Figura 4.12 Clase.

En donde:

- **División Superior:** Contiene el nombre de la Clase
- **División Intermedia:** Contiene los atributos (o variables de instancia) que caracterizan a la Clase (pueden ser private, protected o public).
- **División Inferior:** Contiene los métodos u operaciones, los cuales son la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: private, protected o public).

El diseño asociado se logra a través de los atributos y métodos.

4.4.2.1.1 Atributos y Métodos.

- **Atributos:**

Los atributos o características de una Clase pueden ser de tres tipos, los que definen el grado de comunicación y visibilidad de ellos con el entorno, estos son:

- **public (+,):** Indica que el atributo será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- **private (-,):** Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos lo pueden accesar).
- **protected (#,):** Indica que el atributo no será accesible desde fuera de la clase, pero sí podrá ser accesado por métodos de la clase además de las subclases que se deriven (ver herencia).

❖ **Métodos:**

Los métodos u operaciones de una clase son la forma en como ésta interactúa con su entorno, éstos pueden tener las características:

- **public (+,):** Indica que el método será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- **private (-,):** Indica que el método sólo será accesible desde dentro de la clase (sólo otros métodos de la clase lo pueden accesar).
- **protected(#,):** Indica que el método no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de métodos de las subclases que se deriven (ver herencia).

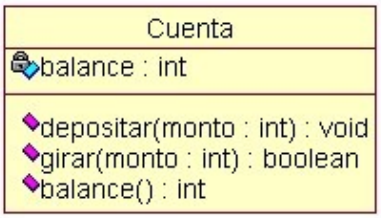


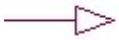
Figura 4.13 Clase con atributos y métodos.

4.4.2.2 Relaciones entre Clases.

Ahora ya definido el concepto de Clase, es necesario explicar como se pueden interrelacionar dos o más clases (cada uno con características y objetivos diferentes).
Antes es necesario explicar el concepto de cardinalidad de relaciones: En UML, la cardinalidad de las relaciones indica el grado y nivel de dependencia, se anotan en cada extremo de la relación y éstas pueden ser:

- uno o muchos: 1..* (1..n)
- 0 o muchos: 0..* (0..n)
- número fijo: m (m denota el número).

a) Herencia (Especialización/Generalización).



Indica que una subclase hereda los métodos y atributos especificados por una Súper Clase, por ende la Subclase además de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la Súper Clase (public y protected), ejemplo:

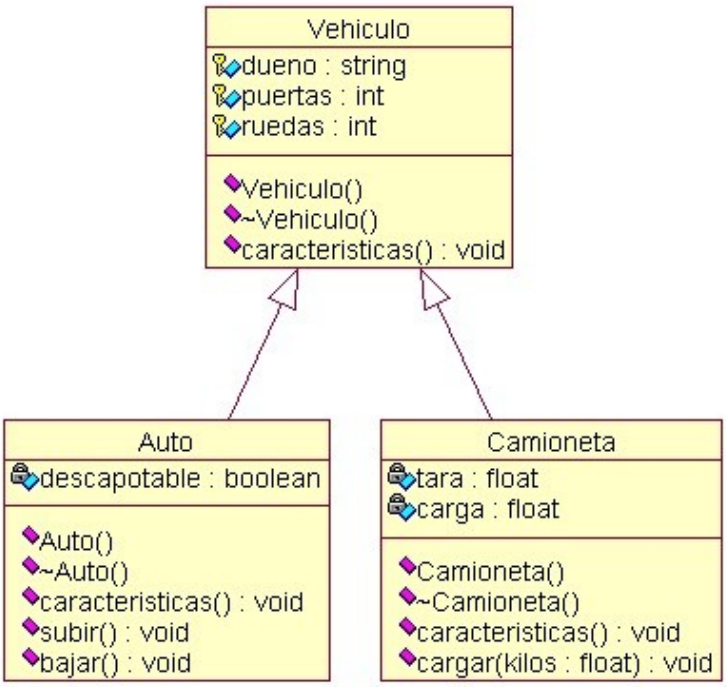


Figura 4.14 Herencia o Generalización.

En la figura se especifica que Auto y Camión heredan de Vehículo, es decir, Auto posee las Características de Vehículo (Precio, VelMax, etc) además posee algo particular que es Descapotable, en cambio Camión también hereda las características de Vehículo (Precio, VelMax, etc) pero posee como particularidad propia Acoplado, Tara y Carga.

Cabe destacar que fuera de este entorno, lo único "visible" es el método Características aplicable a instancias de Vehículo, Auto y Camión, pues tiene definición publica, en cambio atributos como Descapotable no son visibles por ser privados.

b) Agregación.

Para modelar objetos complejos, n bastan los tipos de datos básicos que proveen los lenguajes: enteros, reales y secuencias de caracteres. Cuando se requiere componer objetos que son instancias de clases definidas por el desarrollador de la aplicación, tenemos dos posibilidades:

- **Por Valor:** Es un tipo de relación estática, en donde el tiempo de vida del objeto incluido esta condicionado por el tiempo de vida del que lo incluye. Este tipo de relación es comúnmente llamada Composición (el Objeto base se construye a partir del objeto incluido, es decir, es "parte/todo").

- **Por Referencia:** Es un tipo de relación dinámica, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye. Este tipo de relación es comúnmente llamada Agregación (el objeto base utiliza al incluido para su funcionamiento).

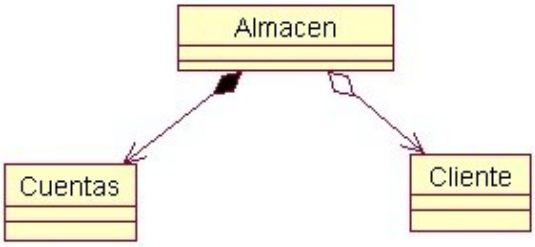


Figura 4.15 Agregación y composición.

La flecha en este tipo de relación indica la navegabilidad del objeto referenciado. Cuando no existe este tipo de particularidad la flecha se elimina.

c) Asociación.

La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre si. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.

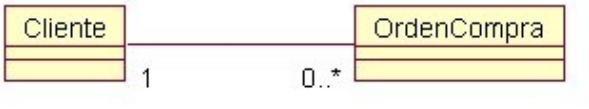


Figura 4.16 Asociación.

d) Dependencia o Instanciación (Uso).

Representa un tipo de relación muy particular, en la que una clase es instanciada (su instanciación es dependiente de otro objeto/clase). Se denota por una flecha punteada.

El uso más particular de este tipo de relación es para denotar la dependencia que tiene una clase de otra, como por ejemplo una aplicación grafica que instancia una ventana (la creación del Objeto Ventana esta condicionado a la instanciación proveniente desde el objeto Aplicación).

Figura 4.17 Dependencia.

Cabe destacar que el objeto creado (en este caso la Ventana gráfica) no se almacena dentro del objeto que lo crea (en este caso la Aplicación).

4.4.2.3 Casos Particulares.

a) Clase Abstracta.

Una clase abstracta se denota con el nombre de la clase y de los métodos con letra "itálica". Esto indica que la clase definida no puede ser instanciada pues posee métodos abstractos (aún no han sido definidos, es decir, sin implementación). La única forma de utilizarla es definiendo subclases, que implementan los métodos abstractos definidos.

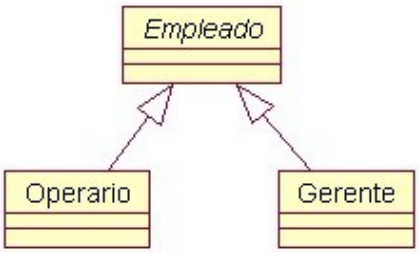


Figura 4.18 Clase abstracta.

b) Clase parametrizada.

Una clase parametrizada se denota con un subcuadro en el extremo superior de la clase, en donde se especifican los parámetros que deben ser pasados a la clase para que esta pueda ser instanciada. El ejemplo más típico es el caso de un Diccionario en donde una llave o palabra tiene asociado un significado, pero en este caso las llaves y elementos pueden ser genéricos. La genericidad puede venir dada de un Template (como en el caso de C++) o bien de alguna estructura predefinida (especialización a través de clases).

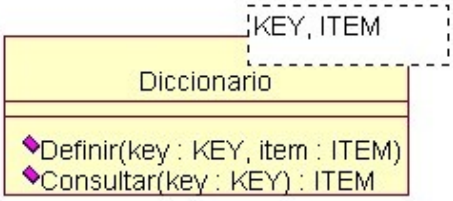


Figura 4.19 Clase parametrizada.

4.5 JAVA, EL LENGUAJE DE PROGRAMACION ELEGIDO.

Java surgió en 1991 cuando un grupo de ingenieros de *Sun Microsystems* trataron de diseñar un nuevo lenguaje de programación destinado a electrodomésticos. La reducida potencia de cálculo y memoria de los electrodomésticos llevó a desarrollar un lenguaje sencillo capaz de generar código de tamaño muy reducido.
Debido a la existencia de distintos tipos de CPUs y a los continuos cambios, era importante conseguir una herramienta independiente del tipo de CPU utilizada. Desarrollaron un código "neutro" que no depende del tipo de electrodoméstico, el cual se ejecuta sobre una "*máquina hipotética o virtual*" denominada ***Java Virtual Machine*** (***JVM***). Es la *JVM* quien interpreta el código neutro convirtiéndolo a código particular de la CPU utilizada. Esto permitía lo que luego se ha convertido en el principal lema del lenguaje: "*Write Once, Run Everywhere*" o "programarlo una vez, correrlo en donde sea".

A pesar de los esfuerzos realizados por sus creadores, ninguna empresa de electrodomésticos se interesó por el nuevo lenguaje.
Java, como lenguaje de programación para computadores, se introdujo a finales de 1995. La clave fue la incorporación de un intérprete *Java* en el programa Netscape Navigator, versión 2.0, produciendo una verdadera revolución en Internet. *Java 1.1* apareció a principios de 1997, mejorando sustancialmente la primera versión del lenguaje.

Al programar en *Java* no se parte de cero. Cualquier aplicación que se desarrolle "cuelga" (o se apoya, según como se quiera ver) en un gran número de clases preexistentes. Algunas de ellas las ha podido hacer el propio usuario, otras pueden ser comerciales, pero siempre hay un número muy importante de clases que forman parte del propio lenguaje (el ***API*** o ***Application Programming Interface*** de ***Java***). *Java* incorpora muchos aspectos que en cualquier otro lenguaje son extensiones propiedad de empresas de software o fabricantes de ordenadores (threads, ejecución remota, componentes, seguridad, acceso a bases de datos, etc.). Por eso es un lenguaje ideal para aprender la informática moderna, porque incorpora todos estos conceptos de un modo estándar, mucho más sencillo y claro que con las citadas extensiones de otros lenguajes. Esto es consecuencia de haber sido diseñado más recientemente y por un único equipo.

El principal objetivo del lenguaje *Java* es llegar a ser el "nexo universal" que conecte a los usuarios con la información, esté ésta situada en el ordenador local, en un servidor de *Web*, en una base de datos o en cualquier otro lugar. *Java* es un lenguaje muy completo (se está convirtiendo en un macro-lenguaje: *Java 1.0* tenía 12 packages; *Java 1.1* tenía 23; *Java 1.2* tenía 59 y así sucesivamente). En cierta forma, al tener Java tantos packages dentro de sí, casi todo depende de casi todo. Por ello, hay que aprenderlo de modo iterativo: primero una visión muy general, que se va refinando en sucesivas iteraciones. Una forma de hacerlo es empezar con un ejemplo completo en el que ya aparecen algunas de las características más importantes.

4.5.1 ELECCIÓN DE JAVA.

La elección del lenguaje de programación se basó en dos criterios:

1. Las características y requerimientos de la aplicación.
2. Las características especiales del lenguaje.

Sabemos que una de las características esenciales que debe cumplir nuestra aplicación es la capacidad de ser adaptable y escalable. Además, la solución propuesta debe estar basada en el análisis y diseño orientado a objetos. Esto nos determina el primer criterio de evaluación: el lenguaje seleccionado debe ser uno que cumpla con el paradigma orientado a objetos y que facilite la adaptabilidad y la escalabilidad.

Aunque bien es cierto que parte de la adaptabilidad y la escalabilidad requeridas serán dadas de manera implícita por el simple hecho de utilizar el paradigma orientado a objetos, en algún momento será necesario que el encargado del mantenimiento del sistema necesite agregar nuevos componentes o modificar alguno de ellos.
En este caso debemos considerar la facilidad que el lenguaje presenta para realizar estas tareas.

Por tanto, el lenguaje elegido para realizar el desarrollo del sistema es por supuesto el lenguaje java. La compañía ***Sun*** describe el lenguaje ***Java*** como "*simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico*". Además de una serie de halagos por parte de *Sun* hacia su propia criatura, el hecho es que todo ello describe bastante bien el lenguaje ***Java***.

La tecnología Java proporciona un entorno de desarrollo multiplataforma y admite varios dispositivos, desde servidores hasta teléfonos celulares y tarjetas inteligentes. La tecnología de Java unifica la infraestructura de negocio creando una plataforma ininterrumpida, segura y conectada en red para los usuarios.

Entre las características y ventajas de Java se encuentran las siguientes:

- **Movilidad Y Seguridad:** La plataforma Java proporciona las bases para una verdadera movilidad. Java ofrece el vehículo ideal para el desarrollo y despliegue de soluciones móviles e inalámbricas.
- **Entorno De Desarrollo:** Java permite el desarrollo sencillo y rápido, a la vez que reduce el número de errores que desperdician tiempo y recursos.
- **Servicios Web:** Java y XML son los lenguajes de computación más extendidos y aceptados mundialmente.
- **Compatibilidad Con Plataformas:** Java proporciona una plataforma de desarrollo segura, abierta, robusta, viable y flexible que permite:

  - Reducir costos.
  - Poner nuevas versiones del sistema software más rápidamente.
  - Obtener todas las ventajas de una flexibilidad máxima.
  - Desarrollar aplicaciones robustas para dispositivos de usuario.
  - No depende de ningún sistema operativo para trabajar.

4.6 HERRAMIENTA DE DESARROLLO: NETBEANS.

El Entorno de Desarrollo Integrado (IDE) **NetBeans** es un entorno de programación para varios lenguajes, incluyendo a Java y C++. Este desarrollo es de fuente abierta (open source), es decir, se proporciona el código fuente del entorno para que se pueda modificar de acuerdo a ciertos parámetros de licencia.

Figura 4.20 Logotipo oficial del proyecto NetBeans.

NetBeans es también una **plataforma de ejecución** de aplicaciones, es decir, facilita la escritura de aplicaciones Java, proporcionando una serie de *servicios* comunes, que a su vez están disponibles a través del IDE. En esta sección del capítulo, nos centramos en el IDE como aplicación, sin entrar en detalles de esa plataforma de ejecución.

Desde la página de NetBeans se puede descargar el entorno, y acceder a su documentación, foros y otros recursos.
El entorno requiere una instalación separada del JDK de Java (la intenta detectar automáticamente al instalarse).

4.6.1 NETBEANS IDE.

El **IDE NetBeans** es un IDE escrito completamente en Java usando la plataforma NetBeans.

La versión actual y estable es NetBeans IDE 6.1, la cual fue lanzada en Julio de 2008.

Todas las funciones del IDE son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. NetBeans contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente.

Sun Studio, Sun Java Studio Enterprise, y Sun Java Studio Creator de Sun Microsystems han sido todos basados en el IDE NetBeans.
El IDE NetBeans está licenciado bajo la SPL (Sun Public Licence).

La versión de NetBeans IDE utilizada para el desarrollo de esta tesis es la 5.5.1 por considerarla bastante estable y con las funciones necesarias para lograr nuestro diseño.

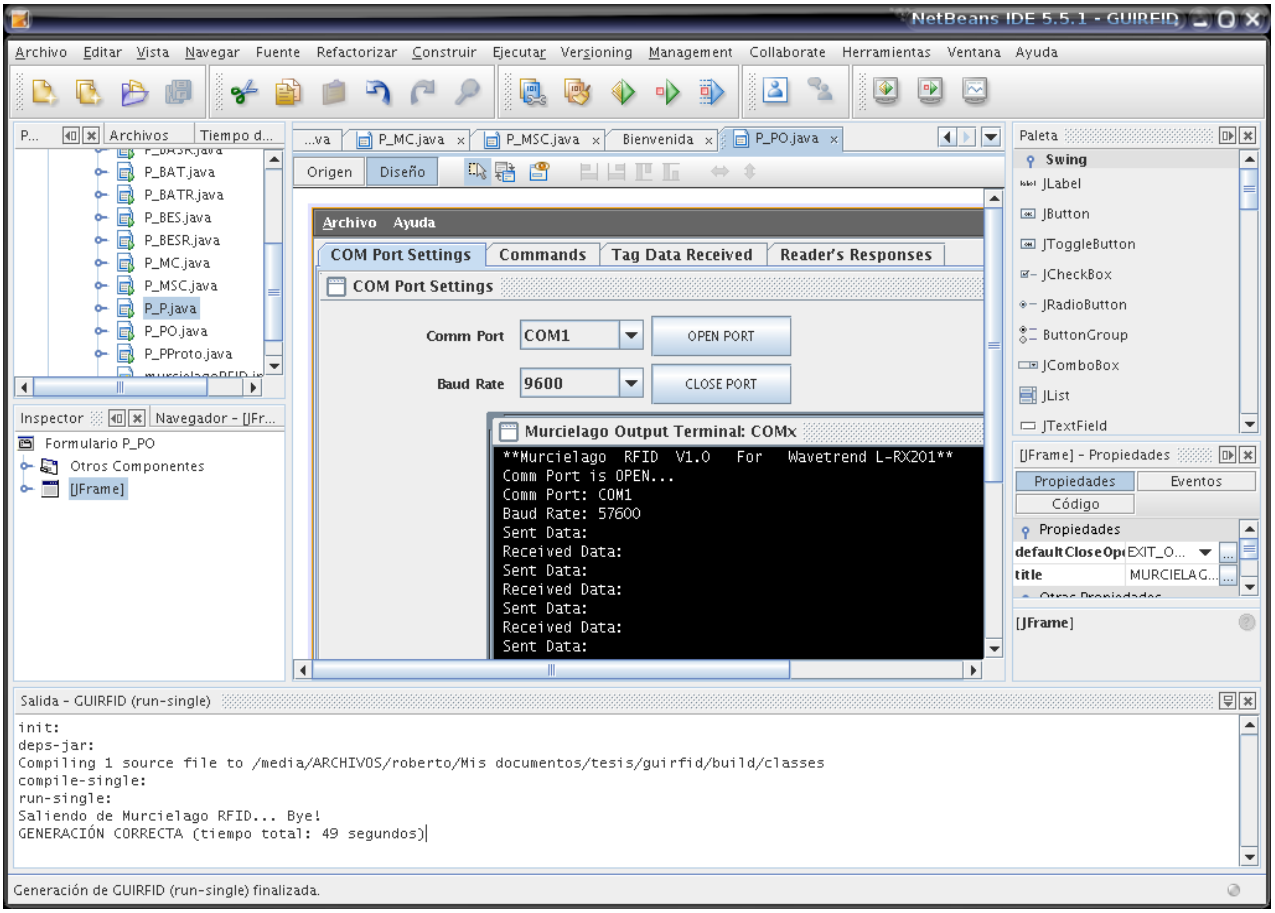


Figura 4.21 NetBeans IDE 5.5.1

4.8 SISTEMAS DE GESTIÓN DE BASE DE DATOS.

Los Sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y *DataBase Management System*, su expresión inglesa.

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información.

4.8.1 SQL.

SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos informática. El nombre de SQL es una abreviatura de Structured Query Language (Lenguaje de estructuras de Consultas).

Es un lenguaje informático que se puede utilizar para interaccionar con una o más bases de datos. En efecto, SQL trabaja con un tipo específico de base de datos, llamada base de datos relacional. El programa informático que controla la base de datos se denomina sistema de gestión de base de datos (database management system) o DBMS.

El nombre Structured Query Language es realmente y en cierta medida inapropiado. En primer lugar, SQL es mucho más que una herramienta de consulta, aunque ese fue su propósito original y recuperar datos sigue siendo una de sus funciones más importantes. SQL se utiliza para controlar todas las funciones que un DBMS proporciona a sus usuarios, incluyendo:

- Definición de Datos. SQL permite a un usuario definir la estructura y organización de los datos almacenados y de las relaciones entre ellos.
- Recuperación de Datos. Permite a un usuario o a un programa de aplicación recuperar los datos almacenados de la base de datos y utilizarlos.
- Manipulación de Datos. Permite a un usuario o a un programa de aplicación actualizar la base de datos añadiendo nuevos datos, suprimiendo datos antiguos y modificando datos previamente almacenados.
- Control de acceso. Puede ser utilizado para restringir la capacidad de un usuario para recuperar, añadir y modificar datos, protegiendo así los datos almacenados frente a accesos no autorizados.
- Compartición de Datos. Se utiliza para coordinar la compartición de datos por parte de usuarios concurrentes, asegurando que no interfieren unos con otros.
- Integridad de Datos. Define restricciones de integridad en la base de datos, protegiéndola contra corrupciones debidas a actualizaciones inconsistentes o a fallos del sistema.

Por tanto SQL es un lenguaje completo de control e ínteractuación con un sistema de gestión de base de datos. SQL es parte integral de un sistema de gestión de base de datos, un lenguaje y una herramienta para comunicarse con el DBMS.

- SQL es un lenguaje de consultas interactivas. Los usuarios escriben ordenes para recuperar datos y mostrarlos en la pantalla, proporcionando una herramienta conveniente y fácil de utilizar para consultas ad hoc de la base de datos.
- SQL es un lenguaje de programación de base de datos. Los programadores insertan órdenes en sus programas de aplicación para acceder a los datos de la base. Tanto los programas escritos por el usuario como los programas de utilidad de la base de datos(

tales como los escritores de informe y las herramientas de entrada de datos) utilizan esta técnica para acceso a la base de datos.

- SQL es un lenguaje de administración de base de datos. El administrador de la BD es responsable de gestionar una base de datos en un minicomputador o en un maxicomputador utiliza SQL para definir la estructura de la base de datos y para controlar el acceso a los datos almacenados.
- SQL es un lenguaje cliente / servidor. Los programas de computador personal utilizan SQL para comunicarse sobre una red de área local con servidores de base de datos que almacenan los datos compartidos. Minimizando el trafico por la red y permite que tanto los PCs como los Servidores efectúen mejor su trabajo.
- SQL es un lenguaje de base de datos distribuidos. Los sistemas de gestión de base de datos distribuidos utilizan SQL para ayudar a distribuir datos a través de muchos sistemas informáticos.
- SQL es un lenguaje de pasarela de base de datos. En una red informática con una mezcla de diferentes productos DBMS, SQL se utiliza a menudo en una pasarela(gateway) que permite que nuestro producto de DBMS se comunique con otro producto.

SQL ha emergido por tanto como una herramienta potente y útil para enlazar personas y sistemas informáticos a los datos almacenados en una base de datos relacional.

4.8.2 MYSQL, EL MANEJADOR DE BASE DE DATOS ELEGIDO.

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. Actualmente, y desde abril de 2008, Sun Microsystems desarrolla MySQL como software libre en un esquema de licenciamiento dual: por un lado lo ofrece bajo la GNU/GPL, pero, empresas que quieran incorporarlo en productos privativos pueden comprar a la empresa una licencia que les permita ese uso. Está desarrollado en su mayor parte en ANSI C.

Existen varias APIs que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySQL, incluyendo C, C++, C#, Pascal, Delphi (via dbExpress), Eiffel, Smalltalk, Java (con una implementación nativa del driver de Java), Lisp, Perl, PHP, Python, Ruby, Gambas, FreeBASIC, y Tcl; cada uno de estos utiliza una API específica. También existe un interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL.

MySQL funciona sobre múltiples plataformas, incluyendo AIX, BSD, FreeBSD, HP-UX, GNU/Linux, Mac OS X, NetBSD, Novell Netware, OpenBSD, OS/2 Warp, QNX, SGI IRIX, Solaris, SunOS, SCO OpenServer, SCO UnixWare, Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Vista y otras versiones de Windows.

Entre las características principales de MySQL destacan:

- Completo soporte para operadores y funciones en cláusulas select y where.
- Completo soporte para cláusulas group by y order by, soporte de funciones de agrupación
- Seguridad: ofrece un sistema de contraseñas y privilegios seguro mediante verificación basada en el host y el tráfico de contraseñas está cifrado al conectarse a un servidor.
- Soporta gran cantidad de datos. MySQL Server tiene bases de datos de hasta 50 millones de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).

- Los clientes se conectan al servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows se pueden conectar usando named pipes y en sistemas Unix usando ficheros socket Unix.

Para nuestra tesis, se utiliza MySQL 5.0.32 para Linux, distribución comunitaria. Esta versión se encuentra actualizada hasta agosto de 2008.

4.8.3 MODELO ENTIDAD-RELACIÓN.

El Modelo Entidad-Relación es un concepto de modelado para bases de datos, propuesto por Peter Chen en 1976, mediante el cual se pretende 'visualizar' los objetos que pertenecen a la Base de Datos como entidades (esto es similar al modelo de Programación Orientada a Objetos) las cuales tienen unos atributos y se vinculan mediante relaciones.

Es una representación conceptual de la información. Mediante una serie de procedimientos se puede pasar del modelo E-R a otros, como por ejemplo el modelo relacional.

El modelado entidad-relación es una técnica para el modelado de datos utilizando diagramas entidad relación. No es la única técnica pero sí la más utilizada. Brevemente consiste en los siguientes pasos:

1. Se parte de una descripción textual del problema o sistema de información a automatizar (los requisitos).
2. Se hace una lista de los sustantivos y verbos que aparecen.
3. Los sustantivos son posibles entidades o atributos.
4. Los verbos son posibles relaciones.
5. Analizando las frases se determina la cardinalidad de las relaciones y otros detalles.
6. Se elabora el diagrama (o diagramas) entidad-relación.
7. Se completa el modelo con listas de atributos y una descripción de otras restricciones que no se pueden reflejar en el diagrama.

Dado lo rudimentario de esta técnica se necesita cierto entrenamiento y experiencia para lograr buenos modelos de datos.

El modelado de datos no acaba con el uso de esta técnica. Son necesarias otras técnicas para lograr un modelo directamente implementable en una base de datos. Brevemente:

- Transformación de relaciones múltiples en binarias.
- Normalización de una base de datos de relaciones (algunas relaciones pueden transformarse en atributos y viceversa).
- Conversión en tablas (en caso de utilizar una base de datos relacional).

Formalmente, los diagramas E-R son un lenguaje gráfico para describir conceptos. Informalmente, son simples dibujos o gráficos que describen la información que trata un sistema de información y el software que lo automatiza.

4.9 SISTEMAS OPERATIVOS.

Desde su creación, las computadoras han utilizado el sistema de codificación de instrucciones basados en el sistema de numeración binaria. En el origen de las computadoras (hace más de cuarenta años), los sistemas operativos no existían y la introducción de un programa para ser ejecutado se convertía en una tarea complicada que solo podía ser realizado por expertos. Esto hacia que las computadoras fueran muy complejas y de difícil uso, exigiendo contar con amplios conocimientos técnicos para operarlas.

En consecuencia, se buscaron los medios más apropiados para que el usuario pueda operar la computadora con un entorno, lenguaje y operación bien definido para hacer un verdadero uso y explotación de esta. Surgen entonces los sistemas operativos.

Un sistema operativo es el encargado de brindar al usuario de manera clara y sencilla de operar, interpretar, codificar y emitir las órdenes al procesador central para que este realice las tareas necesarias y especificas para completar una orden; es el instrumento indispensable para hacer de la computadora un objeto útil y eficiente.

Es entonces que un sistema operativo se define como el conjunto de procedimientos manuales y automáticos, que permiten a un grupo de usuarios compartir una instalación de computadora eficazmente.

4.9.1 FUNCIONES DE LOS SISTEMAS OPERATIVOS.

Las principales funciones de los sistemas operativos son las siguientes:

- Interpretar los comandos que permiten al usuario comunicarse con el procesador central.
- Coordinar y manipular el hardware de la computadora, como la memoria, las impresoras, las unidades de disco, unidades de entrada / salida, el teclado o el Mouse.
- Organizar los archivos en diversos dispositivos de almacenamiento, como discos flexibles, discos duros, discos compactos o cintas magnéticas.
- Gestionar los errores de hardware y la pérdida de datos.
- Servir de base para la creación de software.

4.9.2 CATEGORÍA DE LOS SISTEMAS OPERATIVOS.

Existen diferentes formas de clasificar a los sistemas operativos, entre ellas se pueden mencionar las siguientes:

- Multitareas.
- Monotarea.
- Monousuario.
- Multiusuario.
- Secuencia por lotes.
- Tiempo Real (Runtime).
- Tiempo compartido:

4.9.3 DEBIAN LINUX, EL SISTEMA OPERATIVO ELEGIDO.

Linux es la denominación de un sistema operativo y el nombre de un núcleo. Es uno de los paradigmas del desarrollo de software libre (y de código abierto), donde el código fuente está disponible públicamente y cualquier persona, con los conocimientos informáticos adecuados, puede libremente estudiarlo, usarlo, modificarlo y redistribuirlo.

El término *Linux* estrictamente se refiere al núcleo Linux, el cual tiene sus bases más importantes heredadas de su padre, el sistema operativo Unix. Linux utiliza primordialmente filosofía y metodologías libres (también conocido como GNU/Linux) y que está formado mediante la combinación del núcleo Linux con las bibliotecas y herramientas del proyecto GNU y de muchos otros proyectos/grupos de software (libre o no libre). El núcleo no es parte oficial

del proyecto GNU (el cual posee su propio núcleo en desarrollo, llamado Hurd), pero es distribuido bajo los términos de la licencia GNU GPL.

Debian GNU/Linux es la principal distribución Linux del proyecto Debian, que basa su principio y fin en el software libre.

Creada por el proyecto Debian en el año 1993, la organización responsable de la creación y mantenimiento de la misma distribución, centrado en el núcleo de Linux y utilidades GNU. Éste también mantiene y desarrolla sistemas GNU basados en otros núcleos (Debian GNU/Hurd, Debian GNU/NetBSD y Debian GNU/kFreeBSD).

Nace como una apuesta por separar en sus versiones el software libre del software no libre. El modelo de desarrollo es independiente a empresas, creado por los propios usuarios, sin depender de ninguna manera de necesidades comerciales. Debian no vende directamente su software, lo pone a disposición de cualquiera en Internet, aunque sí permite a personas o empresas distribuir comercialmente este software mientras se respete su licencia.

Figura 4.22 Logo Debian Linux.

La decisión de elegir a Debian como nuestro sistema operativo sobre otras distribuciones Linux (además del simple hecho de que la mayoría de los sistemas operativos basados en Linux son de libre distribución) se tomó gracias a las siguientes características:

- La disponibilidad en varias arquitecturas. La versión estable incluye soporte para 11 plataformas:

  - i386 – x86-32 y 64 bits
  - amd64 – x86-64
  - alpha – DEC Alpha
  - sparc – Sun SPARC
  - arm – ARM architecture
  - powerpc – Arquitectura PowerPC
  - hppa – Arquitectura HP PA-RISC
  - ia64 – Arquitectura Intel Itanium (IA-64)

- mips, mipsel – Arquitectura MIPS (big-endian y little-endian)
- s390 – Arquitectura IBM ESA/390 y z/Architecture
- m68k – Arquitectura Motorola 68k en Amiga, Atari, Macintosh, y varios sistemas embebidos VME

- Una amplia colección de software disponible. La versión 4.0 viene con 18733 paquetes.
- Un grupo de herramientas para facilitar el proceso de instalación y actualización del software (APT, Aptitude, dpkg, Synaptic, etc).
- Su compromiso con los principios y valores involucrados en el movimiento del Software Libre.
- No tiene marcado ningún entorno gráfico en especial, pudiéndose instalar, ya sean: GNOME, KDE, Xfce, Fluxbox, Enlightenment o algún otro.
- Es la distribución más robusta y estable, con lo que se consigue continuidad y seguridad en nuestro sistema.

La última versión estable de Debian es la 4.0 (Etch). La última actualización de esta versión se publicó el 26 de julio de 2008, la cual se utiliza para el completo desarrollo de la presente tesis.

Teniendo definidas las herramientas tecnológicas de diseño, estamos listos para comenzar con el análisis y diseño de nuestro sistema de control en el siguiente capítulo V.

CAPÍTULO V: ANÁLISIS Y DISEÑO DEL SISTEMA.

5.1 MODELO DE REQUISITOS.

El modelo de requisitos tiene como objetivo delimitar el sistema y capturar la funcionalidad que debe ofrecer desde la perspectiva del usuario. Este modelo puede funcionar como un contrato entre el desarrollador y el cliente o usuario del sistema, y por consiguiente proyectando lo que el cliente desea según la percepción del desarrollador. Por lo tanto, es esencial que los clientes puedan comprender este modelo, el cual pretende cumplir con el objetivo descrito en esta tesis para lograr el análisis y diseño del sistema de control para los Lectores RFID, que en este caso el cliente será referido al Centro de Diseño y Manufactura y a la Torre de Ingeniería por ser parte de este proyecto.

El modelo de requisitos es el primer modelo a desarrollarse, sirviendo de base para la formación de todos los demás modelos en el desarrollo de software. En general, cualquier cambio en la funcionalidad del sistema es más fácil de hacer y con menores consecuencias a este nivel, que posteriormente. El modelo de requisitos que desarrollaremos se basa en la metodología *Objectory* (Ivar Jacobson, 1992), basada principalmente en el modelo de *casos de uso*. Actualmente esta metodología es parte del *Proceso Unificado de Rational* (RUP) de IBM. Recordemos que, tanto la metodología Objectory como la OMT (Object Modeling Technique, James Rumbaugh) además de la OOD (Object Oriented Design, Grady Booch), forman lo que se conoce como notación UML (Unified Modeling Languaje).

Cabe señalar que UML no es un método de desarrollo, simplemente son herramientas que nos servirán para modelar nuestro sistema de control.

El modelo de casos de uso es la base para los demás modelos, como se describió anteriormente en el Capítulo IV.

Así, las fases para realizar el análisis y diseño del sistema de control son las siguientes:

- **Requisitos**: El modelo de requisitos es expresado a través del modelo de casos de uso y la definición de los actores en el sistema así como las funciones del mismo, el cual se desarrollará en cooperación con el modelo de interfaces, como se verá más adelante.
- **Construcción**: La fase de construcción del sistema comprende las siguientes etapas:
    - **Análisis**: La funcionalidad especificada por el modelo de casos de uso se estructura en el modelo de análisis, que es estable con respecto a cambios, siendo un modelo lógico independiente del ambiente de implementación. Es en esta parte donde se realiza toda la documentación de los distintos casos de uso, flujos, subflujos y excepciones (si existen) con un formato específico, el cual veremos más adelante en este capítulo.
    - **Diseño**: La funcionalidad de los casos de uso ya estructurada por el análisis es realizada por el modelo de diseño, adaptándose al ambiente de implementación real y refinándose aún más. Para iniciar la etapa de diseño, se realizará la identificación y el diseño de los diagramas de clases, a partir de un modelo conceptual, así como los diagramas de entidad-relación para nuestra base de datos. Se dará paso entonces al diseño de interfaces y el prototipo final del sistema.
    - **Implementación**: Las estructuras de Análisis y Diseño son traducidos mediante el código fuente de algún lenguaje de programación.
    - **Pruebas**: Los casos de uso son ejecutados a través de las pruebas de componentes y pruebas de integración.

- **Documentación**: El modelo de casos de uso debe ser documentado a lo largo de las diversas actividades, dando lugar a distintos documentos como son el manual de usuario, el manual de administración, etc.

El diagrama de la Figura 5.1 ilustra los distintos modelos. Describiremos los detalles y la notación más adelante.

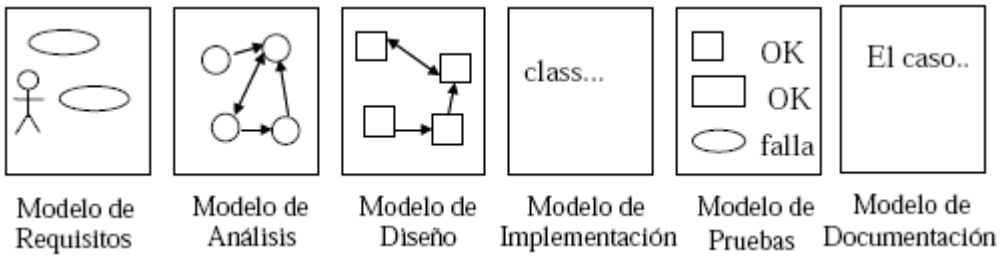


Figura 5.1 Dependencia de los distintos modelos del proceso de software del modelo de casos de uso.

De ellas, la fase de Construcción es la que va a consumir la mayor parte del esfuerzo y del tiempo en un proyecto de desarrollo. Para llevarla a cabo se va a adoptar un enfoque iterativo, tomando en cada iteración un subconjunto de los requisitos (agrupados según casos de uso) y llevándolo a través del análisis y diseño hasta la implementación y pruebas, tal y como se muestra en la Figura 5.2. El sistema se va perfeccionando en cada repetición. Con esta aproximación se consigue disminuir el grado de complejidad que se trata en cada ciclo, y se tiene pronto en el proceso una parte del sistema funcionando. Para esta tesis sólo me enfocaré en las fases de Análisis y Diseño dentro de la fase de Construcción, por ser los objetivos principales.

El propósito del modelo de requisitos es comprender completamente el problema y sus implicaciones. Todos los modelos no solamente se verifican contra el modelo de requisitos, sino que también se desarrollan directamente de él. El modelo de requisitos sirve también como base para el desarrollo de las instrucciones operacionales y los manuales ya que todo lo que el sistema deba hacer se describe aquí desde la perspectiva del usuario. El modelo de requisitos no es un proceso mecánico, el analista debe interactuar constantemente con el cliente para completar la información faltante, y así clarificar ambigüedades e inconsistencias. El analista debe separar entre los requisitos verdaderos y las decisiones relacionadas con el diseño e implementación. Se debe indicar cuales aspectos son obligatorios y cuales son opcionales para evitar restringir la flexibilidad de la implementación. Durante el diseño se debe extender el modelo de requisitos con especificaciones de rendimiento y protocolos de interacción con sistemas externos, al igual que provisiones sobre modularidad y futuras extensiones. En ciertas ocasiones ya se puede incluir aspectos de desarrollo, como el uso de lenguajes de programación particulares.

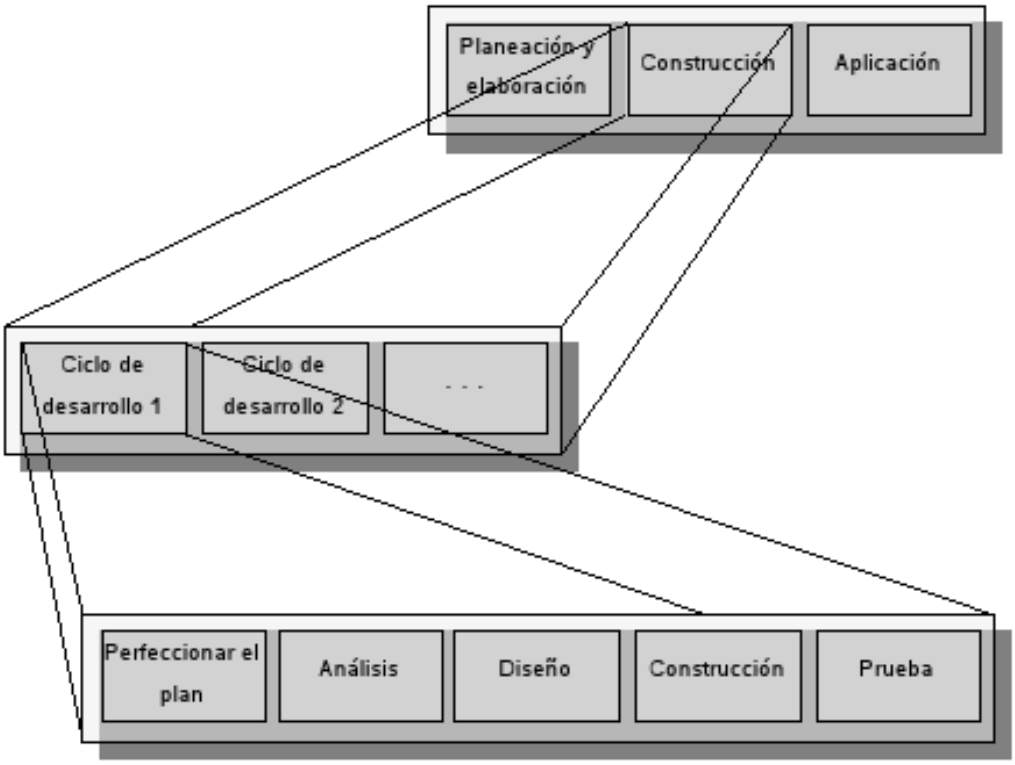


Figura 5.2 Desarrollo iterativo en la fase de Construcción.

Se iniciará entonces con el modelo de requisitos y el desarrollo de los modelos posteriores para el *Análisis y Diseño de un Sistema de Control para los Lectores RFID Wavetrend L-RX201* el cual es el tema de la presente tesis. Para tal meta se mostrará inicialmente una *descripción del problema*.

5.1.1 DESCRIPCIÓN DEL PROBLEMA.

La descripción del problema es una descripción muy preliminar de necesidades que sirve únicamente como punto de inicio para comprender los requisitos del sistema. Se trata aquí de simular una descripción preparada por un cliente la cual debe evolucionar por medio del modelo de requisitos para lograr la especificación final del sistema a desarrollarse. La descripción del problema debe ser una descripción de necesidades y no una propuesta para una solución. La descripción inicial puede ser incompleta e informal. No hay razón para esperar que la descripción inicial del problema, preparada sin un análisis completo, sea correcta.

La descripción del problema para nuestro sistema de control de los Lectores RFID Wavetrend es la siguiente:

El Sistema de Control para los Lectores RFID Wavetrend L-RX201 es un sistema que permite al operador controlar y configurar los Lectores enviando los comandos correspondientes para cada acción específica que se desee realicen estos dispositivos RFID, además de consultar información referente al sistema y a los usuarios del sistema con su correspondiente registro de etiqueta electrónica o Tag. Se desea tener tanto un ambiente gráfico de control como una terminal para el envío de comandos.

El sistema presenta en su pantalla principal un mensaje de bienvenida y los campos de *login (usuario)* y *password (contraseña)* para poder tener acceso al sistema de control. Estos datos, al introducirse en los campos correspondientes, deben validarse antes de permitir el acceso. Un usuario no registrado no podrá tener acceso al sistema.

Una vez registrado el usuario (a través del administrador del sistema) y después de haberse validado los registros de *usuario* y *contraseña* se tiene acceso a la pantalla inicial de trabajo (Framework) en donde se pueden seleccionar las siguientes actividades:

- Configuración del puerto de control.
- Control y configuración de los Lectores RFID (envío de comandos)
- Consulta de datos registrados sobre usuarios y Tags.
- Manipulación de datos registrados (rol Administrador)
- Consulta de información sobre la red de Lectores RFID

La configuración del puerto de control nos permite seleccionar la velocidad de transmisión a través del puerto serie (COMx), la cual debe ser la misma velocidad configurada en cada uno de los Lectores RFID en la red, según el manual de éstas. También nos permite seleccionar el puerto serie a través del cual se va a controlar la red de Lectores RFID, en caso de tener más de un puerto serie en nuestra PC.
El control y la configuración de los Lectores RFID se realiza a través de la selección (por medio de un Radio Button) del comando que se desea enviar a la red, además de seleccionar (por medio de Spinners) la red y el Lector o Lectores RFID a quienes se les desea enviar el comando.

Cuando ya se tiene lista la configuración del puerto de control y comando deseados, se envían (por medio de un Button) hacia la red de Lectores RFID.

Tanto los comandos como las respuestas de los Lectores RFID a estos comandos se pueden ir visualizando en pantalla en la sección de respuestas de los Lectores (Reader´s Responses) ubicada en la parte inferior derecha de nuestra aplicación gráfica.

La consulta de datos registrados sobre usuarios y Tags se encuentra en la barra de menús en la sección *Herramientas*, submenú *Consultar Información,* el cual despliega las opciones *Consultar usuario* y *Consultar Tag*. Al seleccionar alguna de las dos opciones se abre una subpantalla en la cual se puede realizar la búsqueda por nombre del usuario o por número de Tag. Si el usuario o el Tag no se encuentran registrados, la búsqueda regresará un mensaje de no existencia de registro.

La manipulación de datos registrados sólo puede ser realizada por el administrador del sistema. Esta actividad se encuentra en la barra de menús en la sección *Herramientas,* submenú *Modificar Información*, el cual despliega las opciones *Añadir nuevo usuario*, *Añadir nuevo Tag, Eliminar Usuario, Eliminar Tag, Modificar Información Usuario* y *Modificar Información Tag.* Al seleccionar alguna de las opciones se abre una subpantalla en la cual se puede realizar la inserción, eliminación o modificación de datos según se desee.

La consulta de información sobre la red de Lectores muestra el número de Lectores RFID conectadas en red con el sistema.

Nótese lo informal y limitado de esta descripción, la cual estaremos refinando a lo largo del capítulo.

5.1.2 FUNCIONES DEL SISTEMA.

Las funciones del sistema son lo que éste deberá hacer. A continuación se muestra una síntesis de las funciones del sistema consensuadas entre el cliente o dueño del negocio y el analista. La clasificación de las funciones es arbitraria.
Una tabla donde se sintetizan las funcionalidades requeridas del sistema consta de tres columnas:

La primera, denominada *Referencia*, que hace las veces de un identificador que nos ayuda a realizar el seguimiento de la funcionalidad a través de los casos de uso, además de ser una manera de tipificar funcionalidades.

La segunda columna titulada *Función*, muestra una descripción de la funcionalidad requerida. Debemos de observar que en la tabla se listan las funcionalidades deseadas por los dueños del negocio, en esta fase no se presenta una lista definitiva.

Por último, la tercer columna llamada *Categoría* ayuda a clasificar las funcionalidades a fin de establecer prioridades entre ellas e identificar las que pudieran pasar inadvertidas. Se utilizan tres tipos de categorías:

- **Evidente.** Significa que la funcionalidad debe realizarse y el usuario debería saber que se ha realizado.
- **Oculta.** Debe realizarse, aunque no es visible para los usuarios. Esto aplica a servicios técnicos subyacentes.
- **Opcional.** Su inclusión no repercute significativamente en el costo ni en otras funciones.

REFERENCIA	FUNCION	CATEGORIA
R1.1	El usuario deberá proporcionar un identificador y una contraseña para poder utilizar el sistema, de acuerdo al nivel de usuario.	Evidente
R1.2	El sistema deberá permitir asignar las autorizaciones de uso a los usuarios.	Evidente

Tabla 5.1 Funciones de validación de usuarios (R1).

REFERENCIA	FUNCION	CATEGORIA
R.2.1	El sistema deberá conocer en todo momento la posición de los Lectores RFID Wavetrend en la red.	Oculta
R.2.2	El sistema deberá conocer el número de Lectores Wavetrend en la red y si alguna de ellas ha fallado en conexión.	Evidente
R.2.3	El sistema deberá permitir controlar a los Lectores Wavetrend sólo a personal autorizado.	Evidente

R.2.4	El sistema deberá permitir realizar la configuración de los Lectores sólo a personal autorizado.	Evidente
R.2.5	El sistema será capaz de recibir información desde los Lectores Wavetrend.	Evidente
R.2.6	El sistema será capaz de enviar información hacia la red de Lectores Wavetrend.	Evidente
R.2.7	El sistema será capaz de reconocer los comandos enviados por los Lectores Wavetrend e informar al usuario sobre lo referente a dicho comando.	Evidente
R.2.8	El sistema será capaz de reiniciar la red completa de Lectores Wavetrend.	Evidente

Tabla 5.2 Funciones de Control y Configuración (R2).

REFERENCIA	FUNCION	CATEGORIA
R.3.1	El sistema deberá llevar una bitácora de errores registrados en cualquier momento.	Oculta
R.3.2	El sistema deberá tener un catálogo de los tipos de error conocidos.	Oculta
R.3.3	El sistema deberá llevar una bitácora de los usuarios que se han conectado al mismo, registrando rol y hora de entrada.	Oculta
R.3.4	El sistema deberá llevar el registro de los datos enviados hacia la red de Lectores Wavetrend, así como del actor que lo haya realizado y la fecha del envío.	Oculta
R.3.5	El sistema deberá llevar el registro de los datos recibidos de la red de Lectores Wavetrend, así como la fecha de recepción.	Oculta

Tabla 5.3 Funciones de Monitoreo (R3).

REFERENCIA	FUNCION	CATEGORIA
R.4.1	El sistema deberá permitir el acceso al personal con Tag ID registrado.	Evidente
R.4.2	El sistema deberá llevar una bitácora del personal con un Tag ID que haya accesado a través del sistema de control, registrando TagID, Nombre de la persona, área, puerta de acceso, hora de entrada y hora de salida.	Oculta
R.4.3	El sistema deberá denegar el acceso al personal con un Tag ID no registrado y reportarlo en la bitácora de errores, registrando TagID, tipo de error, puerta de acceso y hora de ocurrencia.	Evidente
R.4.4	El sistema deberá enviar una señal hacia el sistema automático de puertas para activar o desactivar la entrada al personal.	Opcional

Tabla 5.4 Funciones extras para control de acceso. (R4).

5.1.3 ATRIBUTOS DEL SISTEMA

Los atributos del sistema son sus características o dimensiones, de ninguna manera funciones. La tabla 5.5 muestra una síntesis de los atributos del sistema consensuadas entre el cliente o dueño del negocio y el analista.

ATRIBUTO	DETALLES Y RESTRICCIONES DE FRONTERA
Metáfora de interfaz	(detalle) Metáfora orientada a ventanas y cuadros de texto. (detalle) Maximiza la navegación utilizando el puntero. (detalle) Deben agregarse de manera fácil nuevos módulos y nuevos dispositivos.
Plataformas de sistema operativo	(detalle) Linux distribución Debian 4.0 (Etch)

Tabla 5.5 Atributos del sistema.

5.1.4 MODELO DE CASOS DE USO.

Tomando en consideración nuestro *Sistema de Control de los Lectores RFID LRX201*, se pueden identificar de la descripción del problema que se tiene al menos un actor, el *Operador*, encargado de controlar a los Lectores con el sistema. Si se analiza un poco más, se puede identificar que los Lectores RFID juegan un papel muy activo con respecto al sistema en desarrollo. A este actor lo llamaremos *Lector RFID*, el cual mantiene una constante interacción con el sistema debido a que envía y recibe información hacia y del sistema, respectivamente. Más aún, podemos identificar un actor ligado al anterior el cual interactúa con el sistema a través de los Lectores al que llamaremos U*suario Tag RFID,* encargado de enviar información con la tarjeta electrónica o Tag al sistema, previamente analizada por los Lectores Wavetrend y posteriormente por el mismo sistema.
Por último tenemos un actor encargado de administrar la base de datos y el sistema mismo en general. A este actor lo llamaremos el *Administrador*. El diagrama de *Delimitación del Sistema* con los actores correspondientes se muestra en la Figura 5.3.

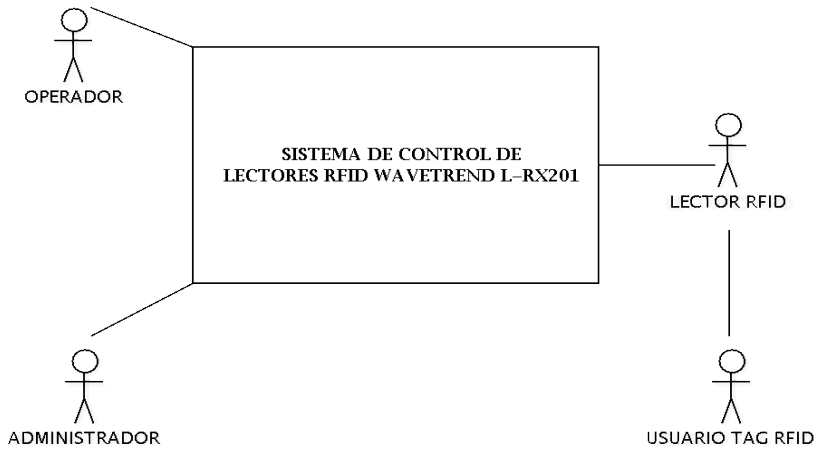


Figura 5.3 Delimitación del sistema de control para los Lectores RFID LRX201.

Volviendo a la distinción entre actor y persona, una misma persona puede jugar el papel del actor *Operador* cuando controla los Lectores RFID y además puede registrar o eliminar información en la base de datos, así como dar mantenimiento a todo el sistema, por ejemplo como *Administrador*, correspondiente a otro actor mostrado en nuestra delimitación.
De nuestro diagrama, el actor *Operador* se considera un actor primario, ya que el sistema se construye pensando en sus actividades, mientras que el actor *Lector RFID y* el *Administrador* son actores secundarios, ya que si no existieran usuarios no habría necesidad de ellos ni del sistema.
El actor *Usuario Tag RFID* debe ser considerado como un actor primario, debido a que si no interactuara con el sistema a través de los Lectores con su Tag (independientemente si se encuentren registrados o no en el sistema) el sistema no tendría sentido ni necesidad, como sucede con el actor *Operador*. Del diagrama anterior se observa que el actor *Lector RFID* funciona sólo como intermediario entre las acciones del actor Usuario Tag RFID y el sistema, por ser un actor pasivo.
En este momento tenemos conocimiento, por lo menos de manera parcial, de qué requerimientos necesita nuestro sistema. Éstos requerimientos aparecen de manera aislada. Después de haber definido los actores de nuestro sistema, se define la funcionalidad propia del mismo por medio de los *casos de uso*.

Lo primero, antes de concentrarnos en lo que serán los casos de uso del sistema, es documentar los actores del mismo. Los actores encontrados y su descripción, se muestran a continuación:

NOMBRE	*Operador*
ROL	Operador del sistema
DESCRIPCION	La persona que controla y configura los Lectores Wavetrend a través del sistema.

Tabla 5.6 Actor Operador.

NOMBRE	*Usuario Tag RFID*
ROL	Usuario de Tag RFID para acceso
DESCRIPCION	La persona que, a través de la interacción entre los Lectores Wavetrend y su respectivo Tag, envía información al sistema.

Tabla 5.7 Actor Usuario RFID.

NOMBRE	*Administrador*
ROL	Administrador del sistema
DESCRIPCION	La persona que configura y da mantenimiento al sistema de control a través del código fuente y da mantenimiento a los catálogos de la base de datos del mismo.

Tabla 5.8 Actor Administrador.

NOMBRE	*Lector RFID*
ROL	Lectores de datos de etiquetas electrónicas (Tags) vía RFID.
DESCRIPCION	Son los encargados de enviar y recibir información hacia y fuera del sistema, respectivamente.

Tabla 5.9 Actor Lector RFID.

Mostraremos a continuación la descripción de los casos de uso.

Como vimos en el capítulo IV, los casos de uso son situaciones de utilización del sistema. Existen técnicas para identificarlos con facilidad. Para el desarrollo de esta tesis se utilizaron algunas preguntas y lluvia de ideas entre el cliente y el analista.
En esta primera fase, no se pretende tener un conocimiento total del sistema (recordemos que estamos siguiendo un proceso iterativo) sino un panorama global. Algunos autores coinciden en que al finalizar la etapa de requisitos debe tenerse al rededor de un 10% ó 20% de avances en los casos de uso.
Los casos de uso pueden escribirse en diferentes formatos y expresarse con diverso grado de detalle y de aceptación de las decisiones concernientes al diseño. La decisión está en manos del analista y diseñador.

Para nuestro tema de tesis se inició con un caso de uso de alto nivel, se pasó por tres versiones del caso de uso (al seguir un proceso iterativo), en cada una de las cuales se agregaba mayor detalle a la descripción. Se muestran entonces los detalles finales de nuestros casos de uso de alto nivel.

Llamamos caso de uso de alto nivel a aquellos que describen el proceso muy brevemente, casi siempre en dos o tres enunciados. Conviene servirse de este tipo de casos de uso durante la fase del modelo de requisitos, a fin de entender rápidamente el grado de complejidad y de funcionalidad del sistema. Estos casos de uso son muy sucintos y vagos en las decisiones de diseño.

Además del grado de detalle, un caso de uso puede clasificarse en primario, secundario u opcional:

- **Casos de uso primarios:** Representan los procesos comunes más importantes.
- **Casos de uso secundarios:** Representan procesos menores o no tan cotidianos.
- **Casos de uso opcionales:** Representan procesos que pueden no abordarse.

A continuación se presentan los casos de uso de alto nivel encontrados. Al tratarse de una descripción sencilla los llamamos, arbitrariamente, casos de uso a nivel 0 (cero). En una siguiente descripción los casos de uso de alto nivel los llamamos arbitrariamente casos de uso a nivel 1. Estos casos de uso a nivel 1 presentan una descripción un poco más detallada para esta fase del modelo de requisitos.

Caso de uso 1: Validar Usuario Sistema de Control
Actores: Administrador, Operador
Tipo: Primario
Descripción nivel 0: El sistema debe verificar la autenticidad del rol del actor antes de poder accesar o realizar alguna actividad en el mismo.
Descripción nivel 1: 1. El actor introduce su login o usuario registrado y su password o clave de acceso en la pantalla principal de acceso al sistema de control. 2. El sistema asigna los permisos necesarios al rol de usuario para actuar dentro del mismo.

Tabla 5.10 Caso de uso Validar Usuario Sistema de Control niveles 0 y 1.

Caso de uso 2: Consultar bitácora de Accesos RFID
Actores: Administrador
Tipo: Primario
Descripción nivel 0: El administrador necesita conocer quien ha accesado a través de los Lectores vía RFID.
Descripción nivel 1: 1. El administrador realiza una petición de consulta a la bitácora de Accesos RFID.

2. El administrador puede indicar, si así lo desea, buscar accesos por fecha, por id de usuario, por id de Lector RFID o por Tag ID.

Tabla 5.11 Caso de uso Consultar bitácora de Accesos RFID niveles 0 y 1.

Caso de uso 3: Consultar Bitácora de Accesos al Sistema de Control
Actores: Administrador
Tipo: Primario
Descripción nivel 0: El administrador necesita conocer los usuarios que han accesado al sistema de control.
Descripción nivel 1: 1. El administrador realiza una petición de consulta a la bitácora de accesos. 2. El administrador puede indicar, si así lo desea, buscar accesos por fecha, por id de usuario o por rol en el sistema.

Tabla 5.12 Caso de uso Consultar bitácora de Accesos al Sistema de Control niveles 0 y 1.

Caso de uso 4: Administración de usuarios
Actores: Administrador
Tipo: Primario
Descripción nivel 0: El administrador necesita llevar un control de los usuarios que utilizan el sistema.
Descripción nivel 1: 1. El administrador da de alta, elimina o modifica un usuario, asignando su contraseña, datos y perfil. 2. El administrador habilita o inhabilita el uso del sistema o el acceso vía RFID a los usuarios registrados y no registrados, según corresponda.

Tabla 5.13 Caso de uso Administración de Usuarios niveles 0 y 1.

Caso de uso 5: Consultar Bitácora de Errores Sistema de Control
Actores: Administrador
Tipo: Primario
Descripción nivel 0: El administrador necesita conocer qué eventos han surgido en el sistema de control.

Descripción nivel 1: 1. El administrador realiza una petición de consulta a la bitácora de errores. 2. El administrador puede indicar, si así lo desea, buscar errores por día, por usuario, por id de Lector RFID, por TagID o por evento.

Tabla 5.14 Caso de uso Consultar Bitácora de Errores Sistema de Control niveles 0 y 1.

Caso de uso 6: Mantenimiento del Sistema de Control
Actores: Administrador
Tipo: Primario
Descripción nivel 0: El administrador necesita dar mantenimiento al sistema de control.
Descripción nivel 1: 1. El administrador revisa los registros (Logs) del sistema o bitácora de eventos para visualizar algún error. 2. En caso de algún error registrado, el administrador revisa la referencia del mismo y analiza el código fuente del sistema de control , registros en la BD o comunicación entre el sistema y la red de Lectores, según corresponda.

Tabla 5.15 Caso de uso Mantenimiento del Sistema de Control niveles 0 y 1.

Caso de uso 7: Mantenimiento Catálogos
Actores: Administrador
Tipo: Primario
Descripción nivel 0: El administrador necesita dar mantenimiento a los catálogos del sistema.
Descripción nivel 1: 1. El administrador revisa el tamaño de las tablas y de uso de espacio en disco referente a los catálogos del sistema 2. El administrador respalda información de las bitácoras y catálogos del sistema y depura lo innecesario.

Tabla 5.16 Caso de uso Mantenimiento Catálogos en el sistema de control niveles 0 y 1.

Caso de uso 8: Configurar Puerto Control
Actores: Operador
Tipo: Primario
Descripción nivel 0: El operador asigna la velocidad de transmisión y el puerto serie a utilizar, con la cual se enviará y se recibirá información hacia y desde la red de Lectores, respectivamente.
Descripción nivel 1: 1. El operador asigna en la PC la velocidad de transmisión especificada en el manual de Lectores Wavetrend, la cual puede variar entre 5 distintos valores, pero que por default está configurada como 57600 baudios. 2. El operador elige el puerto serie de la PC a utilizar para la conexión de la red de Lectores RFID. 3. El operador se encuentra listo para controlar y configurar a los Lectores RFID.

Tabla 5.17 Caso de uso Configurar Puerto Control niveles 0 y 1.

Caso de uso 9: Configurar Lectores
Actores: Operador
Tipo: Primario
Descripción nivel 0: El operador necesita configurar a los Lectores RFID en red, dependiendo de las necesidades requeridas.
Descripción nivel 1: 1. El operador sincroniza la velocidad de transferencia entre el puerto serie de la PC y los Lectores RFID a través del comando de configuración correspondiente. 2. El operador puede entonces configurar a los Lectores RFID según los requerimientos del proyecto. Se puede configurar la auto-recepción de datos de Tags, los identificadores de red y de Lector, el código de sitio, habilitar o deshabilitar la auto-detección en los Lectores, entre otras opciones.

Tabla 5.18 Caso de uso Configurar Lectores niveles 0 y 1.

Caso de uso 10: Controlar Lectores
Actores: Operador
Tipo: Primario
Descripción nivel 0: El operador debe controlar la manera de actuar de los Lectores RFID y el flujo de datos hacia y desde los mismos.
Descripción nivel 1:

1. El operador puede obtener información contenida en los Lectores o en los Tags detectados. 2. El operador puede analizar si cierto Lector RFID está disponible por medio de un barrido de datos o ping. 3. El operador puede reiniciar la red completa de Lectores si así lo requiere.

Tabla 5.19 Caso de uso Controlar Lectores niveles 0 y 1.

Caso de uso 11: Detener auto-detección
Actores: Operador
Tipo: Secundario
Descripción nivel 0: El operador puede detener la auto-detección de Tags en la red de Lectores RFID en caso de ser necesario.
Descripción nivel 1: 1. El operador envía hacia la red de Lectores RFID cierta cantidad de caracteres * para detener la auto-detección de Tags. 2. El operador verifica que se haya detenido la auto-detección de Tags en el log (registro) del sistema correspondiente.

Tabla 5.20 Caso de uso Detener auto-detección niveles 0 y 1.

Caso de uso 12: Consultar Total de Lectores en Red
Actores: Operador
Tipo: Secundario
Descripción nivel 0: El operador puede consultar el total de Lectores RFID en red para llevar un conteo de los mismos o para saber si alguna parte de la red se deshabilitó, según se requiera.
Descripción nivel 1: 1. El operador envía hacia la red de Lectores RFID la petición correspondiente y obtiene el valor total de Lectores en la red. 2. El operador verifica que el total obtenido sea igual al total de Lectores registrados en el sistema, esto para comprobar la integridad de la red.

Tabla 5.21 Caso de uso Consultar Total de Lectores Red en niveles 0 y 1.

Caso de uso 13: Validar Información Recibida
Actores: Usuario Tag RFID
Tipo: Primario
Descripción nivel 0: El usuario con un Tag registrado puede enviar los datos del mismo hacia el sistema de control a través de la red de Lectores RFID para ser validados, debido a una petición de acceso.
Descripción nivel 1: El usuario con un Tag registrado puede hacer una petición de acceso a puertas desde una cierta distancia al Lector RFID, dependiendo de la ganancia configurada en el mismo, la cual puede tomar 2 valores según el manual: alta o baja ganancia (Caso de Uso 9). La información enviada por el Tag se valida dentro del sistema para saber si éste se encuentra registrado y así permitir o denegar la acción.

Tabla 5.22 Caso de uso Validar Información Recibida niveles 0 y 1.

Caso de uso 14: Validar Información Recibida Dispositivo Electrónico
Actores: Usuario Tag RFID
Tipo: Opcional
Descripción nivel 0: El usuario con un Tag registrado puede enviar los datos del mismo hacia el sistema de control a través de la red de Lectores RFID para ser validados, debido a la movilidad de equipo tecnológico.
Descripción nivel 1: El personal de la institución puede intentar extraer algún equipo tecnológico, tal como una laptop, escaner, osciloscopio, etc. del inmueble en cuestión, lo que ocasiona que si el Tag dentro del equipo se aleja del rango de frecuencia (Caso de Uso 9) (especificado en metros y configurable para cada Lector RFID) éste enviará la información a través de la red de Lectores RFID para ser validada. La información enviada por el Tag se valida dentro del sistema para saber si éste se encuentra registrado y pertenece al edificio o es ajeno a la institución.

Tabla 5.23 Caso de uso Validar Información Recibida Dispositivo Electrńico niveles 0 y 1.

Se han descrito los casos de uso y los actores del sistema, a continuación, en la Figura 5.4 se muestra su representación gráfica en UML, con base en el respectivo tema para UML de nuestro Capítulo IV.

Una vez que se tiene el modelo de casos de uso que representa la funcionalidad deseada de nuestro sistema, es momento de hacer un prototipo que nos sirva como base para expandir los casos de uso encontrados, referenciándolos a una posible forma en cómo serán percibidos por los actores cuando actúen con el producto del software. Por tanto, la siguiente sección nos muestra el modelo de interfaces para los casos de uso, teoría que nos servirá para realizar un prototipo de nuestro sistema de control en la sección de diseño.

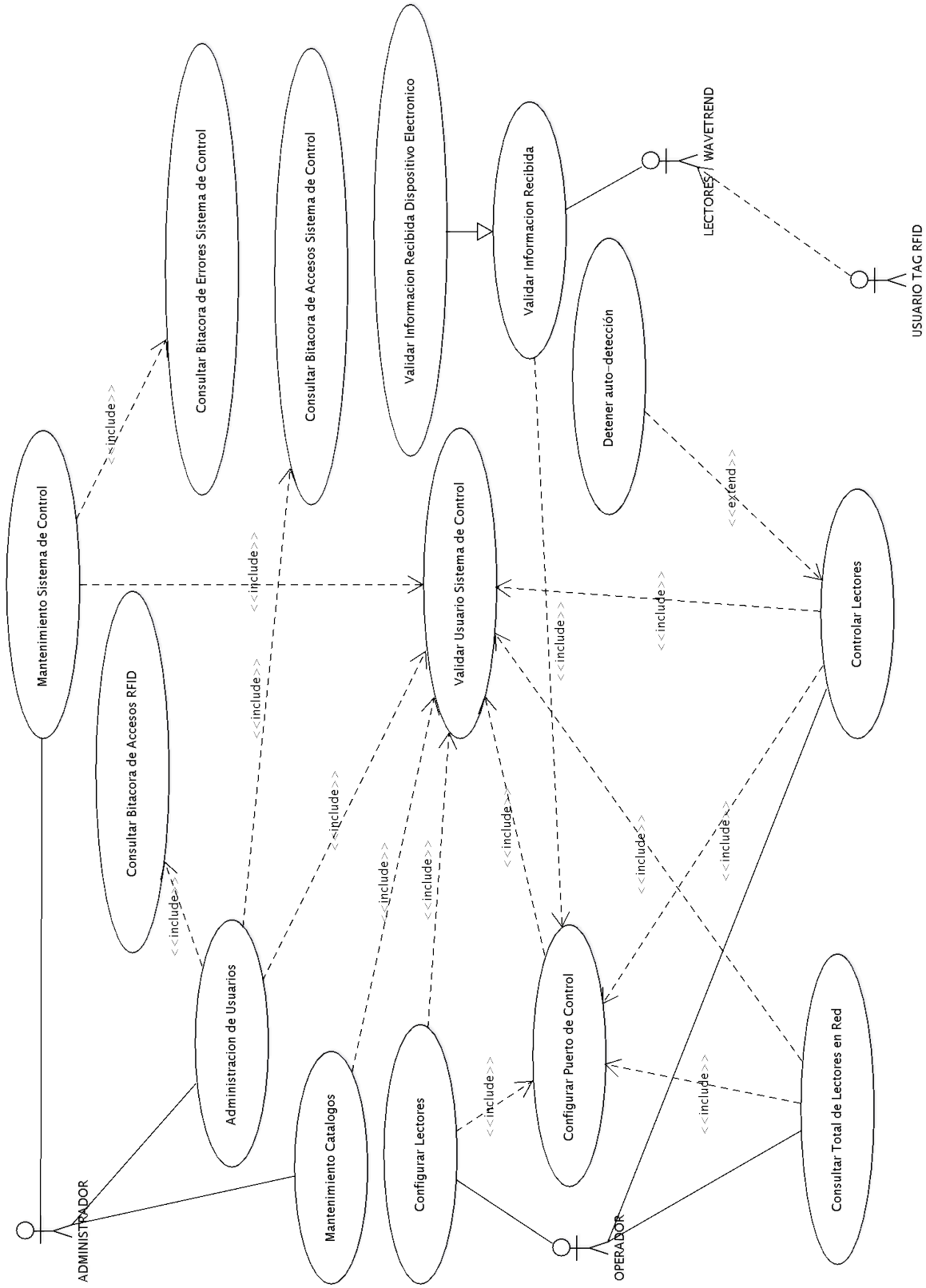


Figura 5.4 Diagrama de casos de uso para nuestro sistema en estudio.

5.1.5 MODELO DE INTERFACES PARA LOS CASOS DE USO.

En esta sección se presenta el modelo de interfaces que se utilizará para la construcción del prototipo del sistema. Estos diseños pueden hacerse en papel o aprovechar una herramienta que simplifique la tarea del diseño de pantallas. El objetivo primordial es la lógica de "navegación" la cual debe basarse en el modelo de casos de uso más que la sofisticación del diseño gráfico.

El *modelo de interfaces* describe la presentación de información entre los actores y el sistema. Se especifica en detalle cómo se verán las interfaces de usuario al ejecutar cada uno de los casos de uso. Si se trata de *Interfaz Humano Computadora* ("HCI - Human Computer Interface") se pueden usar esquemas de cómo vería el usuario las pantallas cuando se ejecuta cada caso de uso.

Normalmente, un prototipo funcional de requisitos mostrando las interfaces de usuario es una estrategia importante. Esto ayuda al usuario a visualizar los casos de uso según serán mostrados por el sistema a ser construido. Tal enfoque elimina muchas posibilidades de malos entendimientos. Cuando se diseñan las interfaces de usuario, es esencial tener a los usuarios involucrados, siendo esencial que las interfaces reflejen la visión lógica del sistema. Esto es realmente uno de los principios fundamentales del diseño de interfaces humanas, donde debe existir consistencia entre la imagen conceptual del usuario y el comportamiento real del sistema. Si las interfaces son protocolos de hardware, se puede referir a los diferentes estándares, como protocolos de comunicación. Estas descripciones de interfaces son por lo tanto partes esenciales de las descripciones de los casos de uso y las deben acompañar.

En la fase de construcción se realizará el diseño de interfaces para nuestro sistema de control.

5.2 CONSTRUCCIÓN DEL SISTEMA.

Una vez concluido el modelo de requisitos y que los casos de uso han sido identificados, clasificados y descritos, comienza una fase de transición importante: la fase de construcción. Es en esta fase donde nuestro esquema de desarrollo iterativo comienza. Debemos escoger algunos casos de uso (o todos, dependiendo del tamaño y complejidad de los mismos y las decisiones del cliente) y comenzamos a construir, empezando por el análisis, llevándolos al diseño, construyéndolos y probándolos; para después llevarlos a un segundo grado de especialización, hacerlos recorrer de nueva cuenta todo el ciclo de desarrollo y así sucesivamente hasta que alcancemos la funcionalidad total deseada, pero recordando que sólo nos enfocaremos (por ser el objetivo de esta tesis) en las fases de análisis y diseño en esta etapa de construcción.

5.2.1 ANÁLISIS DEL SISTEMA.

Iniciaremos el análisis de nuestro sistema expandiendo los casos de uso. Este proceso es descrito más a fondo, a diferencia de los casos de uso de alto nivel.

5.2.1.1 Expansión de los casos de uso seleccionados.

La diferencia básica entre un caso de uso expandido y un caso de uso de alto nivel consiste en que el primero tiene una sección destinada al curso normal de los eventos, que los describe paso por paso, además de tener flujos secundarios con respecto al principal, descritos de la misma manera. Durante esta fase, conviene escribir en formato expandido los casos de uso más importantes y de mayor influencia; en cambio, los menos importantes pueden posponerse hasta el ciclo de desarrollo en el cual van a ser abordados.

Existen varios formatos para expandir los casos de uso. La decisión nuevamente pertenece al analista y diseñador. Para el desarrollo de la presente tesis se utilizó el formato propuesto por Craig Larman y adaptado por el Dr. Alfredo Weitzenfeld, debido a que este formato se adapta a las necesidades del proyecto, presentando las secciones del documento de manera resumida y ordenada y contiene además toda la información necesaria para definir el caso de uso a un nivel detallado para fácilmente iniciar después de su lectura el proceso de diseño.

A continuación (como lo vimos en el Capítulo IV) se muestra el formato utilizado, el cual a diferencia del autor, se agregó alguna información adicional con respecto al formato original.

Caso de Uso:	Nombre del caso de uso
Actores:	Actores primarios y secundarios que interaccionan con el caso de uso.
Tipo:	Tipo de caso de uso: tal y como se definió en los casos de uso niveles 0 y 1. Tipo de flujo: Básico, Inclusión, Extensión, Generalización.
Propósito:	Razón de ser del caso de uso. Se tomará la descripción nivel 0.
Descripción:	Resumen del caso de uso. Se toma como base la descripción nivel 1.
Precondiciones:	Condiciones que deben satisfacerse para poder ejecutar el caso de uso.
Flujo Principal:	El flujo de eventos más importante del caso de uso, donde dependiendo de las acciones de los actores se continuará con alguno de los subflujos.
Subflujos:	Los flujos secundarios del caso de uso, numerados como (S-1), (S-2), etc.
Excepciones:	Excepciones que pueden ocurrir durante el caso de uso, numerados como (E-1), (E-2), etc.

Tabla 5.24 Documentación Casos de Uso.

A continuación se muestran algunos de los casos de uso seleccionados, expandidos mediante el formato anteriormente explicado. Todos los casos de uso expandidos pueden consultarse en el **Anexo A** del presente trabajo.

Caso de Uso 1	Validar Usuario Sistema de Control
Actores	*Administrador, Operador*
Tipo	Primario, Inclusión
Propósito	El sistema debe verificar la autenticidad del rol del actor antes de poder accesar o realizar alguna actividad en el mismo.
Descripción	Este caso de uso es iniciado por los actores *Administrador* y *Operador* introduciendo su login o usuario registrado y su password o clave de acceso en la pantalla principal de acceso al sistema de control. El sistema asigna los permisos necesarios al rol de usuario para actuar dentro del mismo.
Precondiciones	El usuario administrador existe por default en la base de datos.
Flujo Principal	Se presenta al actor la Pantalla de Acceso al Sistema (P-AS). El usuario ingresa su login y password y puede seleccionar entre las opciones

	"Aceptar" o "Cancelar". Si la actividad seleccionada es "Aceptar", se valida la información ingresada con los registros de los usuarios del sistema (S-1) (E-1) y se accesa a la Pantalla Principal del Sistema (P-P). Si la actividad seleccionada es "Cancelar", se limpian los campos de texto y se recarga la Pantalla de Acceso al Sistema (P-AS).
Subflujos	**(S-1) AsignarPermisosSistema** Se asignan los permisos correspondientes en el sistema, dependiendo del rol de usuario registrado y se permite el acceso a la Pantalla Principal del Sistema (P-P).
Excepciones	**(E-1) Error de Acceso:** login/password incorrectos o usuario no registrado. Por favor, intente de nuevo. Se limpian los campos de texto y se recarga la Pantalla Principal (P-1). Después de 3 intentos de validación se sale del sistema. En caso de no existir el usuario operador, el administrador deberá ingresar al sistema y se entra al caso de uso 4 Administración de Usuarios, subflujo AltaUsuario.

Caso de Uso 4	Administración de Usuarios
Actores	*Administrador*
Tipo	Primario, Básico
Propósito	El administrador necesita llevar un control de los usuarios que utilizan el sistema.
Descripción	Este caso de uso es iniciado por el actor *Administrador*. El administrador puede dar de alta, eliminar o modificar algún usuario del sistema, asignando su login, contraseña, datos y perfil. También puede habilitar o inhabilitar el uso del sistema a estos usuarios. Cabe señalar que el usuario administrador se encuentra registrado por default en la base de datos del sistema. El administrador puede también dar de alta, modificar o eliminar algún usuario con Tag RFID asignado, registrando su Tag ID, datos y perfil, así como habilitar o inhabilitar el acceso vía RFID a los usuarios con Tag ID registrados. Es importante señalar que la administración de un *Usuario Tag RFID* también incluye a dispositivos electrónicos con Tag RFID asignado, lo cual es opcional.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control, subflujo AsignarPermisosSistema.
Flujo Principal	Al administrador Se le presenta la Pantalla para la Administración de Usuarios (P-AU) a través de la barra de herramientas en la pestaña "Administración" sub-opción "Usuarios" desde la Pantalla Principal (P-P), con lo que se despliega la pantalla para administración antes nombrada, donde puede elegir entre las opciones Registrar **(S1)**, Modificar **(S2)** o Eliminar **(S3)** algún usuario. También puede habilitar o inhabilitar el uso del sistema a estos usuarios **(S4)**.
Subflujos	**(S1) AltaUsuario** Al administrador se le presentan dos pestañas de selección de usuario, dependiendo de si se desea administrar a un usuario de sistema o

usuario de Tag RFID.
El administrador podrá ingresar los datos necesarios desde la P-AU, tales como número de cuenta, nombre, dependencia, login y password e incluso número de Tag ID, entre otros, dependiendo del tipo de usuario a registrar. Al finalizar el llenado de información, el administrador activará la acción <<Registrar>> para que el nuevo registro se almacene en la Base de Datos del sistema **(E1)**. Al finalizar, aparecerá un mensaje emergente con la leyenda: "El usuario se ha dado de alta con éxito" **(E2)**. El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Si el administrador activa la acción <<Limpiar>>, los datos accesados no son registrados, desaparecen y se procede ingresar nuevos datos. Las acciones <<Modificar>>, <<Cancelar> y <<Eliminar>> se encuentran deshabilitadas para el subflujo AltaUsuario.

(S2) ModificarUsuario
Al administrador se le presentan dos pestañas de selección de usuario, dependiendo de si se desea administrar a un usuario de sistema o usuario de Tag RFID.
El administrador podrá modificar los datos de un usuario desde la P-AU, tales como número de cuenta, nombre, dependencia, login y password e incluso número de Tag ID, entre otros, dependiendo del tipo de usuario a modificar, seleccionándolo a través de un Jlist. Los datos del usuario a modificar se buscan en el sistema y son mostrados en la P-AU, donde el administrador podrá modificar los datos que desee. Al finalizar la modificación de información, el administrador activará la acción <<Modificar>> **(E1)** para que los cambios se almacenen en la Base de Datos del sistema. Al finalizar, aparecerá un mensaje emergente con la leyenda: "El usuario se ha modificado con éxito" **(E2)**. El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Si el administrador activa la acción <<Cancelar>> los cambios no se realizarán. Aparecerá un mensaje con la leyenda: "El usuario NO ha sido modificado". El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Las acciones <<Limpiar>> y <<Registrar>> se encuentran deshabilitadas para el subflujo ModificarUsuario.

(S3) EliminarUsuario
Al administrador se le presentan dos pestañas de selección de usuario, dependiendo de si se desea administrar a un usuario de sistema o usuario de Tag RFID.
El administrador podrá eliminar a un usuario desde la P-AU, seleccionándolo a través de un Jlist. Los datos del usuario a eliminar se buscan en el sistema y son mostrados en la P-AU, El administrador revisa que sea el usuario al que se desea eliminar y activará la acción <<Eliminar>>, para que el usuario sea eliminado del sistema. Previo a esto, aparecerá una ventana emergente para confirmar la eliminación, con el mensaje: "¿Desea eliminar al usuario ####?". Si el administrador activa la acción <<Aceptar>> se procede con la eliminación de usuario, por lo que aparecerá un mensaje con la leyenda: "El usuario se ha eliminado con éxito". El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Si el

	administrador activa la acción <<Cancelar>> de la ventana emergente, la eliminación de usuario no se realizará. Aparecerá un mensaje con la leyenda: “El usuario NO ha sido eliminado”. El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Las acciones <<Limpiar>> y <<Registrar>> se encuentran deshabilitadas para el subflujo EliminarUsuario. **(S4) HabilitarInhabilitarUsuario** Al administrador se le presentan dos pestañas de selección de usuario, dependiendo de si se desea administrar a un usuario de sistema o usuario de Tag RFID. El administrador podrá habilitar/inhabilitar a un usuario desde la P-AU, seleccionándolo a través de un Jlist. Los datos del usuario a habilitar/inhabilitar se buscan en el sistema y son mostrados en la P-AU. El administrador revisa que sea el usuario al que se desea habilitar/inhabilitar y en la sección “Datos de acceso” en la P-AU se encuentra un JcomboBox con las opciones SI/NO y con la pregunta ¿Desea habilitar al usuario? con lo que se elige “SI” para habilitarlo y “NO” para inhabilitarlo. Al finalizar la habilitación/inhabilitación, el administrador activará la acción <<Modificar>> **(E1)** para que el cambio se almacene en la Base de Datos del sistema. Al finalizar, aparecerá un mensaje emergente con la leyenda: “El usuario se ha modificado con éxito” **(E2)**. El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Si el administrador activa la acción <<Cancelar>> los cambios no se realizarán. Aparecerá un mensaje con la leyenda: “El usuario NO ha sido modificado”. El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Las acciones <<Limpiar>> y <<Registrar>> se encuentran deshabilitadas para el subflujo Habilitar/InhabilitarUsuario.
Excepciones	**(E1) Error en el registro:** falta llenar los campos obligatorios. Los campos señalados con un asterisco * son requeridos. El error se describe en una ventana emergente con la opción “Aceptar”. Al aceptar la opción la ventana emergente desaparece. Se regresa a la pantalla anterior para que el administrador llene los datos requeridos. **(E2) Error en el registro:** el usuario con n° de cuenta **####** o número de tag **####** (dependiendo el tipo de usuario) ya existe. El error se describe en una ventana emergente con la opción “Aceptar”. Al aceptar la opción la ventana emergente desaparece. Se regresa a la pantalla anterior para que el administrador corrija los datos ingresados.

Caso de Uso 6	Mantenimiento del Sistema de Control
Actores	Administrador
Tipo	Primario, Básico
Propósito	El administrador necesita dar mantenimiento al sistema de control.
Descripción	Este caso de uso lo inicia el actor *Administrador.* El administrador revisa los registros (Logs) del sistema o bitácora de eventos para visualizar algún error. En caso de algún error registrado, el administrador revisa la referencia del mismo y analiza el código fuente del sistema de control , registros en la BD o comunicación entre el sistema y la red de Lectores, según corresponda.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control y posteriormente haber ejecutado el caso de uso Consultar Bitácora de Errores Sistema de Control.
Flujo Principal	El administrador consulta la bitácora de errores en el sistema de control (Caso de Uso 5) y dependiendo del error el administrador podrá darle seguimiento, eligiendo la opción Mantenimiento del Sistema a través de la barra de herramientas en la pestaña “Mantenimiento” sub-opción “Mantenimiento del Sistema” desde la Pantalla Principal (P-P). Al administrador se le presenta una pantalla emergente (P-MSC) donde encontrará las opciones de mantenimiento: Analizar Excepciones Java del Sistema **(S1)** y Depurar Código del Sistema **(S2)**, a través de un JTabbedPane.
Subflujos	**(S1) AnalizarExcepcionesJavaSistema** Al administrador se le presenta la lectura del log de excepciones Java. Si el administrador activa la acción <<Refrescar>>, se presentarán las últimas líneas del log de excepciones de Java dentro de un JTextArea **(E1)**. Si el administrador activa la acción <<Guardar>>, se le presentará un JfileChooser para guardar la salida del log como un archivo de texto para su próximo análisis. Si el administrador activa la acción <<Limpiar>>, la JTextArea se vaciará, quedando en blanco. Todas las acciones se realizan a través de 3 JButton's , ubicados en la parte inferior izquierda de la P-MSC. **(S2) DepurarCódigoSistema** Al administrador se le presenta un JTextArea. donde podrá seleccionar, a través de un JFileChooser, el archivo Java a editar, activando la acción <<Abrir>>. El archivo aparecerá en la JTextArea para ser editado. Si los cambios realizados son correctos, el administrador podrá activar la acción <<Guardar>> para salvar los cambios al archivo Java. Aparecerá un JDialog con el mensaje: “Se guardó el archivo Java NombreArchivo.java”. Si se desea compilar el código fuente guardado para generar el archvo clase, se puede realizar activando la acción <<Compilar>>. Aparecera el mensaje: “Se ha compilado el archivo JavaNombreArchivo.java con éxito”. Si desea NO guardar los cambios, podrá activar la acción <<Cancelar>> para cerrar el archivo sin modificar. Se limpiará la JTextArea y aparecerá un JDialog con el mensaje: “No se realizaron cambios al archivo Java NombreArchivo.java”. Todas las acciones se realizan a través de 4 JButton's, ubicados en la parte inferior izquierda de la P-MSC.
Excepciones	**(E1) Error en la lectura de log:** archivo no encontrado.

Caso de Uso 8	Configurar Puerto Control
Actores	Operador
Tipo	Primario, Inclusión
Propósito	El operador asigna la velocidad de transmisión y el puerto serie a utilizar, con el cual se enviará y se recibirá información hacia y desde la red de Lectores, respectivamente.
Descripción	Este caso de uso lo inicia el actor *Operador*. El operador asigna en la PC la velocidad de transmisión especificada en el manual de Lectores Wavetrend, la cual puede variar entre 5 distintos valores, pero que por default está configurada como 57600 baudios. El operador elige el puerto serie de la PC a utilizar para la conexión de la red de Lectores RFID. El operador se encuentra listo para controlar y configurar a los Lectores RFID.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control.
Flujo Principal	El operador elige la pestaña COM Port Settings desde la Pantalla Principal del Sistema (P-P) donde puede elegir el puerto serie a utilizar (COM1-COM4) a través de un *JComboBox .* El operador también elige la velocidad de transmisión, la cual varía entre 5 distintos valores en baudios: 9600, 19200, 38400, 57600 *y 115000,* esto a través de un *jComboBox.* Una vez que el operador eligió el puerto serie (Comm Port) y la velocidad de transmisión (Baud Rate) deseados, puede intentar abrir o cerrar el puerto serie para iniciar la transmisión y recepción de datos hacia y desde la red de Lectores RFID, respectivamente **(E1) (E2)**. Para esto se cuenta con dos *jToggleButton,* uno para abrir el puerto serie (OPEN PORT) y otro para cerrarlo (CLOSE PORT).
Subflujos	Ninguno
Excepciones	**(E1) Error al Abrir el Puerto de Control:** el puerto COMx elegido está siendo usado por otra aplicación. Intente desocuparlo o elija otro puerto. Este error se registra en la Bitácora de errores del Sistema (Caso de Uso 5) y su respectiva excepción en Java. **(E2) Error De Velocidad de Transmisión:** la velocidad de transmisión elegida no coincide con los datos recibidos desde la red de Lectores. Intente de nuevo. Este error se registra en la Bitácora de errores del Sistema (Caso de Uso 5) y su respectiva excepción en Java.

5.2.2 DISEÑO DEL SISTEMA.

Después de la etapa de análisis del sistema, donde se explicaron detalladamente los casos de uso para que el analista pueda comenzar a desarrollar con la simple lectura de la documentación, pasamos a la última etapa de nuestra fase de construcción y en sí la más importante para el desarrollo de nuestra tesis y en general de cualquier desarrollo de software: la fase de diseño. Es aquí donde nuestro diseño será orientado a objetos. Es aquí donde diseñaremos nuestra base de datos a utilizar en conjunto con el sistema de control RFID y es en esta etapa donde diseñaremos el prototipo del sistema de control, todo esto con formatos finales, producto de las iteraciones de la metodología de Proceso Unificado.

5.2.2.1 Modelo del dominio del problema.

El modelo del dominio del problema define un modelo de clases común para todos los involucrados, analistas al igual que clientes. Este modelo de clases consiste de los objetos del dominio del problema, o sea objetos que tienen una correspondencia directa en el área de la aplicación. Como los usuarios y clientes deberían reconocer todos los conceptos, se puede desarrollar una terminología común al razonar sobre los casos de uso, y por lo tanto disminuyendo la probabilidad de malos entendimientos entre el analista y el usuario. Al discutirlo, se evolucionará el modelo del dominio del problema.

El propósito principal del dominio del problema es formar una base común de entendimiento del desarrollo y no para definir el sistema completo. Por lo tanto, se pueden aprovechar algunas de las heurísticas de los métodos anteriores para la identificación de los objetos en el dominio del problema, logrando un glosario o diccionario de clases que sirve como común denominador a todos los componentes del sistema, incluyendo a las diversas personas involucradas a lo largo del desarrollo.
Aunque es suficiente describir el dominio del problema en término de objetos o clases, es posible refinar más aún mediante la inclusión de asociaciones, atributos, herencia y operaciones, siempre y cuando esto ayude a comprender mejor el problema y no se vuelva un esfuerzo demasiado grande durante esta etapa.

En las siguientes secciones identificaremos las clases del dominio del problema junto con aspectos adicionales, cómo asociaciones y atributos. Identificar muchos objetos o conceptos constituye la esencia del análisis y diseño orientado a objetos y el esfuerzo se recompensa con los resultados obtenidos durante la fase de diseño e implementación. La principal cualidad del modelo del dominio del problema es que representa cosas del mundo real, no componentes software. En términos prácticos, un mapa mental.

5.2.2.2 Identificación de Clases.

La identificación de clases del dominio del problema se obtiene principalmente de algún documento textual que describa el sistema. Aunque pudiéramos tomar como punto de partida los documentos desarrollados para el modelo de casos de uso, a menudo **la descripción original del problema** es **suficiente**. Se comienza este proceso mediante la identificación de las clases candidatas, explícitas o implícitas, a las que se refiera la descripción del problema. Para ello se extraen todos los sustantivos de la descripción del problema o de algún otro documento similar, de acuerdo a las siguientes consideraciones:

- Los sustantivos en la descripción del problema son los posibles candidatos a clases de objetos.
- Durante esta etapa, se debe identificar entidades físicas al igual que entidades conceptuales.
- No se debe tratar de diferenciar entre clases y atributos durante ésta etapa.
- Dado que no todas las clases se describen de manera explícita, siendo algunas implícitas en la aplicación, será necesario añadir clases que pueden ser identificadas por nuestro conocimiento del área.
- Se debe revisar los pronombres en la descripción del problema para asegurar que no se haya perdido ningún sustantivo descrito de forma implícita.
- Para facilitar la identificación de clases, se subrayan todos los sustantivos de la descripción del problema. En el caso del sistema de reservaciones de vuelos, partimos de la descripción del problema y subrayamos todos los sustantivos, como se ve a continuación (utilizamos nuestra descripción del problema inicial):

El Sistema de Control para los Lectores RFID Wavetrend L-RX201 es un sistema que permite al operador controlar y configurar los Lectores enviando los comandos correspondientes para cada acción específica que se desee realicen estos dispositivos RFID, además de consultar información referente al sistema y a los usuarios del sistema con su correspondiente registro de etiqueta electrónica o Tag. Se desea tener tanto un ambiente gráfico de control como una terminal para el envío de comandos.

El sistema presenta en su pantalla principal un mensaje de bienvenida y los campos de *login (usuario)* y *password (contraseña)* para poder tener acceso al sistema de control. Estos datos, al introducirse en los campos correspondientes, deben validarse antes de permitir el acceso. Un usuario no registrado no podrá tener acceso al sistema.

Una vez registrado el usuario (a través del administrador del sistema) y después de haberse validado los registros de *usuario* y *contraseña* se tiene acceso a la pantalla inicial de trabajo (Framework) en donde se pueden seleccionar las siguientes actividades:

- Configuración del puerto de control.
- Control y configuración de los Lectores RFID (envío de comandos)
- Consulta de datos registrados sobre usuarios y Tags.
- Manipulación de datos registrados (rol Administrador)
- Consulta de información sobre la red de Lectores RFID

La configuración del puerto de control nos permite seleccionar la velocidad de transmisión a través del puerto serie (COMx), la cual debe ser la misma velocidad configurada en cada uno de los Lectores RFID en la red, según el manual de éstas. También nos permite seleccionar el puerto serie a través del cual se va a controlar la red de Lectores RFID, en caso de tener más de un puerto serie en nuestra PC.
El control y la configuración de los Lectores RFID se realiza a través de la selección (por medio de un Radio Button) del comando que se desea enviar a la red, además de seleccionar (por medio de Spinners) la red y el Lector o Lectores RFID a quienes se les desea enviar el comando.

Cuando ya se tiene lista la configuración del puerto de control y comando deseados, se envían (por medio de un Button) hacia la red de Lectores RFID.

Tanto los comandos como las respuestas de los Lectores RFID a estos comandos se pueden ir visualizando en pantalla en la sección de respuestas de los Lectores (Reader´s Responses) ubicada en la parte inferior derecha de nuestra aplicación gráfica.

La consulta de datos registrados sobre usuarios y Tags se encuentra en la barra de menús en la sección *Herramientas*, submenú *Consultar Información,* el cual despliega las opciones *Consultar usuario* y *Consultar Tag*. Al seleccionar alguna de las dos opciones se abre una subpantalla en la cual se puede realizar la búsqueda por nombre del usuario o por número de Tag. Si el usuario o el Tag no se encuentran registrados, la búsqueda regresará un mensaje de no existencia de registro.

La manipulación de datos registrados sólo puede ser realizada por el administrador del sistema. Esta actividad se encuentra en la barra de menús en la sección *Herramientas,* submenú *Modificar Información*, el cual despliega las opciones *Añadir nuevo usuario*, *Añadir nuevo Tag, Eliminar Usuario, Eliminar Tag, Modificar Información Usuario* y *Modificar Información Tag.* Al seleccionar alguna de las opciones se abre una subpantalla en la cual se puede realizar la inserción, eliminación o modificación de datos según se desee.

La consulta de información sobre la red de Lectores muestra el número de Lectores RFID conectadas en red con el sistema.

A partir de estos sustantivos se prepara una lista inicial de clases candidatas, como se muestra en la Tabla 5.25 Se debe excluir clases repetidas, manteniendo todos los nombres en singular.

CLASES CANDIDATAS		
Sistema de control	campos	PC
Lectores RFID	login	respuestas
operador	password	pantalla
comandos	acceso	barra de menús
control	datos	herramientas
configuración	administrador del sistema	submenú
información	actividades	subpantalla
usuarios	puerto de control	búsqueda
Tag	velocidad de transmisión	mensaje
ambiente gráfico	opciones	registro
terminal	puerto serie	inserción
pantalla principal	consulta de datos	eliminación
mensaje de bienvenida	manipulación de datos	modificación
número	red	

Tabla 5.25 Clases candidatas para nuestro sistema de control identificadas de la descripción del problema.

5.2.2.3 Selección de Clases.

A partir de las clases candidatas, se seleccionaron las clases relevantes tomando en cuenta los siguientes consideraciones:

- Todas las clases deben tener sentido en el área de la aplicación, la relevancia al problema debe ser el único criterio para la selección.
- Se deben eliminar clases redundantes, si estas expresan la misma información. La clase más descriptiva debe ser guardada.
- Se deben eliminar clases irrelevantes, que tienen poco o nada que ver con el problema. Esto requiere juicio porque en un contexto una clase puede ser importante mientras que en otro contexto la clase podría no serlo.
- Se deben clarificar las clases imprecisas. Algunas clases pueden tener bordes mal definidos o demasiado generales.
- Se deben eliminar las clases que debieran ser atributos más que clases, cuando los nombres corresponden a propiedades más que entidades independientes.
- Se deben eliminar las clases que debieran ser roles más que clases, cuando los nombres corresponden al papel que juegan las clases más que entidades independientes.
- Se deben eliminar las clases que debieran ser operaciones más que clases, si las entidades representan operaciones que son aplicadas a los objetos y no entidades manipuladas por si mismas.
- Se deben eliminar las clases que corresponden a construcciones de implementación si están alejadas del mundo real, por lo cual deben ser eliminadas del análisis. No se necesita una clase para representar una entidad física. Por ejemplo: subrutinas, listas, y arreglos, son clases típicas de implementación.
- Se debe eliminar clases que correspondan aspectos de interfaces de usuario y no de la aplicación.
- Se deben eliminar clases que correspondan a todo un sistema completo.
- Se deben eliminar clases que correspondan a actores del sistema.

- Se deben agregar clases implícitas que no aparezcan en la descripción del problema. Tomaremos algunas de las clases candidatas del sistema de control identificadas anteriormente y seleccionamos las que mejor se apliquen a nuestro problema.

Las clases identificadas se muestran en la Tabla 5.26 Nótese que se agregaron nuevas clases, que no aparecían en la descripción del problema. Esto se hizo para lograr un dominio más completo. En general, distintos analistas identificarían listas similares de clases, aunque siempre con alguna variación. Algunos sustantivos se engloban en una idea general para lograr así las clases identificadas:

CLASES IDENTIFICADAS	
AccesoSistema FrameCargandoRFID FramePrincipal AdministrarUsuarios BitácoraAccesoSistema	BitacoraAccesosTag BitácoraErroresSistema MantenimientoCatálogos MantenimientoSistemaControl

Tabla 5.26 Clases identificadas para nuestro Sistema de Control de Lectores RFID.

5.2.2.4 Diagrama de clases.

Después de haber identificado y seleccionado las clases, se debe construir el diagrama de clases para el dominio del problema. Este diagrama se muestra en la Figura 5.5 y puede ayudar a identificar clases adicionales, y servirá como base para encontrar las atributos y asociaciones entre ellas. Obsérvese que todas las clases se generalizan de la clase JFrame.

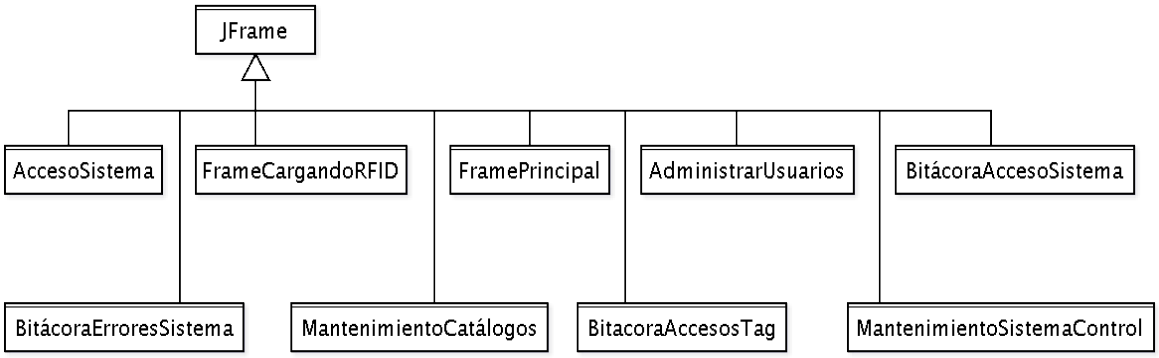


Figura 5.5 Diagrama de Clases identificadas para nuestro Sistema de Control.

5.2.2.5 Diccionario de Clases.

El diccionario de clases describe textualmente las clases identificadas durante el modelo del dominio del problema. Este diccionario sirve como un glosario de términos y se muestra a continuación:

AccesoSistema: Clase JFrame que servirá para validar el acceso login / password al sistema de control y asignar los permisos correspondientes, dependiendo si el usuario es Administrador u Operador.

FrameCargandoRFID: Clase JFrame que servirá para mostrar el avance de carga en porcentaje del sistema de control.

FramePrincipal: Clase JFrame principal. A partir de ella se lanzarán las demás clases a partir de métodos Java.

AdministrarUsuarios: Clase JFrame que servirá para dar de alta, modificar o dar de baja a los usuarios tanto del sistema como a usuarios Tag.

BitacoraAccesoSistema: Clase JFrame para ingresar a los registros en la Base de Datos de los usuarios que han accesado al sistema de control.

BitacoraAccesosTag: Clase JFrame para ingresar a los registros en la Base de Datos de los usuarios que han accesado a puertas a través de los Lectores RFID utilizando su Tag registrado.

BitacoraErroresSistema: Clase JFrame para ingresar a los registros en la Base de Datos de los errores que han surgido en el sistema de control.

MantenimientoCatálogos: Clase JFrame para realizar el mantenimiento correspondiente a los catálogos de la Base de Datos, realizar respaldos y depuraciones.

MantenimientoSistemaControl: Clase JFrame para realizar el mantenimiento correspondiente al sistema de control, a través del código fuente de las clases existentes.

Partiendo de las clases candidatas, encontramos varios sustantivos que al final resultaron atributos o métodos. Complementando con algunos atributos y métodos pensados y algunos otros extraídos de los casos de uso expandidos se tiene a continuación en las Figuras 5.6 y 5.7 el diagrama de clases con atributos y con métodos, respectivamente.

Figura 5.6 Diagrama de Clases con Atributos para nuestro Sistema de Control.

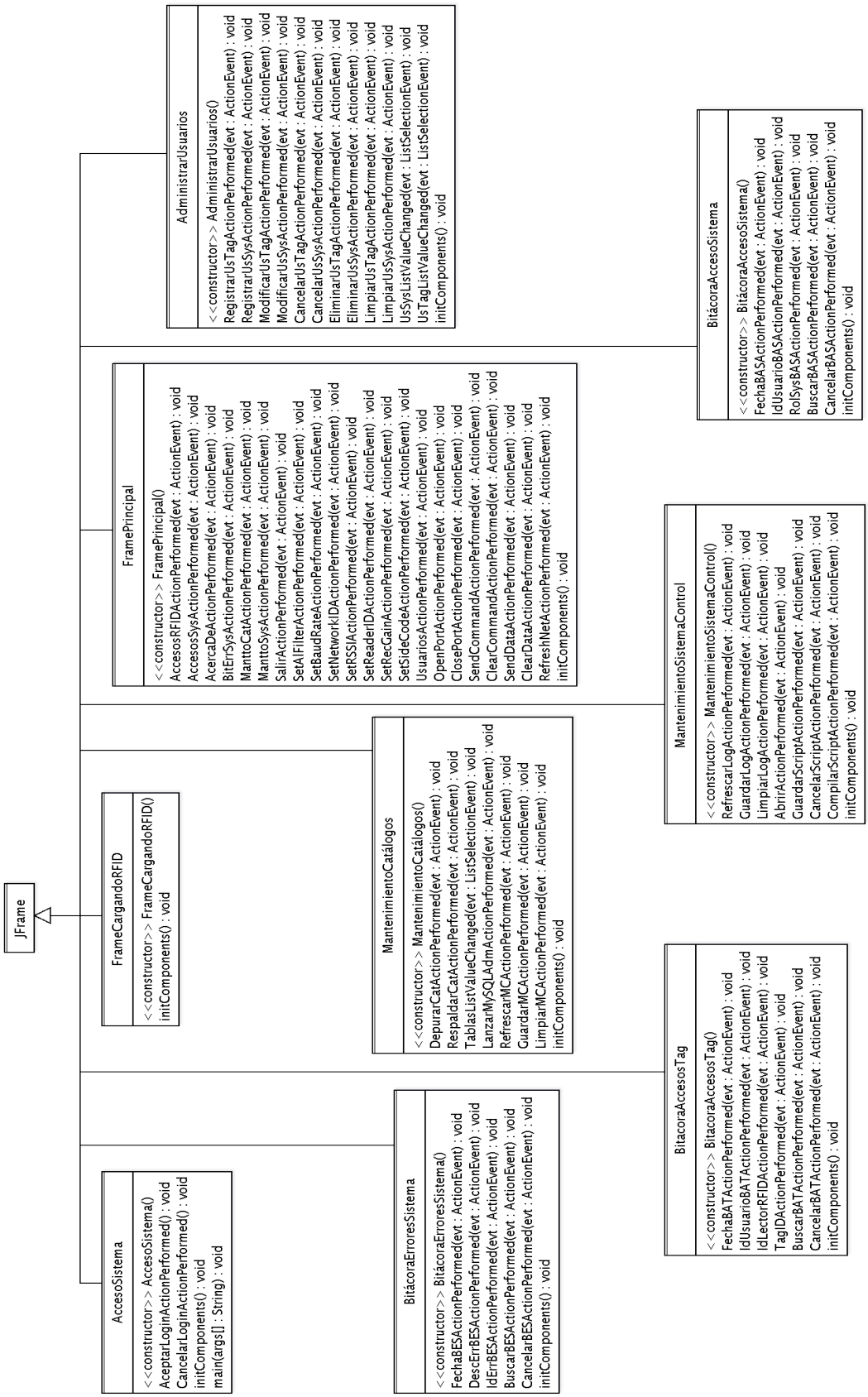


Figura 5.7 Diagrama de Clases con Métodos para nuestro Sistema de Control.

5.2.2.6 Diseño de la base de datos.

El diseño de la base de datos a utilizar en nuestro sistema de control se llevó a cabo con base en el modelo Entidad / Relación y utilizando las reglas de normalización. Se llegó a la tercera forma normal, basándonos en los siguientes conceptos:

- El proceso de normalización de bases de datos consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional.

- Las bases de datos relacionales se normalizan para:

  - Evitar la redundancia de los datos.
  - Evitar problemas de actualización de los datos en las tablas.
  - Proteger la integridad de los datos.

- En el modelo relacional es frecuente llamar tabla a una entidad, aunque para que una tabla sea considerada como una entidad tiene que cumplir con algunas restricciones:

  - Cada columna debe tener su nombre único.
  - No puede haber dos filas iguales. No se permiten los duplicados.
  - Todos los datos en una columna deben ser del mismo tipo.

- En general, las primeras tres formas normales son suficientes para cubrir las necesidades de la mayoría de las bases de datos. La tercera forma normal es un objetivo generalmente aceptado para eliminar la redundancia de un diseño de base de datos. El creador de estas 3 primeras formas normales y 12 en total fue Edgar F. Codd.

- **Primera Forma Normal**. Todos los atributos deben tener un solo valor para cada instancia. Si un atributo tiene múltiples valores, se crea una entidad adicional y lo relaciona con la entidad original mediante una relación M:1

- **Segunda forma Normal**. Un atributo debe de ser dependiente del identificador único completo. Si un atributo no es dependiente del identificador único completo, esta fuera de lugar y deberá ser movido.

- **Tercera Forma Normal**. Ningún atributo no-UID(UID- identificador único) puede ser dependiente de otro atributo no-UID. Si un atributo depende de otro atributo no-UID, es necesario mover ambos, el atributo dependiente y el atributo del que depende, a una nueva entidad relacionada con la entidad actual.

Con base en lo anterior, se tiene el resultado del modelo para el diseño de la Base de Datos a utilizar en la figura 5.8. Se utilizó para ello el software de libre distribución DBDesigner en su versión 4:

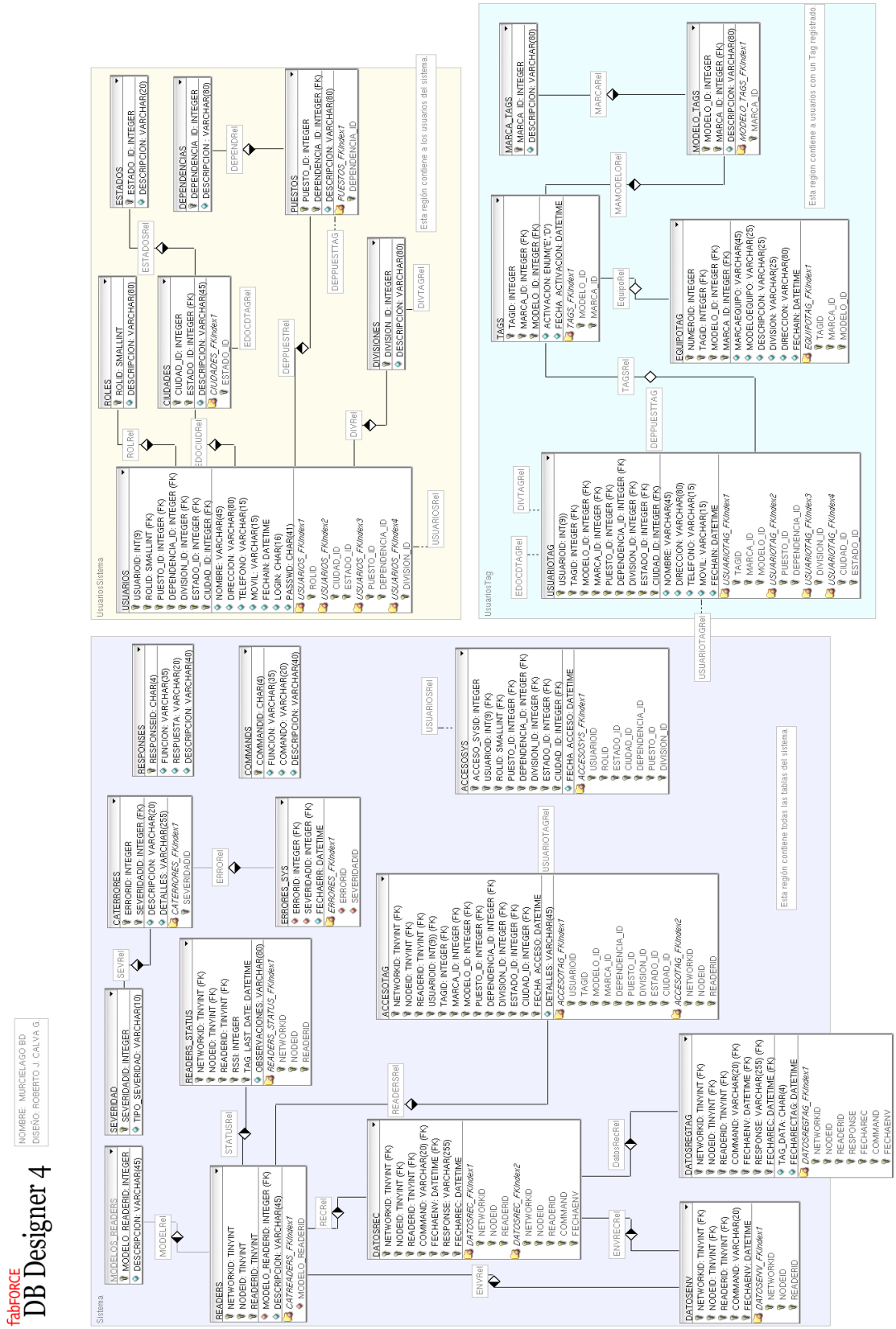


Figura 5.8 Modelo Entidad – Relación de la Base de Datos MURCIELAGOBD con atributos.

A continuación describimos las tablas:

NOMBRE	DEFINICIÓN
MODELOS_READERS	Catálogo que guarda los distintos modelos de Lectores RFID.
READERS	Catálogo de direcciones en la red de Lectores.
READERS_STATUS	Tabla que proporciona el estatus actual de los Lectores.
DATOSENV	Tabla que guarda los datos enviados a la red de Lectores.
DATOSREC	Tabla que guarda los datos recibidos desde los Lectores.
DATOSREGTAG	Tabla que guarda los datos de Tags recibidos.
ACCESOTAG	Tabla que guarda los datos de acceso con Tag registrado.
SEVERIDAD	Catálogo de severidad de errores.
CATERRORES	Catálogo de errores identificados en el sistema de control.
ERRORES_SYS	Tabla que guarda los errores encontrados en el sistema.
RESPONSES	Catálogo de Respuestas para los Lectores L-RX-201.
COMMANDS	Catálogo de Comandos para los Lectores L-RX-201.
ACCESOSYS	Tabla que guarda los datos de acceso al sistema de control.
ROLES	Catálogo de roles en el sistema de control.
USUARIOS	Catálogo de usuarios del sistema de control.
ESTADOS	Catálogo de Estados de residencia de los usuarios.
CIUDADES	Catálogo de Ciudades de residencia de los usuarios.
DEPENDENCIAS	Catálogo de Dependencias en la UNAM.
PUESTOS	Catálogo de Puestos en la UNAM.
DIVISIONES	Catálogo de Divisiones en la UNAM.
MARCA_TAGS	Catálogo de marcas de Tags.
MODELO_TAGS	Catálogo de modelos de Tags.
TAGS	Tabla de activación o desactivación de Tags.
EQUIPOTAG	Catálogo de equipos con un Tag registrado.
USUARIOTAG	Catálogo de usuarios con un Tag registrado.

Tabla 5.27 Descripción de las Tablas de la Base de Datos.

El Diccionario de Datos se muestra en el **Anexo E** del presente trabajo.

5.2.2.7 Prototipo del sistema.

Como apoyo en la descripción de los casos de uso expandidos y del modelo de interfaces, mostraremos el prototipo del sistema diseñado a través de la herramienta de libre distribución NetBeans de Sun Microsystems que nos ayudará para tener una buena visión del proyecto, comenzando con la pantalla inicial. Dado que el caso de uso principal del sistema es que todo usuario deba haberse firmado, la primer pantalla será la clásica de inicio de sesión login/password como se muestra en la Figura 5.9. Si la llave login/password es correcta, el usuario del sistema podrá entrar a él, dependiendo del rol correspondiente (*Operador* o *Administrador*), de lo contrario el acceso será denegado. Aparecerá una pantalla de carga del sistema (Figura 5.10) antes de accesar a la Pantalla Principal.

Figura 5.9 Pantalla de Acceso al Sistema (P-AS).

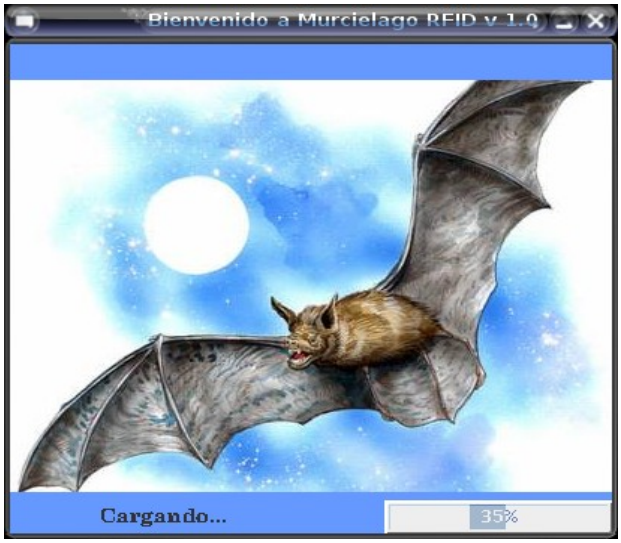


Figura 5.10 Pantalla de Carga del Sistema (FrameCargandoRFID).

Si el usuario es Administrador (el usuario administrador existe por default en la base de datos del sistema como administrador/administrador), se cargarán los permisos correspondientes en la Pantalla Principal del Sistema (P-P) con las opciones “Mantenimiento” y “Administración” en la barra de herramientas, además de las etiquetas de control y configuración, como se observa en la Figura 5.11:

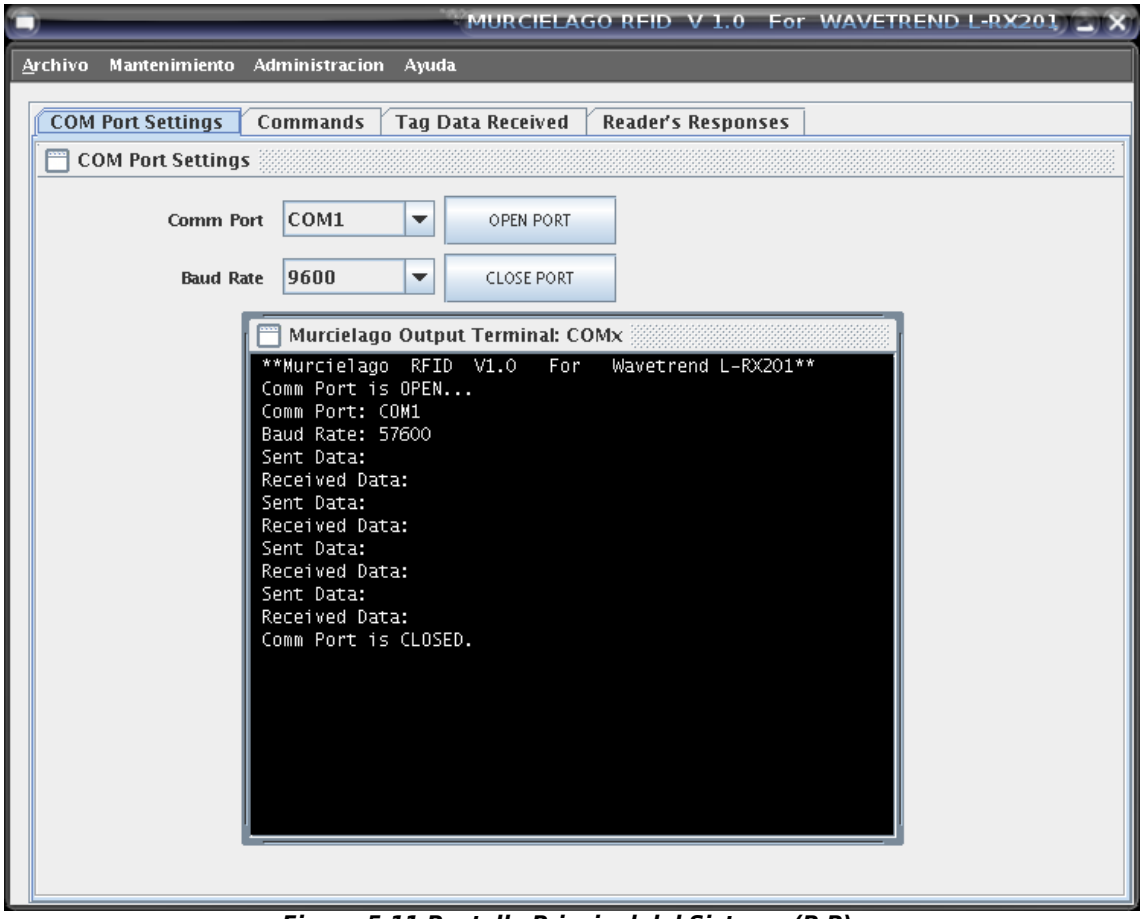


Figura 5.11 Pantalla Principal del Sistema (P-P).

Si el usuario es Operador , se cargarán los permisos correspondientes en la Pantalla Principal del Sistema para Operador (P-PO) donde las opciones “Mantenimiento” y “Administración” NO EXISTIRAN en la barra de herramientas, pues el usuario Operador sólo tiene permisos sobre la utilización de las etiquetas de control y configuración del sistema, como se puede observar en la Figura 5.12. En la Figura se observa la pestaña “COM Port Settings”, a través de la cual el Operador configurará el puesto serie a utilizar para el control de la red de Lectores RFID y podrá observar el resultado a través de la terminal de salida “COMx”.

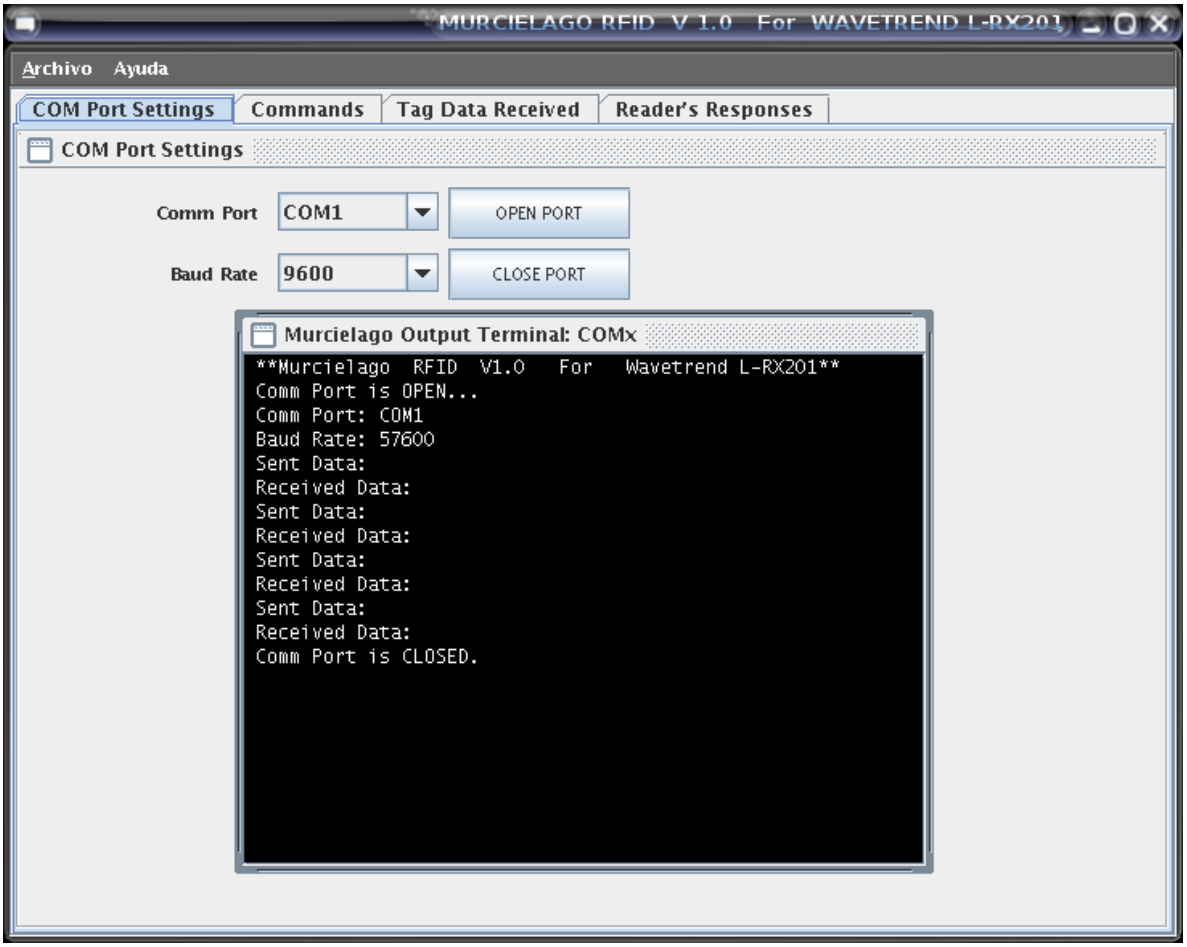


Figura 5.12 Pantalla Principal del Sistema para Operador (P-PO).

Continuando con el usuario Operador, se tiene en la Pantalla Principal una pestaña de nombre "Commands" (Fig. 5.13). Es en esta sección del sistema donde el usuario Operador podrá controlar y configurar a la red de Lectores Wavetrend L-RX201 e incluso observar en la terminal de salida todo lo que pudiera ocurrir al enviar y recibir comandos. Se tienen además dos pestañas más, una para observar los datos de Tag recibidos (Tag Data Received) y la siguiente para observar las respuestas del Lector(es) a cada comando enviado (Reader's Responses).

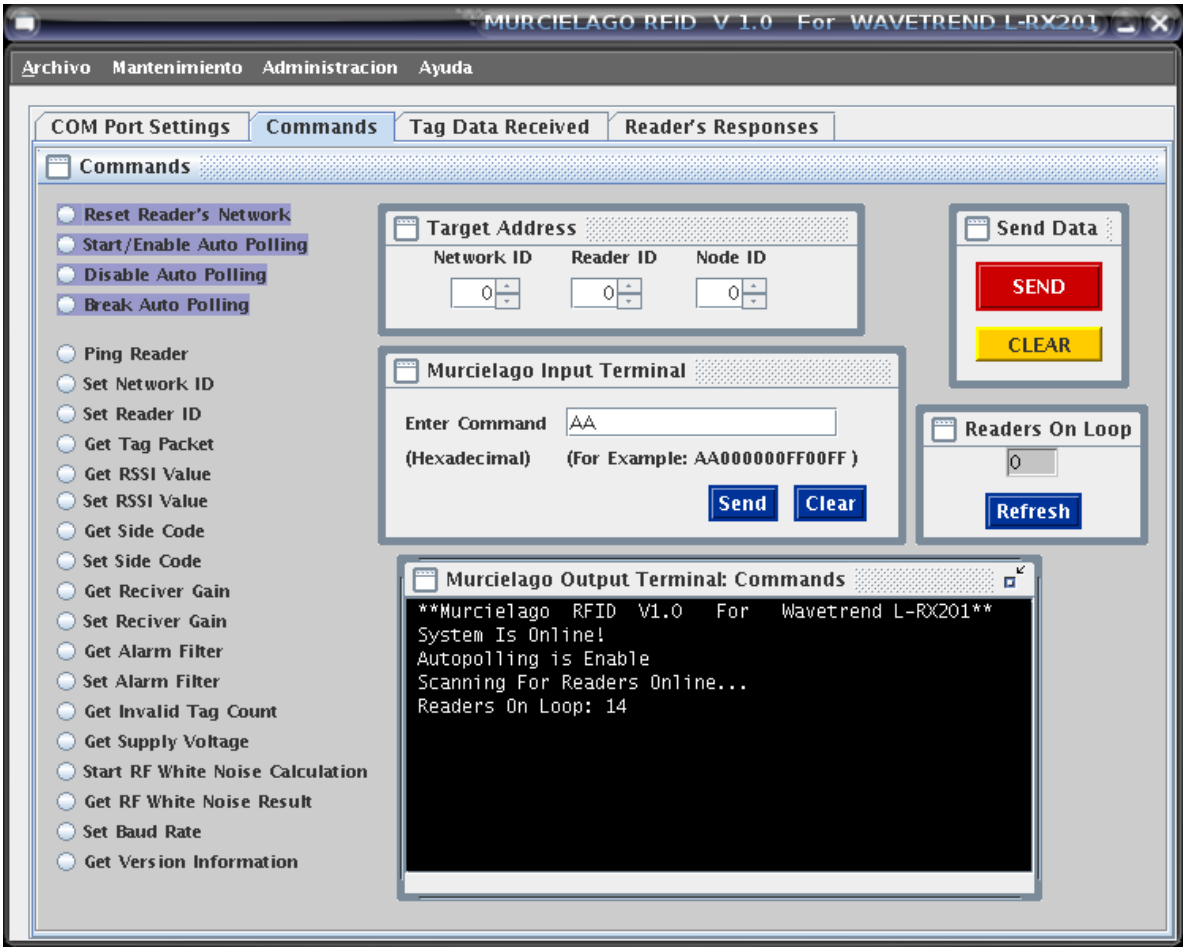


Figura 5.13 Comandos de Control y Configuración de Lectores L-RX201.

Las diversas interfaces de uso se muestran en el **Anexo B** referente al prototipo del sistema, dependiendo del tipo de rol, ya sea operador o administrador.

Como comparación, se muestra en la Figura 5.14 el software comercial de la marca Wavetrend, llamado "Reader Network Analyzer" para el control de Lectores de la misma marca, para modelos L-RX200, L-RX201 y W-RX201. Se observa que el software comercial sólo nos permite el control y configuración de la red de Lectores RFID, es decir, no existe opción de almacenar los datos enviados y recibidos y tampoco se tiene una llave login/password para asegurarnos que alguna persona no autorizada manipule nuestra red de Lectores, algunas características que nuestro sistema de control sí posee.

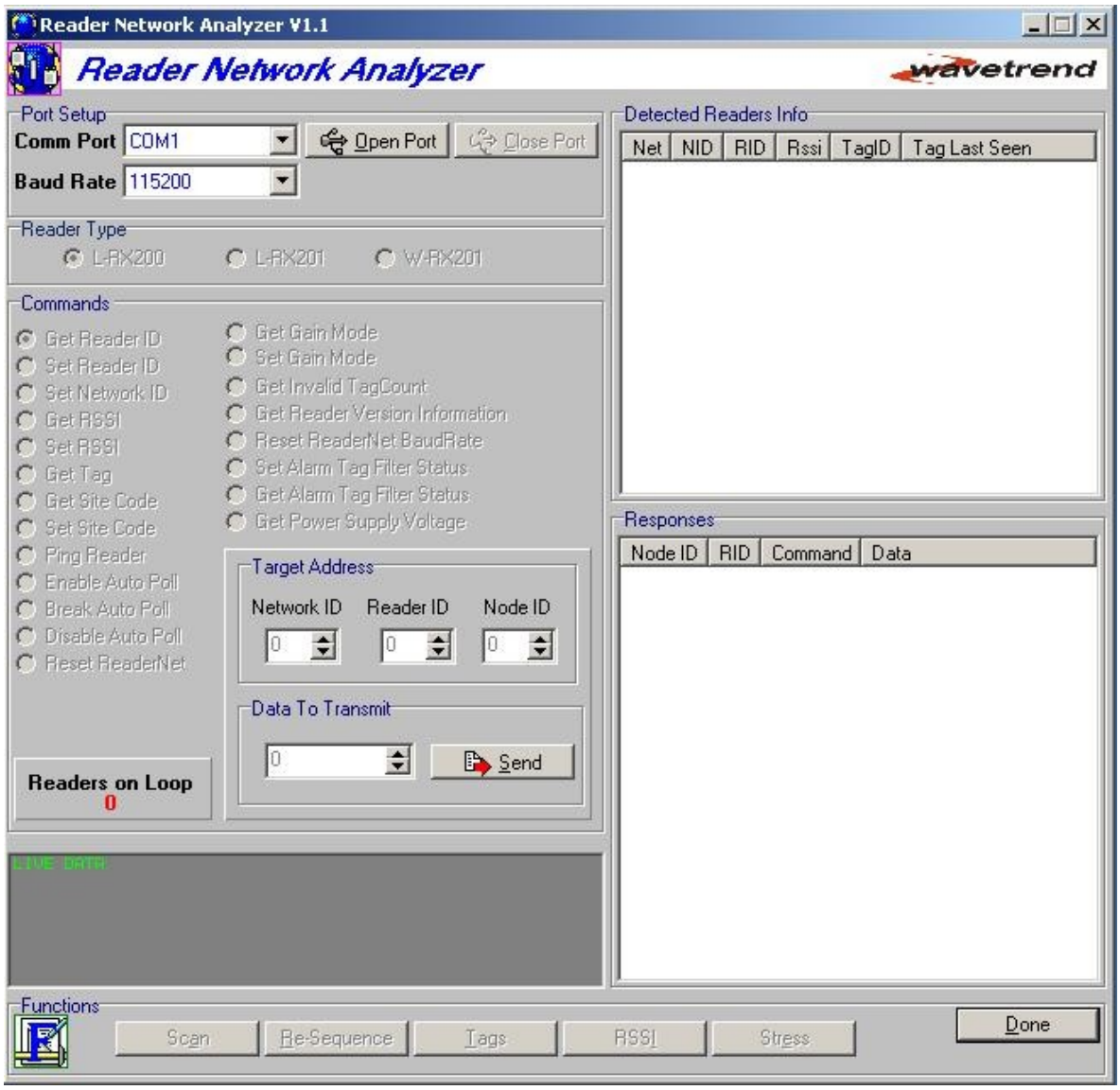


Figura 5.14 Software comercial de la marca Wavetrend para el control y configuración de la red de Lectores RFID de la misma marca.

CONCLUSIONES.

El análisis y diseño realizado nos lleva a una solución de desarrollo de software que cumple con las necesidades de un sistema de control completo y que a diferencia del software comercial y propiedad de Wavetrend Technologies, el sistema de control diseñado presenta una mejor interfaz gráfica y una mejor distribución de los componentes, así como la administración directa de usuarios y del propio sistema a través del mismo. Lo más importante del sistema de control es su filosofía: al ser un software libre y de código abierto, bajo la licencia GNU GPL, le permitirá que toda la comunidad de desarrolladores de software libre en el mundo participen en este proyecto y así lograr obtener mejores versiones del sistema de control en un tiempo muy corto, además de no tener que pagar miles de dólares por un sistema de control, con lo que se logra un ahorro considerable para el Centro de Diseño y Manufactura (CDM) y para la Torre de Ingeniería en la UNAM. Esta filosofía está muy lejos de implementarse en México. Muy pocas empresas lo hacen y a diferencia de países como India, Brasil y Argentina, los cuales son países en vías de desarrollo y en el caso de la India ya una potencia mundial, han crecido enormemente en cuanto a software se refiere y se han independizado en algún porcentaje de grandes monopolios como Microsoft, por ejemplo. En el caso de la India, al paso de los años se ha convertido en el mayor productor de software y han cambiado casi al 100% sus sistemas por software libre.

El desarrollo de este trabajo me dio la posibilidad de conocer las tecnologías que son en este momento líderes en el desarrollo de software alrededor del mundo. Por ejemplo, MySQL, un gestor de bases de datos de fuente abierta y con parte comunitaria de libre distribución, dio un paso importante en cuanto a crecimiento al ser adquirida en abril de 2008 por SUN Microsystems, un líder de IT y al igual que Java, el lenguaje orientado a objetos por excelencia mejor conocido también propiedad de SUN es el lenguaje más utilizado debido a que permite programar casi cualquier aplicación y puede correr hasta en un refrigerador (recordando la ideología de casas inteligentes). Y qué decir del sistema operativo Linux, una de las maravillas del software del siglo pasado. Debian GNU/Linux, es una de las distribuciones basadas en Linux más importantes, influyentes y estables que existen. Además, es el proyecto con filosofía de software libre y código abierto más puro que existe.

Sin lugar a dudas, el presente trabajo de tesis me abrió un camino dirigido a estas grandes filosofías del software actual y también a ver al software como servicio, no solamente como producto.

El sistema se diseñó basándonos en el modelo RUP (Rational Unified Process) debido a las ventajas de ser un modelo cíclico, siempre iterativo y envolvente, lo cual nos permite interactuar de forma constante con el cliente y así llegar al objetivo deseado. La ventaja de este modelo con respecto a otros es que se toman en cuenta los problemas que siempre ha presentado la elaboración de software, y al ser un conjunto de metodologías adaptables al contexto y necesidades de cada organización o de cada proyecto, nos permite la personalización de acuerdo a necesidades.

Con el análisis y diseño del sistema de control RFID y con el prototipo mostrado, además de todo el código fuente generado, se proporciona un gran avance para que se continúe con este proyecto en una siguiente fase de desarrollo. Por desgracia, no pude lograr desarrollar el sistema de control y quedó sólo en análisis y diseño, debido a intereses personales del Centro de Diseño y Manufactura y de Torre de Ingeniería, los cuales realizan proyectos que pueden vender después. Al enterarse que la presente tesis se basaba en la filosofía del software libre y que a su vez iba a ser una tesis, me retiraron los tags y Lectores argumentando que los iban a utilizar en otro proyecto y que me las prestarían después, con lo cual perdí un año de desarrollo de la presente tesis esperando los Lectores RFID.

La ideología del CDM y de algunos investigadores de la Torre de Ingeniería, al menos en mi caso, es lo que atrasa el desarrollo de software e incluso de hardware en nuestro país. Mientras que en Europa, Estados Unidos, India y algunos países de sudamérica lo que gobierna es la comunidad de software libre, quienes como comunidad comparten sus ideas y códigos para crear software que todos pueden utilizar y compartir y agregarlo o adaptarlo a sus necesidades y que los tiempos de desarrollo son mínimos, en México aún se cree, sin generalizar, en herramientas propietarias y de fuente cerrada por el temor y desconocimiento tanto de software libre como de las licencias GPL.

Con el presente trabajo, se ha logrado recopilar toda la información necesaria para que la tecnología RFID pueda ser desarrollada en nuestro país utilizando los tags y lectores Wavetrend, pudiéndose implementar en nuestra Universidad, recordando que los inventarios tanto de laboratorios como de nuestras bibliotecas pueden lograrse fácilmente y reforzar la seguridad con esta tecnología y por supuesto con nuestro sistema de control.

El análisis y diseño para el sistema de control de Lectores Wavetrend RFID en la presente tesis, además del prototipo de sistema y código fuente, es suficiente para que en un posterior estudio se pueda proceder con el desarrollo y pruebas del sistema, siempre y cuando se faciliten los Lectores y tags para continuar con el proceso de desarrollo y lograr así una independencia de software en esta novedosa rama de la Identificación por Radiofrecuencia.

BIBLIOGRAFÍA.

- Applying UML and Patterns. Craig Larman. Prentice Hall. 2001
- Ingeniería de Software Orientada a Objetos con UML, Java e Internet.Alfredo Weitzenfeld. Thomson Learning. ITAM. 2004
- An Introduction to Object-Oriented Analysis and Design. Craig Larman. Prentice Hall. 2001
- El Lenguaje Unificado de Modelado. Grady Booch. James Rumbaugh. Ivar Jacobson. Addison Wesley. 2000
- Entendiendo UML: La guía del desarrollador. Paul Harmon y Mark Watson. Morgan Kauffman Publishers, Inc. 1998.
- MySQL para Windows y Linux. Cesar Pérez. Alfaomega. 2004
- Java Cómo Programar. Harvey & Paul Deitel. Prentice Hall. 2003
- Sistemas de Comunicación Digitales y Analógicos. Leon Couch. Prentice Hall. 2003
- Microcontroladores PIC. 1a Parte. José Angulo. Mc Graw Hill. 2000
- Sistemas Digitales. Principios y Aplicaciones. Ronald Tocci. Pearson Education. 2003
- Manual de Administración de Linux. Steve Shah. Mc Graw Hill. 2005
- UNIX y Linux. Guía Práctica. Sebastián Sánchez. Alfaomega. 2002
- Wavetrend User Manual For the L-RX201. Wavetrend Technologies.
- GNU General Public License Version 2. GNU Project. June 1991.

REFERENCIAS.

- **PAGINA OFICIAL DE JAVA:**
  java.sun.com
- **API DE JAVA:**
  java.sun.com/j2se/1.5.0/docs/**api**/
- **PAGINA OFICIAL DE MySQL:**
  www.mysql.com
- **PAGINA OFICIAL DE DEBIAN GNU/LINUX:**
  www.debian.org
- **PAGINA OFICIAL DE ARGO UML:**
  argouml.tigris.org
- **PAGINA OFICIAL DE NETBEANS:**
  www.netbeans.org
- **WIKIPEDIA:**
  es.wikipedia.org
- **FREE SOFTWARE FOUNDATION y GNU Project:**
  http://www.fsf.org/ www.gnu.org

- es.wikipedia.org/wiki/RUP
- es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/
- www.ibm.com/software/awdtools/rup
- http://pisuerga.inf.ubu.es/lsi/Invest/Java/Tuto/I_4.htm
- http://sinetgy.org/~jgb/articulos/software-servicio/
- http://es.wikipedia.org/wiki/Fases_del_desarrollo_de_software
- http://es.wikipedia.org/wiki/Software_libre
- http://www.sinopticos.com/rs_485.html
- http://docs.sun.com/app/docs/doc/8175965/6mlcsc4ac?l=es&a=view
- http://www.revista.unam.mx/vol.1/num2/art4/
- http://www.technologies.cl/HXWNEW01.exe?1,1,30,NEW,65
- http://www.qualitrain.com.mx/otrosArt/java-net.htm
- *es.wikipedia.org/wiki/Radiofrecuencia*
- http://www.debian-administration.org/

ANEXO A: Casos de uso expandidos.

Caso de Uso 1	Validar Usuario Sistema de Control
Actores	*Administrador*, *Operador*
Tipo	Primario, Inclusión
Propósito	El sistema debe verificar la autenticidad del rol del actor antes de poder accesar o realizar alguna actividad en el mismo.
Descripción	Este caso de uso es iniciado por los actores *Administrador* y *Operador* introduciendo su login o usuario registrado y su password o clave de acceso en la pantalla principal de acceso al sistema de control. El sistema asigna los permisos necesarios al rol de usuario para actuar dentro del mismo.
Precondiciones	El usuario administrador existe por default en la base de datos.
Flujo Principal	Se presenta al actor la Pantalla de Acceso al Sistema (P-AS). El usuario ingresa su login y password y puede seleccionar entre las opciones “Aceptar” o “Cancelar”. Si la actividad seleccionada es “Aceptar”, se valida la información ingresada con los registros de los usuarios del sistema (S-1) (E-1) y se accesa a la Pantalla Principal del Sistema (P-P). Si la actividad seleccionada es “Cancelar”, se limpian los campos de texto y se recarga la Pantalla de Acceso al Sistema (P-AS).
Subflujos	**(S-1) AsignarPermisosSistema** Se asignan los permisos correspondientes en el sistema, dependiendo del rol de usuario registrado y se permite el acceso a la Pantalla Principal del Sistema (P-P).
Excepciones	**(E-1) Error de Acceso:** login/password incorrectos o usuario no registrado. Por favor, intente de nuevo. Se limpian los campos de texto y se recarga la Pantalla Principal (P-1). Después de 3 intentos de validación se sale del sistema. En caso de no existir el usuario operador, el administrador deberá ingresar al sistema y se entra al caso de uso 4 Administración de Usuarios, subflujo AltaUsuario.

Caso de Uso 2	Consultar bitácora de Accesos RFID
Actores	*Administrador*
Tipo	Primario, Inclusión
Propósito	El administrador necesita conocer quién ha accesado a través de los Lectores vía RFID.
Descripción	Este caso de uso lo inicia el actor *Administrador.* El administrador realiza una petición de consulta a la bitácora de Accesos RFID. El administrador puede indicar, si así lo desea, buscar accesos por fecha, por id de usuario, por id de Lector RFID o por Tag ID.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control, subflujo AsignarPermisosSistema.
Flujo Principal	El administrador elige la opción consultar bitácora de accesos RFID a través de la barra de herramientas en la pestaña "Administración" sub-opción "Accesos RFID" desde la Pantalla Principal (P-P). Al administrador se le presenta una pantalla emergente (P-BAT) para realizar la consulta de accesos RFID, a través de la cual podrá realizar una petición a la base de datos, buscando accesos por fecha, por id de usuario, por id de Lector RFID o por Tag ID, seleccionando la llave de búsqueda deseada a través de un JradioButton y activando la acción <<Buscar>> **(E1)**. Si el administrador elige realizar la búsqueda por fecha, podrá hacerlo a través de 3 JComboBox para seleccionar la fecha inicial y 3 más para la fecha final **(E3)**. Si decide buscar por o por id de Lector, lo podrá hacer seleccionando los datos correspondientes a través de 3 JSpinner, seleccionando Network ID, Reader ID y Node ID. Si decide buscar por id de usuario o por TagID, podrá hacerlo a través de un JTextField para que pueda ingresar los datos requeridos. En cualquiera de las opciones podrá activar la acción <<Buscar> **(E2)** y podrá visualizar los datos obtenidos en una nueva pantalla (P_BATR) **(S1)**, o activar la acción <<Cancelar> con lo cual la pantalla P-BAT desaparecerá.
Subflujos	**(S1) ResultadoBitacoraTag** Aparecerán los datos solicitados, ordenados por última fecha y hora a través de la pantalla emergente P-BATR, con opciones de impresión de datos si el administrador activa la acción <<Imprimir>> o con la opción de cerrar la pantalla emergente si activa la acción <<Salir>>.
Excepciones	**(E1) Error de selección:** debe elegir alguna opción a través del botón de selección. Aparecerá un jDialog con el mensaje. Al aceptarlo, se puede regresar a la P-BAT para una nueva selección. **(E2) Error en la búsqueda:** no se encontraron registros. Aparecerá un jDialog con el mensaje. Al aceptarlo, se puede regresar a la P-BAT para una nueva búsqueda. **(E3) Error en la fecha:** La fecha inicial es mayor que la fecha final. Aparecerá un jDialog con el mensaje. Al aceptarlo, se puede regresar a la P-BAT para corregir los datos seleccionados.

Caso de Uso 3	Consultar Bitácora de Accesos al Sistema de Control
Actores	Administrador
Tipo	Primario, Inclusión
Propósito	El administrador necesita conocer los usuarios que han accesado al sistema de control.
Descripción	Este caso de uso lo inicia el actor *Administrador.* El administrador realiza una petición de consulta a la bitácora de accesos. El administrador puede indicar, si así lo desea, buscar accesos por fecha, por usuario o por rol en el sistema.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control.
Flujo Principal	El administrador elige la opción consultar bitácora de accesos al sistema de control a través de la barra de herramientas en la pestaña "Administración" sub-opción "Accesos al Sistema" desde la Pantalla Principal (P-P). Al administrador se le presenta una pantalla emergente (P-BAS) para realizar la consulta de accesos al sistema, a través de la cual podrá realizar una petición a la base de datos, buscando accesos por fecha, por id de usuario o por rol, seleccionando la llave de búsqueda deseada a través de un JradioButton y activando la acción <<Buscar>> **(E1)**. Si el administrador elige realizar la búsqueda por fecha, podrá hacerlo a través de 3 JComboBox para seleccionar la fecha inicial y 3 más para la fecha final **(E3)**. Si decide buscar por id de usuario, podrá hacerlo a través de un JTextField para que pueda ingresar los datos requeridos. Si decide buscar por rol de usuario, lo podrá hacer seleccionando el rol a través de un JComoBox. En cualquiera de las opciones podrá activar la acción <<Buscar> **(E2)** y podrá visualizar los datos obtenidos en una nueva pantalla (P_BASR) **(S1)**, o activar la acción <<Cancelar> con lo cual la pantalla P-BAS desaparecerá.
Subflujos	**(S1) ResultadoBitacoraSys** Aparecerán los datos solicitados, ordenados por última fecha y hora a través de la pantalla emergente P-BASR, con opciones de impresión de datos si el administrador activa la acción <<Imprimir>> o con la opción de cerrar la pantalla emergente si activa la acción <<Salir>>.
Excepciones	**(E1) Error de selección:** debe elegir alguna opción a través del botón de selección. Aparecerá un jDialog con el mensaje. Al aceptarlo, se puede regresar a la P-BAS para una nueva selección. **(E2) Error en la búsqueda:** no se encontraron registros. Aparecerá un jDialog con el mensaje. Al aceptarlo, se puede regresar a la P-BAS para una nueva búsqueda. **(E3) Error en la fecha:** La fecha inicial es mayor que la fecha final. Aparecerá un jDialog con el mensaje. Al aceptarlo, se puede regresar a la P-BAS para corregir los datos seleccionados.

Caso de Uso 4	Administración de Usuarios
Actores	*Administrador*
Tipo	Primario, Básico
Propósito	El administrador necesita llevar un control de los usuarios que utilizan el sistema.

Descripción	Este caso de uso es iniciado por el actor *Administrador*. El administrador puede dar de alta, eliminar o modificar algún usuario del sistema, asignando su login, contraseña, datos y perfil. También puede habilitar o inhabilitar el uso del sistema a estos usuarios. Cabe señalar que el usuario administrador se encuentra registrado por default en la base de datos del sistema. El administrador puede también dar de alta, modificar o eliminar algún usuario con Tag RFID asignado, registrando su Tag ID, datos y perfil, así como habilitar o inhabilitar el acceso vía RFID a los usuarios con Tag ID registrados. Es importante señalar que la administración de un *Usuario Tag RFID* también incluye a dispositivos electrónicos con Tag RFID asignado, lo cual es opcional.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control, subflujo AsignarPermisosSistema.
Flujo Principal	Al administrador Se le presenta la Pantalla para la Administración de Usuarios (P-AU) a través de la barra de herramientas en la pestaña "Administración" sub-opción "Usuarios" desde la Pantalla Principal (P-P), con lo que se despliega la pantalla para administración antes nombrada, donde puede elegir entre las opciones Registrar **(S1)**, Modificar **(S2)** o Eliminar **(S3)** algún usuario. También puede habilitar o inhabilitar el uso del sistema a estos usuarios **(S4)**.
Subflujos	**(S1) AltaUsuario** Al administrador se le presentan dos pestañas de selección de usuario, dependiendo de si se desea administrar a un usuario de sistema o usuario de Tag RFID. El administrador podrá ingresar los datos necesarios desde la P-AU, tales como número de cuenta, nombre, dependencia, login y password e incluso número de Tag ID, entre otros, dependiendo del tipo de usuario a registrar. Al finalizar el llenado de información, el administrador activará la acción <<Registrar>> para que el nuevo registro se almacene en la Base de Datos del sistema **(E1)**. Al finalizar, aparecerá un mensaje emergente con la leyenda: "El usuario se ha dado de alta con éxito" **(E2)**. El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Si el administrador activa la acción <<Limpiar>>, los datos accesados no son registrados, desaparecen y se procede ingresar nuevos datos. Las acciones <<Modificar>>, <<Cancelar> y <<Eliminar>> se encuentran deshabilitadas para el subflujo AltaUsuario. **(S2) ModificarUsuario** Al administrador se le presentan dos pestañas de selección de usuario, dependiendo de si se desea administrar a un usuario de sistema o usuario de Tag RFID. El administrador podrá modificar los datos de un usuario desde la P-AU, tales como número de cuenta, nombre, dependencia, login y password e incluso número de Tag ID, entre otros, dependiendo del tipo de usuario a modificar, seleccionándolo a través de un Jlist. Los datos del usuario a modificar se buscan en el sistema y son mostrados en la P-AU, donde el administrador podrá modificar los datos que desee. Al finalizar la modificación de información, el administrador activará la acción <<Modificar>> **(E1)** para que los cambios se almacenen en la Base de Datos del sistema. Al finalizar, aparecerá un mensaje emergente con la leyenda: "El usuario se ha modificado con éxito" **(E2)**. El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU,

	donde los campos son limpiados para poder realizar alguna nueva acción. Si el administrador activa la acción <<Cancelar>> los cambios no se realizarán. Aparecerá un mensaje con la leyenda: “El usuario NO ha sido modificado”. El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Las acciones <<Limpiar>> y <<Registrar>> se encuentran deshabilitadas para el subflujo ModificarUsuario. **(S3) EliminarUsuario** Al administrador se le presentan dos pestañas de selección de usuario, dependiendo de si se desea administrar a un usuario de sistema o usuario de Tag RFID. El administrador podrá eliminar a un usuario desde la P-AU, seleccionándolo a través de un Jlist. Los datos del usuario a eliminar se buscan en el sistema y son mostrados en la P-AU, El administrador revisa que sea el usuario al que se desea eliminar y activará la acción <<Eliminar>>, para que el usuario sea eliminado del sistema. Previo a esto, aparecerá una ventana emergente para confirmar la eliminación, con el mensaje: “¿Desea eliminar al usuario ####?”. Si el administrador activa la acción <<Aceptar>> se procede con la eliminación de usuario, por lo que aparecerá un mensaje con la leyenda: “El usuario se ha eliminado con éxito”. El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Si el administrador activa la acción <<Cancelar>> de la ventana emergente, la eliminación de usuario no se realizará. Aparecerá un mensaje con la leyenda: “El usuario NO ha sido eliminado”. El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Las acciones <<Limpiar>> y <<Registrar>> se encuentran deshabilitadas para el subflujo EliminarUsuario. **(S4) Habilitar/InhabilitarUsuario** Al administrador se le presentan dos pestañas de selección de usuario, dependiendo de si se desea administrar a un usuario de sistema o usuario de Tag RFID. El administrador podrá habilitar/inhabilitar a un usuario desde la P-AU, seleccionándolo a través de un Jlist. Los datos del usuario a habilitar/inhabilitar se buscan en el sistema y son mostrados en la P-AU. El administrador revisa que sea el usuario al que se desea habilitar/inhabilitar y en la sección “Datos de acceso” en la P-AU se encuentra un JcomboBox con las opciones SI/NO y con la pregunta ¿Desea habilitar al usuario? con lo que se elige “SI” para habilitarlo y “NO” para inhabilitarlo. Al finalizar la habilitación/inhabilitación, el administrador activará la acción <<Modificar>> **(E1)** para que el cambio se almacene en la Base de Datos del sistema. Al finalizar, aparecerá un mensaje emergente con la leyenda: “El usuario se ha modificado con éxito” **(E2)**. El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Si el administrador activa la acción <<Cancelar>> los cambios no se realizarán. Aparecerá un mensaje con la leyenda: “El usuario NO ha sido modificado”. El mensaje desaparece al activar la acción <<Aceptar>> y se regresa a la P-AU, donde los campos son limpiados para poder realizar alguna nueva acción. Las acciones <<Limpiar>> y

	<<Registrar>> se encuentran deshabilitadas para el subflujo Habilitar/InhabilitarUsuario.
Excepciones	**(E1) Error en el registro:** falta llenar los campos obligatorios. Los campos señalados con un asterisco * son requeridos. El error se describe en una ventana emergente con la opción "Aceptar". Al aceptar la opción la ventana emergente desaparece. Se regresa a la pantalla anterior para que el administrador llene los datos requeridos. **(E2) Error en el registro:** el usuario con n° de cuenta #### o número de tag #### (dependiendo el tipo de usuario) ya existe. El error se describe en una ventana emergente con la opción "Aceptar". Al aceptar la opción la ventana emergente desaparece. Se regresa a la pantalla anterior para que el administrador corrija los datos ingresados.

Caso de Uso 5	Consultar Bitácora de Errores Sistema de Control
Actores	Administrador
Tipo	Primario, Inclusión
Propósito	El administrador necesita conocer qué eventos han surgido en el sistema de control.
Descripción	Este caso de uso lo inicia el actor *Administrador, el cual* realiza una petición de consulta a la bitácora de errores. El administrador puede indicar, si así lo desea, buscar errores por tipo de error, por fecha o por descripción.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control, subflujo AsignarPermisosSistema.
Flujo Principal	El administrador elige la opción consultar bitácora de errores en el sistema de control a través de la barra de herramientas en la pestaña "Mantenimiento" sub-opción "Errores en el Sistema" desde la Pantalla Principal (P-P). Al administrador se le presenta la Pantalla para consultar la Bitácora de Errores al Sistema (P-BES), a través de la cual podrá realizar una petición a la base de datos, buscando errores por fecha, por palabra clave o por ID de error, seleccionando la llave de búsqueda deseada a través de un JradioButton y activando la acción <<Buscar>> **(E1)**. Si el administrador elige realizar la búsqueda por fecha, podrá hacerlo a través de 3 JComboBox para seleccionar la fecha inicial y 3 más para la fecha final **(E3)**. Si decide buscar por palabra clave, podrá hacerlo a través de un JTextField para que pueda ingresar los datos requeridos. Si decide buscar por ID de error, lo podrá hacer seleccionando el rol a través de un JComoBox. En cualquiera de las opciones podrá activar la acción <<Buscar> **(E2)** y podrá visualizar los datos obtenidos en una nueva pantalla (P_BESR) **(S1)**, o activar la acción <<Cancelar> con lo cual la pantalla P-BES desaparecerá.
Subflujos	**(S1) ResultadoBitacoraErr** Aparecerán los datos solicitados, ordenados por última fecha y hora a través de la pantalla emergente P-BESR, con opciones de impresión de datos si el administrador activa la acción <<Imprimir>> o con la opción de cerrar la pantalla emergente si activa la acción <<Salir>>.

Excepciones	**(E1) Error de selección:** debe elegir alguna opción a través del botón de selección. Aparecerá un jDialog con el mensaje. Al aceptarlo, se puede regresar a la P-BES para una nueva selección. **(E2) Error en la búsqueda:** no se encontraron registros. Aparecerá un jDialog con el mensaje. Al aceptarlo, se puede regresar a la P-BES para una nueva búsqueda. **(E3) Error en la fecha:** La fecha inicial es mayor que la fecha final. Aparecerá un jDialog con el mensaje. Al aceptarlo, se puede regresar a la P-BES para corregir los datos seleccionados.

Caso de Uso 6	Mantenimiento del Sistema de Control
Actores	Administrador
Tipo	Primario, Básico
Propósito	El administrador necesita dar mantenimiento al sistema de control.
Descripción	Este caso de uso lo inicia el actor *Administrador.* El administrador revisa los registros (Logs) del sistema o bitácora de eventos para visualizar algún error. En caso de algún error registrado, el administrador revisa la referencia del mismo y analiza el código fuente del sistema de control , registros en la BD o comunicación entre el sistema y la red de Lectores, según corresponda.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control y posteriormente haber ejecutado el caso de uso Consultar Bitácora de Errores Sistema de Control.
Flujo Principal	El administrador consulta la bitácora de errores en el sistema de control (Caso de Uso 5) y dependiendo del error el administrador podrá darle seguimiento, eligiendo la opción Mantenimiento del Sistema a través de la barra de herramientas en la pestaña “Mantenimiento” sub-opción “Mantenimiento del Sistema” desde la Pantalla Principal (P-P). Al administrador se le presenta una pantalla emergente (P-MSC) donde encontrará las opciones de mantenimiento: Analizar Excepciones Java del Sistema **(S1)** y Depurar Código del Sistema **(S2)**, a través de un JTabbedPane.
Subflujos	**(S1) AnalizarExcepcionesJavaSistema** Al administrador se le presenta la lectura del log de excepciones Java. Si el administrador activa la acción <<Refrescar>>, se presentarán las últimas líneas del log de excepciones de Java dentro de un JTextArea **(E1)**. Si el administrador activa la acción <<Guardar>>, se le presentará un JfileChooser para guardar la salida del log como un archivo de texto para su próximo análisis. Si el administrador activa la acción <<Limpiar>>, la JTextArea se vaciará, quedando en blanco. Todas las acciones se realizan a través de 3 JButton's , ubicados en la parte inferior izquierda de la P-MSC. **(S2) DepurarCódigoSistema** Al administrador se le presenta un JTextArea. donde podrá seleccionar, a través de un JFileChooser, el archivo Java a editar, activando la acción <<Abrir>>. El archivo aparecerá en la JTextArea para ser editado. Si los

	cambios realizados son correctos, el administrador podrá activar la acción <<Guardar>> para salvar los cambios al archivo Java. Aparecerá un JDialog con el mensaje: “Se guardó el archivo Java NombreArchivo.java”. Si se desea compilar el código fuente guardado para generar el archvo clase, se puede realizar activando la acción <<Compilar>>. Aparecera el mensaje: “Se ha compilado el archivo JavaNombreArchivo.java con éxito”. Si desea NO guardar los cambios, podrá activar la acción <<Cancelar>> para cerrar el archivo sin modificar. Se limpiará la JTextArea y aparecerá un JDialog con el mensaje: “No se realizaron cambios al archivo Java NombreArchivo.java”. Todas las acciones se realizan a través de 4 JButton's, ubicados en la parte inferior izquierda de la P-MSC.
Excepciones	**(E1) Error en la lectura de log:** archivo no encontrado.

Caso de Uso 7	Mantenimiento Catálogos
Actores	Administrador
Tipo	Primario, Básico
Propósito	El administrador necesita dar mantenimiento a los catálogos del sistema.
Descripción	Este caso de uso lo inicia el actor *Administrador, el cual* revisa el tamaño de las tablas y de uso de espacio en disco referente a los catálogos del sistema. El administrador respalda información de las bitácoras y catálogos del sistema y depura lo innecesario.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control.
Flujo Principal	El administrador elige la opción Mantenimiento de Catálogos en el sistema de control a través de la barra de herramientas en la pestaña “Mantenimiento” sub-opción “Mantenimiento Catálogos” desde la Pantalla Principal (P-P). Al administrador se le presenta una pantalla para Mantenimiento de Catálogos (P-MC) **(S1)**, Depurar Registros del Sistema **(S2)**, revisar el log de eventos de la base de datos **(S3)** e incluso lanzar la herramienta administrativa **(S4)**.
Subflujos	**(S1) RespaldarRegistrosSistema** Desde la P-MC el administrador accede a la pestaña “Tablas”. Podrá elegir la tabla a respaldar, a través de un JList ubicada en la parte superior izquierda de la pantalla. Las propiedades de la tabla elegida, tales como nombre, tamaño y sistema de archivos en donde se encuentra se muestran en la parte superior derecha de la pantalla. El administrador podrá entonces respaldar la tabla seleccionada, activando la acción <<Respaldar>>, a través de un JButton. Aparecerá un jDialog con el mensaje: “La tabla se ha respaldado con éxito”**(E1)**. **(S2) DepurarRegistrosSistema** Desde la P-MC el administrador accede a la pestaña “Tablas”. Podrá elegir la tabla a depurar, a través de un JList ubicada en la parte superior izquierda de la pantalla. Las propiedades de la tabla elegida, tales como nombre, tamaño y sistema de archivos en donde se encuentra se muestran en la parte superior derecha de la pantalla. El administrador podrá entonces depurar la tabla seleccionada en la sección Depuración

	de Tabla ubicada en la parte central inferior de la pantalla, estableciendo primero la fecha y hora inicial y final a depurar y activando la acción <<Depurar>>, a través de un JButton. Aparecerá un jDialog con el mensaje "La tabla se ha depurado con éxito"**(E2)**. **(S3) RevisarLogBaseDatos** Desde la P-MC el administrador accede a la pestaña "Log". Se presenta una JTextArea con la salida de los últimos registros en el log de la base de datos. E administrador podrá revisar los eventos que han sucedido en la base de datos y podrá refrescar la pantalla, guardar la salida del log e incluso limpiar la pantalla si asi lo desea, al activar la acción <<Refrescar>>, <<Guardar>> o <<Limpiar>>, respectivamente, a través de 3 JButtons ubicados en la parte inferior izquierda de la pantalla. **(S4) LanzarHerramientaAdministrativa** Desde la P-MC el administrador accede a la pestaña "Propiedades". Se presenta información general de la versión del manejador de base de datos, versión del sistema operativo, nombre del esquema y ubicación del registro de eventos en la base de datos. También se presenta una nota importante, donde se indica: "Para mayor información y administración completa de la base de datos utilice MySQL Administration Tool". El administrador podrá entonces lanzar la aplicación administrativa al activar la acción <<Lanzar>>, a través de un JButton.
Excepciones	**(E1) Error al respaldar:** la tabla NO ha sido respaldada de forma correcta. Aparecerá un jDialog con el mensaje. Al aceptarlo, se puede regresar a la P-MC. **(E2) Error al depurar:** la fecha y hora final es menor a la fecha y hora inicial. Revise los datos y vuelva a intentarlo. Aparecerá un jDialog con el mensaje. Al aceptarlo, se puede regresar a la P-MC.

Caso de Uso 8	Configurar Puerto Control
Actores	Operador
Tipo	Primario, Inclusión
Propósito	El operador asigna la velocidad de transmisión y el puerto serie a utilizar, con el cual se enviará y se recibirá información hacia y desde la red de Lectores, respectivamente.
Descripción	Este caso de uso lo inicia el actor *Operador*. El operador asigna en la PC la velocidad de transmisión especificada en el manual de Lectores Wavetrend, la cual puede variar entre 5 distintos valores, pero que por default está configurada como 57600 baudios. El operador elige el puerto serie de la PC a utilizar para la conexión de la red de Lectores RFID. El operador se encuentra listo para controlar y configurar a los Lectores RFID.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control.

Flujo Principal	El operador elige la pestaña COM Port Settings desde la Pantalla Principal del Sistema (P-P) donde puede elegir el puerto serie a utilizar (COM1-COM4) a través de un *JComboBox .* El operador también elige la velocidad de transmisión, la cual varía entre 5 distintos valores en baudios: 9600, 19200, 38400, 57600 *y 115000,* esto a través de un *jComboBox.* Una vez que el operador eligió el puerto serie (Comm Port) y la velocidad de transmisión (Baud Rate) deseados, puede intentar abrir o cerrar el puerto serie para iniciar la transmisión y recepción de datos hacia y desde la red de Lectores RFID, respectivamente **(E1) (E2)**. Para esto se cuenta con dos *jToggleButton,* uno para abrir el puerto serie (OPEN PORT) y otro para cerrarlo (CLOSE PORT).
Subflujos	Ninguno
Excepciones	**(E1) Error al Abrir el Puerto de Control:** el puerto COMx elegido está siendo usado por otra aplicación. Intente desocuparlo o elija otro puerto. Este error se registra en la Bitácora de errores del Sistema (Caso de Uso 5) y su respectiva excepción en Java. **(E2) Error De Velocidad de Transmisión:** la velocidad de transmisión elegida no coincide con los datos recibidos desde la red de Lectores. Intente de nuevo. Este error se registra en la Bitácora de errores del Sistema (Caso de Uso 5) y su respectiva excepción en Java.

Caso de Uso 9	Configurar Lectores
Actores	Operador
Tipo	Primario, Básico
Propósito	El operador necesita configurar a los Lectores RFID en red, dependiendo de las necesidades requeridas.
Descripción	Este caso de uso lo inicia el actor *Operador*. El operador, de ser necesario, sincroniza la velocidad de transferencia entre el puerto serie de la PC y los Lectores RFID a través del comando de configuración correspondiente. La velocidad de transferencia por default en los Lectores RFID es de 57600 baudios. El operador puede entonces configurar a los Lectores RFID según los requerimientos del proyecto. Se puede configurar la auto-recepción de datos de Tags, los identificadores de red y de Lector, el código de sitio, habilitar o deshabilitar la auto-detección en los Lectores, entre otras opciones.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control y posteriormente el caso de uso Configurar Puerto Control.
Flujo Principal	El operador, desde la Pantalla Principal del Sistema (P-P), elige el tipo de configuración a realizar (comandos que inician con “Set”, “Enable” y “Disable”) a través de *jRadioButtons,* dentro de la pestaña “Commands” **(S1) (S2) (S3) (S4) (S5) (S6) (S7) (S8) (S9)**. Al terminar de elegir las configuraciones a realizar, en la terminal de salida (Murciélago Output Terminal) aparecerá el mensaje “Hay comandos de configuración pendientes para <<comando(s) de configuración elegido(s)>>, pulse SEND en la sección “Send Data” para que los comandos de configuración tengan efecto”. El operador deberá elegir entre las opciones <<SEND>> o <<CLEAR>> a través de dos *jButtons*. ubicados en la sección “Send

	Data”. Posterior a esto, el operador elige la acción <<SEND>> en la sección “Send Data” (en el caso de haber aceptado configuraciones a realizar) para enviar los cambios al Lector o Lectores correspondientes **(E1)** . Al final, la terminal de salida nos indicará los cambios realizados con el siguiente mensaje: “Cambios realizados con éxito.” **(E2)**. Si el operador no desea enviar los cambios, podrá activar la acción <<CLEAR>> en la sección “Send Data”. Aparecerá el mensaje “El comando de configuración <<comando(s) de configuración elegido(s) y cancelado(s)>> ha sido cancelado. No existen opciones de configuración pendientes.”. La terminal de salida se encuentra en el ángulo inferior derecho de la Pantalla.
Subflujos	**(S1) StartEnableAutoPolling** El operador seleccionará esta opción (a través de un jRadioButton) si desea iniciar/habilitar el autopoleo de los Lectores en búsqueda de tags. Esta opción se almacenará en la memoria EEPROM de los Lectores RFID. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S2) DisableAutoPolling** El operador seleccionará esta opción (a través de un jRadioButton) si desea deshabilitar el autopoleo de los Lectores en búsqueda de tags. Esta opción se almacenará en la memoria EEPROM de los Lectores RFID. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S1) SetNetworkID** El operador seleccionará esta opción (a través de un jRadioButton) si desea cambiar el Network ID de algún Lector o red de Lectores. Esta opción se almacenará en la memoria EEPROM de los Lectores RFID. Al elegir el jRadioButton aparecerá una JOptionPane para que el operador ingrese el nuevo valor de Network ID a enviar . Si activa la acción <<Aceptar>> se almacenará en alguna variable el nuevo valor a enviar **(E3)**. Si activa la acción <<Cancelar>> se cancelará el envío pendiente para esta petición. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S2) SetReaderID** El operador seleccionará esta opción (a través de un jRadioButton) si desea cambiar el Reader ID de algún Lector o Lectores. Esta opción se almacenará en la memoria EEPROM de los Lectores RFID. Al elegir el jRadioButton aparecerá una JOptionPane para que el operador ingrese el nuevo valor de Reader ID a enviar. Si activa la acción <<Aceptar>> se almacenará en alguna variable el nuevo valor a enviar **(E3)**. Si activa la acción <<Cancelar>> se cancelará el envío pendiente para esta petición. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S3) SetRSSIValue** El operador seleccionará esta opción (a través de un jRadioButton) si desea cambiar el valor de RSSI (Received Signal Strength Indicator) de algún Lector o Lectores. Esta opción se almacena en la memoria EEPROM de los Lectores RFID. Al elegir el jRadioButton aparecerá una JOptionPane para que el operador ingrese el nuevo valor de RSSI a enviar **(E3)**. Si

	activa la acción <<Aceptar>> se almacenará en alguna variable el nuevo valor a enviar. Si activa la acción <<Cancelar>> se cancelará el envío pendiente para esta petición. **(S4) SetSideCode** El operador seleccionará esta opción (a través de un jRadioButton) si desea cambiar el valor del Side Code (Código de Sitio) de algún Lector o Lectores. Esta opción se almacenará en la memoria EEPROM de los Lectores RFID. Al elegir el jRadioButton aparecerá una JOptionPane para que el operador ingrese el nuevo valor del nuevo Side Code a enviar. Si activa la acción <<Aceptar>> se almacenará en alguna variable el nuevo valor a enviar. Si activa la acción <<Cancelar>> se cancelará el envío pendiente para esta petición. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S5) SetReciverGain** El operador seleccionará esta opción (a través de un jRadioButton) si desea cambiar el valor de Reciver Gain (Ganancia de Recepción) de algún Lector o Lectores. Esta opción se almacenará en la memoria EEPROM de los Lectores RFID. Al elegir el jRadioButton aparecerá una JOptionPane para que el operador elija el nuevo valor de la nueva Reciver Gain a enviar. Si activa la acción <<Aceptar>> se almacenará en alguna variable el nuevo valor a enviar. Si activa la acción <<Cancelar>> se cancelará el envío pendiente para esta petición. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S6) SetAlarmFIlter** El operador seleccionará esta opción (a través de un jRadioButton) si desea cambiar el valor de Alarm FIlter (Filtrado de Alarma) de algún Lector o Lectores. Esta opción se almacenará en la memoria EEPROM de los Lectores RFID. Al elegir el jRadioButton aparecerá una JOptionPane para que el operador elija el nuevo valor de la nueva AlarmFIlter a enviar. Si activa la acción <<Aceptar>> se almacenará en alguna variable el nuevo valor a enviar. Si activa la acción <<Cancelar>> se cancelará el envío pendiente para esta petición. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S7) SetBaudRate** El operador seleccionará esta opción (a través de un jRadioButton) si desea cambiar el valor de Baud Rate (Velocidad de Transmisión en baudios) de algún Lector o Lectores. Esta opción se almacenará en la memoria EEPROM de los Lectores RFID. Al elegir el jRadioButton aparecerá una JOptionPane para que el operador elija el nuevo valor de el nuevo Baud Rate a enviar. Si activa la acción <<Aceptar>> se almacenará en alguna variable el nuevo valor a enviar. Si activa la acción <<Cancelar>> se cancelará el envío pendiente para esta petición. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal.
Excepciones	**(E1) Error al enviar datos:** No se han elegido opciones de control o configuración a realizar. **(E2) Error de configuración:** No surtieron efecto los siguientes

	cambios: <<configuración o configuraciones elegida(s)>>. Error en las opciones de configuración. Revise el manual de control y configuración de Lectores RFID Wavetrend para más detalles. **(E3) Error en el valor:** debe elegir un valor entre 0 y 255.

Caso de Uso 10	Controlar Lectores
Actores	Operador
Tipo	Primario, Básico
Propósito	El operador debe controlar la manera de actuar de los Lectores RFID y el flujo de datos hacia y desde los mismos.
Descripción	Este caso de uso lo inicia el actor *Operador*, el cual puede obtener información contenida en los Lectores o en los Tags detectados y puede analizar si cierto Lector RFID está disponible por medio de un barrido de datos o ping, además de tener la posibilidad de reiniciar la red completa de Lectores si así lo requiere, entre otras funciones de control.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control y posteriormente el caso de uso Configurar Puerto Control.
Flujo Principal	El operador, desde la Pantalla Principal del Sistema (P-P), elige el tipo de configuración a realizar (comandos que inician con "Get", entre otros) a través de *jRadioButtons* dentro de la pestaña "Commands" **(S1) (S2) (S3) (S4) (S5) (S6) (S7) (S8) (S9) (S10) (S11)**. Al terminar de seleccionar las opciones de control a realizar, en la terminal de salida (Murciélago Output Terminal) aparecerá el mensaje "Hay comandos de control pendientes para <<comando(s) de control elegido(s)>>, pulse SEND en la sección "Send Data" de la Pantalla Principal del Sistema para que los comandos de control o de consulta tengan efecto". El operador deberá elegir entre las opciones <<SEND>> o <<CLEAR>> a través de dos *jButton en la sección "Send Data"*. Posterior a esto, el operador elige la acción <<SEND>> en la sección "Send Data" (en el caso de haber aceptado comandos de control o de consulta a realizar) para enviar los comandos al Lector o Lectores correspondientes **(E1)** . Al final, la terminal de salida nos indicará las consultas o control realizadas con el siguiente mensaje: "Opciones de control o de consulta realizadas con éxito ." **(E2)**. Si el operador no desea enviar los comandos de control o de consulta, podrá activar la acción <<CLEAR>> en la sección "Send Data". Aparecerá el mensaje "El comando de control o consulta <<comando(s) de control o consulta elegido(s) y cancelado(s)>> ha sido cancelado. No existen opciones de control o consulta pendientes.". La terminal de salida se encuentra en el ángulo inferior derecho de la Pantalla.
Subflujos	**(S1) ResetReadersNetwork** El operador seleccionará esta opción (a través de un jRadioButton) si desea reiniciar toda la red de Lectores. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S2) PingReader** El operador seleccionará esta opción (a través de un jRadioButton) si desea revisar la correcta comunicación de algún Lector o Lectores. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal.

	Se regresa al flujo principal. **(S3) GetTagPacket** El operador seleccionará esta opción (a través de un jRadioButton) si desea obtener información de algún Lector o Lectores que hayan recibido a su vez información de algún tag. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S4) GetRSSIvalue** El operador seleccionará esta opción (a través de un jRadioButton) si desea obtener el valor de RSSI (Received Signal Strength Indicator) de algún Lector o Lectores. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S5) GetSideCode** El operador seleccionará esta opción (a través de un jRadioButton) si desea obtener el valor del Side Code (Código de Sitio) de algún Lector o Lectores. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S6) GetReciverGain** El operador seleccionará esta opción (a través de un jRadioButton) si desea obtener el valor de Reciver Gain (Ganancia de Recepción) de algún Lector o Lectores. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S7) GetAlarmFilter** El operador seleccionará esta opción (a través de un jRadioButton) si desea obtener el valor de Alarm FIlter (Filtrado de Alarma) de algún Lector o Lectores. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S8) GetInvalidTagCount** El operador seleccionará esta opción (a través de un jRadioButton) si desea obtener el valor de tags leídos erróneamente. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S9) GetSupplyVoltage** El operador seleccionará esta opción (a través de un jRadioButton) si desea obtener el valor de voltaje de alimentación suministrado a algún Lector o Lectores. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S10) StartRFwhiteNoiseCalculation** El operador seleccionará esta opción (a través de un jRadioButton) si desea calcular el ruido blanco en la señal RF de algún Lector o Lectores. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal. **(S11) GetRFwhiteNoiseCalculation** El operador seleccionará esta opción (a través de un jRadioButton) si desea obtener el resultado del cálculo de ruido blanco en la señal RF de algún Lector o Lectores almacenado en memoria. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo

	principal. **(S12) GetVersionInformation** El operador seleccionará esta opción (a través de un jRadioButton) si desea obtener la versión del firmware de algún Lector o Lectores. Aparecerá el mensaje correspondiente en la Murcielago Output Terminal. Se regresa al flujo principal.
Excepciones	**(E1) Error al enviar datos:** No se han elegido opciones de control o configuración a realizar. **(E2) Error de control:** No surtieron efecto los siguientes comandos de control: <<comando(s) de control elegido(s)>>. Error en las opciones de control. Revise el manual de control y configuración de Lectores RFID Wavetrend para más detalles.

Caso de Uso 11	Detener auto-detección
Actores	Operador
Tipo	Secundario, Extensión
Propósito	El operador puede detener la auto-detección de Tags en la red de Lectores RFID en caso de ser necesario.
Descripción	Este caso de uso lo inicia el actor *Operador*, el cual envía hacia la red de Lectores RFID cierta cantidad de caracteres * (asterisco) para detener la auto-detección de Tags. El operador verifica que se haya detenido la auto-detección de Tags en el log (registro) del sistema correspondiente.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control y posteriormente el caso de uso Configurar Puerto Control.
Flujo Principal	El operador elige la opción “Break Auto Polling” a través de un *jRadioButton* en la sección “Commands” desde la Pantalla Principal del Sistema (P-P) y en la terminal de salida (Murciélago Output Terminal) aparecerá el mensaje “Se ha seleccionado la opción “Break Auto Polling”. Pulse SEND en la sección “Send Data” de la Pantalla Principal del Sistema para que se interrumpa el auto-poleo en la red de Lectores”. La terminal de salida se encuentra en el ángulo inferior derecho de la sección “Commands”. Posterior a esto, el operador elige la acción SEND en la sección “Send Data” de la Pantalla Principal del Sistema para enviar la secuencia de asteriscos y espacios necesarios para detener el auto-poleo o auto-detección de Tags **(E1)** a través de un *jButton*. Al final, la terminal de salida nos indicará el siguiente mensaje: “Se ha detenido la auto-detección de Lectores con éxito.” **(E2)**. Si desea cancelar el envío, el operador debe activar la acción “CLEAR” a través de un JButton en la sección “Send Data“.
Subflujos	Ninguno
Excepciones	**(E1) Error al enviar datos:** No se han elegido opciones de control o configuración a realizar. **(E2) Error de control:** No se ha detenido la auto-detección. Intente enviar de nuevo el comando correspondiente.

Caso de Uso 12	Consultar Total de Lectores en Red
Actores	Operador
Tipo	Secundario, Básico
Propósito	El operador puede consultar el total de Lectores RFID en red para llevar un conteo de los mismos o para saber si alguna parte de la red se deshabilitó, según se requiera.
Descripción	Este caso de uso lo inicia el actor *Operador*, el cual envía hacia la red de Lectores RFID la petición correspondiente y obtiene el valor total de Lectores en la red. El operador verifica que el total obtenido sea igual al total de Lectores registrados en el sistema, esto para comprobar la integridad de la red.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control y posteriormente el caso de uso Configurar Puerto Control.
Flujo Principal	El operador elige la acción REFRESH en la sección "Readers On Loop" para enviar el comando correspondiente, esto desde la Pantalla Principal del Sistema (P-P),seccion "Commands" y en la terminal de salida (Murciélago Output Terminal) aparecerá el mensaje "Escaneando Lectores en Red... Lectores en Red: #de Lectores " **(E1)** activada la acción través de un *jButton.*
Subflujos	Ninguno
Excepciones	**(E1) Error al obtener el total de lectores:** por favor, verifique la integridad de la red.

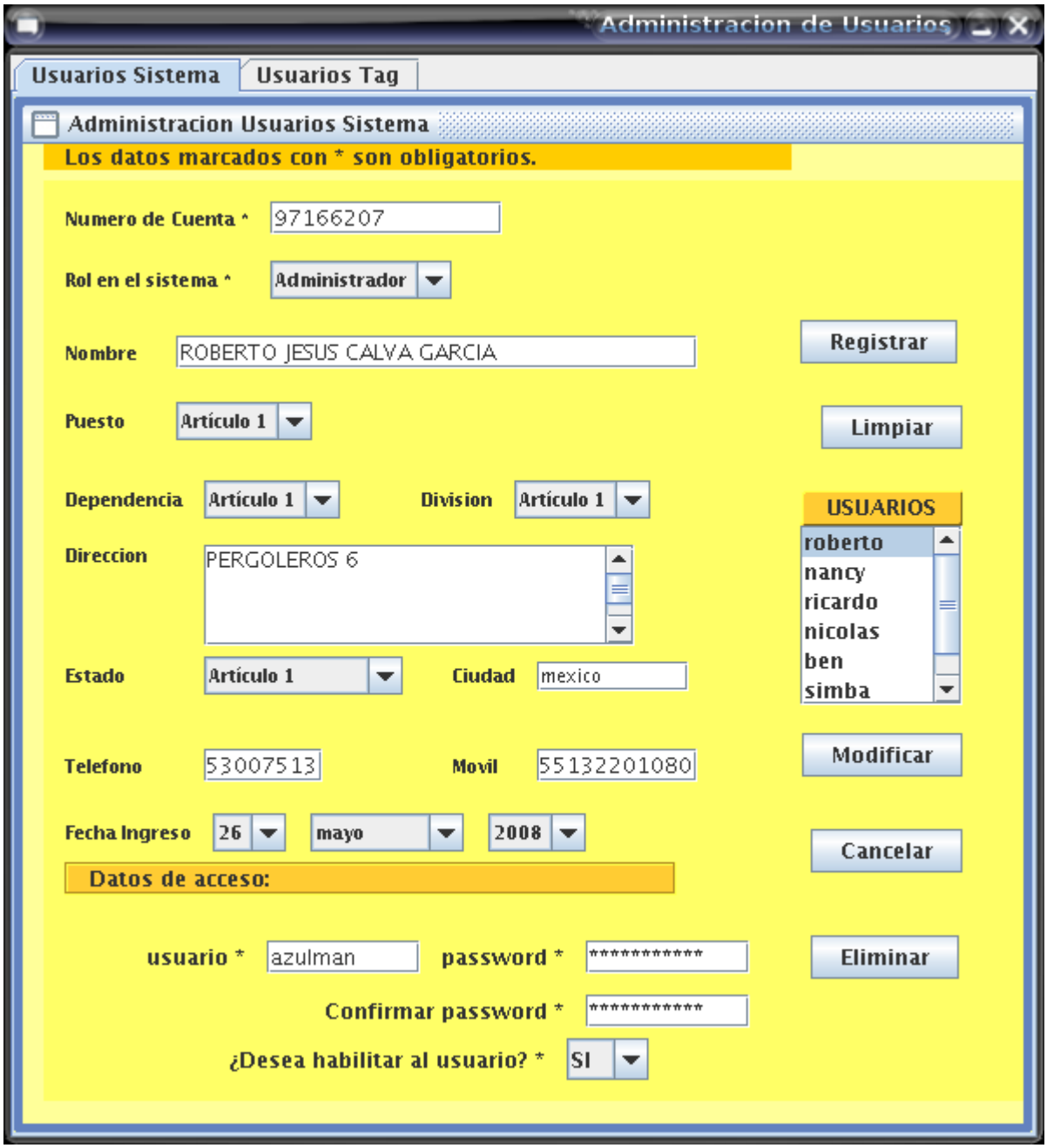
Caso de Uso 13	Validar Información Recibida
Actores	Usuario Tag RFID
Tipo	Primario, Básico
Propósito	El usuario con un Tag registrado puede enviar los datos del mismo hacia el sistema de control a través de la red de Lectores RFID para ser validados, debido a una petición de acceso.
Descripción	Este caso de uso lo inicia el actor *Usuario Tag RFID*. El usuario con un Tag registrado puede hacer una petición de acceso a puertas desde una cierta distancia al Lector RFID, dependiendo de la ganancia configurada en el mismo, la cual puede tomar 2 valores según el manual: alta o baja ganancia (Caso de Uso 9). La información enviada por el Tag se valida dentro del sistema para saber si éste se encuentra registrado y así permitir o denegar la acción.

Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control y posteriormente el caso de uso Configurar Puerto Control.
Flujo Principal	EL usuario con un Tag registrado realiza una petición de acceso por alguna puerta con Lector RFID asignado al acercar la tarjeta electrónica o Tag. La información del Tag es recibida por el Lector RFID y enviada al sistema de control para ser evaluada. Si el Tag se encuentra registrado en el sistema y habilitado para su acceso, se enviará una señal a un sistema externo y ajeno del sistema de control para abrir la puerta correspondiente **(S1)(E1)**.
Subflujos	(S1)
Excepciones	**(E1) Error de acceso:** el usuario con Tag ID ###### no se encuentra registrado en el sistema.

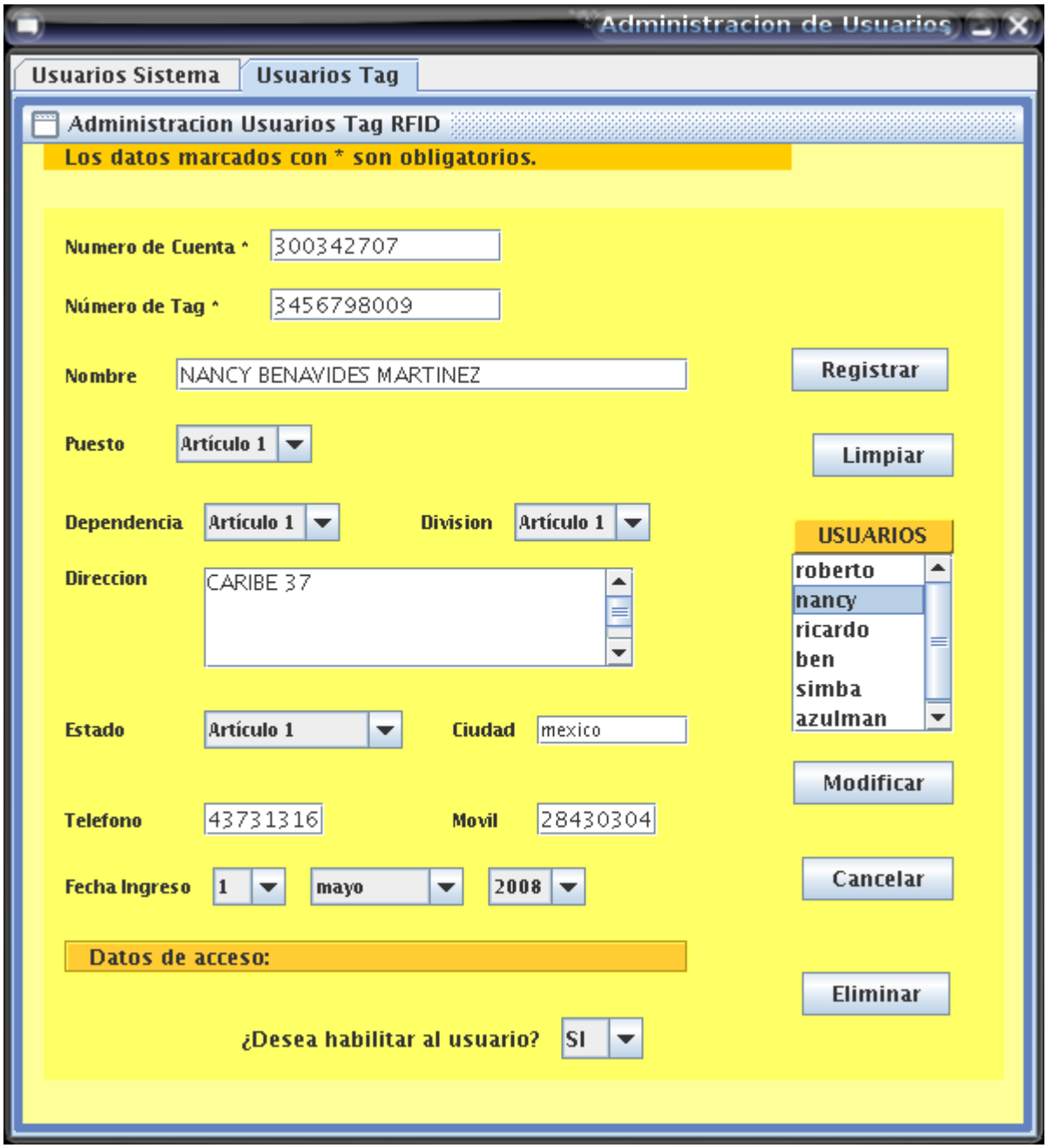
Caso de Uso 14	Validar Información Recibida Dispositivo Electronico
Actores	Usuario Tag RFID
Tipo	Opcional, Generalización
Propósito	El usuario con un Tag registrado puede enviar los datos del mismo hacia el sistema de control a través de la red de Lectores RFID para ser validados, debido a la movilidad de equipo tecnológico.
Descripción	Este caso de uso lo inicia el actor *Usuario Tag RFID*. El personal de la institución puede intentar extraer algún equipo tecnológico, tal como una laptop, escaner, osciloscopio, etc. del inmueble en cuestión, lo que ocasiona que si el Tag dentro del equipo se aleja del rango de frecuencia (Caso de Uso 9) (especificado en metros y configurable para cada Lector RFID) éste enviará la información a través de la red de Lectores RFID para ser validada. La información enviada por el Tag se valida dentro del sistema para saber si éste se encuentra registrado y pertenece al edificio o es ajeno a la institución.
Precondiciones	Se requiere haber ejecutado anteriormente el caso de uso Validar Usuario Sistema de Control y posteriormente el caso de uso Configurar Puerto Control.
Flujo Principal	El personal de la institución intenta extraer algún dispositivo electrónico , el cuál tiene un Tag registrado en su interior. Esto provocará que, al salir del rango de frecuencia, el Tag RFID deje de enviar información hacia la red de Lectores, lo que debe provocar que el sistema de control deje de validarla y por este motivo envíe una señal que active alguna alarma en la institución.
Subflujos	Ninguno
Excepciones	Ninguna

ANEXO B: Prototipo del sistema.

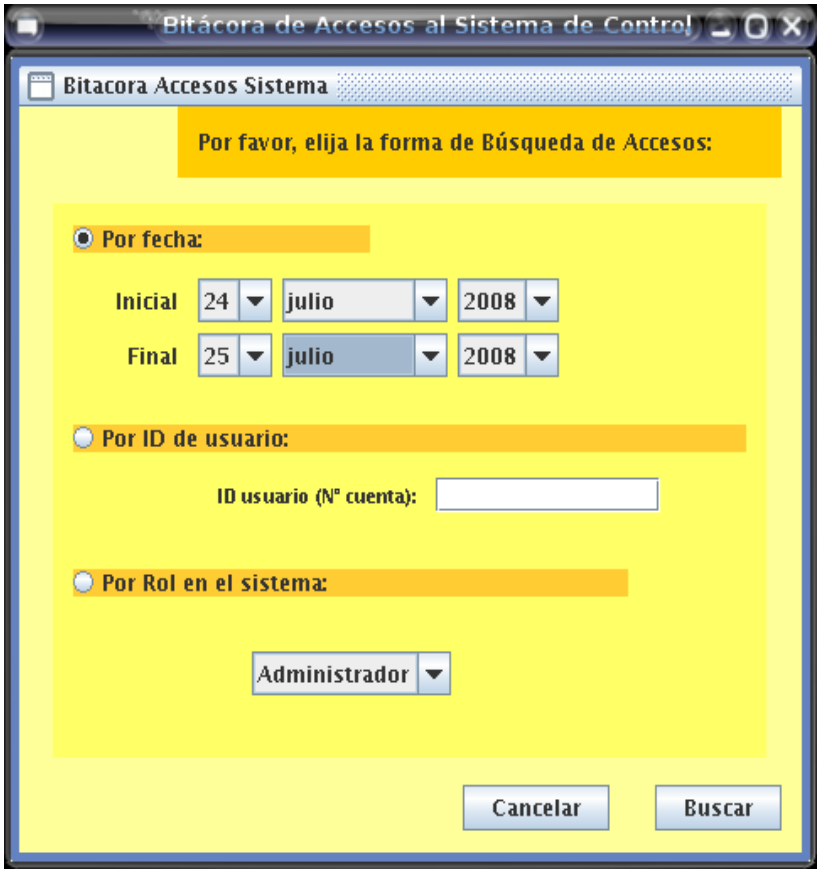
En esta sección veremos las distintas pantallas para el uso del sistema de control para los Lectores RFID Wavetrend.



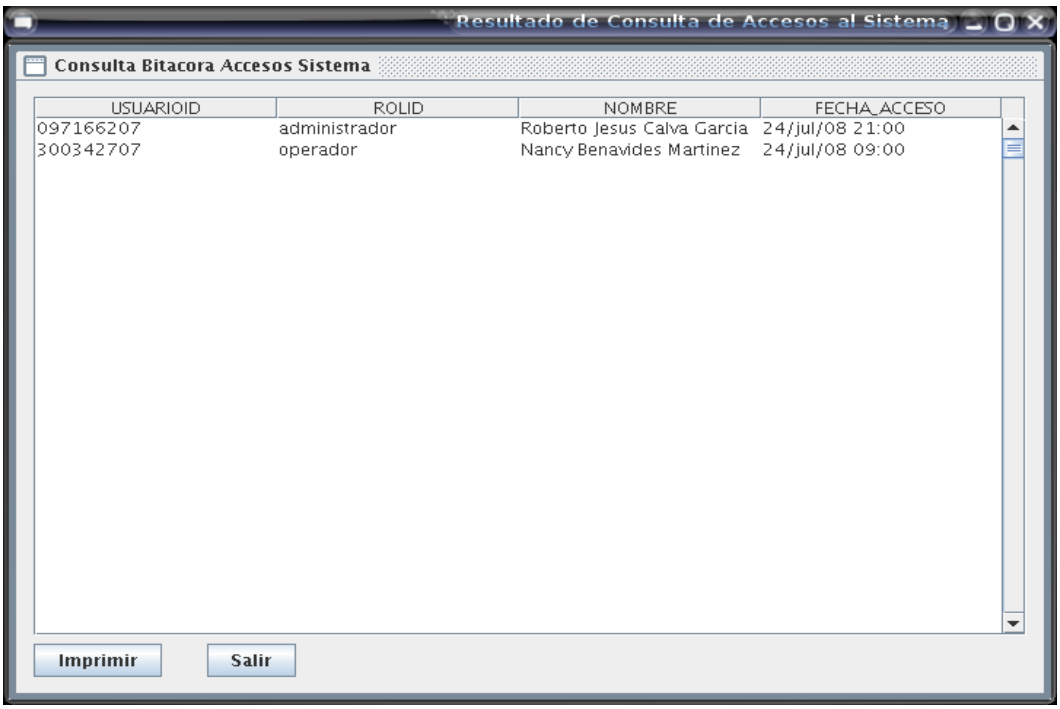
Pantalla Para Administración de Usuarios del Sistema de Control



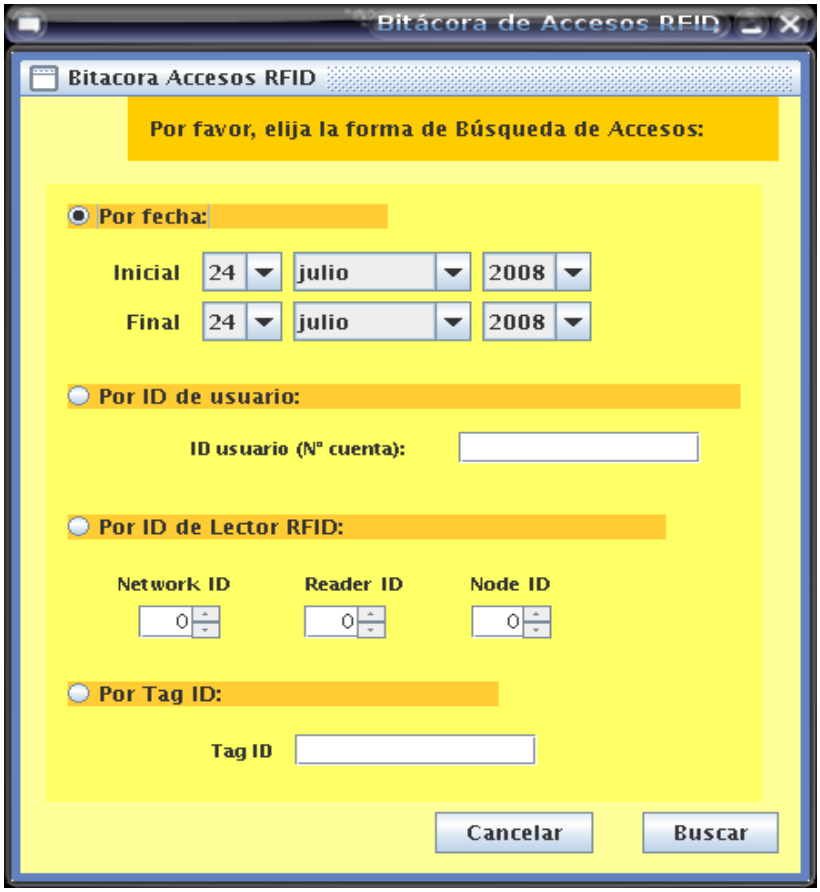
Pantalla Para Administración de Usuarios con Tag RFID



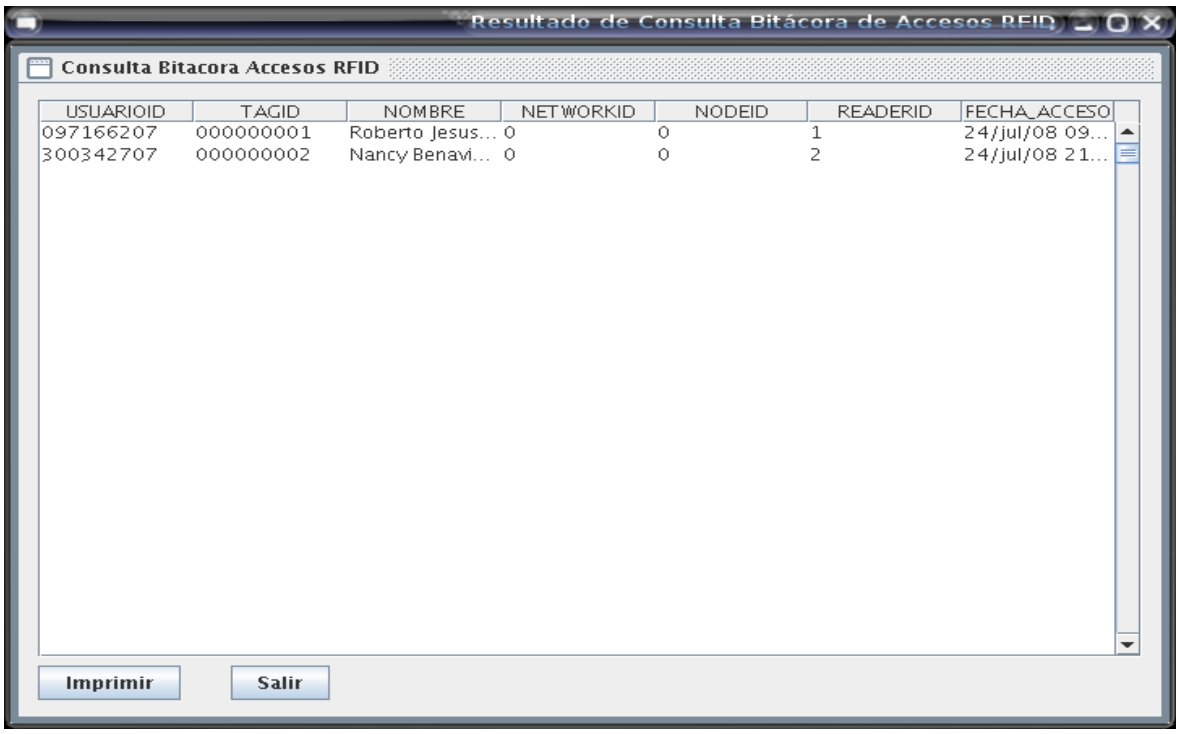
Bitácora de Accesos al Sistema de Control



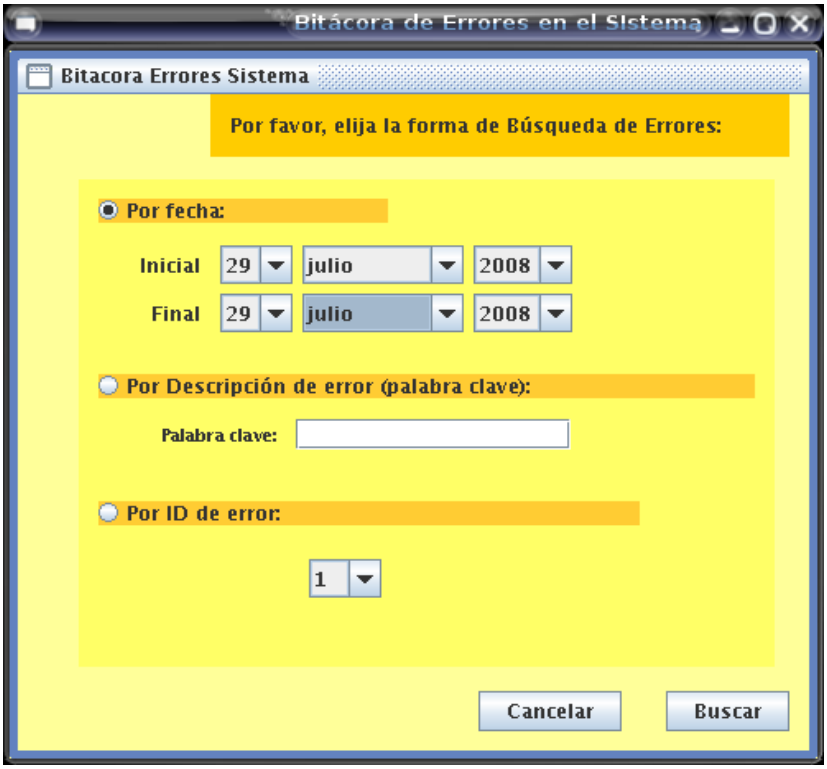
Resultado de Búsqueda Bitácora de Accesos al Sistema de Control



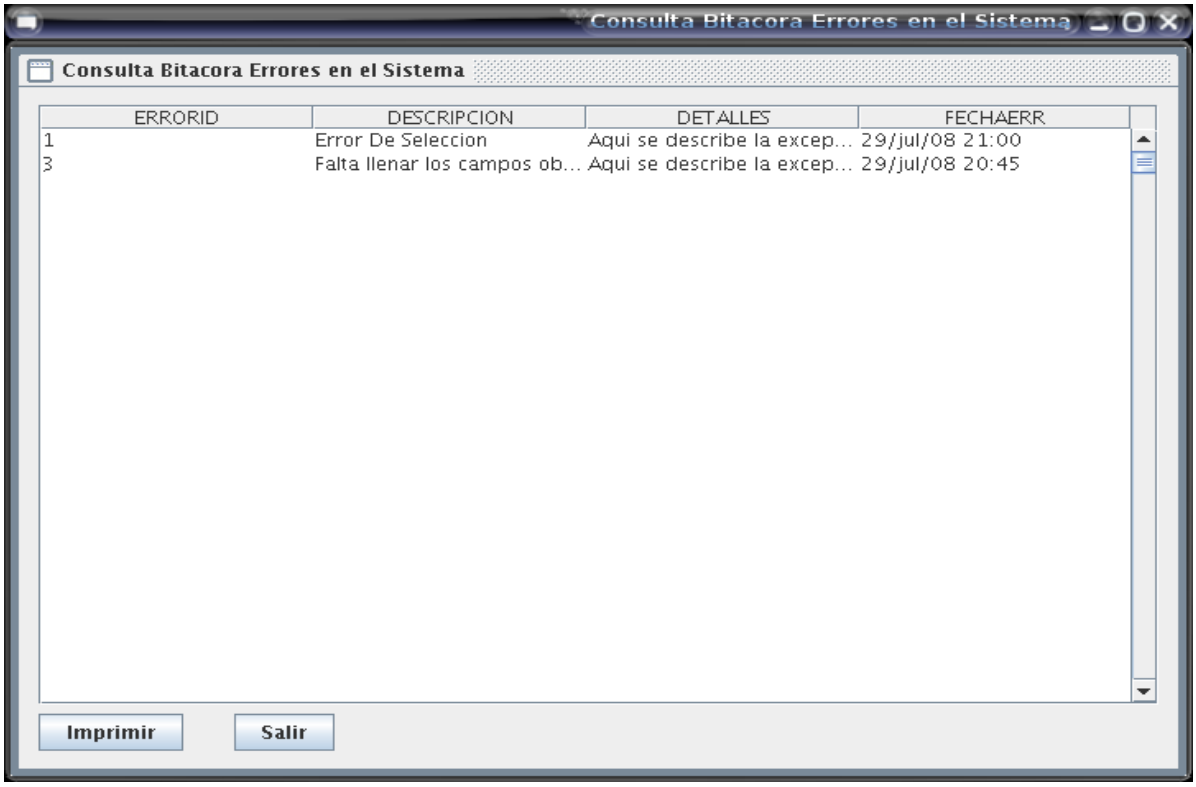
Bitácora de Accesos RFID



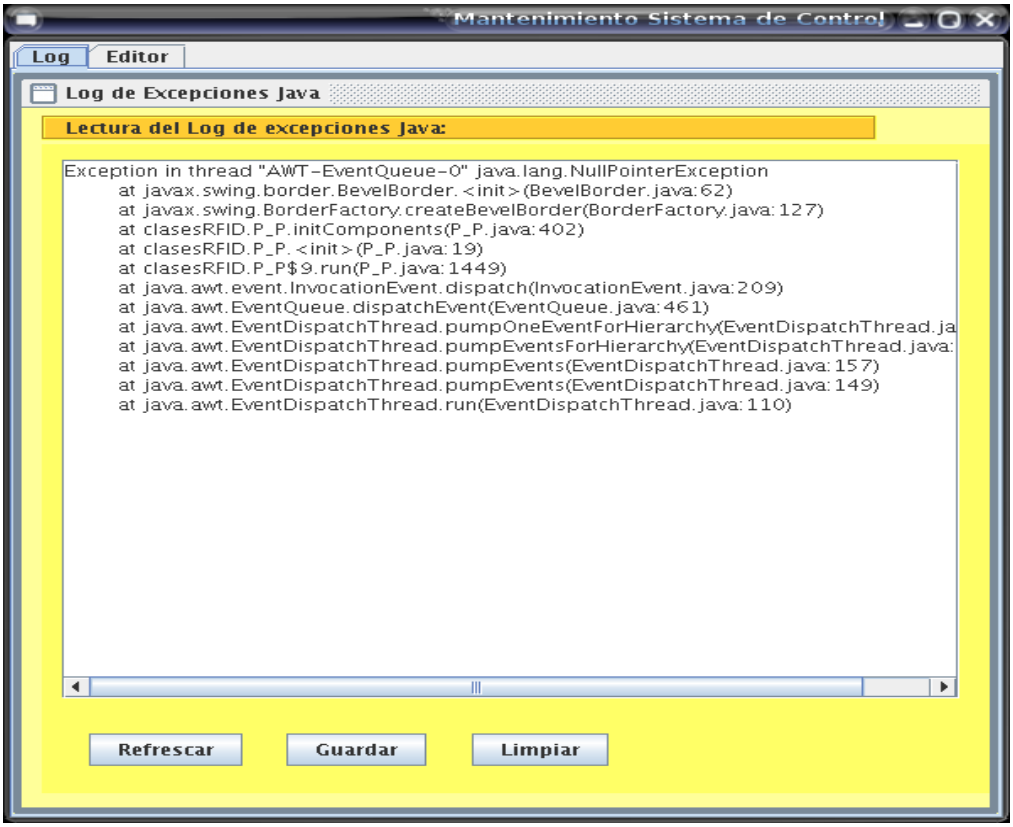
Resultado de Consulta Bitácora de Accesos RFID



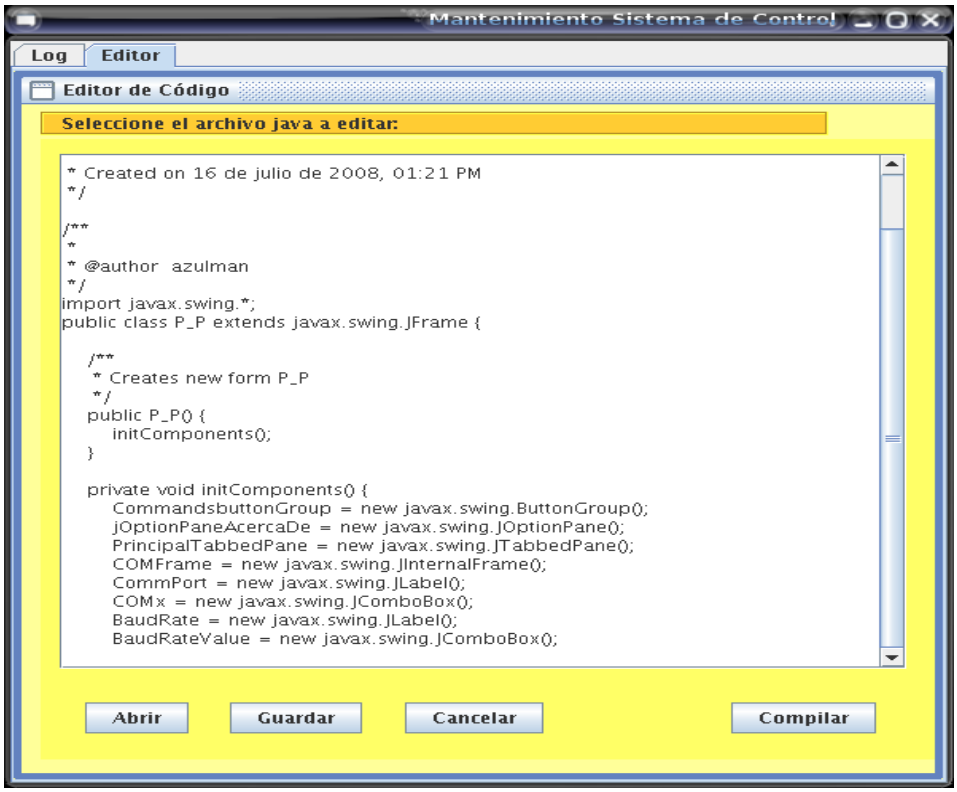
Bitácora de Errores Sistema de Control



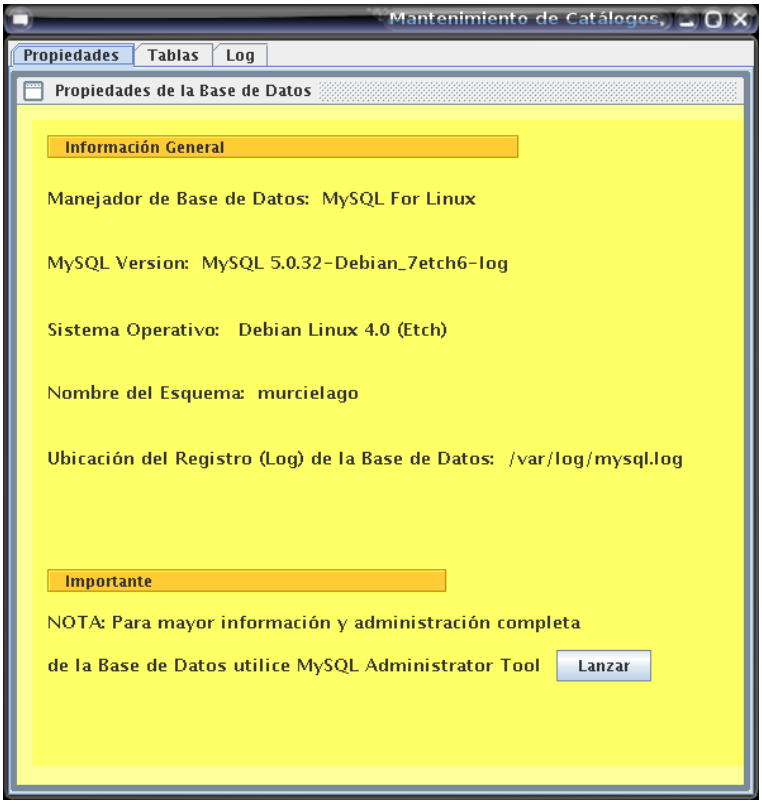
Resultado de Búsqueda Bitácora de Errores Sistema de Control



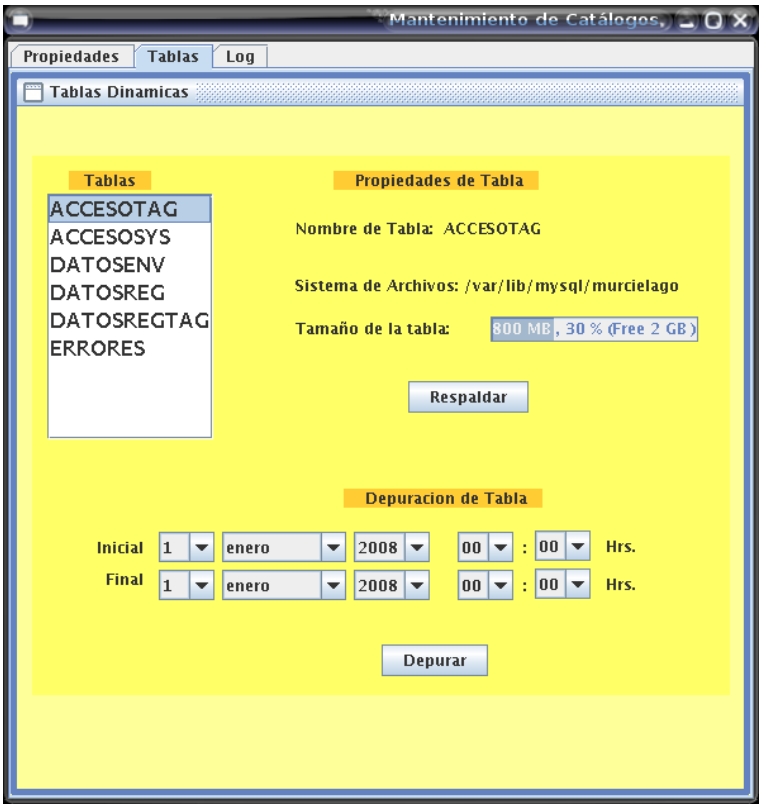
Pantalla Mantenimiento Sistema de Control: Log de sistema



Pantalla Mantenimiento Sistema de Control: Editor de código



Pantalla Mantenimiento Catálogos: Propiedades



Pantalla Mantenimiento Catálogos: Tablas

ANEXO C: Código del sistema.

Debido al gran tamaño del código fuente, en este apartado sólo se muestran algunas clases interesantes y no tan extensas. El el CD anexo se puede encontrar el código fuente completo, así como la licencia GNU GPL, algunos datos interesantes y el prototipo del sistema.

a) P_AS.java

El siguiente código se utilizó para la autentificación de usuarios que quieran accesar al sistema de control. Se realiza una conexión a la base de datos y se le asignan los permisos correspondientes, dependiendo si el usuario es operador o administrador.

```
//P_AS.java
//Clase para autentificar a un usuario que desea accesar al sistema y asignarle permisos.
package clasesRFID;
import java.sql.*;

public class P_AS extends javax.swing.JFrame {

    public P_AS() {
        initComponents();
    }

    private void initComponents() {
        jOptionPaneErrorLogin = new javax.swing.JOptionPane();
        LoginPanel = new javax.swing.JPanel();
        MurcielagoRFID = new javax.swing.JLabel();
        ForWave = new javax.swing.JLabel();
        LoginLabel = new javax.swing.JLabel();
        GetLogin = new javax.swing.JTextField();
        AceptarLogin = new javax.swing.JButton();
        CancelarLogin = new javax.swing.JButton();
        PassLabel = new javax.swing.JLabel();
        GetPassword = new javax.swing.JPasswordField();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setTitle("Acceso a Murcielago RFID");
        setAlwaysOnTop(true);
        setBackground(new java.awt.Color(153, 153, 238));
        setResizable(false);
        LoginPanel.setBackground(new java.awt.Color(204, 204, 204));
        MurcielagoRFID.setBackground(new java.awt.Color(153, 153, 238));
        MurcielagoRFID.setFont(new java.awt.Font("Dialog", 1, 18));
        MurcielagoRFID.setText("MURCIELAGO RFID V 1.0");

        ForWave.setText("For WAVETREND L-RX201");

        LoginLabel.setText("usuario");

        AceptarLogin.setText("Entrar");
        AceptarLogin.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                AceptarLoginActionPerformed(evt);
            }
        });

        CancelarLogin.setText("Cancelar");
        CancelarLogin.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                CancelarLoginActionPerformed(evt);
            }
        });
```

```
        PassLabel.setText("password");

        org.jdesktop.layout.GroupLayout LoginPanelLayout = new
org.jdesktop.layout.GroupLayout(LoginPanel);
        LoginPanel.setLayout(LoginPanelLayout);
        LoginPanelLayout.setHorizontalGroup(
            LoginPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(LoginPanelLayout.createSequentialGroup()
                .add(106, 106, 106)
                .add(MurcielagoRFID)
                .addContainerGap(196, Short.MAX_VALUE))
            .add(org.jdesktop.layout.GroupLayout.TRAILING, LoginPanelLayout.createSequentialGroup()
                .addContainerGap(93, Short.MAX_VALUE)
                .add(LoginPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
                    .add(LoginPanelLayout.createSequentialGroup()
                        .add(LoginPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                            .add(LoginLabel)
                            .add(PassLabel))
                        .add(17, 17, 17)
                        .add(LoginPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING, false)
                            .add(GetPassword)
                            .add(GetLogin, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 150, Short.MAX_VALUE))
                        .add(34, 34, 34)
                        .add(LoginPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                            .add(AceptarLogin)
                            .add(CancelarLogin))
                        .add(30, 30, 30))
                    .add(ForWave))
                .add(53, 53, 53))
        );
        LoginPanelLayout.setVerticalGroup(
            LoginPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(LoginPanelLayout.createSequentialGroup()
                .add(LoginPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(LoginPanelLayout.createSequentialGroup()
                        .addContainerGap()
                        .add(MurcielagoRFID, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE, 45,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)
                        .add(ForWave))
                    .add(LoginPanelLayout.createSequentialGroup()
                        .add(165, 165, 165)
                        .add(LoginPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                            .add(AceptarLogin)
                            .add(GetLogin, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                            .add(LoginLabel))))
                .add(21, 21, 21)
                .add(LoginPanelLayout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                    .add(CancelarLogin)
                    .add(GetPassword, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                    .add(PassLabel))
                .addContainerGap(154, Short.MAX_VALUE))
        );
        org.jdesktop.layout.GroupLayout layout = new org.jdesktop.layout.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(LoginPanel, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING, LoginPanel,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
```

```
            );
            pack();
        }//fin initComponents

        private void AceptarLoginActionPerformed(java.awt.event.ActionEvent evt) {

            Connection con = null;
           Statement stmt = null;
            String login, passwd, sentenciaSQL, sentenciaSQLencrypt, passwordEncrypt;
            char[] pass;
            login = GetLogin.getText(); // login capturado desde JTextField
            pass = GetPassword.getPassword(); // password capturado desde JPasswordField
            passwd = new String(pass);  // password convertido de char a string para su comparación
            System.out.println("login: "+login);
            System.out.println("password: "+passwd);

            // Se intenta cargar el controlador jdbc para MySQL:
            try {
                    Class.forName("com.mysql.jdbc.Driver");
           } catch(java.lang.ClassNotFoundException err) {
                    System.out.println("Clase no encontrada: " + err);
                   }

            // Se intenta la conexión a la BD:
           try {
                    con = DriverManager.getConnection("jdbc:mysql://localhost/murcielago_test",
"murcielago","murcielago123");
                       stmt = con.createStatement();

                    // Se encripta el password
                    sentenciaSQLencrypt = "SELECT password('"+passwd+"') as passwordEncrypt";
                       ResultSet rsEncrypt = stmt.executeQuery(sentenciaSQLencrypt);
                    if(rsEncrypt.next()) {
                            passwordEncrypt = rsEncrypt.getString("passwordEncrypt");
                            System.out.println("passwordEncriptado: "+passwordEncrypt); // Se imprime en pantalla el
passwd encriptado

                            //Se compara la encriptacion con lo almacenado en la BD:
                            sentenciaSQL = "SELECT * FROM LOGINUSUARIO WHERE login='"+login+"' AND
passwd='"+passwordEncrypt+"'";
                            ResultSet rs = stmt.executeQuery(sentenciaSQL);
                            System.out.println(sentenciaSQL);
                            if(rs.next()) {
                                    System.out.println("Usuario Aceptado");
                                    // aqui va sentencia para cerrar ventana login
                                    String arg [] = new String [0];
                                    P_P.main(arg);
                            }
                            else {
                            jOptionPaneErrorLogin.showMessageDialog(P_AS.this,"Error de Acceso: login/password
incorrectos o usuario no " +
                            "registrado."+" \n Por favor, intente de nuevo.","Error de Acceso al
Sistema",jOptionPaneErrorLogin.ERROR_MESSAGE);
                                     GetLogin.setText(null);
                            GetPassword.setText(null);
                            System.out.println("usuario NO aceptado");
                                    }
                                    rs.close();
                                    stmt.close();
                                    con.close();
                    }
                    else {
                                    System.out.println("error al encriptar password");
                                    }
           } catch(java.sql.SQLException error) {
                       System.out.println("Se produjo error de conexión a la BD murcielago: "+error);
                       }
```

```
    }

    private void CancelarLoginActionPerformed(java.awt.event.ActionEvent evt) {

        GetLogin.setText(null);
        GetPassword.setText(null);
    }


    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new P_AS().setVisible(true);
            }
        });
    }

    // Declaración de varibales -no modificar
    private javax.swing.JButton AceptarLogin;
    private javax.swing.JButton CancelarLogin;
    private javax.swing.JLabel ForWave;
    private javax.swing.JTextField GetLogin;
    private javax.swing.JPasswordField GetPassword;
    private javax.swing.JLabel LoginLabel;
    private javax.swing.JPanel LoginPanel;
    private javax.swing.JLabel MurcielagoRFID;
    private javax.swing.JLabel PassLabel;
    private javax.swing.JOptionPane jOptionPaneErrorLogin;
    // Fin de declaración de variables

}
```

b) LeerReader.java

El siguiente código java puede ser utilizado para la lectura de datos a través del puerto serie y se anexa como una sugerencia para la etapa de desarrollo.

```
//LeerReader.java
//Clase para leer respuestas de los Readers Wavetrend.

import java.io.*;
import java.util.*;
import javax.comm.*;
import javax.swing.*;

public class LeerReader implements Runnable, SerialPortEventListener {
static CommPortIdentifier portId;
static Enumeration portList;

InputStream inputStream;
SerialPort serialPort;
Thread readThread;

int i = -1;
char c = ' ';
byte[] bytesTag = new byte[32];
byte[] readBuffer = new byte[64];

public static void main(String[] args) {
portList = CommPortIdentifier.getPortIdentifiers();

try
    {
    portId = CommPortIdentifier.getPortIdentifier("COM1");
    LeerReader reader = new LeerReader();
```

```
        }
    catch (NoSuchPortException nspe)
        {
        System.out.println("Puerto COM1 no encontrado");
        }
    catch (UnsupportedCommOperationException ucoe)
        {
        System.out.println("Umbral (threshold) no disponible.");
        }

    }

    public LeerReader() throws UnsupportedCommOperationException {

        // Si el puerto no esta en uso, se intenta abrir
        try {
            serialPort = (SerialPort) portId.open("LecturaSimpleApp", 2000);
            } catch (PortInUseException e) {}

        //Se recibe información del puerto y se obtiene un canal de entrada
        try {
            serialPort.enableReceiveThreshold(12);
            System.out.println("Puerto '" + serialPort.getName() + "'con configuracion:");
            System.out.println("\tTamaño del buffer de entrada: " + serialPort.getInputBufferSize());
            System.out.println("\tRecibir Enmarcado Permitido? " + serialPort.isReceiveFramingEnabled());
            System.out.println("\tRecibir byte enmarcado: " + serialPort.getReceiveFramingByte());
            System.out.println("\tRecibir umbral (Threshold) Permitido? " +
serialPort.isReceiveThresholdEnabled());
            System.out.println("\tRecibir umbral (Threshold): " + serialPort.getReceiveThreshold());
            System.out.println("\tRecibir Timeout permitido? " + serialPort.isReceiveTimeoutEnabled());
            System.out.println("\tRecibir Timeout: " + serialPort.getReceiveTimeout());
            inputStream = serialPort.getInputStream();
            } catch (IOException e) {}

        try {
            serialPort.addEventListener(this);
            } catch (TooManyListenersException e) {}
            serialPort.notifyOnDataAvailable(true);

        // Se fijan los parámetros de comunicación del puerto
        try {
            serialPort.setSerialPortParams(57600,
            SerialPort.DATABITS_8,
            SerialPort.STOPBITS_1,
            SerialPort.PARITY_NONE);
            } catch (UnsupportedCommOperationException e) {}
            readThread = new Thread(this);
            readThread.start();
    }// fin del constructor


    public void run() {
        try {
            Thread.sleep(20000);
            } catch (InterruptedException e) {}
    }// fin metodo run de Runnable


    //Se da un caso para el evento de datos disponibles en el puerto serie
    public void serialEvent(SerialPortEvent evento) {
        System.out.println("Metodo incorporado serialEvent()");
        switch(evento.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:

    //Se recuperan los bytes provenientes del tag
    try {
    while ((i = inputStream.read()) != -1) {
            c = (char)i;
        if(c == '!'){
           int numBytes = inputStream.read(bytesTag);
           String datosTag = new String(bytesTag);
           if(bytesTag[0] == 42 && bytesTag[1] == 42){
            String tagId = new String(bytesTag, 15, 4);
            System.out.println("Byte 1 Y 2 de bytesTag: [*][*]= " + "["+bytesTag[0]+"]"+ "["+bytesTag[1]+"]");
            System.out.println("Numero de Bytes en bytesTag: " + numBytes);
            System.out.println("4 bytes de TagID en Decimal: " +
"["+bytesTag[15]+"]"+"["+bytesTag[16]+"]"+"["+bytesTag[17]+"]"+"["+bytesTag[18]+"]" );
            System.out.println("TagID: " + tagId.trim());
           }
        }
      }
    } catch (IOException e) {}


    //Si existen, se recuperan las respuestas de los Lectores Wavetrend:
    try {
        while (inputStream.available() > 0) {
        System.out.println("Available bytes: " + inputStream.available());
        int numBytes = inputStream.read(readBuffer);
        System.out.println("Numero de Bytes leidos: " + numBytes);
        }
        System.out.println("\"" + new String(readBuffer).trim() + "\"");
    } catch (IOException e) {}


    //Se intenta cerrar el puerto
    try {
       inputStream.close();
       }catch( IOException e ) {}
    break;
       }// fin instruccion switch
     }// fin metodo serialEvent
    }// fin clase LeerReader
```

---

c) EscribirReader.java

El siguiente código java puede ser utilizado para la escritura de datos a través del puerto serie y se anexa como una sugerencia para la etapa de desarrollo. Nótese que los comandos se encuentran en la variable String “mensaje”.

```
    //EscribirReader.java
    //Clase para escribir comandos y enviarlos a los Readers Wavetrend.
    import java.io.*;
    import java.util.*;
    import javax.comm.*;

    public class EscribirReader {
        static Enumeration listaPuertos;
        static CommPortIdentifier idPuerto;
        public static String mensaje[] = {"ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*" +
```

```
"ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*" +
"ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*" +
"ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*" +
"ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*" +
"ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*ÿ*\u002A", "\u00AA\u0000\u0000\u0000\u00FF\u0000\u00FF",
"\u00AA\u0000\u0000\u0000\u0001\u0001\u0000", "\u00AA\u0000\u0000\u0000\u0001\u0002\u0003",
"\u00AA\u0000\u0000\u0000\u0001\u0003\u0002", "\u00AA\u0000\u0000\u0000\u0001\u0004\u0005",
"\u00AA\u0000\u0000\u0000\u0001\u0005\u0004", "\u00AA\u0000\u0000\u0000\u0001\u0006\u0007",
"\u00AA\u0000\u0000\u0000\u0001\u0007\u0006", "\u00AA\u0000\u0000\u0000\u0001\u0008\u0009",
"\u00AA\u0000\u0000\u0000\u0001\u0009\u0008"};
static SerialPort puertoSerie;
static OutputStream salida;
int comando;

 public EscribirReader(int comando) {
    listaPuertos = CommPortIdentifier.getPortIdentifiers();

    while( listaPuertos.hasMoreElements() ) {
        idPuerto = (CommPortIdentifier)listaPuertos.nextElement();
        if( idPuerto.getPortType() == CommPortIdentifier.PORT_SERIAL ) {
                if( idPuerto.getName().equals("COM1") ) {
                  if( idPuerto.getName().equals("/dev/term/a") ) {

  // Si el puerto no está en uso, se intenta abrir
                    try {
                        puertoSerie = ( SerialPort )idPuerto.open("AplEscritura",2000);
                    } catch( PortInUseException e ) {}

  // Se obtiene un canal de salida
                    try {
                        salida = puertoSerie.getOutputStream();
                    } catch( IOException e ) {}

 // Se fijan los parámetros de comunicación del puerto
                    try {
                        puertoSerie.setSerialPortParams( 57600,
                        SerialPort.DATABITS_8,
                        SerialPort.STOPBITS_1,
                        SerialPort.PARITY_NONE );
                    } catch( UnsupportedCommOperationException e ) {}

// Se envía el mensaje
                    try {
                        salida.write( mensaje[comando].getBytes() );
                        System.out.println(mensaje[comando]);
                    } catch( IOException e ) {}
                    try {
                        salida.close();
                    }catch( IOException e ) {}
                }
            }
        }
    }
  }
}
```

ANEXO D: Comandos de control de Lectores RFID Wavetrend L-RX201.

Reset Network Command.

La función de este comando es reiniciar la red entera de Lectores RFID y restablecer las direcciones NODE ID. La dirección NODE ID en el paquete command tiene el valor 255 (broadcast value) para asegurarse que la red entera de lectores será reiniciada. Únicamente el el Lector 1 responderá con un paquete response.

Command						
0xAA	0x00	0x00	0x00	0xFF	0x00	Checksum
Response						
0x55	0x00	NetworkID	ReceiverID	0x01	0x00	Checksum

Start / Enabling Polling Mode Command

La función de este comando es configurar al Lector 1 con auto-poleo automático. Esto hará que se envíen auto-peticiones de lectura de tags a toda la red de Lectores.

Command						
0xAA	0x00	NetworkID	ReceiverID	NodeID	0x01	Checksum
Response						
0x55	0x00	NetworkID	ReceiverID	NodeID	0x01	Checksum

Disable Auto Polling Command

La función de este comando es deshabilitar futuros auto-poleos después del reinicio de la red, reiniciando el flag de auto-polling en la memoria EEPROM. Este comando puede direccionarse directamente al Lector 1 o a todo el broadcast.

Command						
0xAA	0x00	NetworkID	ReceiverID	NodeID	0x02	Checksum
Response						
0x55	0x00	NetworkID	ReceiverID	NodeID	0x02	Checksum

Ping Reader Command

El comando Ping es usado simplemente para revisar si un Lector en la red funciona correctamente. El tipo de error se muestra más abajo.

Command						
0xAA	0x00	NetworkID	ReceiverID	NodeID	0x03	Checksum

Response							
0x55	0x01	NetworkID	ReceiverID	NodeID	0x03	ErrorNumber	Checksum

ErrorNumber	Descripción
0	No errors encountered
1	Unknown reader command received
2	Tag Table underflow error
3	Command Packet checksum error
4	RF Module - Unknown command response
5	RF Module - Unknown general response
6	RF Module - Re-sync failure
7	RF Module - Command response failure
8	RF Module - Receive response failure
9	No response packet received from polled reader

Set Network ID Command

La función de este comando es asignar el Network ID de un Lector y almacenarlo el la memoria EEPROM.

Command							
0xAA	0x01	NetworkID	ReceiverID	NodeID	0x04	NewNetworkID	Checksum
Response							
0x55	0x00	NetworkID	ReceiverID	NodeID	0x04	Checksum	

Set Reader ID Command

La función de este comando es asignar el Reader ID de un Lector y almacenarlo el la memoria EEPROM.

Command							
0xAA	0x01	NetworkID	ReceiverID	NodeID	0x05	NewReaderID	Checksum
Response							
0x55	0x00	NetworkID	ReceiverID	NodeID	0x05	Checksum	

Get Tag Packet Command

Este es el comando más usado en el sistema. Su función es hacer peticiones al Lector para ver si éste ha detectado algún tag. Si no es leído ningún tag, un paquete vacío en el respectivo byte de la respuesta es regresado. Los bytes de datos recibidos de un tag se muestran más abajo.

Command
0xAA 0x00 NetworkID ReceiverID NodeID 0x06 Checksum
Response (vacío)
0x55 0x00 NetworkID ReceiverID NodeID 0x06 Checksum
Response (tag leído)
0x55 DataLength NetworkID ReceiverID NodeID 0x06 Data Checksum

Data	Función/Valor	Data	Función/Valor
1	!	17	Tag ID MSB
2	*	18	Tag ID
3	*	19	Tag ID
4	Interval	20	Tag ID LSB
5	Reed Switch Counter	21	Type of tag flag
6	Firmware version	22	Reader ID
7	B	23	RSSI signal strength
8	C	24	Checksum
9	Movement switch counter	25	20H (reserved)
10	Age byte MSB	26	Alarm byte
11	Age byte	27	Node ID
12	Age byte	28	Network ID
13	Age byte LSB	29	Reader Set RSSI Value
14	Site code MSB	30	Firmware Version
15	Site code	31	LF
16	Site code LSB	32	CR

Set RSSI Value Command

Este comando asignará el valor RSSI al tag y lo almacenará en la memoria EEPROM. Los valores de RSSI van desde 0 – 255. 0 es el valor más sensitivo de señal de radiofrecuencia.

Command
0xAA 0x01 NetworkID ReceiverID NodeID 0x07 NewRSSI Checksum
Response
0x55 0x00 NetworkID ReceiverID NodeID 0x07 Checksum

Get RSSI Value Command

Este comando regresa el valor de RSSI que se está utilizando actualmente y el cual está almacenado en la memoria EEPROM.

Command
0xAA 0x00 NetworkID ReceiverID NodeID 0x08 Checksum
Response
0x55 0x01 NetworkID ReceiverID NodeID 0x08 RSSI Checksum

Set Site Code Command

el Site Code (Código de Sitio) es un grupo de 3 bytes asignados a cada tag. Su función es habilitar al Lector para filtrar tags que no pertenezcan al Side Code que se está monitoreando. Este valor es almacenado en la memoria EEPROM. Cuando un Site Code es asignado con valor 0, todos los tags son reportados.

Command
0xAA 0x03 NetworkID ReceiverID NodeID 0x09 Site1 Site2 Site3 Checksum
Response
0x55 0x00 NetworkID ReceiverID NodeID 0x09 Checksum

Get Site Code Command

Este comando regresa el valor de Site Code.

Command
0xAA 0x00 NetworkID ReceiverID NodeID 0x0A Checksum
Response
0x55 0x03 NetworkID ReceiverID NodeID 0x0A Site1 Site2 Site3 Checksum

Set Receiver Gain Command

Este comando configura la ganancia del módulo de RF en 2 diferentes niveles.

Command
0xAA 0x01 NetworkID ReceiverID NodeID 0x0B Gain Checksum
Response
0x55 0x00 NetworkID ReceiverID NodeID 0x0B Checksum

gain=0 (rango corto de detección de tags) **gain=1** (rango largo de detección de tags)

Get Receiver Gain Command

Este comando obtiene la ganancia del módulo de RF.

Command						
0xAA	0x00	NetworkID	ReceiverID	NodeID	0x0C	Checksum
Response						
0x55	0x01	NetworkID	ReceiverID	NodeID	0x0C	Gain	Checksum

gain=0 (rango corto de detección de tags) **gain=1** (rango largo de detección de tags)

Set Alarm Tag Filter Status Command

Este comando filtrará los tags con alguna condición de alarma. Los valores en el byte Status se muestran más abajo.

Command							
0xAA	0x01	NetworkID	ReceiverID	NodeID	0x0D	Status	Checksum
Response							
0x55	0x00	NetworkID	ReceiverID	NodeID	0x0D	Checksum	

Status = 0 - Report all tags
Status = 1 - Report only tags with an Alarm condition
Status = 2 - Report only tags without any Alarm condition

Get Alarm Tag Filter Status Command

Este comando regresará el actual valor para la condición de filtro de tags con alarma.

Command							
0xAA	0x00	NetworkID	ReceiverID	NodeID	0x0E	Checksum	
Response							
0x55	0x01	NetworkID	ReceiverID	NodeID	0x0E	Status	Checksum

Status = 0 - Report all tags
Status = 1 - Report only tags with an Alarm condition
Status = 2 - Report only tags without any Alarm condition

Reset Network Baud Rate Command

Este comando nos sirve para establecer la velocidad de transmisión de los Lectores RFID. Este comando únicamente acepta el valor del broadcast (es decir, se envía hacia todos los Lectores en todas las redes) y no hay valores de respuesta. Un cambio no adecuado puede ocasionar la pérdida de comunicación entre la PC y la red de Lectores. EL valor por default es de 115200 baudios.

Command
0xAA 0x01 0xFF 0xFF 0xFF 0xFF Rate Checksum

Rate 0 = 115200 baud
Rate 1 = 57600 baud
Rate 2 = 38800 baud
Rate 3 = 19200 baud
Rate 4 = 9600 baud

Start Environmental Noise Level Value Calculation

Este comando establece al Lector en modo evaluación para calcular el nivel ruido blanco en el entorno a los 433.92 Mhz. El resultado será obtenido después de aproximadamente 40 segundos. Durante este período no serán decodificados datos de tags. Una vez terminado el cálculo de ruido blanco, el Lector regresará a su operación normal.

Command
0xAA 0x00 NetworkID ReceiverID NodeID 0x11 Checksum
Response
0x55 0x00 NetworkID ReceiverID NodeID 0x11 Checksum

Get Environmental Noise Level Value

Este comando regresará el valor calculado de ruido blanco en el ambiente (valores entre 0 – 255). Este comando sólo puede ser utilizado después de haber calculado el nivel de ruido en el ambiente.

Command
0xAA 0x00 NetworkID ReceiverID NodeID 0x12 Checksum
Response
0x55 0x01 NetworkID ReceiverID NodeID 0x12 Noise Checksum

ANEXO E: Diseño de la Base de Datos.

A continuación se presenta el diccionario de datos:

ESTADOS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
ESTADO_ID	INTEGER	NOT NULL	Número ID del Estado de la Rep.	si	no
DESCRIPCION	VARCHAR(20)	NOT NULL	Descripción del Estado de la Rep.	no	no

MARCA_TAGS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
MARCA_ID	INTEGER	NOT NULL	Número ID de la marca del tag	si	no
DESCRIPCION	VARCHAR(20)	NOT NULL	Descripción de la marca del tag	no	no

DEPENDENCIAS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
DEPENDENCIA_ID	INTEGER	NOT NULL	Número ID de la dependencia	si	no
DESCRIPCION	VARCHAR(80)	NOT NULL	Descripción de la dependencia	no	no

DIVISIONES					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
DIVISION_ID	INTEGER	NOT NULL	Número ID de la división	si	no
DESCRIPCION	VARCHAR(80)	NOT NULL	Descripción de la división	no	no

ROLES					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
ROLID	INTEGER	NOT NULL	Número ID del rol en el sistema	si	no
DESCRIPCION	VARCHAR(80)	NOT NULL	Descripción del rol en el sistema	no	no

SEVERIDAD					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
SEVERIDADID	INTEGER	NOT NULL	Número ID de severidad del error	si	no
TIPO_SEVERIDAD	VARCHAR(10)	NOT NULL	Descripción de severidad del error	no	no

MODELOS_READERS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
MODELO_READERID	INTEGER	NOT NULL	Número ID del modelo de Lector	si	no
DESCRIPCION	VARCHAR(45)	NOT NULL	Descripción del modelo de Lector	no	no

RESPONSES					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
RESPONSEID	CHAR(4)	NOT NULL	Núm. ID de la respuesta del Lector	si	no
FUNCION	VARCHAR(35)	NOT NULL	Función de la respuesta del Lector	no	no
RESPUESTA	VARCHAR(20)	NOT NULL	Respuesta del Lector RFID	no	no
DESCRIPCION	VARCHAR(40)	NOT NULL	Descripción del comando	no	no

COMMANDS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
COMMANDID	CHAR(4)	NOT NULL	Núm. ID del comando del Lector	si	no
FUNCION	VARCHAR(35)	NOT NULL	Función del comando del Lector	no	no
COMANDO	VARCHAR(20)	NOT NULL	Comando del Lector RFID	no	no
DESCRIPCION	VARCHAR(40)	NOT NULL	Descripción del comando	no	no

READERS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
NETWORKID	TINYINT	NOT NULL	ID de Red del Lector	si	no
NODEID	TINYINT	NOT NULL	ID de nodo del Lector	si	no
READERID	TINYINT	NOT NULL	ID interno del Lector	si	no
MODELO_READERID	INTEGER	NOT NULL	Número ID del modelo de Lector	no	si
DESCRIPCION	VARCHAR(45)	NULL	Descripción del Lector	no	no

PUESTOS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
PUESTO_ID	INTEGER	NOT NULL	Puesto del usuario del sistema	si	no
DEPENDENCIA_ID	INTEGER	NOT NULL	Número ID de la dependencia	si	si
DESCRIPCION	VARCHAR(80)	NOT NULL	Descripción del puesto	no	no

READERS_STATUS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
NETWORKID	TINYINT	NOT NULL	ID de Red del Lector	si	si
NODEID	TINYINT	NOT NULL	ID de nodo del Lector	si	si
READERID	TINYINT	NOT NULL	ID interno del Lector	si	si
RSSI	INTEGER	NOT NULL	Received Signal Strength Indicator	si	no
TAG_LAST_DATE	DATETIME	NOT NULL	Ultima fecha de registro de tag	si	no
OBSERVACIONES	VARCHAR(80)	NULL	Observación sobe el estado del Lector	no	no

CATERRORES					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
ERRORID	INTEGER	NOT NULL	ID del error	si	no
SEVERIDADID	INTEGER	NOT NULL	Número ID de severidad del error	si	si
DESCRIPCION	VARCHAR(20)	NULL	Descripción del error	no	no
DETALLES	VARCHAR(255)	NULL	Detalles del error	no	no

MODELO_TAGS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
MODELO_ID	INTEGER	NOT NULL	ID del modelo de tag	si	no
MARCA_ID	INTEGER	NOT NULL	Número ID de la marca del tag	si	si
DESCRIPCION	VARCHAR(80)	NULL	Descripción del modelo de tag	no	no

ERRORES_SYS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
ERRORID	INTEGER	NOT NULL	ID del error	si	si
SEVERIDADID	INTEGER	NOT NULL	Número ID de severidad del error	si	si
FECHAERR	DATETIME	NULL	Descripción del error	no	no

CIUDADES					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
CIUDAD_ID	INTEGER	NOT NULL	Número ID de la Ciudad de la Rep.	si	no
ESTADO_ID	INTEGER	NOT NULL	Número ID del Estado de la Rep.	si	si
DESCRIPCION	VARCHAR(45)	NOT NULL	Descripción de la Ciudad de la Rep.	no	no

DATOSENV					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
NETWORKID	TINYINT	NOT NULL	ID de Red del Lector	si	si
NODEID	TINYINT	NOT NULL	ID de nodo del Lector	si	si
READERID	TINYINT	NOT NULL	ID interno del Lector	si	si
COMMAND	VARCHAR(20)	NOT NULL	Comando enviado al Lector	si	no
FECHAENV	DATETIME	NOT NULL	Fecha de envío del comando	si	no

DATOSREC					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
NETWORKID	TINYINT	NOT NULL	ID de Red del Lector	si	si
NODEID	TINYINT	NOT NULL	ID de nodo del Lector	si	si
READERID	TINYINT	NOT NULL	ID interno del Lector	si	si
COMMAND	VARCHAR(20)	NOT NULL	Comando enviado al Lector	si	si

FECHAENV	DATETIME	NOT NULL	Fecha de envío del comando	si	si
RESPONSE	VARCHAR(255)	NOT NULL	Respuesta del Lector	si	no
FECHAREC	DATETIME	NOT NULL	Fecha de respuesta del Lector	si	no

USUARIOS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
USUARIOID	INT(9)	NOT NULL	ID de usuario (No Cuenta)	si	no
ROLID	INTEGER	NOT NULL	Número ID del rol en el sistema	si	si
PUESTO_ID	INTEGER	NOT NULL	Puesto del usuario del sistema	si	si
DEPENDENCIA_ID	INTEGER	NOT NULL	Número ID de la dependencia	si	si
DIVISION_ID	INTEGER	NOT NULL	Número ID de la división	si	si
ESTADO_ID	INTEGER	NOT NULL	Número ID del Estado de la Rep.	si	si
CIUDAD_ID	INTEGER	NOT NULL	Número ID de la Ciudad de la Rep.	si	si
NOMBRE	VARCHAR(45)	NOT NULL	Nombre del usuario del sistema	no	no
DIRECCION	VARCHAR(80)	NOT NULL	Dirección del usuario del sistema	no	no
TELEFONO	VARCHAR(15)	NULL	Teléfono del usuario del sistema	no	no
MOVIL	VARCHAR(15)	NULL	Móvil del usuario del sistema	no	no
FECHAIN	DATETIME	NOT NULL	Fecha de ingreso del usuario	no	no
LOGIN	CHAR(16)	NOT NULL	Login del usuario del sistema	no	no
PASSWD	CHAR(41)	NOT NULL	Password del usuario del sistema	no	no

TAGS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
TAGID	INTEGER	NOT NULL	ID del tag	si	no
MARCA_ID	INTEGER	NOT NULL	Número ID de la marca del tag	si	si
MODELO_ID	INTEGER	NOT NULL	ID del modelo de tag	si	si
ACTIVACION	ENUM('E','D')	NOT NULL	Valor de activación/desactivación	no	no
FECHA_ACTIVACION	DATETIME	NOT NULL	Fecha de activación del tag	no	no

ACCESOSYS					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
ACCESO_SYSID	INTEGER	NOT NULL	ID de usuario (No Cuenta)	si	no
USUARIOID	INT(9)	NOT NULL	ID de usuario (No Cuenta)	si	si
ROLID	INTEGER	NOT NULL	Número ID del rol en el sistema	si	si
PUESTO_ID	INTEGER	NOT NULL	Puesto del usuario del sistema	si	si
DEPENDENCIA_	INTEGER	NOT NULL	Número ID de la dependencia	si	si

ID					
DIVISION_ID	INTEGER	NOT NULL	Número ID de la división	si	si
ESTADO_ID	INTEGER	NOT NULL	Número ID del Estado de la Rep.	si	si
CIUDAD_ID	INTEGER	NOT NULL	Número ID de la Ciudad de la Rep.	si	si
FECHA_ACCESO	DATETIME	NOT NULL	Fecha de ingreso del usuario	no	no

EQUIPOTAG					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
NUMEROID	INTEGER	NOT NULL	ID del equipo	si	no
TAGID	INTEGER	NOT NULL	ID del tag	si	si
MODELO_ID	INTEGER	NOT NULL	ID del modelo de tag	si	si
MARCA_ID	INTEGER	NOT NULL	Número ID de la marca del tag	si	si
MARCAEQUIPO	VARCHAR(45)	NOT NULL	Marca del equipo	no	no
MODELOEQUIPO	VARCHAR(25)	NOT NULL	Modelo del equipo	no	no
DESCRIPCION	VARCHAR(25)	NULL	Descripción del equipo	no	no
DIVISION	VARCHAR(25)	NOT NULL	División a la que pertenece	no	no
DIRECCION	VARCHAR(80)	NOT NULL	Dirección donde se encuentra el equipo	no	no
FECHAIN	DATETIME	NOT NULL	Fecha de ingreso del equipo	no	no

DATOSRECTAG					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
NETWORKID	TINYINT	NOT NULL	ID de Red del Lector	si	si
NODEID	TINYINT	NOT NULL	ID de nodo del Lector	si	si
READERID	TINYINT	NOT NULL	ID interno del Lector	si	si
COMMAND	VARCHAR(20)	NOT NULL	Comando enviado al Lector	si	si
FECHAENV	DATETIME	NOT NULL	Fecha de envío del comando	si	si
RESPONSE	VARCHAR(255)	NOT NULL	Respuesta del Lector	si	si
FECHAREC	DATETIME	NOT NULL	Fecha de respuesta del Lector	si	si
TAG_DATA	CHAR(4)	NOT NULL	Datos recibidos del tag	no	no
FECHARECTAG	DATETIME	NOT NULL	Fecha de recepción de datos	no	no

USUARIOTAG					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
USUARIOID	INT(9)	NOT NULL	ID de usuario (No Cuenta)	si	no
TAGID	INTEGER	NOT NULL	Número ID del rol en el sistema	si	si
MODELO_ID	INTEGER	NOT NULL	ID del modelo de tag	si	si
MARCA_ID	INTEGER	NOT NULL	Número ID de la marca del tag	si	si

PUESTO_ID	INTEGER	NOT NULL	Puesto del usuario del sistema	si	si
DEPENDENCIA_ID	INTEGER	NOT NULL	Número ID de la dependencia	si	si
DIVISION_ID	INTEGER	NOT NULL	Número ID de la división	si	si
ESTADO_ID	INTEGER	NOT NULL	Número ID del Estado de la Rep.	si	si
CIUDAD_ID	INTEGER	NOT NULL	Número ID de la Ciudad de la Rep.	si	si
NOMBRE	VARCHAR(45)	NOT NULL	Nombre del usuario del sistema	no	no
DIRECCION	VARCHAR(80)	NOT NULL	Dirección del usuario del sistema	no	no
TELEFONO	VARCHAR(15)	NULL	Teléfono del usuario del sistema	no	no
MOVIL	VARCHAR(15)	NULL	Móvil del usuario del sistema	no	no
FECHAIN	DATETIME	NOT NULL	Fecha de ingreso del usuario	no	no

ACCESOTAG					
Atributo	**Tipo Dato**	**Opción NULL**	**Definición**	**PK**	**FK**
NETWORKID	TINYINT	NOT NULL	ID de Red del Lector	si	si
NODEID	TINYINT	NOT NULL	ID de nodo del Lector	si	si
READERID	TINYINT	NOT NULL	ID interno del Lector	si	si
USUARIOID	INT(9)	NOT NULL	ID de usuario (No Cuenta)	si	si
TAGID	INTEGER	NOT NULL	Número ID del rol en el sistema	si	si
MODELO_ID	INTEGER	NOT NULL	ID del modelo de tag	si	si
MARCA_ID	INTEGER	NOT NULL	Número ID de la marca del tag	si	si
PUESTO_ID	INTEGER	NOT NULL	Puesto del usuario del sistema	si	si
DEPENDENCIA_ID	INTEGER	NOT NULL	Número ID de la dependencia	si	si
DIVISION_ID	INTEGER	NOT NULL	Número ID de la división	si	si
ESTADO_ID	INTEGER	NOT NULL	Número ID del Estado de la Rep.	si	si
CIUDAD_ID	INTEGER	NOT NULL	Número ID de la Ciudad de la Rep.	si	si
FECHA_ACCESO	DATETIME	NOT NULL	Fecha de ingreso del usuario	si	no
DETALLES	VARCHAR(45)	NULL	Detalles del acceso vía tag	no	no

Scripts y sentencias SQL:

Creación Base de Datos:

```
azulman@debian-azulin:~$ mysql -u root -p

mysql> create database murcielago_test;

mysql> use murcielago_test;
```

Creación de Tablas (Ordenadas por FK):

```
CREATE TABLE ESTADOS (
  ESTADO_ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  DESCRIPCION VARCHAR(20) NOT NULL,
  PRIMARY KEY(ESTADO_ID)
)
TYPE=InnoDB;

CREATE TABLE MARCA_TAGS (
  MARCA_ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  DESCRIPCION VARCHAR(20) NULL,
  PRIMARY KEY(MARCA_ID)
)
TYPE=InnoDB;

CREATE TABLE DEPENDENCIAS (
  DEPENDENCIA_ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  DESCRIPCION  VARCHAR(80) NOT NULL,
  PRIMARY KEY(DEPENDENCIA_ID)
)
TYPE=InnoDB;

CREATE TABLE DIVISIONES (
  DIVISION_ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  DESCRIPCION VARCHAR(80) NOT NULL,
  PRIMARY KEY(DIVISION_ID)
)
TYPE=InnoDB;

CREATE TABLE ROLES (
  ROLID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  DESCRIPCION VARCHAR(80) NOT NULL,
  PRIMARY KEY(ROLID)
)
TYPE=InnoDB;


CREATE TABLE SEVERIDAD (
  SEVERIDADID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  TIPO_SEVERIDAD VARCHAR(10) NOT NULL,
  PRIMARY KEY(SEVERIDADID)
);

CREATE TABLE MODELOS_READERS (
  MODELO_READERID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  DESCRIPCION VARCHAR(45) NOT NULL,
  PRIMARY KEY(MODELO_READERID)
)
TYPE=InnoDB;

CREATE TABLE RESPONSES (
  RESPONSEID CHAR(4) NOT NULL,
  FUNCION VARCHAR(35) NOT NULL,
  RESPUESTA VARCHAR(20) NOT NULL,
  DESCRIPCION VARCHAR(40) NOT NULL,
  PRIMARY KEY(RESPONSEID)
)
TYPE=InnoDB;

CREATE TABLE COMMANDS (
  COMMANDID CHAR(4) NOT NULL,
  FUNCION VARCHAR(35) NOT NULL,
  COMANDO VARCHAR(20) NOT NULL,
  DESCRIPCION VARCHAR(40) NOT NULL,
  PRIMARY KEY(COMMANDID)
)
```

```
TYPE=InnoDB;

CREATE TABLE READERS (
  NETWORKID TINYINT UNSIGNED NOT NULL,
  NODEID TINYINT UNSIGNED NOT NULL,
  READERID TINYINT UNSIGNED NOT NULL,
  MODELO_READERID INTEGER UNSIGNED NOT NULL,
  DESCRIPCION VARCHAR(45) NULL,
  PRIMARY KEY(NETWORKID, NODEID, READERID),
  INDEX CATREADERS_FKIndex1(MODELO_READERID),
  FOREIGN KEY(MODELO_READERID)
    REFERENCES MODELOS_READERS(MODELO_READERID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
)
TYPE=InnoDB;

CREATE TABLE PUESTOS (
  PUESTO_ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  DEPENDENCIA_ID INTEGER UNSIGNED NOT NULL,
  DESCRIPCION VARCHAR(80) NOT NULL,
  PRIMARY KEY(PUESTO_ID, DEPENDENCIA_ID),
  INDEX PUESTOS_FKIndex1(DEPENDENCIA_ID),
  FOREIGN KEY(DEPENDENCIA_ID)
    REFERENCES DEPENDENCIAS(DEPENDENCIA_ID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
)
TYPE=InnoDB;

CREATE TABLE READERS_STATUS (
  NETWORKID TINYINT UNSIGNED NOT NULL,
  NODEID TINYINT UNSIGNED NOT NULL,
  READERID TINYINT UNSIGNED NOT NULL,
  RSSI INTEGER UNSIGNED NOT NULL,
  TAG_LAST_DATE DATETIME NOT NULL,
  OBSERVACIONES VARCHAR(80) NULL,
  PRIMARY KEY(NETWORKID, NODEID, READERID, RSSI, TAG_LAST_DATE),
  INDEX READERS_STATUS_FKIndex1(NETWORKID, NODEID, READERID),
  FOREIGN KEY(NETWORKID, NODEID, READERID)
    REFERENCES READERS(NETWORKID, NODEID, READERID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
);

CREATE TABLE CATERRORES (
  ERRORID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  SEVERIDADID INTEGER UNSIGNED NOT NULL,
  DESCRIPCION VARCHAR(20) NULL,
  DETALLES VARCHAR(255) NULL,
  PRIMARY KEY(ERRORID, SEVERIDADID),
  INDEX CATERRORES_FKIndex1(SEVERIDADID),
  FOREIGN KEY(SEVERIDADID)
    REFERENCES SEVERIDAD(SEVERIDADID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
)
TYPE=InnoDB
AUTO_INCREMENT = 1;


CREATE TABLE MODELO_TAGS (
  MODELO_ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  MARCA_ID INTEGER UNSIGNED NOT NULL,
  DESCRIPCION VARCHAR(80) NULL,
  PRIMARY KEY(MODELO_ID, MARCA_ID),
  INDEX MODELO_TAGS_FKIndex1(MARCA_ID),
```

```
  FOREIGN KEY(MARCA_ID)
    REFERENCES MARCA_TAGS(MARCA_ID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
)
TYPE=InnoDB;

CREATE TABLE ERRORES_SYS (
  ERRORID INTEGER UNSIGNED NOT NULL,
  SEVERIDADID INTEGER UNSIGNED NOT NULL,
  FECHAERR DATETIME NULL,
  INDEX ERRORES_FKIndex1(ERRORID, SEVERIDADID),
  FOREIGN KEY(ERRORID, SEVERIDADID)
    REFERENCES CATERRORES(ERRORID, SEVERIDADID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
)
TYPE=InnoDB
AUTO_INCREMENT = 1;

CREATE TABLE CIUDADES (
  CIUDAD_ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  ESTADO_ID INTEGER UNSIGNED NOT NULL,
  DESCRIPCION VARCHAR(45) NULL,
  PRIMARY KEY(CIUDAD_ID, ESTADO_ID),
  INDEX CIUDADES_FKIndex1(ESTADO_ID),
  FOREIGN KEY(ESTADO_ID)
    REFERENCES ESTADOS(ESTADO_ID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
)
TYPE=InnoDB;

CREATE TABLE DATOSENV (
  NETWORKID TINYINT UNSIGNED NOT NULL,
  NODEID TINYINT UNSIGNED NOT NULL,
  READERID TINYINT UNSIGNED NOT NULL,
  COMMAND VARCHAR(20) NOT NULL,
  FECHAENV DATETIME NOT NULL,
  PRIMARY KEY(NETWORKID, NODEID, READERID, COMMAND, FECHAENV),
  INDEX DATOSENV_FKIndex1(NETWORKID, NODEID, READERID),
  FOREIGN KEY(NETWORKID, NODEID, READERID)
    REFERENCES READERS(NETWORKID, NODEID, READERID)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
)
TYPE=InnoDB;

CREATE TABLE DATOSREC (
  NETWORKID TINYINT UNSIGNED NOT NULL,
  NODEID TINYINT UNSIGNED NOT NULL,
  READERID TINYINT UNSIGNED NOT NULL,
  COMMAND VARCHAR(20) NOT NULL,
  FECHAENV DATETIME NOT NULL,
  RESPONSE VARCHAR(255) NOT NULL,
  FECHAREC DATETIME NOT NULL,
  PRIMARY KEY(NETWORKID, NODEID, READERID, COMMAND, FECHAENV, RESPONSE, FECHAREC),
  INDEX DATOSREC_FKIndex1(NETWORKID, NODEID, READERID),
  INDEX DATOSREC_FKIndex2(NETWORKID, NODEID, READERID, COMMAND, FECHAENV),
  FOREIGN KEY(NETWORKID, NODEID, READERID)
    REFERENCES READERS(NETWORKID, NODEID, READERID)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(NETWORKID, NODEID, READERID, COMMAND, FECHAENV)
    REFERENCES DATOSENV(NETWORKID, NODEID, READERID, COMMAND, FECHAENV)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
```

```
)
TYPE=InnoDB;

CREATE TABLE USUARIOS (
  USUARIOID INT(9) UNSIGNED NOT NULL,
  ROLID INTEGER UNSIGNED NOT NULL,
  PUESTO_ID INTEGER UNSIGNED NOT NULL,
  DEPENDENCIA_ID INTEGER UNSIGNED NOT NULL,
  DIVISION_ID INTEGER UNSIGNED NOT NULL,
  ESTADO_ID INTEGER UNSIGNED NOT NULL,
  CIUDAD_ID INTEGER UNSIGNED NOT NULL,
  NOMBRE VARCHAR(45) NOT NULL,
  DIRECCION VARCHAR(80) NOT NULL,
  TELEFONO VARCHAR(15) NULL,
  MOVIL VARCHAR(15) NULL,
  FECHAIN DATETIME NOT NULL,
  LOGIN CHAR(16) NOT NULL,
  PASSWD CHAR(41) NOT NULL,
  PRIMARY KEY(USUARIOID, ROLID, PUESTO_ID, DEPENDENCIA_ID, DIVISION_ID, ESTADO_ID, CIUDAD_ID),
  INDEX USUARIOS_FKIndex1(ROLID),
  INDEX USUARIOS_FKIndex2(CIUDAD_ID, ESTADO_ID),
  INDEX USUARIOS_FKIndex3(PUESTO_ID, DEPENDENCIA_ID),
  INDEX USUARIOS_FKIndex4(DIVISION_ID),
  FOREIGN KEY(ROLID)
    REFERENCES ROLES(ROLID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
  FOREIGN KEY(CIUDAD_ID, ESTADO_ID)
    REFERENCES CIUDADES(CIUDAD_ID, ESTADO_ID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
  FOREIGN KEY(PUESTO_ID, DEPENDENCIA_ID)
    REFERENCES PUESTOS(PUESTO_ID, DEPENDENCIA_ID)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
  FOREIGN KEY(DIVISION_ID)
    REFERENCES DIVISIONES(DIVISION_ID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
)
TYPE=InnoDB;

CREATE TABLE TAGS (
  TAGID INTEGER UNSIGNED NOT NULL,
  MARCA_ID INTEGER UNSIGNED NOT NULL,
  MODELO_ID INTEGER UNSIGNED NOT NULL,
  ACTIVACION ENUM('E','D') NOT NULL,
  FECHA_ACTIVACION DATETIME NOT NULL,
  PRIMARY KEY(TAGID, MARCA_ID, MODELO_ID),
  INDEX TAGS_FKIndex1(MODELO_ID, MARCA_ID),
  FOREIGN KEY(MODELO_ID, MARCA_ID)
    REFERENCES MODELO_TAGS(MODELO_ID, MARCA_ID)
      ON DELETE CASCADE
      ON UPDATE CASCADE
)
TYPE=InnoDB;


CREATE TABLE ACCESOSYS (
  ACCESO_SYSID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  USUARIOID INT(9) UNSIGNED NOT NULL,
  ROLID SMALLINT UNSIGNED NOT NULL,
  PUESTO_ID INTEGER UNSIGNED NOT NULL,
  DEPENDENCIA_ID INTEGER UNSIGNED NOT NULL,
  DIVISION_ID INTEGER UNSIGNED NOT NULL,
  ESTADO_ID INTEGER UNSIGNED NOT NULL,
  CIUDAD_ID INTEGER UNSIGNED NOT NULL,
```

```
    FECHA_ACCESO DATETIME NOT NULL,
    PRIMARY KEY(ACCESO_SYSID, USUARIOID, ROLID, PUESTO_ID, DEPENDENCIA_ID, DIVISION_ID, ESTADO_ID,
CIUDAD_ID),
    INDEX ACCESOSYS_FKIndex1(USUARIOID, ROLID, ESTADO_ID, CIUDAD_ID, DEPENDENCIA_ID, PUESTO_ID,
DIVISION_ID),
    FOREIGN KEY(USUARIOID, ROLID, ESTADO_ID, CIUDAD_ID, DEPENDENCIA_ID, PUESTO_ID, DIVISION_ID)
      REFERENCES USUARIOS(USUARIOID, ROLID, ESTADO_ID, CIUDAD_ID, DEPENDENCIA_ID, PUESTO_ID,
DIVISION_ID)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
  )
  TYPE=InnoDB;

  CREATE TABLE EQUIPOTAG (
    NUMEROID INTEGER UNSIGNED NOT NULL,
    TAGID INTEGER UNSIGNED NOT NULL,
    MODELO_ID INTEGER UNSIGNED NOT NULL,
    MARCA_ID INTEGER UNSIGNED NOT NULL,
    MARCAEQUIPO VARCHAR(45) NOT NULL,
    MODELOEQUIPO VARCHAR(25) NOT NULL,
    DESCRIPCION VARCHAR(25) NULL,
    DIVISION VARCHAR(25) NOT NULL,
    DIRECCION VARCHAR(80) NOT NULL,
    FECHAIN DATETIME NOT NULL,
    PRIMARY KEY(NUMEROID, TAGID, MODELO_ID, MARCA_ID),
    INDEX EQUIPOTAG_FKIndex1(TAGID, MARCA_ID, MODELO_ID),
    FOREIGN KEY(TAGID, MARCA_ID, MODELO_ID)
      REFERENCES TAGS(TAGID, MARCA_ID, MODELO_ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
  )
  TYPE=InnoDB;

  CREATE TABLE DATOSREGTAG (
    NETWORKID TINYINT UNSIGNED NOT NULL,
    NODEID TINYINT UNSIGNED NOT NULL,
    READERID TINYINT UNSIGNED NOT NULL,
    COMMAND VARCHAR(20) NOT NULL,
    FECHAENV DATETIME NOT NULL,
    RESPONSE VARCHAR(255) NOT NULL,
    FECHAREC DATETIME NOT NULL,
    TAG_DATA CHAR(4) NOT NULL,
    FECHARECTAG DATETIME NOT NULL,
    PRIMARY KEY(NETWORKID, NODEID, READERID, COMMAND, FECHAENV, RESPONSE, FECHAREC),
    INDEX DATOSREGTAG_FKIndex1(NETWORKID, NODEID, READERID, RESPONSE, FECHAREC, COMMAND,
FECHAENV),
    FOREIGN KEY(NETWORKID, NODEID, READERID, RESPONSE, FECHAREC, COMMAND, FECHAENV)
      REFERENCES DATOSREC(NETWORKID, NODEID, READERID, RESPONSE, FECHAREC, COMMAND,
FECHAENV)
        ON DELETE CASCADE
        ON UPDATE CASCADE
  )
  TYPE=InnoDB;


  CREATE TABLE USUARIOTAG (
    USUARIOID INT(9)) UNSIGNED NOT NULL,
    TAGID INTEGER UNSIGNED NOT NULL,
    MODELO_ID INTEGER UNSIGNED NOT NULL,
    MARCA_ID INTEGER UNSIGNED NOT NULL,
    PUESTO_ID INTEGER UNSIGNED NOT NULL,
    DEPENDENCIA_ID INTEGER UNSIGNED NOT NULL,
    DIVISION_ID INTEGER UNSIGNED NOT NULL,
    ESTADO_ID INTEGER UNSIGNED NOT NULL,
    CIUDAD_ID INTEGER UNSIGNED NOT NULL,
    NOMBRE VARCHAR(45) NOT NULL,
    DIRECCION VARCHAR(80) NOT NULL,
```

```
    TELEFONO VARCHAR(15) NULL,
    MOVIL VARCHAR(15) NULL,
    FECHAIN DATETIME NOT NULL,
    PRIMARY KEY(USUARIOID, TAGID, MODELO_ID, MARCA_ID, PUESTO_ID, DEPENDENCIA_ID, DIVISION_ID,
ESTADO_ID, CIUDAD_ID),
    INDEX USUARIOTAG_FKIndex1(TAGID, MARCA_ID, MODELO_ID),
    INDEX USUARIOTAG_FKIndex2(PUESTO_ID, DEPENDENCIA_ID),
    INDEX USUARIOTAG_FKIndex3(DIVISION_ID),
    INDEX USUARIOTAG_FKIndex4(CIUDAD_ID, ESTADO_ID),
    FOREIGN KEY(TAGID, MARCA_ID, MODELO_ID)
      REFERENCES TAGS(TAGID, MARCA_ID, MODELO_ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY(PUESTO_ID, DEPENDENCIA_ID)
      REFERENCES PUESTOS(PUESTO_ID, DEPENDENCIA_ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY(DIVISION_ID)
      REFERENCES DIVISIONES(DIVISION_ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY(CIUDAD_ID, ESTADO_ID)
      REFERENCES CIUDADES(CIUDAD_ID, ESTADO_ID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
  )
  TYPE=InnoDB;


  CREATE TABLE ACCESOTAG (
    NETWORKID TINYINT UNSIGNED NOT NULL,
    NODEID TINYINT UNSIGNED NOT NULL,
    READERID TINYINT UNSIGNED NOT NULL,
    USUARIOID INT(9)) UNSIGNED NOT NULL,
    TAGID INTEGER UNSIGNED NOT NULL,
    MARCA_ID INTEGER UNSIGNED NOT NULL,
    MODELO_ID INTEGER UNSIGNED NOT NULL,
    PUESTO_ID INTEGER UNSIGNED NOT NULL,
    DEPENDENCIA_ID INTEGER UNSIGNED NOT NULL,
    DIVISION_ID INTEGER UNSIGNED NOT NULL,
    ESTADO_ID INTEGER UNSIGNED NOT NULL,
    CIUDAD_ID INTEGER UNSIGNED NOT NULL,
    FECHA_ACCESO DATETIME NOT NULL,
    DETALLES VARCHAR(45) NULL,
    PRIMARY KEY(NETWORKID, NODEID, READERID, USUARIOID, TAGID, MARCA_ID, MODELO_ID, PUESTO_ID,
DEPENDENCIA_ID, DIVISION_ID, ESTADO_ID, CIUDAD_ID, FECHA_ACCESO),
    INDEX ACCESOTAG_FKIndex1(USUARIOID, TAGID, MODELO_ID, MARCA_ID, DEPENDENCIA_ID, PUESTO_ID,
DIVISION_ID, ESTADO_ID, CIUDAD_ID),
    INDEX ACCESOTAG_FKIndex2(NETWORKID, NODEID, READERID),
    FOREIGN KEY(USUARIOID, TAGID, MODELO_ID, MARCA_ID, DEPENDENCIA_ID, PUESTO_ID, DIVISION_ID,
ESTADO_ID, CIUDAD_ID)
      REFERENCES USUARIOTAG(USUARIOID, TAGID, MODELO_ID, MARCA_ID, DEPENDENCIA_ID, PUESTO_ID,
DIVISION_ID, ESTADO_ID, CIUDAD_ID)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION,
    FOREIGN KEY(NETWORKID, NODEID, READERID)
      REFERENCES READERS(NETWORKID, NODEID, READERID)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
  )
  TYPE=InnoDB;
```

Inserción de datos para el usuario administrador:

```
INSERT INTO DEPENDENCIAS (DEPENDENCIA_ID, DESCRIPCION ) VALUES('1','Torre Ingenieria');
INSERT INTO DEPENDENCIAS (DEPENDENCIA_ID, DESCRIPCION ) VALUES('2','Centro Diseño y Manufactura');

INSERT INTO DIVISIONES (DIVISION_ID, DESCRIPCION) VALUES('1', 'Division Ingenieria Electrica');
INSERT INTO DIVISIONES (DIVISION_ID, DESCRIPCION) VALUES('1', 'Division Ingenieria Mecanica');

INSERT INTO ESTADOS (ESTADO_ID, DESCRIPCION) VALUES('1','Distrito Federal');


INSERT INTO PUESTOS (PUESTO_ID, DEPENDENCIA_ID, DESCRIPCION) VALUES('1','1','Director');
INSERT INTO PUESTOS (PUESTO_ID, DEPENDENCIA_ID, DESCRIPCION) VALUES('2','1','Profesor');
INSERT INTO PUESTOS (PUESTO_ID, DEPENDENCIA_ID, DESCRIPCION) VALUES('3','1','Estudiante');


INSERT INTO ROLES (ROLID, DESCRIPCION) VALUES('1','administrador','Usuarios con permisos para administrar
el sistema de control');
INSERT INTO ROLES (ROLID, DESCRIPCION) VALUES('2','operador','Usuarios con permisos para controlar y
configurar la red de Lectores RFID');


INSERT INTO USUARIOS (USUARIOID, ROLID, PUESTO_ID, DEPENDENCIA_ID, DIVISION_ID, ESTADO_ID,
CIUDAD_ID, NOMBRE, DIRECCION, TELEFONO, MOVIL, FECHAIN, LOGIN, PASSWD)
VALUES('000000001','1','1','1','1','1','1','administrador','Ciudad
Universitaria','','','now()','administrador',PASSWORD("administrador"));

INSERT INTO USUARIOS (USUARIOID, ROLID, PUESTO_ID, DEPENDENCIA_ID, DIVISION_ID, ESTADO_ID,
CIUDAD_ID, NOMBRE, DIRECCION, TELEFONO, MOVIL, FECHAIN, LOGIN, PASSWD)
VALUES('097166207','1','3','2','1','1','1','Roberto Jesus Calva Garcia','Pergoleros
6','55734716','5513221080','now()','azulman',PASSWORD("electronico"));
```