UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Análisis y Diseño de un Sistema de Explotación de Datos aplicado a Vending Machines utilizando J2EE y J2ME

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN
P R E S E N T A

GUILLERMO RESÉNDIZ INFANZÓN

DIRECTOR DE TESIS: Dr. Luis Andrés Buzo de la Peña

MÉXICO, D.F. 2008

A la UNAM, por permitirme vivir la mejor etapa de mi vida.

A la Facultad de Ingeniería y a mis profesores, por enseñarme una nueva manera de ver el mundo.

A Mamá y Papá, por todo su apoyo, este trabajo lo dedico enteramente a ustedes.

A mis Amigos: Sandra y Esteban, por la ayuda incondicional durante todo este tiempo, gracias por todo.

A Softel y al Dr. Andres Buzo , por la oportunidad de desarrollar este trabajo.

A mi familia, por las pruebas impuestas, por enseñarme a ser responsable de mi persona desde el principio.

A ti porque no importando el tiempo ni el lugar, sigues ahí pacientemente viviendo y esperando.

Introducción

Hoy en día, con la implantación de nuevas tecnologías en Internet, existen muchísimas aplicaciones informáticas las cuales pueden ayudar enormemente en la vida diaria del ser humano. Una de estas posibilidades, son los sistemas de monitoreo, los cuales arrojan datos que para su mayor aprovechamiento son tratados por diversas aplicaciones, permitiéndonos aprovechar y medir la calidad con la que se pueden ofrecer diversos productos y servicios. Una de estas posibilidades es la orientada a las maquinas electrónicas expendedoras de productos de diversa índole, llamadas también Vending Machines. Estas maquinas, desde su nacimiento hace poco más de un siglo, han ido abriendo la brecha en lo que se refiere a las ventas sin intervención humana, haciendo mucho más eficientes las mismas y sobre todo, gracias a la variedad de tipos que existen en el mercado de estas maquinas, se puede vender casi de todo, ampliando las posibilidades aun mas.

He aquí donde radica el problema a resolver, muchas de estas maquinas necesitan ser monitoreadas para obtener los mejores resultados en cuanto a desempeño de las mismas, todo esto en tiempo real, para que la productividad de estos aparatos aumente lo más posible y sobre todo, se aproveche al máximo los beneficios que nos ofrece esta tecnología. Las maquinas son monitoreadas hoy en día utilizando intervención humana , lo que arroja en pérdidas económicas importantes debido a la pérdida de tiempo que esto implica y sobre todo , a que en ocasiones las personas encargadas suelen hacer fraudes al momento de hacer corte en cada una de las Vending machines monitoreadas. A su vez, las empresas dedicadas a estas máquinas, no solamente necesitan tener acceso a la información de inventario, sino también a organizar las maquinas de manera que se puedan registrar responsables de las mismas.

Esto con ayuda de rutas de Vending Machines y datos pertinentes relevantes para cada persona asignada a una o más máquinas. Esto no solo aplicaría a grandes empresas, sino también a particulares que en un momento dado quisieran empezar en el negocio de las Vending Machines.

De aquí nace la necesidad de implementar un sistema totalmente automatizado para monitorear dichas maquinas a distancia, el cual se buscara sea un sistema Web basado en el estándar J2EE, el cual es ampliamente usado debido a su robustez cuando se implementan sistemas, así como también poder auditar las maquinas expendedoras remotamente con un cliente J2ME embebido en un modulo M2M. Para ello existe una tecnología denominada Telemetría, la cual es utilizada para medir magnitudes a grandes distancias mediante redes inalámbricas principalmente.

Esta tecnología es utilizada muy comúnmente en los transbordadores, satélites y demás artefactos utilizados en el espacio. Se echará mano de esta tecnología utilizando los llamados MODEM GSM, los cuales nos permiten acceder a Internet por medio de la red GPRs y así enviar los datos recolectados en los clientes (Vending Machines) al servidor central el cual los analizara y los ingresara a una base de datos relacional para así poder ser explotados de una manera mucho más ágil y precisa.

2.- Marco Teórico

Los sistemas inalámbricos se han caracterizado desde que surgieron, de la gran ventaja que nos ofrecen al poder enviar y recibir información casi desde cualquier punto con lo cual las posibilidades de desarrollo de sistemas crecen enormemente.

Existen muchas tecnologías aplicadas a las redes inalámbricas, de las cuales una de las más comunes hoy en día es la GPRS.

GPRS es la tecnología inalámbrica de datos en paquetes más ampliamente soportada en el mundo. Proporciona altas velocidades de transferencia de datos (especialmente útil para conectar a Internet) y se utiliza en las redes GSM.

La red GPRS que más se utiliza en México es Telcel, debido a la cobertura que nos ofrece y la disponibilidad de sus servicios. Se cobra a razón del volumen de datos enviados y recibidos. (Bytes) Los costos que se nos ofrecen por envío de información vía GPRS son

Paquete	KB	Costo	Precio por KB Incluido
Básico	1,000	$115.00	$0.11
Paquete 50	5,000	$230.00	$0.06
Paquete 100	10,000	$345.00	$0.05
Paquete 500	50,000	$575.00	$0.02

Llegando así a costos mensuales como los siguientes:

MB / Mes	*Costo*
15 MB	$300
40 MB	$600
3MB	$200
1MB	$70

Si hacemos un estimativo, cada terminal procesa alrededor de 20 Kb de información y si se tiene una flota de 1000 Vending Machines, el costo por transmisión GPRS sería de alrededor de 600 pesos mensuales con un nivel de holgura muy aceptable (el doble).

Por tal motivo, es posible ver hoy en día que muchos de los sistemas de monitoreo en México utilizan la tecnología GPRS como medio para enviar los datos que se recaudan en las tomas, para así poder ser analizados por una aplicación especializada en ello. Para este envío de datos, se utilizan lo que comúnmente se llama MODEM GSM.

Un MODEM GSM es un dispositivo el cual opera sobre redes gsm, Un MODEM GSM, al igual que un teléfono celular gsm, necesita para funcionar de una SIM card de una compañía telefónica que ofrezca los servicios de una red gsm .Al igual que los módems comunes, los módems gsm utilizan comandos AT para ser configurados, los cuales están ampliamente definidos en los estándares y mediante ellos podemos realizar configuraciones especificas a dichos módems.

En la actualidad existen bastantes proveedores de este tipo de módems, uno de los cuales es el denominado TC65 de la empresa Siemens que incorpora para su mejor desempeño, el lenguaje de programación java 2 MicroEdition para un desarrollo de aplicaciones más robusto.

Al hablar de aplicaciones de monitoreo que envían los datos a un servidor central de procesamiento para que este interprete los mismos mediante ciertos algoritmos, podemos ver que existen un sin fin de tipos de aplicaciones de las que pudiéramos echar mano para la correcta manipulación y presentación de los mismos al usuario final. Entre ellas podemos destacar las aplicaciones Web, que en los últimos tiempos han tomando verdadera fuerza por la flexibilidad que estas presentan.

Para la construcción de dichas aplicaciones , se presentan un sin fin de metodologías y lenguajes , de los cuales tenemos , solo por citar algunos de los ejemplos más comunes , php , perl , cgis , coldfusion , java , etc. .

De las tecnologías antes mencionadas, una de las más importantes debido al crecimiento presentado en los últimos años, es la que respecta a java, más específicamente en su rama Java 2 Enterprise Edition. Dicha división , presenta claramente un estándar para desarrollar aplicaciones web con el lenguaje orientado a objetos Java , de los cuales para su correcto uso se presenta una metodología base para el desarrollo de las mismas , la cual es denominada MVC (Modelo Vista Controlador) el cual es ampliamente usado en las aplicaciones web de los últimos años. En dicho modelo, se hace hincapié en las 3 partes principales de las que consta un sistema web de última generación. Un Modelo, el cual es el que encapsula toda la lógica del negocio, el cual se representa en términos de lenguaje java por POJO (Plain Old Java Object). Un Controlador , el cual se representa en su más sencilla expresión por un Servlet , el cual decide a quien relega las tareas y reenvía las peticiones y por ultimo una Vista , la cual se representa por un jsp (java Server Pages) la cual sirve únicamente para desplegar y/o pedir datos al usuario.

Esta es la base principal de los sistemas web que se realizan en este momento, los cuales tienen una serie de modificaciones sin salirse del mismo MVC, como por ejemplo los tan famosos frameworks de desarrollo como lo son Struts, Java Server Faces o para la persistencia de datos, los llamados ORMS de los cuales podemos citar a Hibernate e iBates solo por citar a los más utilizados.

Gracias al amplio desarrollo de esta tecnología, muchos desarrolladores se inician en el desarrollo no solamente de aplicaciones basadas en las reglas anteriormente propuestas, sino también en el desarrollo de frameworks los cuales hereden las mismas bases pero con la misma meta: mejorar el desarrollo del software y la implantación de sistemas que puedan tener un mejor mantenimiento con miras a ser ampliados a corto y/o a largo plazo.

2.1.- Descripción del protocolo de control de inventario usado en las Vending Machines (DEX)

Las vending machines, tan comunes para nosotros hoy en día, datan del finales del siglo XVIII, en la época en la que se empezaron a introducir maquinas expendedoras de goma de mascar en los Estados Unidos por Thomas Adams Gum Company. Desde aquellas épocas, estas maquinas han experimentado una serie de cambios tecnológicos para prevenir el robo y las descompostura, así como también el poder auditar las ventas y la cantidad de los productos existentes en las maquinas en cualquier momento.

Una de las grandes innovaciones es el uso de estándares para auditar las vending machines y conocer el estado actual mediante un protocolo de comunicaciones. Para ello, se introdujo un protocolo el cual se considera un estándar para las vending machines en América, el cual es llamado DEX.

Las siglas DEX significan Digital Exchange. Este estándar fue creado por la industria de Vending Machines en los años 90 para permitir a las mismas comunicar la información de las transacciones que ellas realizaran a computadoras portátiles. Estas transacciones, por mencionar unos ejemplos son:

- Medidor de ventas (Cash Meter Sales)
- Vend Cont by column (Ventas por selección)
- Price set for each column (Establecer un precio para cada selección)
- Value of Coin in the Coins Box (Monedas disponibles en el monedero)
- Value of Bills in the bill stacker (Billetes disponibles)

.

Este protocolo define un mecanismo para comunicar 2 maquinas para el intercambio de información. Entre las características que este presenta, están las siguientes:

Modo de intercambio de información: Este estándar está diseñado para el intercambio de datos entre dos dispositivos, en el cual se envían de uno a otro en forma de paquetes de información. Los paquetes son enviados, después de llevarse a cabo los 2 HandShake de inicialización, en bloques hasta que no exista más información que enviar y de esta manera la sesión de intercambio termina

Formato de envío de datos: Los datos son enviados asíncronamente, a 9600 bits/segundo y por bytes. Los diseñadores de este estándar, eligieron el envío de información asíncrona puesto que algunos dispositivos no están preparados para enviar información de forma sincronizada.

Bloques de información: Los mensajes son enviados en forma de bloques de longitud variable, el cual no puede ser mayor a los 245 bytes cada uno.

2.2 Proceso de Intercambio de Información utilizando el protocolo DEX

El proceso para iniciar una transmisión de información utilizando el protocolo DEX se divide en 3 partes principales

1. Primer HandShake
2. Segundo HandShake
3. Envío de la Información o tercer HandShake

- **Primer HandShake**

Esta se inicia asumiendo que se está conectado de manera física al dispositivo y se asume el status de maestro. Este status se consigue enviando un código ENQ y se espera la respuesta del esclavo de que se está preparado para iniciar el primer handshake. Inmediatamente después de esta respuesta, la parte que ha tomado el status de maestro, envía la siguiente información en bloque:

- El Comm Id del maestro, usado para identificar esta transmisión.
- Un tipo de operación requerida, usada para definir la operación que se desea realizar (envío o recepción de información)
- Los parámetros de revisión y nivel del estándar que es capaz el maestro de procesar (En este momento, solo existe un solo valor para estos parámetros).

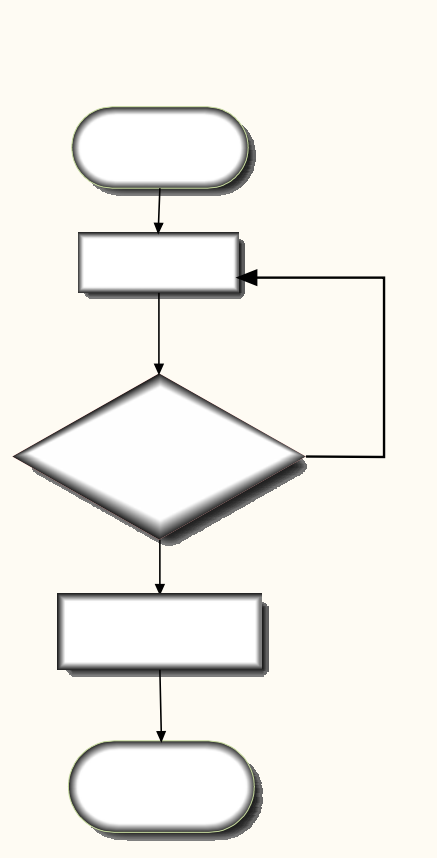


Figura 2.1 Primer Handshake Dex

- **Segundo HandShake**

Para el segundo handshake, el dispositivo que asumió el status de esclavo inicia la comunicación enviando un código ENQ, al cual el maestro responde con un código DLE0 en el cual le avisa que está listo para recibir los siguientes parámetros:

- Un código que indica si el sistema está listo para atender la petición requerida en el primer handshake o el código de error o razón por la que no se puede atender la petición.
- El Comm Id del dispositivo
- Los parámetros de revisión y nivel del estándar que es capaz de manejar el dispositivo (En este momento solamente se puede enviar el código R01L01)

- Un código DEL ETX y el CRC del mensaje completo.

Después del envío de estos parámetros, el dispositivo con status de maestro le responde un código DLE1 con el cual le indica que está listo para el tercer y último paso, que sería el tipo de petición que requirió, a lo que el esclavo le responde un código EOT.

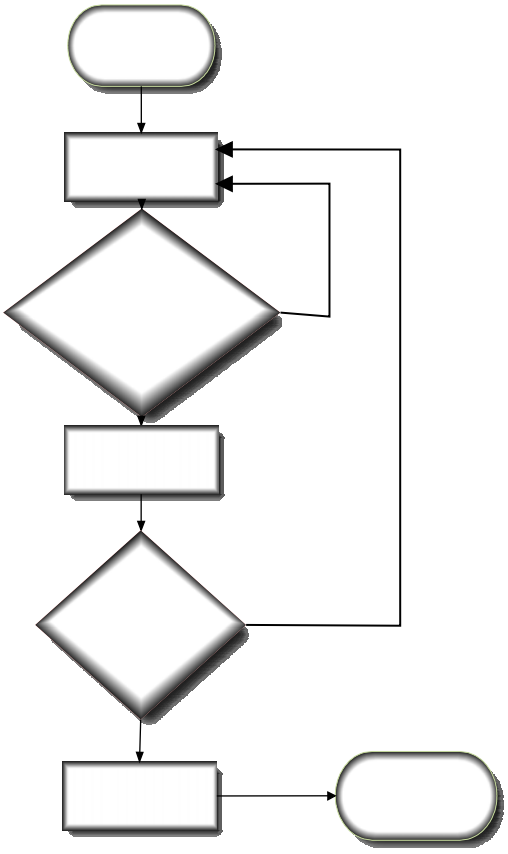


Figura 2.2 Segundo Handshake Dex

- **Última Sesión o Tercer Handshake.**

El dispositivo al que se le requirió una petición asume el status de maestro, enviando un código ENQ avisando que está listo para la realizar la acción solicitada. El dispositivo con status esclavo responde un código DLE0 indicando que está listo con lo cual se iniciara la transmisión del mensaje requerido, el cual vendrá en paquetes los cuales vendrán delimitados por los códigos DXS y DXE (DX Start, DX End) y el CRC-16. Entre cada paquete, el dispositivo que esté recibiendo estará emitiendo un código DLE1 y DLE0 alternadamente, hasta que encuentre el final del mensaje, que será en el momento en el que el dispositivo emisor anuncie el último paquete con un código. En este momento el receptor enviara el código DLE correspondiente, así como también el código EOT.

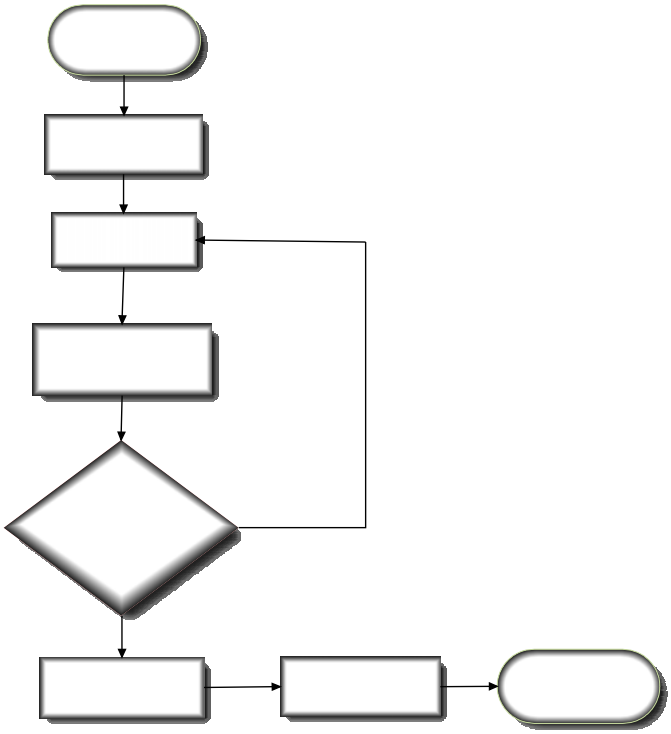


Figura 2.3 Tercer Handshake Dex

2.3- La telemetría y los sistemas m2m

La palabra *telemetría* procede de las palabras griegas *tele* ("lejos") y *metron* ("medida"). La **telemetría** es una tecnología que permite la medición remota de magnitudes físicas y el posterior envío de la información hacia el operador del sistema. En otras palabras, es un conjunto de procedimientos para medir magnitudes físicas y químicas desde una posición distante al lugar donde se producen los fenómenos.

La telemetría es actualmente una industria en rápido crecimiento. El desarrollo Tecnológico de estos últimos años está habilitando numerosas posibilidades para el monitoreo y el control de máquinas y procesos en prácticamente cualquier segmento Industrial, productivo y de servicios.

El envío de información hacia el operador en un sistema de telemetría se realiza típicamente mediante comunicación inalámbrica, aunque también se puede realizar por otros medios.

La telemetría se utiliza en grandes sistemas, tales como las naves espaciales o las plantas químicas, debido a que facilita la monitorización automática y el registro de las mediciones, así como el envío de alertas, con el fin de que el funcionamiento sea seguro y eficiente

En el pasado, la telemetría fue utilizada única y exclusivamente por grandes empresas financiadas por grandes organizaciones. La NASA es un ejemplo de esto, ha utilizado la telemetría desde los inicios de sus programas espaciales, para operar con naves espaciales y satélites
Otras de las principales aplicaciones de la telemetría es la meteorología. Los equipos instalados en sondas y globos meteorológicos permiten obtener mediadas de las capas altas de la atmósfera y realizar mapas que ayudan a predecir el clima.

Si bien existen distintas opciones tecnológicas para implementar un sistema remoto , ha sido sin duda la estandarización y el éxito a nivel mundial de las redes de comunicaciones GSM/GPRS lo que nos ha permitido disponer de una nueva generación de sistemas capaces de dar solución de una manera eficaz a nuestras necesidades de automatización y control en todos los ámbitos y, más concretamente, en aquellos procesos dispersos en extensión geográfica, ya sea a nivel local, nacional o transnacional.

Hoy por hoy nos hallamos en un escenario donde la mayor parte de los actores se han puesto de acuerdo: los operadores móviles —interesados en optimizar el tráfico de sus redes con los servicios de datos—, los fabricantes de equipos industriales y de teléfonos móviles —en los cuales el término M2M ya está encabezando líneas de producto, cada vez con mayor peso específico—, las ingenierías, integradores y desarrolladores —que ven en GPRS un estándar de comunicaciones potente y práctico— y finalmente los usuarios —que estaban a la espera de la evolución, estandarización y comercialización de un sistema competitivo en precio y prestaciones.

M2M (machine to machine, máquina a máquina) es una abreviatura para máquina a máquina, o una tecnología que soporta la comunicación alambica o inalámbrica entre máquinas. Es comúnmente traducida como Machine – To – Machine pero algunas veces es traducida como Man – to – Machine, Machine – to – Man, Machine – to – Mobile, Mobile – to Machine y generalmente se utiliza en la Telemetría.

El campo de aplicación de las soluciones M2M es heterogéneo aunque presentan una serie de componentes básicos comunes en la comunicación entre la plataforma y las máquinas remotas

ADQUISICION DE DATOS. Los datos se recogen a través de sensores y son enviados en la plataforma que reside en el centro de control o a los sistemas de gestión de las empresas.

CONTROL REMOTO. Se basa en la realización de una serie de operaciones sobre la máquina remota.

CONFIGURACIÓN REMOTA. Consiste en el cambio de la programación o de los modos de funcionamiento de las máquinas controladas.

SUPERVISIÓN. La supervisión es necesaria en los sistemas con capacidad para detectar situaciones de alarma. En estos casos, la información que se transmite es crítica.

PRESENTACION DE DATOS. Los datos puedan ser visualizados a través de diversos dispositivos o sistemas.

.

3. Justificación de las Tecnologías a utilizar.

3.1.- El modulo TC65 de Siemens como una opción viable para sistemas m2m.

La diferencia de un modem ordinario con un Modem GSM es que el Modem GSM permite el ejecutar aplicaciones de terceros eliminando la necesidad de controladores externos. Este tipo de módems, tienen implementado también la posibilidad de una conexión óptima a Internet, haciendo estos módems muy útiles en el desarrollo de aplicaciones.

Este tipo de módulos están disponibles a través de muchísimos proveedores, permitiéndonos así elegir el que mejor se ajuste a nuestras necesidades.

En la actualidad existen muchos módulos que se utilizan comúnmente en las aplicaciones m2m. Como en cualquier rama de tecnología, estos ofrecen distintas posibilidades no solamente tecnológicas, si no también basados en el aspecto económico.

En el campo de la telemetría para Vending machines, existen muchas opciones de las cual podemos echar mano para desarrollar una aplicación m2m. De esas opciones, existen un par las cuales son consideradas las más importantes debido al respaldo de las empresas de telecomunicaciones que las crearon y sobre todo de la robustez que ofrecen, las cuales son:

Sony Ericsson GA64

La solución de Sony Ericsson, es un modulo GSM con las siguientes características:

- Lenguaje de Programación para desarrollo de aplicaciones: C
- Memoria de datos para software embebido: 256 Kb
- Protocolos soportados: TCP, UDP, FTP
- Interfaces: 2x RS232, USB, SPI, 2x GPIO,
- Supply voltage: 5 V
- GSM band: 850/900/1800/1900
- GPRS Class: 10

Software

Para el desarrollo de aplicaciones, la solución de Sony ofrece un intérprete de scripts de lenguaje C, el cual no lee sentencias de lenguaje C estándar, todas deben ser portadas a su lenguaje propietario. El modulo puede guardar a lo mas dos scripts, de los cuales uno solo debe estar activo así como también, la RAM interna del equipo (1 Kb) es compartida

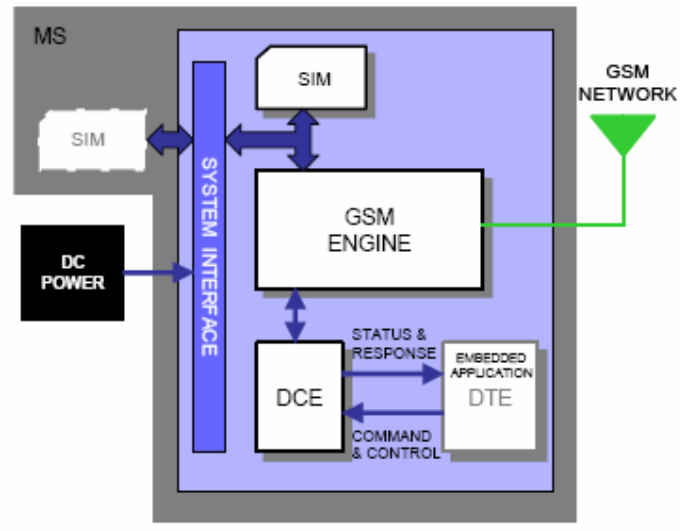


Figura 3.1 Diagrama Sony Ericsson

Siemens TC65

El **TC65** permite a los desarrolladores trabajar con una sofisticada plataforma de software. Este módulo GSM/GPRS cuatri-banda con gran variedad de interfaces, puede ser integrado virtualmente en cualquier aplicación M2M (machine-to-machine).

Con el fin de disminuir el tiempo de desarrollo, el soporte JAVA del módulo **TC65** reduce costos de implementación, ya que no es necesario utilizar software propietario para desarrollar, gracias al plugin para Eclipse (IDE de desarrollo OpenSource).

Las principales características del TC65 son:

- Soporte JAVA, IMP 2.0
- Procesador ARM7
- Memoria flash de 1,7 MB
- Memoria RAM de 400 KB
- Transferencia segura de datos vía OTA (Over the Air)
- Soporte para los protocolos TCP, UDP, HTTP, FTP, SMTP, POP3
- USB 2.0
- SIM card interface, 3 V, 1.8 V
- I2C bus y SPI bus
- 2 x analog in (ADC)
- 1 x analog out (PWM)
- 10 GPIOs

Hardware

La máquina de java y las aplicaciones embebidas corren gracias al procesador ARM7 del que está dotado el equipo.

Software

Este módulo tiene integrado el soporte para Java 2 MicroEdition, así como también Siemens añadió el API para Comandos AT, IO y manejo de su File System . El sistema de archivos puede ser visto a través de una herramienta que ellos proporcionan como una memoria flash cuando el módulo es conectado a la PC. GPIO, SPI e I2C son controlados con comandos AT a través del API implementados por Siemens. El flujo que el módulo tiene cuando se ejecuta una aplicación es el siguiente:

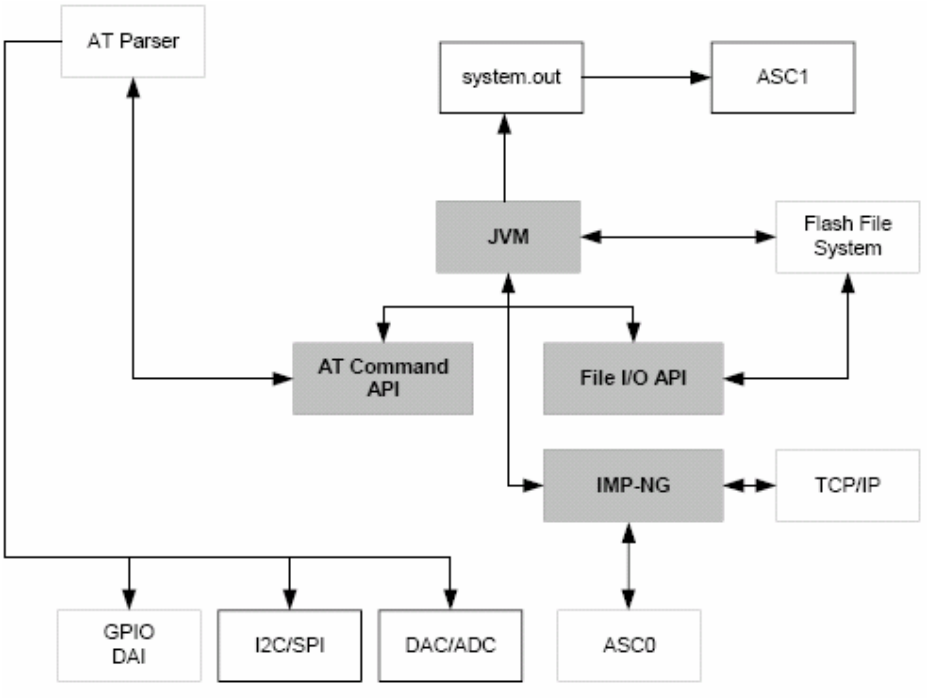


Figura 3.2 Diagrama Siemens TC65

El api para enviar comandos AT es como sigue:
atc.send (string)

Donde string es el comando at a ser enviado, atc el objeto de tipo ATCommand y el método send es el que sirve para ejecutar dicho comando.

La máquina virtual provista en este módulo permite la ejecución de múltiples threads y aparte el módulo tiene para su uso 400 KB de RAM para las aplicaciones.

Este modulo viene con un kit de desarrollo que puede ser agregado en los más comunes IDES de desarrollo actualmente utilizados, como Eclipse.

Por lo anterior, se puede deducir fácilmente que el modulo ofrecido por siemens, ofrece aparte de una plataforma independiente de desarrollo (esto es, se puede desarrollar bajo entornos open source utilizando eclipse y liberarnos de paga por los mismos) con un lenguaje muy extendido como lo es java, en su rama de j2me la cual es en la actualidad una de las más utilizadas para las aplicaciones de telefonía celular.

La opción ofrecida por Siemens nos resulta muy interesante, por lo que con esto se ve justificada la utilización del modulo, no solamente tecnológicamente si no también económicamente, al tener un precio bajo comparando que no se necesitara de comprar licencias para desarrollar el software necesario para que el modulo funcione.

3.2.- J2EE como alternativa para el diseño

Se ha hablado de las posibilidades que ofrece el modulo siemens con la utilización de J2ME para el desarrollo de la aplicación cliente, pero por ahora nos centraremos en el desarrollo de la aplicación servidor.

En épocas recientes, el auge de java ha crecido enormemente. Esto se debe a la gran contribución y apertura que tienen las empresas de tecnologías más importantes para el uso y crecimiento de esta tecnología, sin olvidar el desarrollo open source que es sin duda, uno de los motores por los que java cada día crece y nos ofrece la posibilidad de ser implementado como solución de muchas de las necesidades tecnológicas que se presentan en estos tiempos.

Una de estas ramas de java, es la llamada J2EE o Java 2 Enterprise Edition.la cual es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Java arquitectura de n niveles distribuida, basándose ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones , la cual está definida por una *especificación* que es también considerada informalmente como un estándar debido a que los suministradores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son *conformes a Java EE*.

Java EE incluye varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc., y define como coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, Servlets, portlets (siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios web. Esto permite al desarrollador crear una Aplicación Empresarial que es portable entre plataformas y escalable Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar las transacciones, seguridad, escalabilidad, concurrencia y gestión de los componentes que son desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de las tareas de mantenimiento de bajo nivel.

Otras ventajas con que cuenta J2EE para que se eligiera como pieza fundamental de este proyecto, son la escalabilidad, la posibilidad de tener la lógica del negocio independiente con los clientes, la alta disponibilidad de las aplicaciones y del acceso a las bases de datos, por destacar los más importantes.

Entre los principales beneficios que se han desencadenado con la aplicación de J2EE, encontramos:

- Enriquecimiento y robustecimiento de soluciones.
- Generación de componentes reutilizables dentro del ámbito de los distintos sistemas del proyecto, con lo cual se han reducido en un 50% los tiempos de desarrollo proyectado para funcionalidades similares en tres proyectos simultáneos.
- Disminución de tiempos de investigación y de elaboración de prototipos.
- Reducción de tiempos en la replicación del conocimiento.

Por lo cual , al ver el problema que nos hemos planteado desde el principio , J2EE nos ofrece una serie de características las cuales pueden ser aprovechadas para la construcción de un sistema robusto y de mantenimiento optimo , el cual pueda ser ampliado en sus capacidades de una manera rápida provechando las ventajas que nos ofrece dicho estándar.

4.- Análisis, Diseño e Implementación del Cliente J2ME en la Vending Machine.

4.1 Breve Introducción a J2ME

Sun Microsystems lanzó a mediados de los años 90 el lenguaje de programación Java que, aunque en un principio fue diseñado para generar aplicaciones que controlaran electrodomésticos como lavadoras, frigoríficos, etc., debido a su gran robustez e independencia de la plataforma donde se ejecutase el código, desde sus comienzos se utilizó para la creación de componentes interactivos integrados en páginas Web y programación de aplicaciones independientes. Estos componentes se denominaron applets y casi todo el trabajo de los programadores se dedicó al desarrollo de éstos. Con los años, Java ha progresado enormemente en varios ámbitos como servicios HTTP, servidores de aplicaciones, acceso a bases de datos (JDBC)... Como vemos Java se ha ido adaptando a las necesidades tanto de los usuarios como de las empresas ofreciendo soluciones y servicios tanto a unos como a otros. Debido a la explosión tecnológica de estos últimos años Java ha desarrollado soluciones personalizadas para cada ámbito tecnológico. Sun ha agrupado cada uno de esos ámbitos en una edición distinta de su lenguaje Java. Estas ediciones son Java 2 Estándar Edition, orientada al desarrollo de aplicaciones independientes y de applets, Java 2 Enterprise Edition, enfocada al entorno empresarial y Java 2 Micro Edition, orientada a la programación de aplicaciones para pequeños dispositivos.

Java 2 Platform, Micro Edition (J2ME): Esta versión de Java está enfocada a la aplicación de la tecnología Java en dispositivos electrónicos con capacidades computacionales y gráficas muy reducidas, tales como teléfonos móviles, PDAs o electrodomésticos inteligentes. Esta edición tiene unos componentes básicos que la diferencian de las otras versiones, como el uso de una máquina virtual denominada KVM (Kilo Virtual Machine, debido a que requiere sólo unos pocos Kilobytes de memoria para funcionar) en vez del uso de la JVM clásica, inclusión de un pequeño y rápido recolector de basura (Gargabe Collector)

Los componentes que conforman dicha tecnología son los siguientes:

- **Máquinas Virtuales** (Virtual Machines): Tenemos una serie de máquinas virtuales Java con diferentes requisitos, cada una para diferentes tipos de dispositivos.

- **Configuraciones,** que son un conjunto de clases básicas orientadas a conformar el corazón de las implementaciones para dispositivos de características específicas. Existen 2 configuraciones definidas en J2ME: Connected Limited Device Configuration (CLDC) enfocada a dispositivos con restricciones de procesamiento y memoria, y Connected Device Configuration (CDC) enfocada a dispositivos con más recursos.

- **Perfiles**, que son unas bibliotecas Java de clases específicas orientadas a Implementar funcionalidades de más alto nivel para familias específicas de dispositivos.

Un entorno de ejecución determinado de J2ME se compone entonces de una selección de:

a) Máquina virtual.
b) Configuración.
c) Perfil.
d) Paquetes Opcionales.

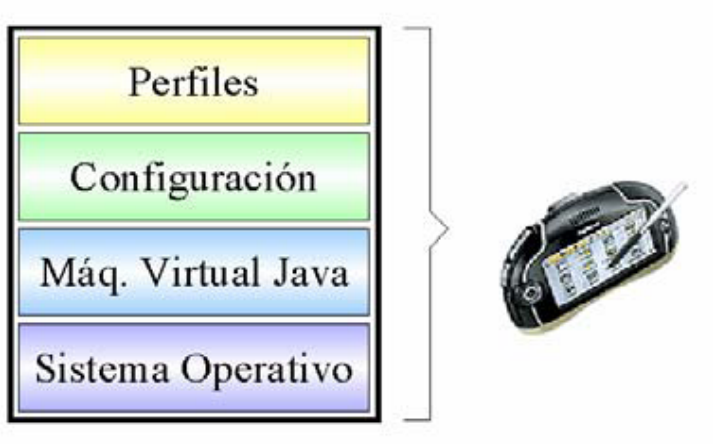


Figura 4.1 Diagrama Configuracion J2ME

.

Una máquina virtual de Java (JVM) es un programa encargado de interpretar código intermedio (bytecode) de los programas Java pre compilados a código máquina ejecutable por la plataforma, efectuar las llamadas pertinentes al sistema operativo subyacente y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java. De esta forma, la JVM proporciona al programa Java independencia de la plataforma con respecto al hardware y al sistema operativo subyacente. Las implementaciones tradicionales de JVM son, en general, muy pesadas en cuanto a memoria ocupada y requerimientos computacionales. J2ME define varias JVMs de referencia adecuadas al ámbito de los dispositivos electrónicos que, en algunos casos, suprimen algunas características con el fin de obtener una implementación menos exigente.

Existen 2 configuraciones CLDC y CDC, cada una con unas características propias que veremos en profundidad más adelante. Como consecuencia, cada una requiere su propia máquina virtual. La VM (Virtual Machine) de la configuración CLDC se denomina KVM y la de la configuración CDC se denomina CVM. Veremos a continuación las características principales de cada una de ellas:

KVM

Su nombre KVM proviene de Kilobyte (haciendo referencia a la baja ocupación de memoria, entre 40Kb y 80Kb). Se trata de una implementación de Máquina Virtual reducida y especialmente orientada a dispositivos con bajas capacidades computacionales y de memoria. La KVM está escrita en lenguaje C, aproximadamente unas 24000 líneas de código, y fue diseñada para ser:

- Pequeña, con una carga de memoria entre los 40Kb y los 80 Kb, dependiendo de la plataforma y las opciones de compilación.
- Alta portabilidad.
- Modulable.
- Lo más completa y rápida posible y sin sacrificar características para las que fue diseñada.

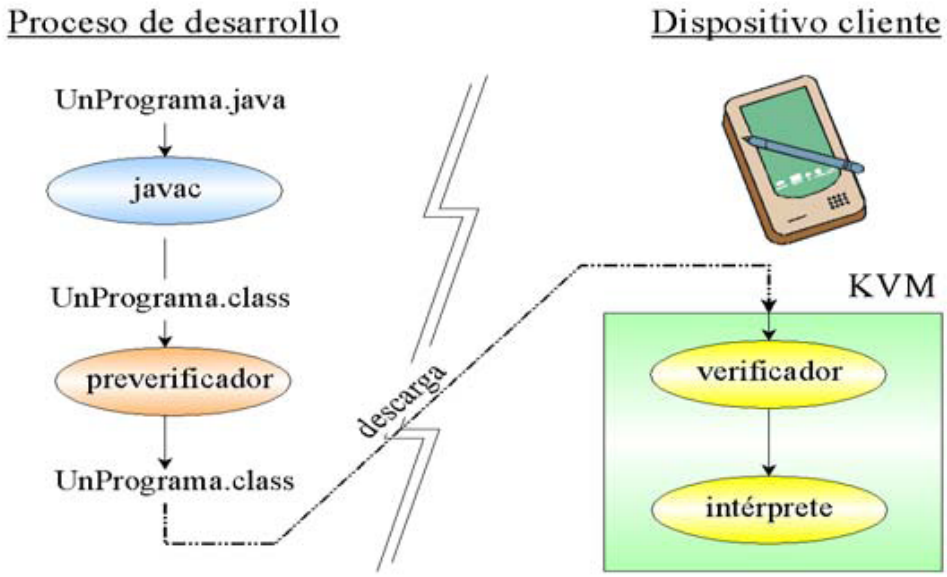


Figura 4.2 Diagrama Desarrollo J2ME

CVM

La CVM (Compact Virtual Machine) ha sido tomada como Máquina Virtual Java de referencia para la configuración CDC y soporta las mismas características que la Máquina Virtual de J2SE. Está orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y en torno a 2Mb o más de memoria RAM. Algunas de las características que presenta esta Máquina Virtual son:

1. Sistema de memoria avanzado.
2. Tiempo de espera bajo para el recolector de basura.
3. Separación completa de la VM del sistema de memoria.
4. Baja ocupación en memoria de las clases.
5. Librerías de seguridad, invocación remota de métodos (RMI), etc.

Configuraciones

Ya hemos mencionado algo anteriormente relacionado con las configuraciones. Para tenerlo bien claro diremos que una configuración es el conjunto mínimo de APIs Java que permiten desarrollar aplicaciones para un grupo de dispositivos. Éstas APIs describen las características básicas, comunes a todos los dispositivos:

- Características soportadas del lenguaje de programación Java.
- Características soportadas por la Máquina Virtual Java.
- Bibliotecas básicas de Java y APIs soportadas.

Como ya hemos visto con anterioridad, existen dos configuraciones en J2ME:

CLDC, orientada a dispositivos con limitaciones computacionales y de memoria y
CDC, orientada a dispositivos con no tantas limitaciones. Ahora veremos un poco más en profundidad cada una de estas configuraciones.

Configuración de dispositivos con conexión, CDC (*Connected Limited Configuration*)

La CDC está orientada a dispositivos con cierta capacidad computacional y de memoria. Por ejemplo, decodificadores de televisión digital, televisores con internet, algunos electrodomésticos y sistemas de navegación en automóviles. CDC usa una Máquina Virtual Java similar en sus características a una de J2SE, pero con limitaciones en el apartado gráfico y de memoria del dispositivo. Ésta Máquina Virtual es la que hemos visto como CVM (Compact Virtual Machine). La CDC está enfocada a dispositivos con las siguientes capacidades:

- Procesador de 32 bits.
- Disponer de 2 Mb o más de memoria total, incluyendo memoria RAM y ROM.
- Poseer la funcionalidad completa de la Máquina Virtual Java2.
- Conectividad a algún tipo de red.

La CDC incluye varios paquetes Java de la edición estándar. Las peculiaridades de la CDC están contenidas principalmente en el paquete javax.microedition.io, que incluye soporte para comunicaciones http y basadas en datagramas.

Configuración de dispositivos limitados con conexión, CLDC (*Connected Limited Device Configuration*).

La CLDC está orientada a dispositivos dotados de conexión y con limitaciones en cuanto a capacidad gráfica, cómputo y memoria. Un ejemplo de éstos dispositivos son: teléfonos móviles, buscapersonas (pagers), PDAs, organizadores personales, etc. CLDC está orientado a dispositivos con ciertas restricciones. Algunas de éstas restricciones vienen dadas por el uso de la KVM, necesaria al trabajar con la CLDC debido a su pequeño tamaño. Los dispositivos que usan CLDC deben cumplir los siguientes requisitos:

- Disponer entre 160 Kb y 512 Kb de memoria total disponible. Como mínimo se debe disponer de 128 Kb de memoria no volátil para la Máquina Virtual Java y las bibliotecas CLDC, y 32 Kb de memoria volátil para la Máquina Virtual en tiempo de ejecución.

- Procesador de 16 o 32 bits con al menos 25 Mhz de velocidad.

- Ofrecer bajo consumo, debido a que estos dispositivos trabajan con suministro de energía limitado, normalmente baterías.

- Tener conexión a algún tipo de red, normalmente sin cable, con conexión intermitente y ancho de banda limitado (unos 9600 bps).

CLDC aporta las siguientes funcionalidades en los dispositivos:

- Un subconjunto del lenguaje Java y todas las restricciones de su Máquina Virtual (KVM).

- Un subconjunto de las bibliotecas Java del núcleo.

- Soporte para E/S básica.

- Soporte para acceso a redes.

- Seguridad.

Perfiles

El perfil es el que define las APIs que controlan el ciclo de vida de la aplicación, interfaz de usuario, etc. Más concretamente, un perfil es un conjunto de APIs orientado a un ámbito de aplicación determinado. Los perfiles identifican un grupo de dispositivos por la funcionalidad que proporcionan (electrodomésticos, teléfonos móviles, etc.) y el tipo de aplicaciones que se ejecutarán en ellos. Las librerías de la interfaz gráfica son un componente muy importante en la definición de un perfil. Aquí nos podemos encontrar grandes diferencias entre interfaces, desde el menú textual de los teléfonos móviles hasta los táctiles de los PDAs.

El perfil establece unas APIs que definen las características de un dispositivo, mientras que la configuración hace lo propio con una familia de ellos. Esto hace que a la hora de construir una aplicación se cuente tanto con las APIs del perfil como de la configuración. Tenemos que tener en cuenta que un perfil siempre se construye sobre una configuración determinada. De este modo, podemos pensar en un perfil como un conjunto de APIs que dotan a una configuración de funcionalidad específica. Ya hemos visto los conceptos necesarios para entender cómo es un entorno de ejecución en Java Micro Edition. Anteriormente vimos que para una configuración determinada se usaba una Máquina Virtual Java específica. Teníamos que con la configuración CDC usábamos la CVM y que con la configuración CLDC usábamos la KVM. Con los perfiles ocurre lo mismo. Existen unos perfiles que construiremos sobre la configuración CDC y otros que construiremos sobre la CLDC. Para la configuración CDC tenemos los siguientes perfiles:

- *Foundation Profile*.
- *Personal Profile*.
- *RMI Profile*.

4.2 Descripción del problema.

Como se ha visto, para la auditoria de las Vending Machine, se necesita de clientes la capacidad de implementar el Protocolo DEX y además con la posibilidad de tener una administración local y remota haciendo uso de Sockets TCP y utilizando el puerto serial del módulo.

Para ello se propone, implementar 2 consolas de administración, una local y una remota. La local será servida por el puerto serial y la remota será servida por un socket tcp, en calidad de servidor. Ambas consolas tendrán la capacidad de servir comandos específicos para modificar la configuración del equipo en cuanto a parámetros de conexión (APN, usuario, password, puertos, etc.) así como también la posibilidad de guardar dicha configuración en un archivo de texto para guardar la configuración que se programe a través de dicha consola.

Para la parte de auditoría, se requiere la implementación de los 3 handshakes del protocolo DEX, además del manejo de archivos en el File System del módulo para el envío de las lecturas al servidor por medio de un socket TCP cliente. Lo anterior se describe en el siguiente diagrama de casos de usos:

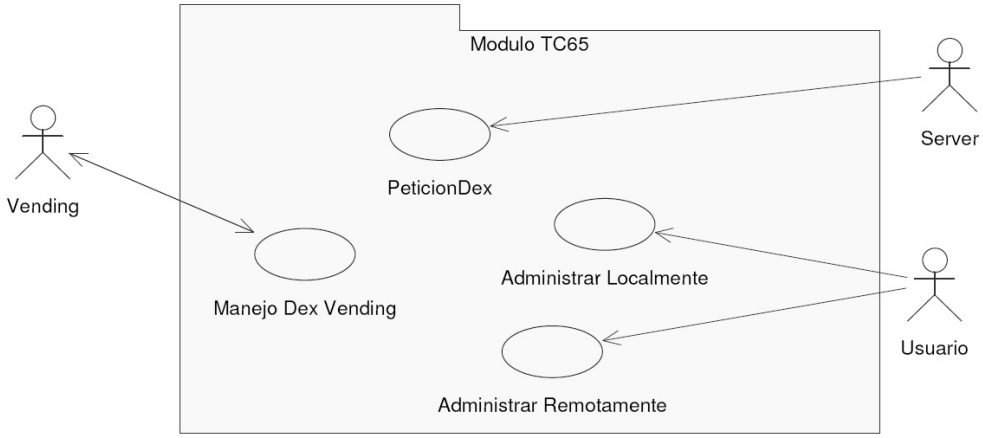


Imagen 4.3 Caso de Usos TC65

Con lo que vemos, en la figura anterior se muestran los componentes más importantes que tendrá cada uno de los clientes en esta etapa así como también las interacciones esperadas. Haciendo uso de UML, procederemos a modelar los componentes propuestos.

4.3 Planteamiento de la solución en la Implementación del Cliente J2ME

Como se detallo en el primer capítulo de este trabajo, se necesita de un cliente robusto, desarrollado bajo J2ME para poder ser implantado como un componente en el MODEM TC65 el cual nos permitiría poder acceder a la red GPRs, configurar parámetros especiales, acceder a la Vending Machine utilizando el protocolo DEX para auditarla y desde luego a transmitir la información obtenida al servidor central. A continuación se muestra un esquema genérico de lo que se busca de este cliente:

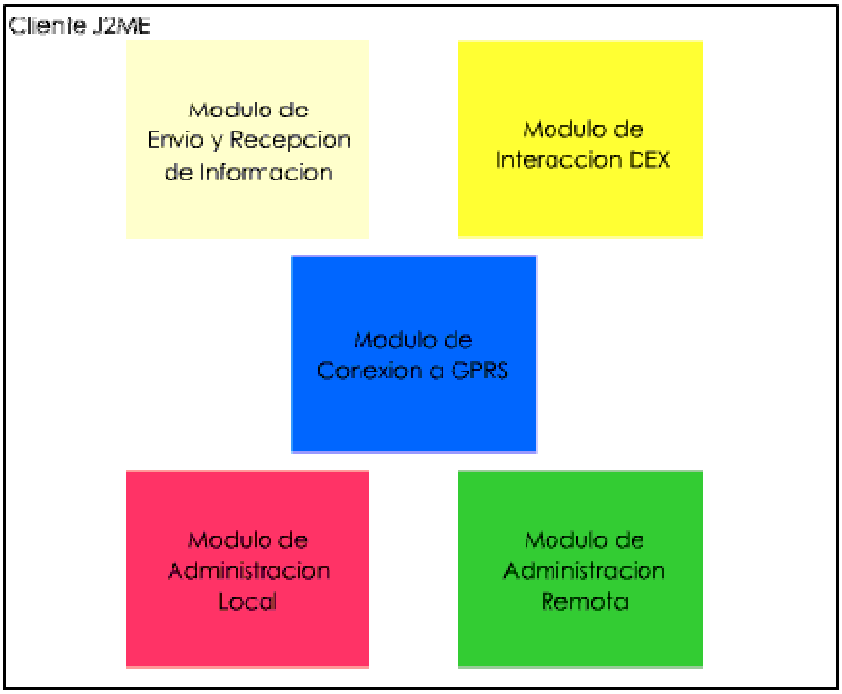


Figura 4.4 Diagrama Modulo J2ME

Como podemos ver en la figura anterior, se describen los módulos siguientes:

Modulo de Conexión a GPRS: Este modulo se necesita manejar las conexiones a la red GPRS del modulo siemens, se tiene que basar principalmente en dos puntos:

- Obtención de conexión con la red GPRs mediante parámetros previamente configurados : Estos parámetros son los necesarios para obtener una conexión a la red GPRs y son principalmente APN , usuario , password , y servidor los cuales pueden ser ingresados mediante un la consola de administración remota o local (detallada en los dos módulos de administración) o en el caso especifico de un archivo de configuración inicial , el cual se cargara en el File System del modulo y será leído por el modulo TC65 y se tomaran los valores asignados en este archivo como variables predefinidas.
- Obtención de conexión en caso de re conexión: Debido a que este modulo debe siempre estar conectado a la red GPRS, se necesita una rutina la cual este constantemente verificando que el modulo esté conectado a la red GPRS y si no lo está, tratar de volver a conectarse para poder estar siendo administrado remotamente el modulo y también poder enviar los datos de lecturas DEX de la Vending Machine.

Modulo de Administración Local: En este modulo se debe permitir la administración local del modulo, mediante comandos predefinidos ya por nosotros , entre los cuales se podrán modificar APN , usuario , password así como también el modo de inicialización del modulo. También se podrán leer variables propias de cuando el modulo se conecte a la red GPRS, como son potencia, ATTACH de SIM, etc., estos últimos mediante el uso de comandos AT propios del modulo siemens, los cuales podrán ser accedidos mediante el api que tiene el mismo. Toda esta administración local, se podrá realizar conectando un cable serial al puerto de la caja electrónica a la que estará conectado el modulo y mediante la cual a una velocidad de bps se podrá acceder a la consola.

Modulo de Administración Remota: En este modulo , se podrá realizar lo mismo que la administración local , pero de manera remota utilizando sockets TCP , los cuales pueden ser configurados en el cliente , para recibir peticiones utilizando un telnet normal y poder ingresar los comandos anteriormente descritos. Cabe destacar que este modulo puede ser configurable para aceptar una sola conexión concurrente.

Modulo de Interacción DEX: Este se espera sea el modulo principal de la aplicación, mediante el cual podremos tener interacción con la Vending machine implementando el protocolo DEX, el cual nos permitirá auditarla y guardar el archivo de la información obtenida en el file system del modem.
Modulo de envío y recepción de información. En este apartado, se requerirá de implementar un socket TCP mediante el cual se envié la información que se genere cuando se audite la Vending Machina. Se basara principalmente en un socket servidor que escuche las peticiones de lectura que le haga el Server y después de ocurrida la lectura, enviar el archivo obtenido, para que posteriormente sea parseado por el Server.

Para lo anterior tenemos el siguiente diagrama de deployment, el cual nos permitirá ver cómo se busca que interactúe el módulo :

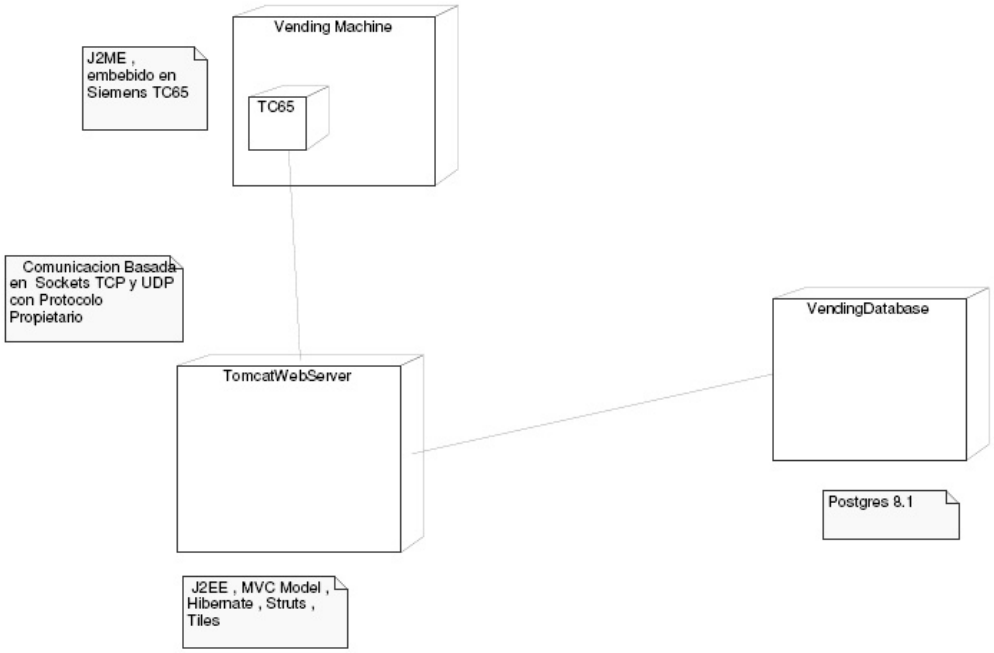


Figura 4.5 Diagrama Deployment

Los archivos obtenidos de la interacción entre el módulo y la vending machine utilizando el protocolo DEX se les conoce como archivos tipo DEX . Cada archivo contiene etiquetas distintas las cuales nos proveen información detallada acerca de la auditoria que se busca en las máquinas

Un ejemplo del archivo que se obtiene cuando se audita la vending machine es el siguiente:

```
DXS*6150210000*VA*V1/1*1
ST*001*0001
ID1*DX00404081514-0284DE3*UNKNOWN 13-9*3912**EVS2.2*
ID4*2*52*50
VA1*13283000*24584*116000*232*0*0
VA2*198780*399*0*0
VA3*0*0*0*0
CA1*00  03041195*MXP-704-0003*0**
CA3*142400*115000*27400*0*16885400*13238400*3675400*0
CA4*21400*0*2913400*84900
CA9*0*1100
CA17*0*50*60*6*78
CA17*1*100*73*17*63
CA17*2*200*66*83*95
CA17*3*500*54*30*179
CA17*4*1000*0*0*0
CA17*5*1000*0*0*0
CA17*6*2000*0*0*0
TA2*0*0*0*0
LS*0100
PA1*1*500*
PA2*841*455700*10*5000*0*0
PA5***20
PA1*2*500*
PA2*1744*943200*11*5500*0*0
PA5***26
PA1*3*500*
PA2*2769*1517400*12*6000*0*0
PA5***24
PA1*4*500*
PA2*2914*1576300*17*8500*0*0
PA5***27
PA1*5*500*
PA2*422*223600*3*1500*0*0
PA5***22
PA1*6*500*
PA2*1359*726900*9*4500*0*0
PA5***30
PA1*7*500*
PA2*1926*1043400*23*11500*0*0
PA1*8*500*
PA2*1557*843600*14*7000*0*0
PA1*9*500*
PA2*1859*1021000*11*5500*0*0
PA1*10*500*
PA2*2068*1122800*28*14000*0*0
PA1*11*500*
PA2*2202*1197300*14*7000*0*0
PA1*12*500*
PA2*2151*1172900*20*10000*0*0
PA1*13*500*
PA2*2772*1438900*60*30000*0*0
PA5***24
LE*0100
EA2*DO*4*613
EA2*CR*1*52
EA3*19
EA7*0*75
MA5*ERROR*ENABLE
```

```
MA5*BAUD*9600
MA5*SEL01*1,2,3*828*899
MA5*SEL02*1,2,3*1695*1803
MA5*SEL03*1,2,3*2716*2900
MA5*SEL04*1,2,3*2835*3006
MA5*SEL05*1,2,3*405*445
MA5*SEL06*1,2,3*1321*1443
MA5*SEL07*5*1695*1710
MA5*SEL08*4*1451*1858
MA5*SEL09*6*1642*1711
MA5*SEL10*7*1756*1790
MA5*SEL11*8*1996*2121
MA5*SEL12*9*1863*2056
MA5*SEL13*1,2,3*2733*2918
MA5*TUBE1**60*73*66*54*0*0*0
MA5*SWITCH*UNLOCK*2,4,5**8
MA5*STATUS*********
MA5*CCOC*0*0*0
MA5*PREV*4231
MA5*LANG*ESP
MA5*TIME*1*050705*1230*OFF
MA5*LITE*0**0000**0000
MA5*RFRG*0**0000**0000*F*350*600*0
MA5*BLC1*0**0000**0000**0
MA5*BLC2*0**0000**0000**0
MA5*DISC*0**0000**0000**0
MA5*OVER*
SD1*000000
G85*1360
SE*88*0001
DXE*1*1
```

Como podemos ver, es un archivo de texto el cual puede ser fácilmente parseado e interpretado de acuerdo a los identificadores que se muestran en el Apéndice B de este trabajo. En cuestiones de tamaño no sobrepasa los 2 Kb.

Cabe aclarar que la comunicación entre los dispositivos remotos y el servidor web se tratarán utilizando VPNs debido a la seguridad que las mismas implementan. Una es una red de información privada que hace uso de una infraestructura pública de telecomunicaciones (internet).

Conecta diferentes segmentos de red o usuarios a una red principal, manteniendo la privacidad a través del uso de un protocolo de túnel o aislamiento así como de otras tecnologías que proveen seguridad. La función principal de una VPN es la de brindar conectividad a una red, a través de una red pública, brindando la integridad de la información.

Ahora bien, de acuerdo a los módulos anteriormente descritos, se propone los siguientes diagramas de clases en los que se ve planteado las interacciones y los componentes de los que constará. Se tratará finalmente de un componente stand-alone basado en J2ME Se propone un MainMidlet como clase principal de arranque la cual será la encargada de inicializar el sistema.

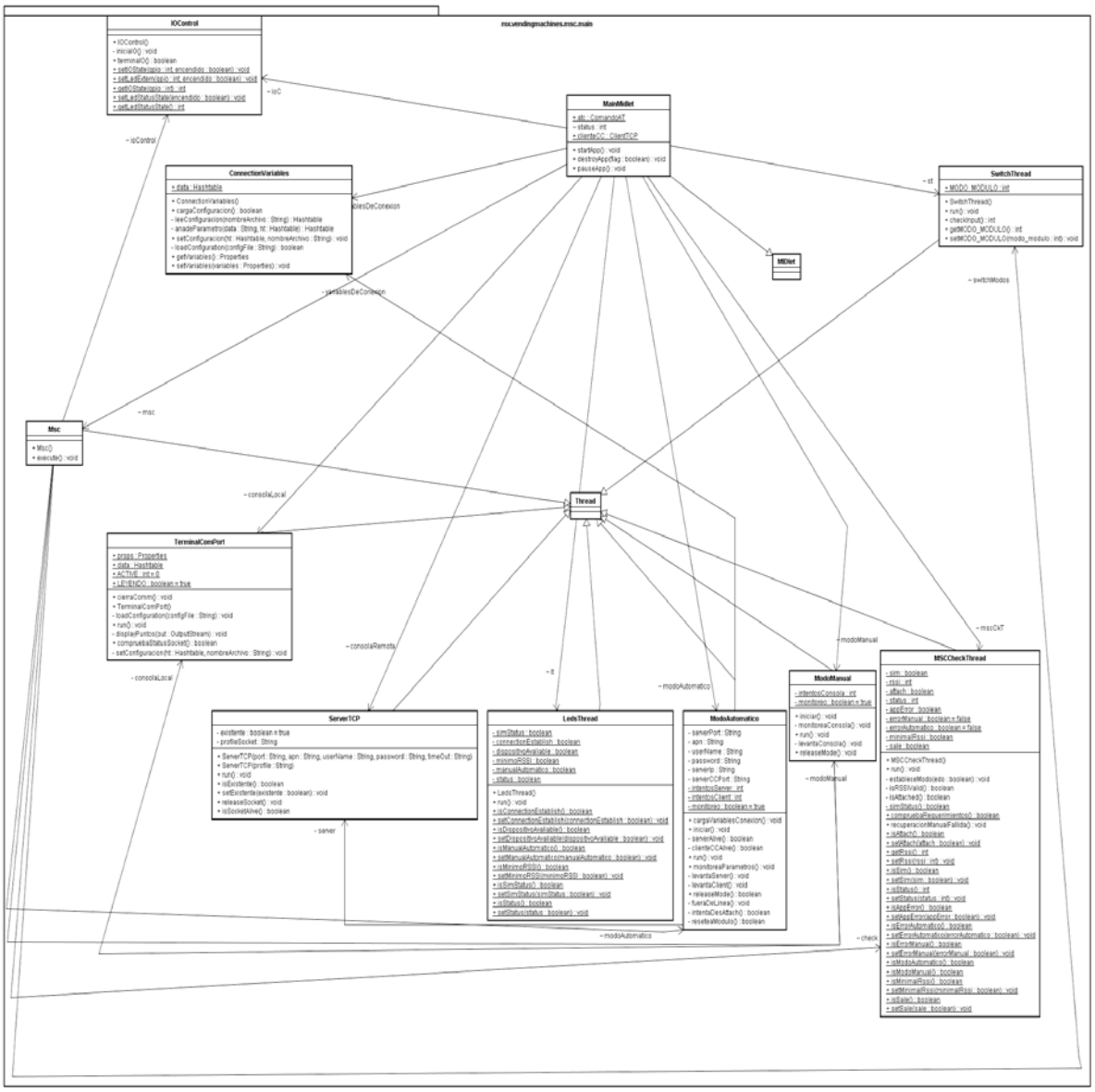


Figura 4.6 Diagrama De Clases 1 : Modulo TC65

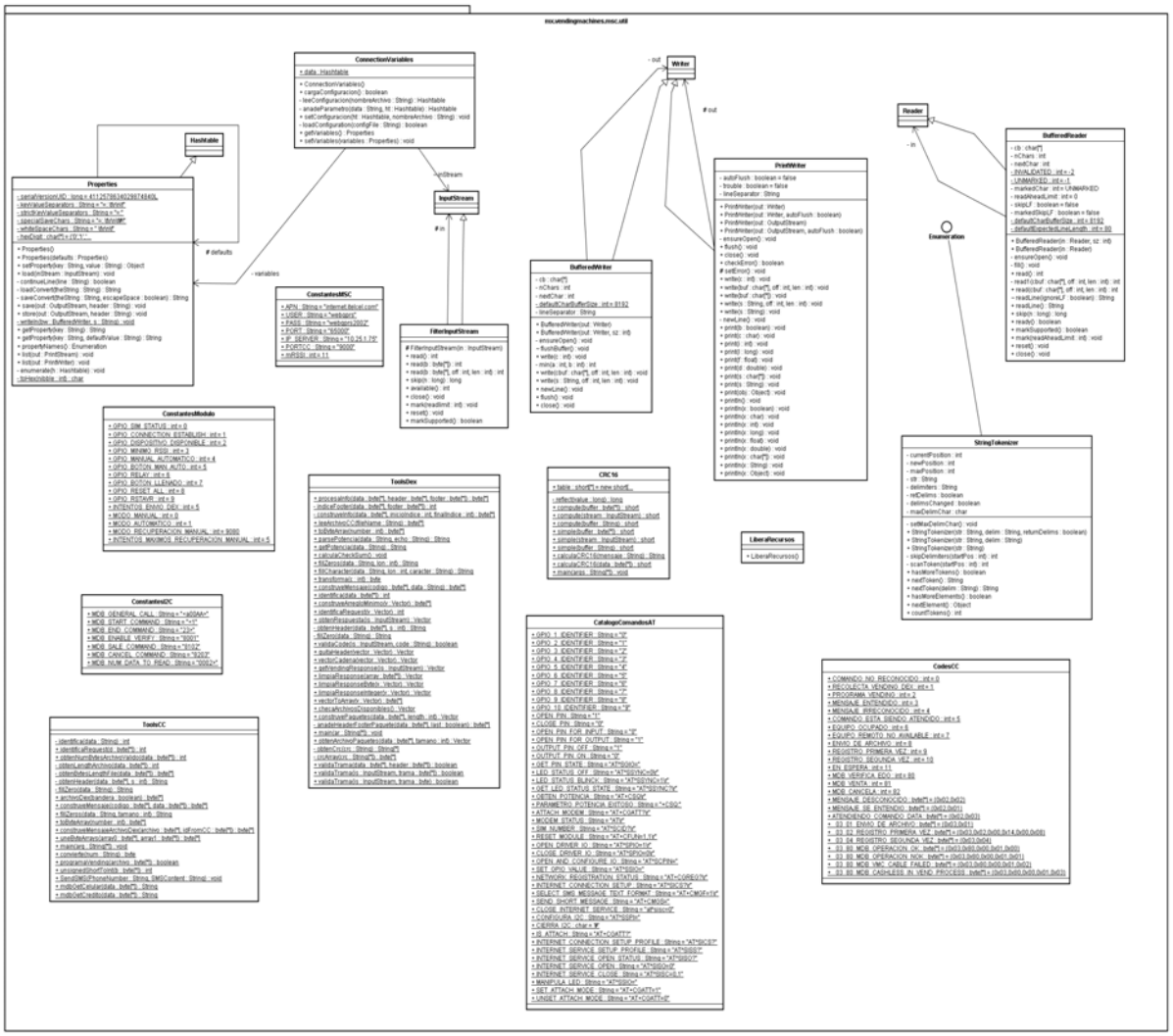


Figura 4.7 Diagrama De Clases 2 : Modulo TC65

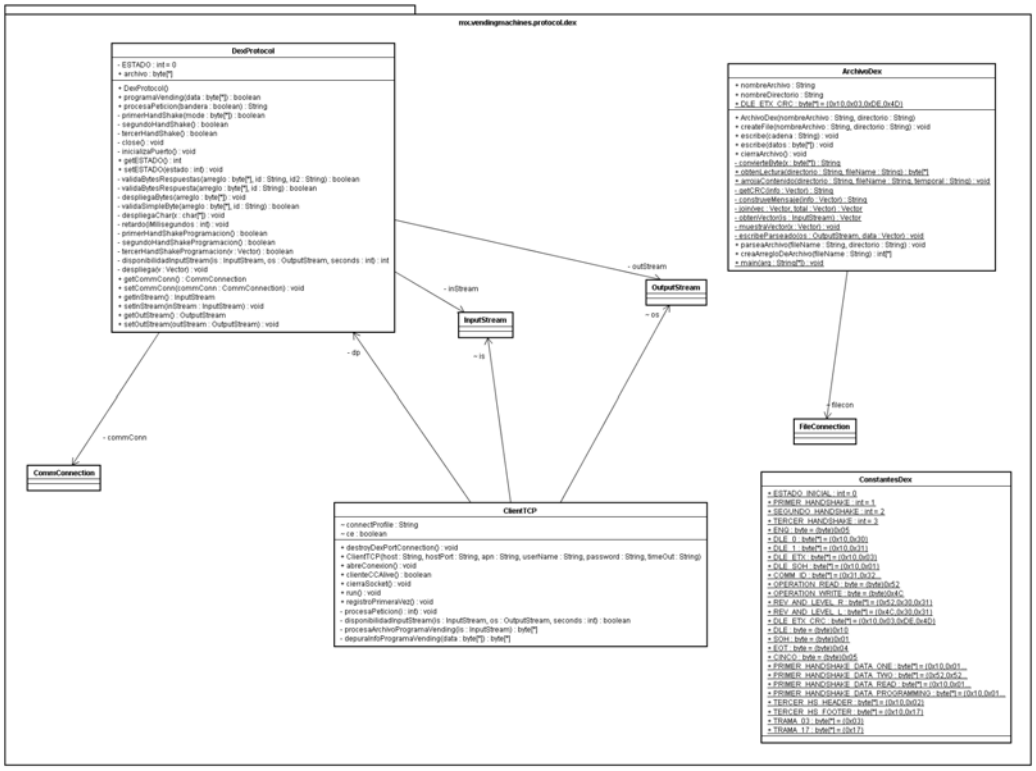


Figura 4.8 Diagrama De Clases 3 : Modulo TC65

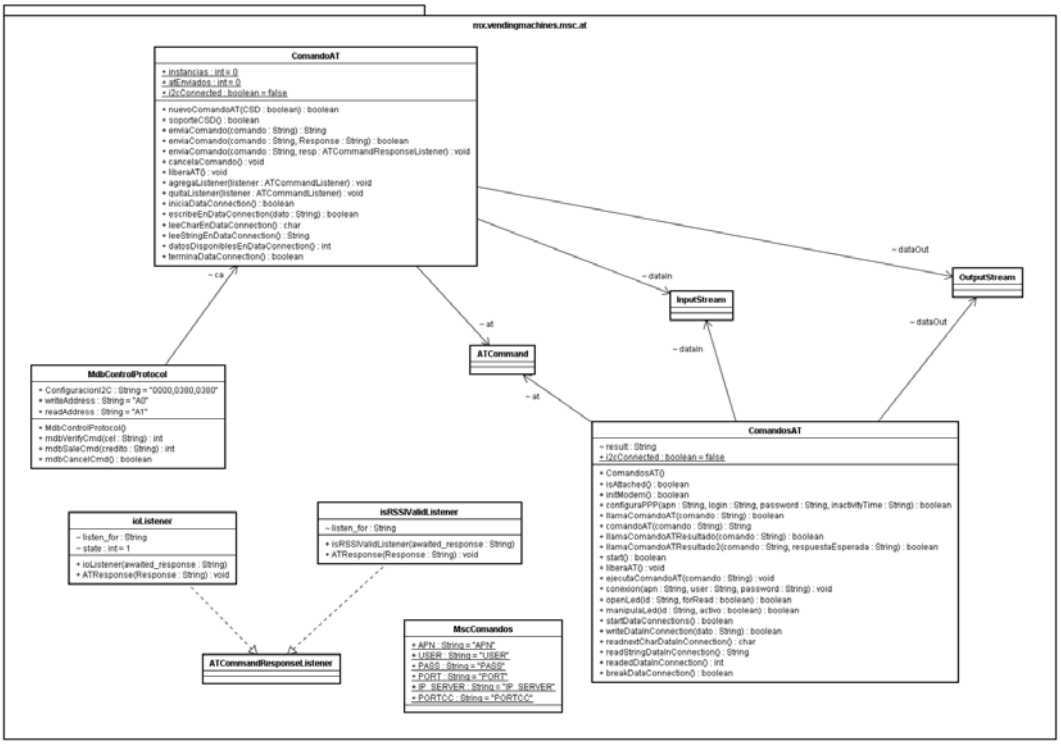


Figura 4.9 Diagrama De Clases 4 : Modulo TC65

5.- Análisis y diseño del servidor J2EE.

5.1 Descripción del problema

Hemos visto que para manejar cada una de las Vending Machines, utilizaremos un modulo TC65 conectado a cada una de ellas y él se encargara de enviar la información que se requiera a través de GPRS con ayuda del software que se ha propuesto. Para manejar la información de cada una de estas Vending Machines, necesitamos de un servidor robusto, que no solamente nos permita enviar las peticiones de lectura de cada una de ellas, sino también el manejo de cada respuesta enviada por las mismas y además poder monitorear el comportamiento de cada una de ellas. Un esquema de lo anterior se puede ver a continuación:

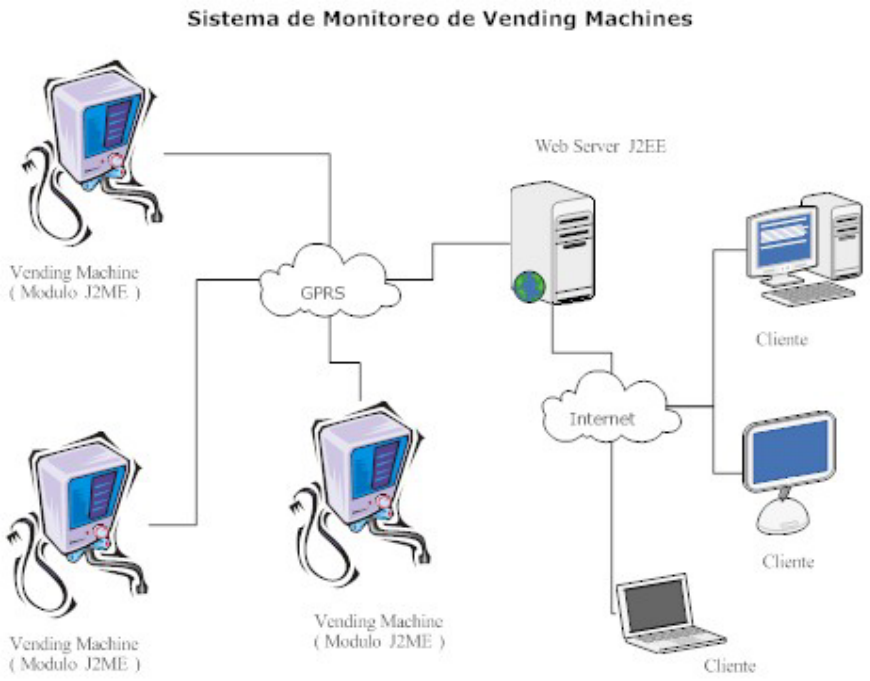


Figura 5.1 Diagrama General Del Sistema

En la figura anterior, vemos que debemos recibir la información de cada una de las Vending Machine que pertenecen a uno o más clientes.

Haciendo una investigación de mercado con algunos proveedores de productos que utilizan las Vending Machine ya sea para promocionar o vender sus productos, se recabaron las siguientes necesidades que presentan en común:

Estas maquinas, cuando nos reporten que se está terminando el producto, habrá que abastecerlas, por lo que será necesario establecer rutas, para que cuando se requiera enviar más producto para este fin, hacerlo de una manera mucho más ordenada para evitar pérdidas de tiempo. Debemos también, saber la ubicación de las maquinas, así como también la IP que tengan para poder enviar peticiones de lectura DEX. Cada una de estas maquinas estará manejada por un encargado el cual a su vez puede tener bajo su responsabilidad una o más maquinas. Este encargado, nos podrá enviar alarmas para reportar cualquier anomalía con la vending machine (monedero, electricidad, mal funcionamiento de SIM) para que se pueda hacer un mejor manejo de la misma y poder tener productos y la maquina disponible para no tener pérdidas económicas. Para ello necesitamos crear reportes de status de las maquinas en cuanto a ventas y a cantidad de producto disponible y a su vez, podemos manejar los catálogos de productos y presentaciones disponibles en cada máquina.

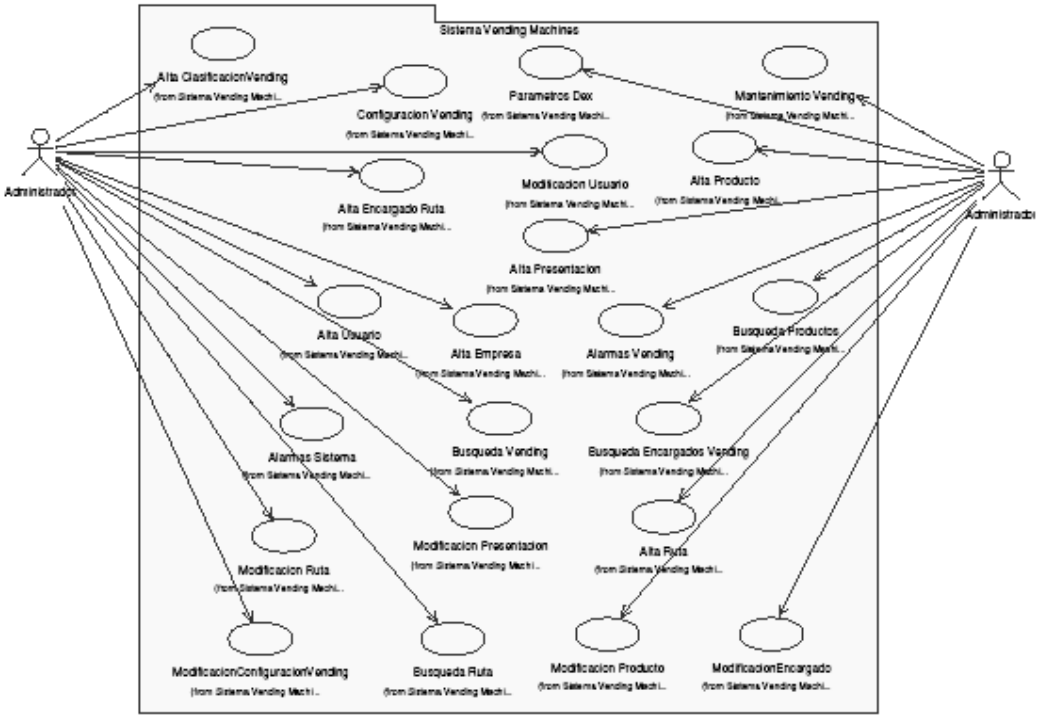
Aparte de lo mencionado anteriormente, se tiene que tener en cuenta los siguientes puntos:

- Manejo de Usuarios
- Niveles de Usuario (Administrador ,
- Reportes
- Configuración de Vending Machines
- Alta de Productos y Presentaciones
- Alarmas
- Manejo de Rutas

Los siguientes diagramas de casos de uso, nos muestran de manera más clara lo anteriormente expuesto:

File: C:\Documents and Settings\memo\Escritorio\tesis\vendingUML.mdl 05:33:34 p.m. Martes, 29 de Enero de 2008 Use Case Diagram: Use Case View / VendingEmpresa Page 1

Figura 5.2 Diagrama de Casos de Uso 1



File: C:\Documents and Settings\memo\Escritorio\tesis\vendingUML.mdl 05:36:26 p.m. Martes, 29 de Enero de 2008 Use Case Diagram: Use Case View / VendingAdministrador Page 1

Figura 5.3 Diagrama de Casos de Uso 2

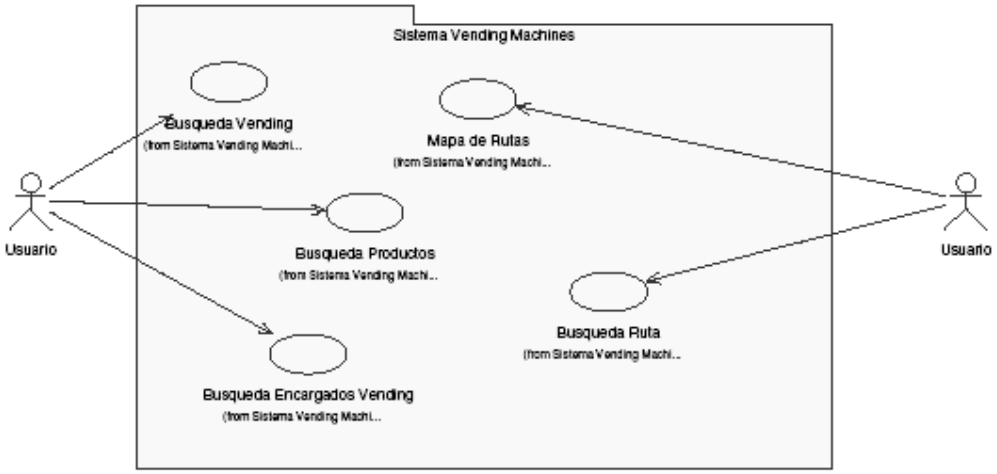


Figura 5.4 Diagrama de Casos de Uso 3

Como se puede observar, ya hemos definido las necesidades que se requiere cubra el servidor, con lo cual podremos diseñar la BD la cual será alimentada por el server de

acuerdo a lo que hemos propuesto. El diagrama Entidad Relación se puede consultar en el **apéndice A**.

5.2 Diseño de pantallas para la aplicación Web

Antes de iniciar cualquier propuesta de Arquitectura de Sistemas, necesitaremos saber qué es lo que se espera del sistema en cuestión de vista final al usuario y así poder tener una idea mucho más clara de cómo va a funcionar el sistema en cuestión de flujos lógicos. A continuación se propondrán las plantillas de cada pantalla que se tendrá en el sistema

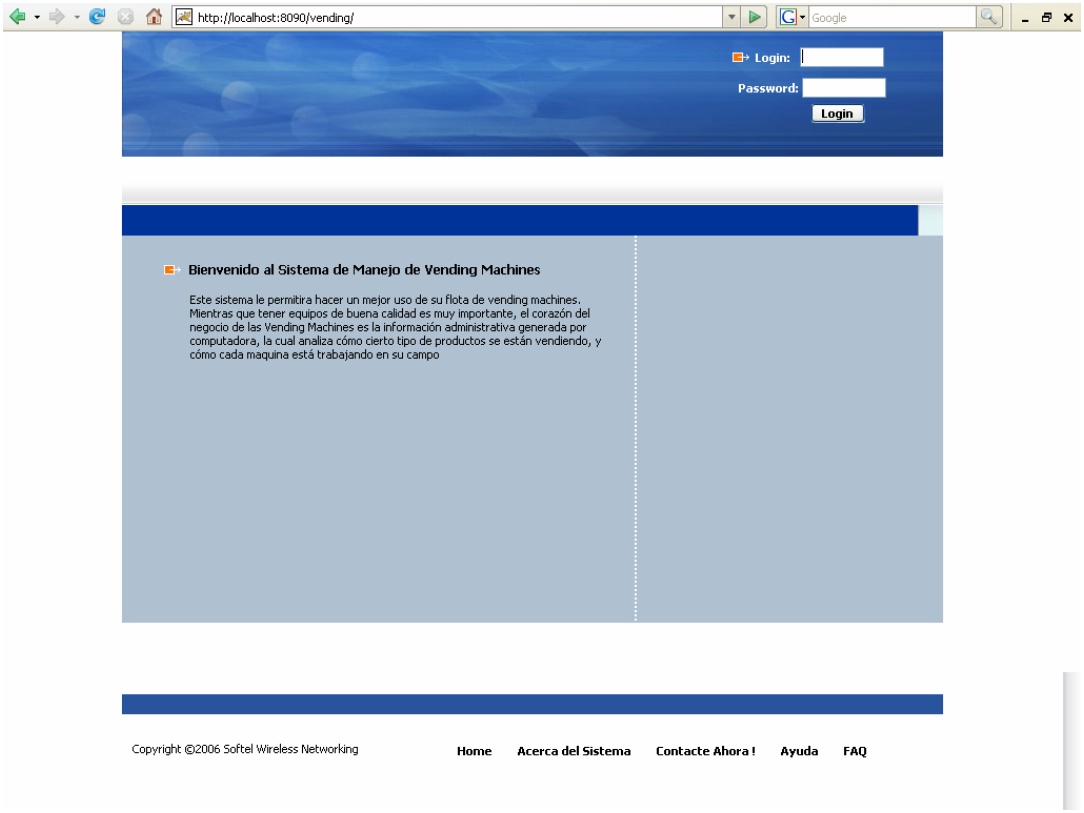


Figura 5.5 Pantalla Login

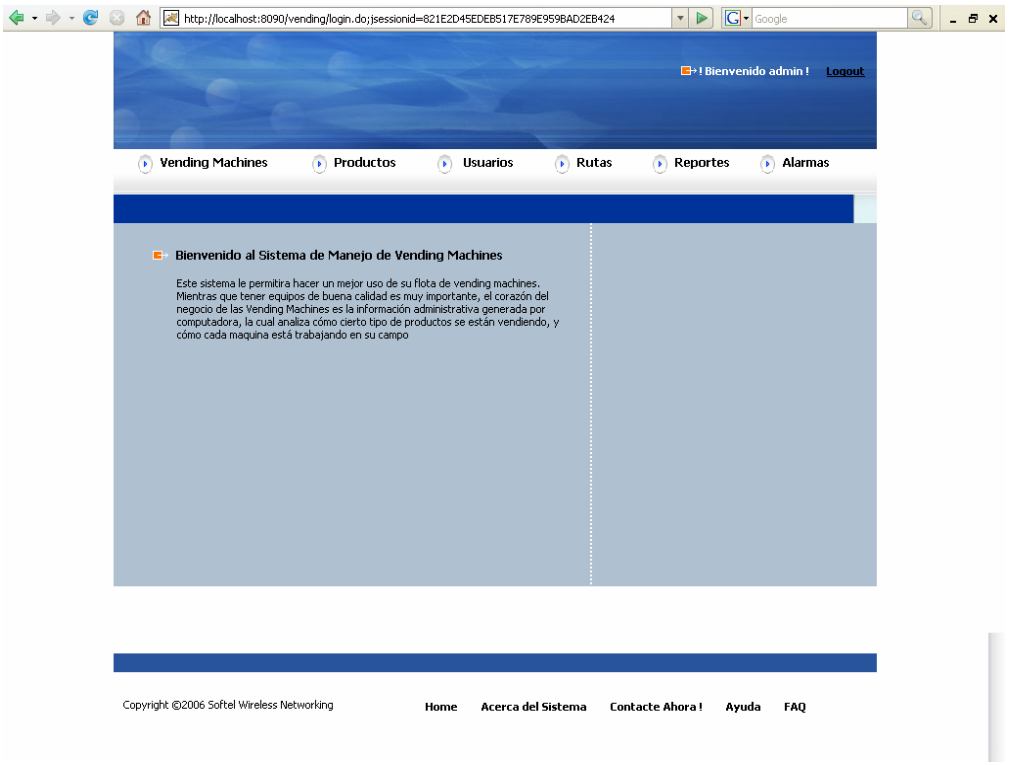


Figura 5.6 Pantalla Bienvenida

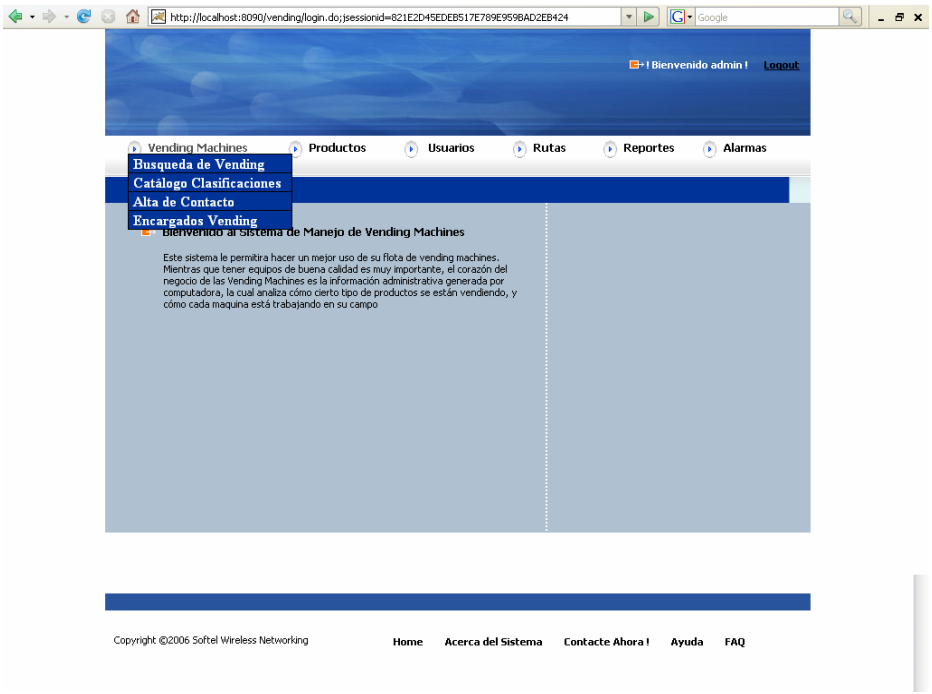


Figura 5.7 Menu Vending Machines

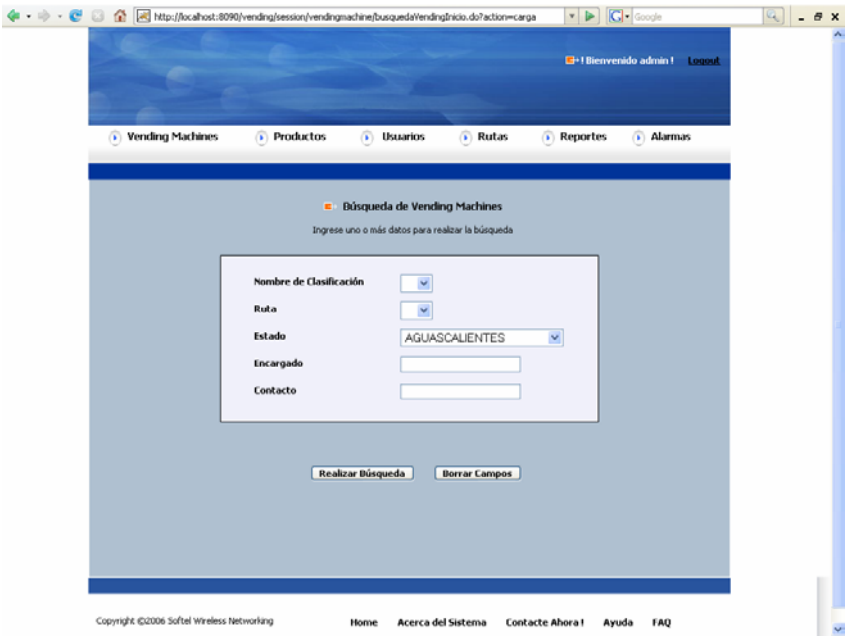


Figura 5.8 Búsqueda Vending Machine

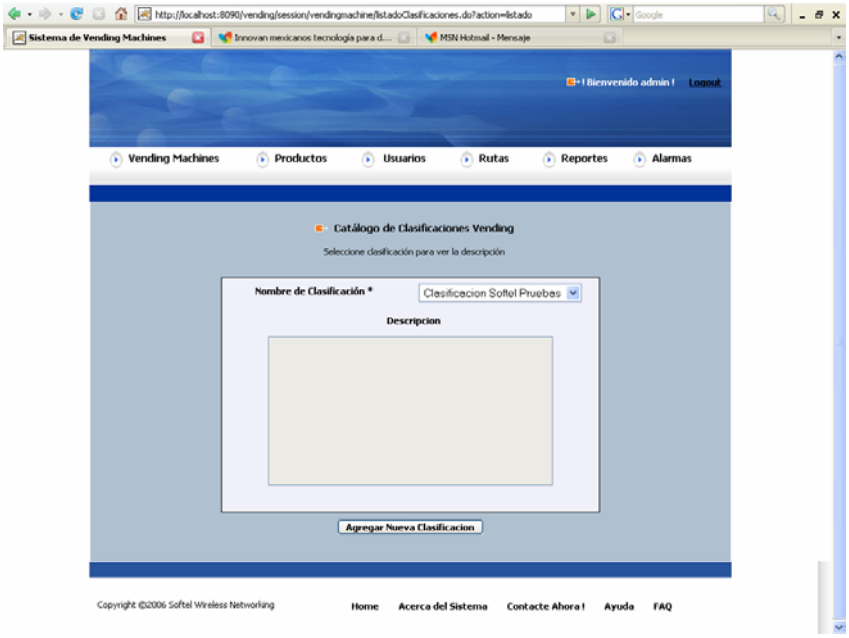


Figura 5.9 Catálogo de Clasificaciones 1

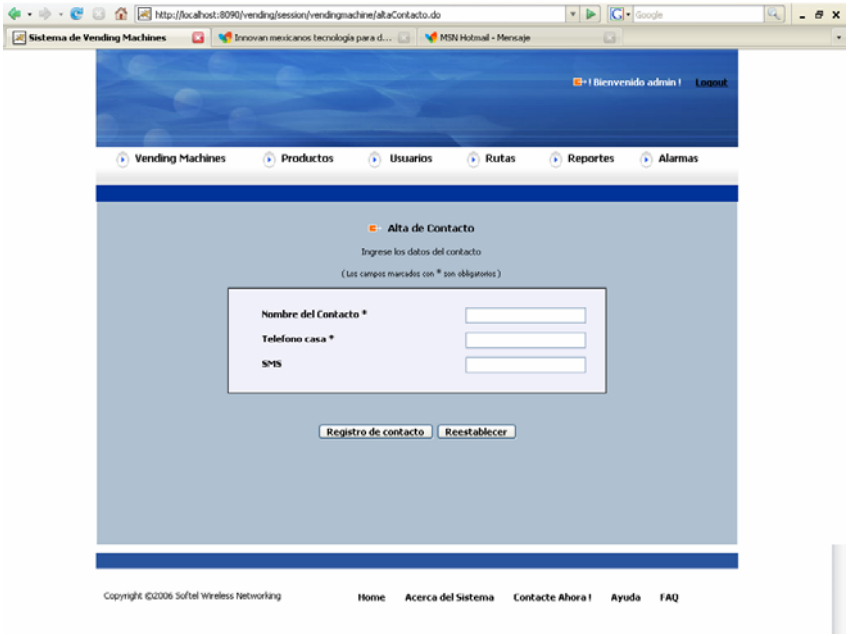


Figura 5.10 Alta de Contacto

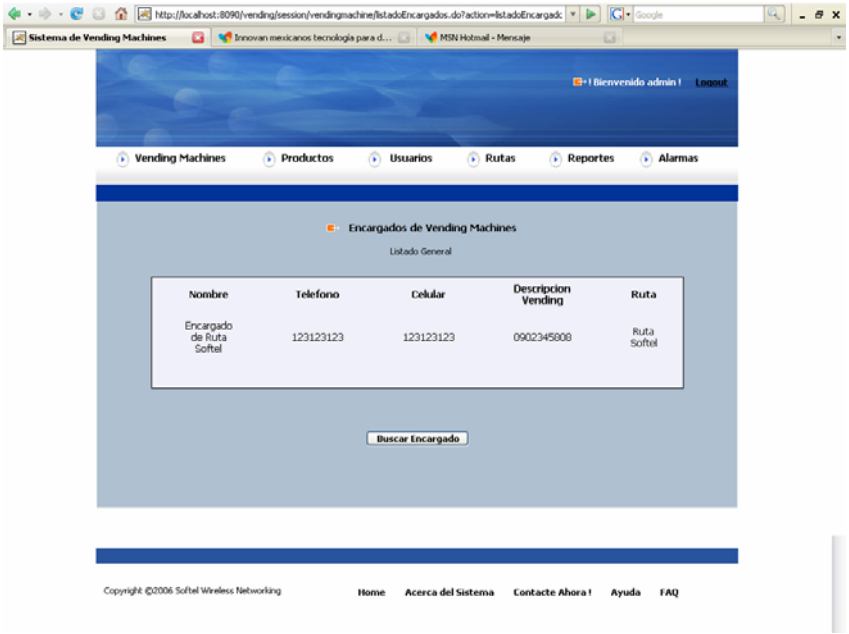


Figura 5.11 Listado de Encargados

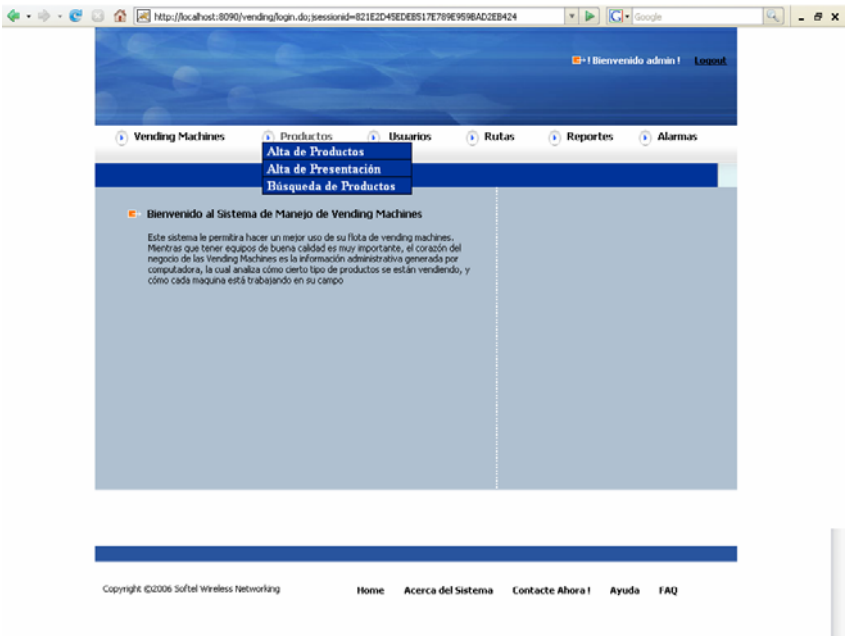


Figura 5.12 Menú Productos

Figura 5.13 Alta de Producto

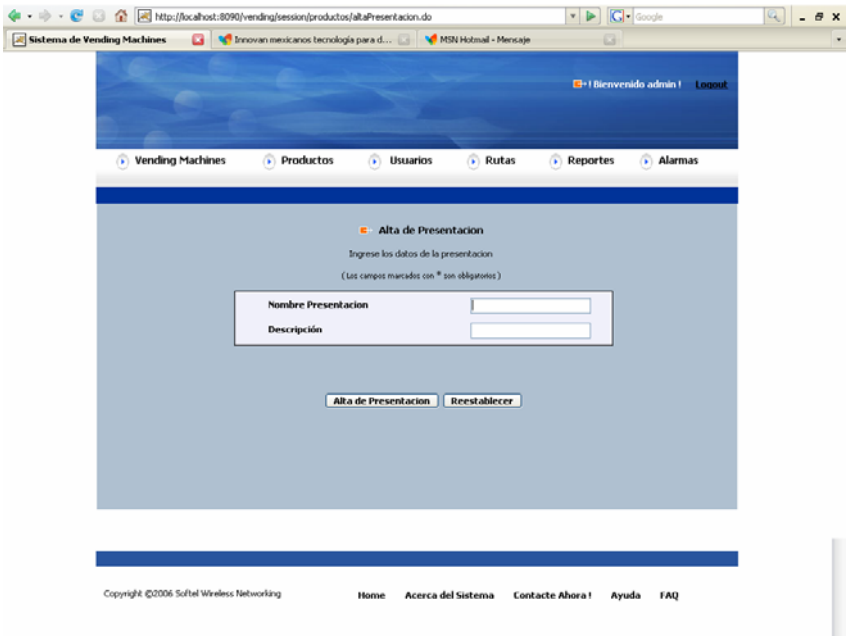


Figura 5.14 Alta de Presentación

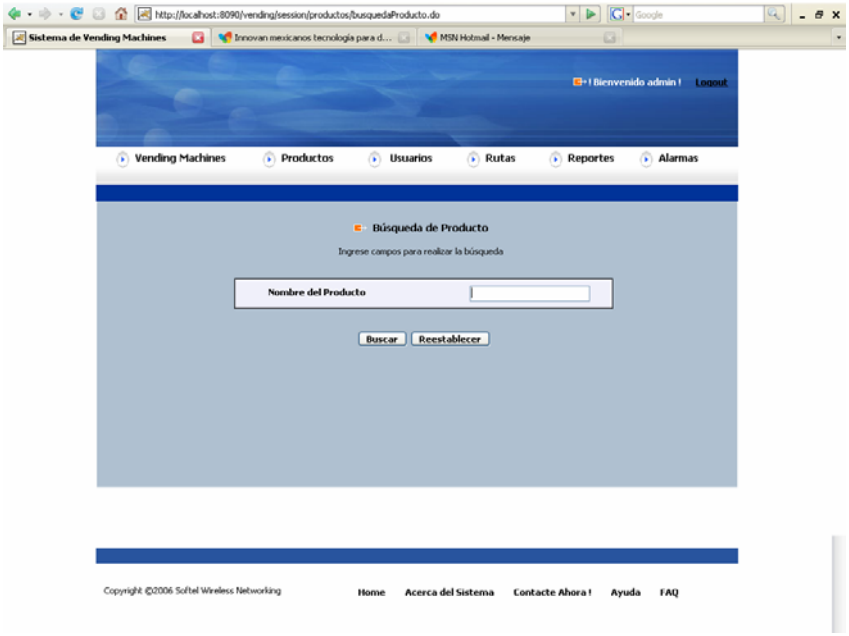


Figura 5.15 Búsqueda de Productos

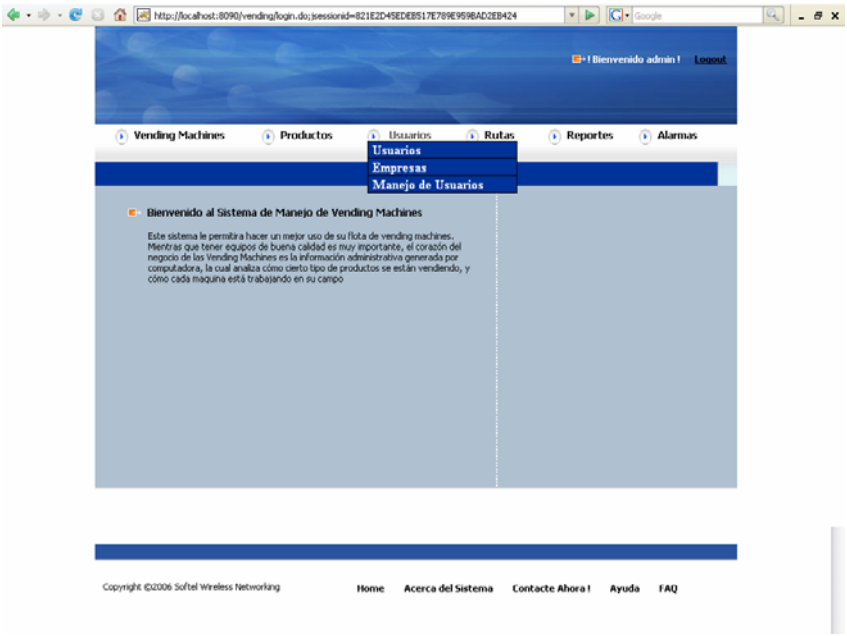


Figura 5.16 Menú Usuarios

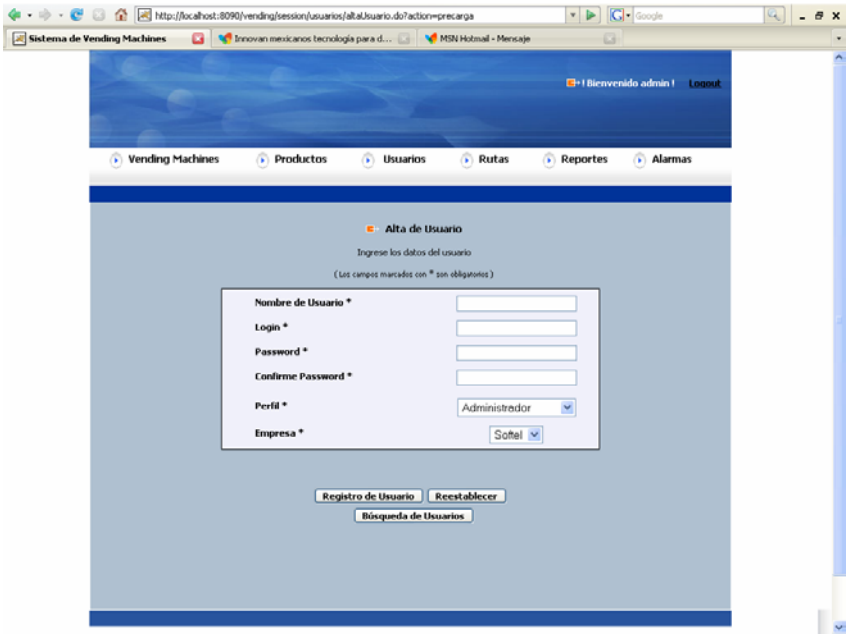


Figura 5.17 Alta de Usuarios

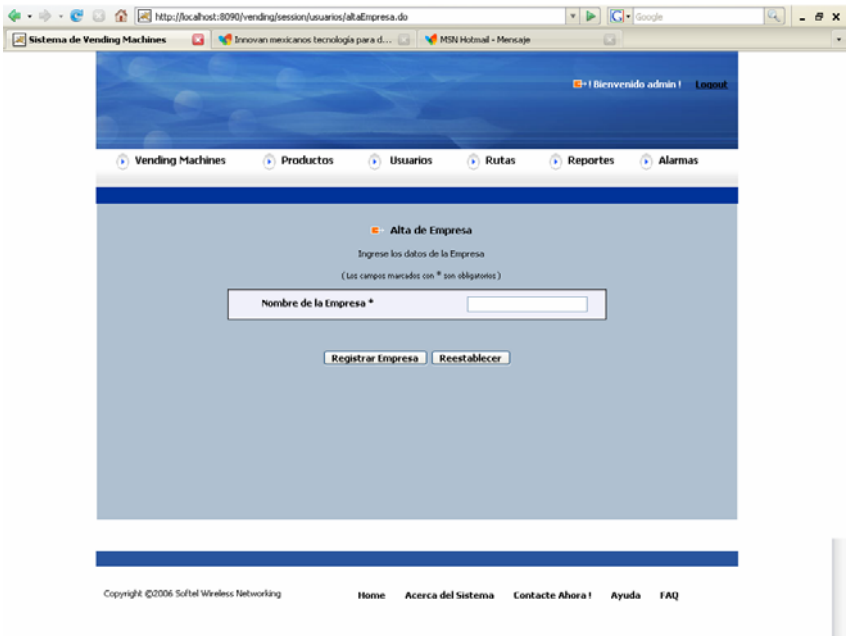


Figura 5.18 Alta de Empresas

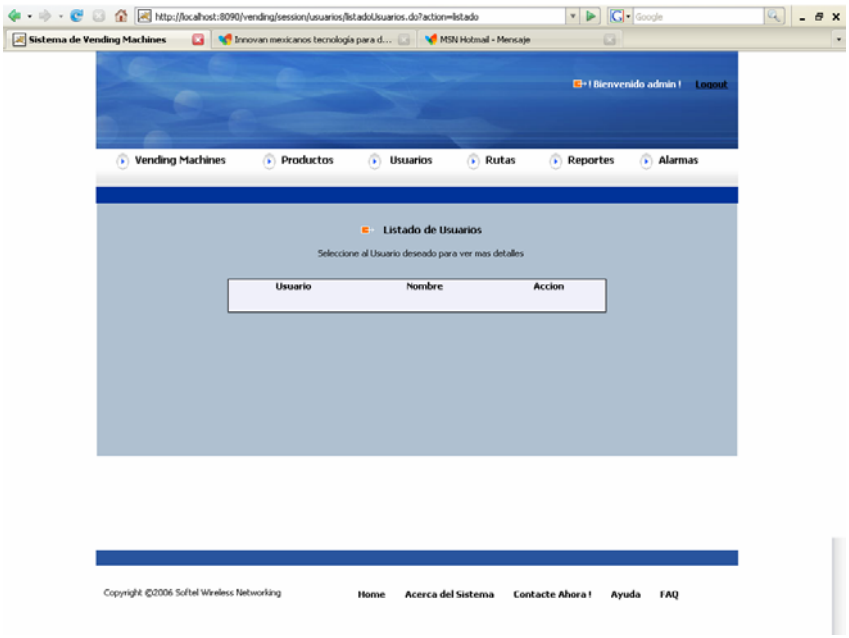


Figura 5.19 Manejo de Usuarios

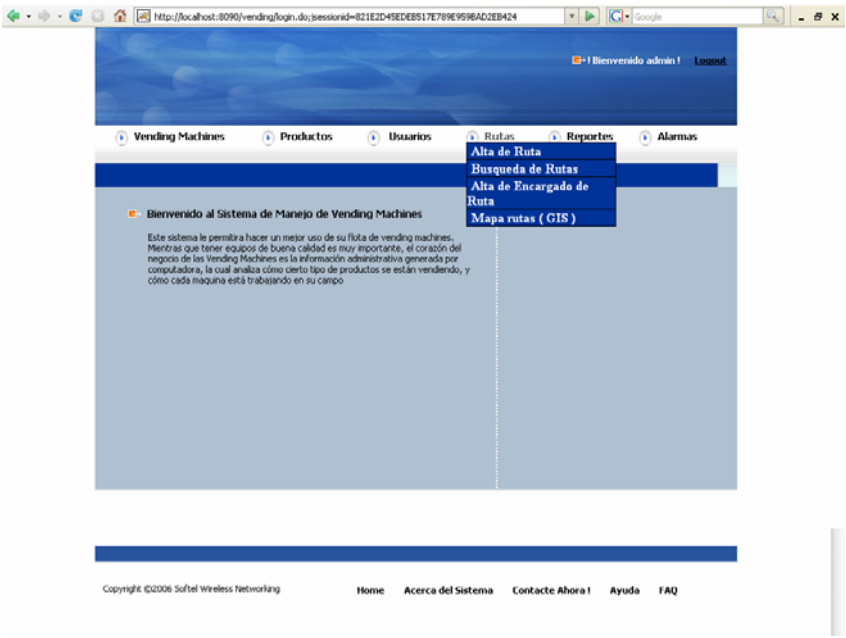


Figura 5.20 Menú de Rutas

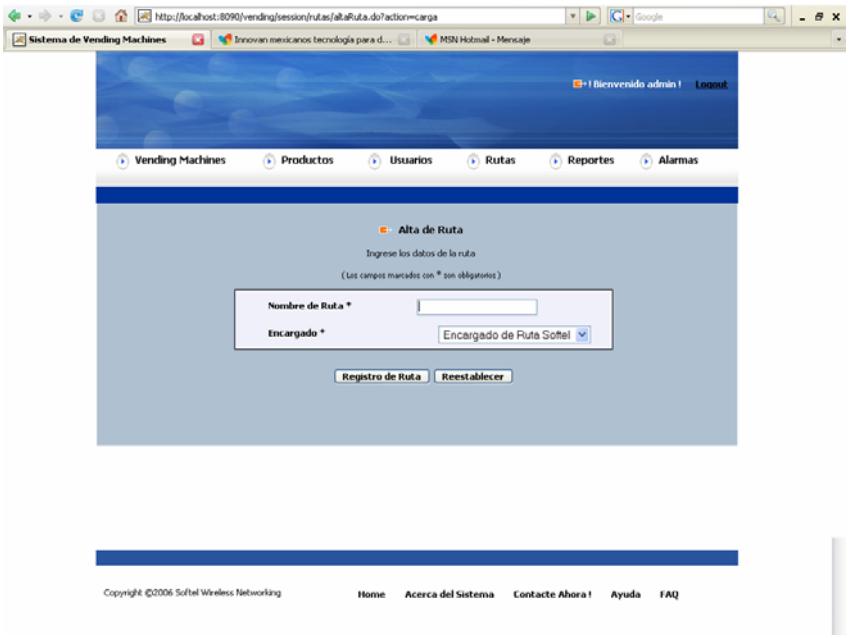


Figura 5.21 Alta de Rutas

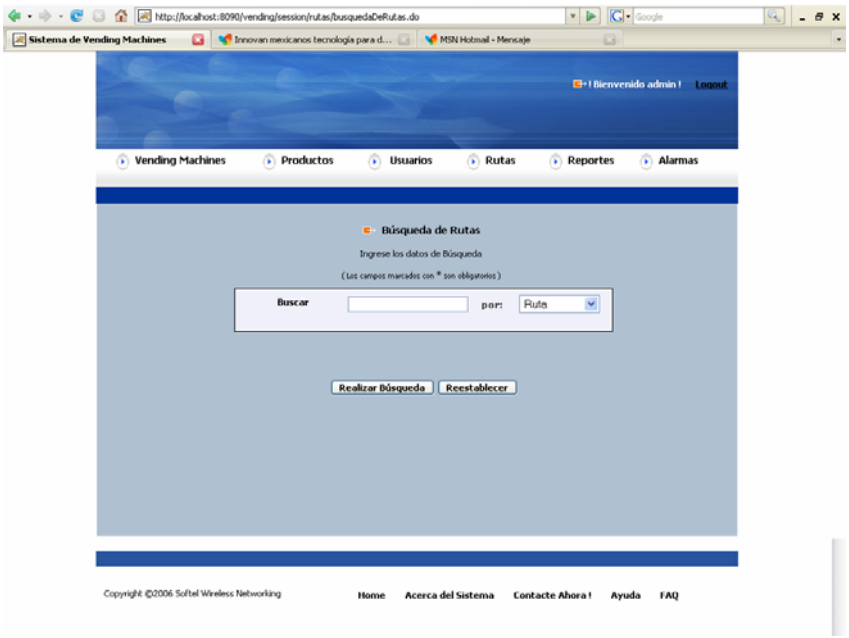


Figura 5.22 Búsqueda de Rutas

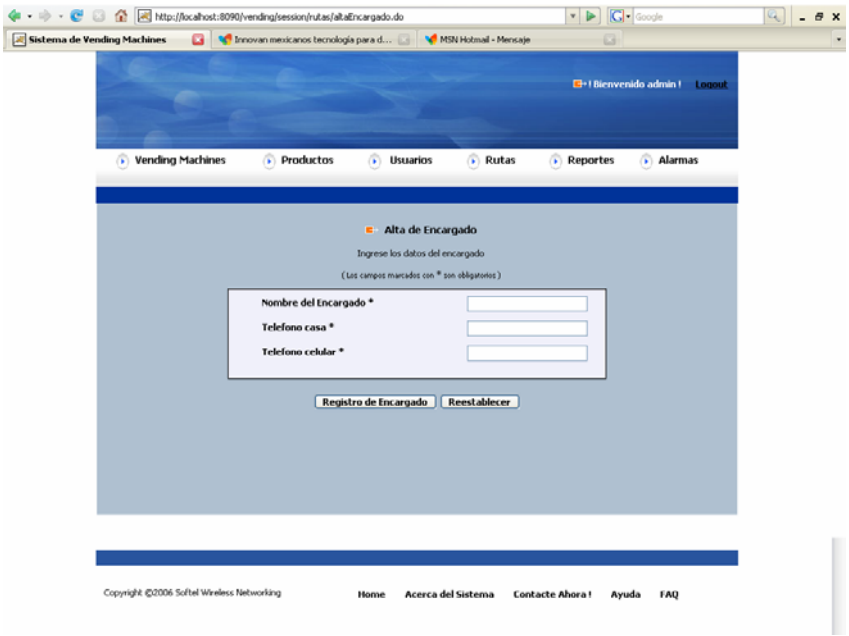


Figura 5.23 Alta de Encargado de Ruta

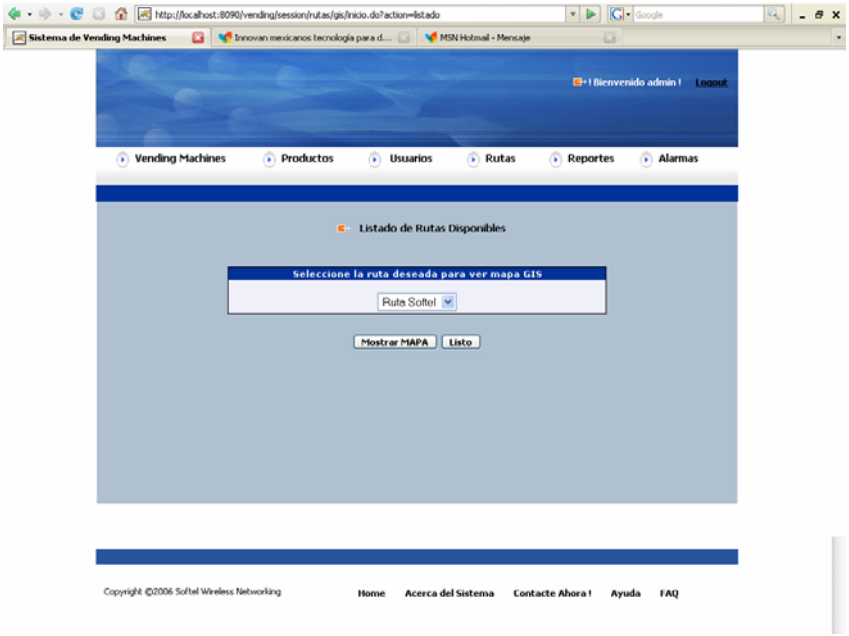


Figura 5.24 Mapa de Rutas

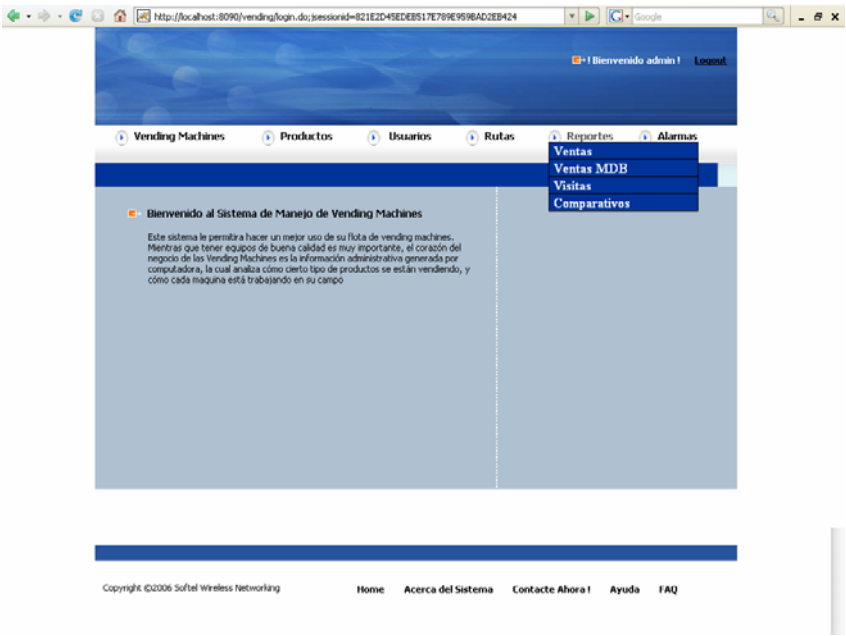


Figura 5.25 Menú de Reportes

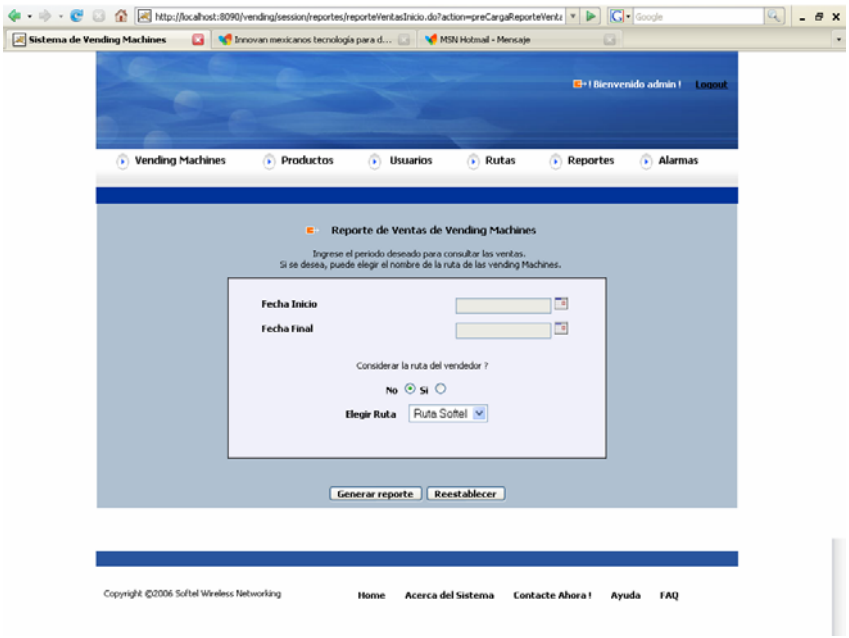


Figura 5.26 Reporte Ventas

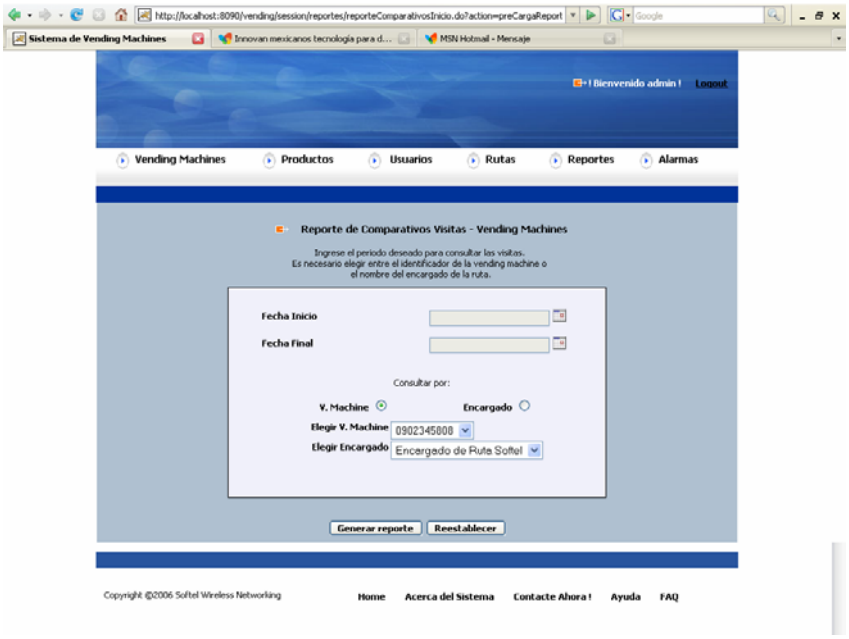


Figura 5.27 Reporte Visitas

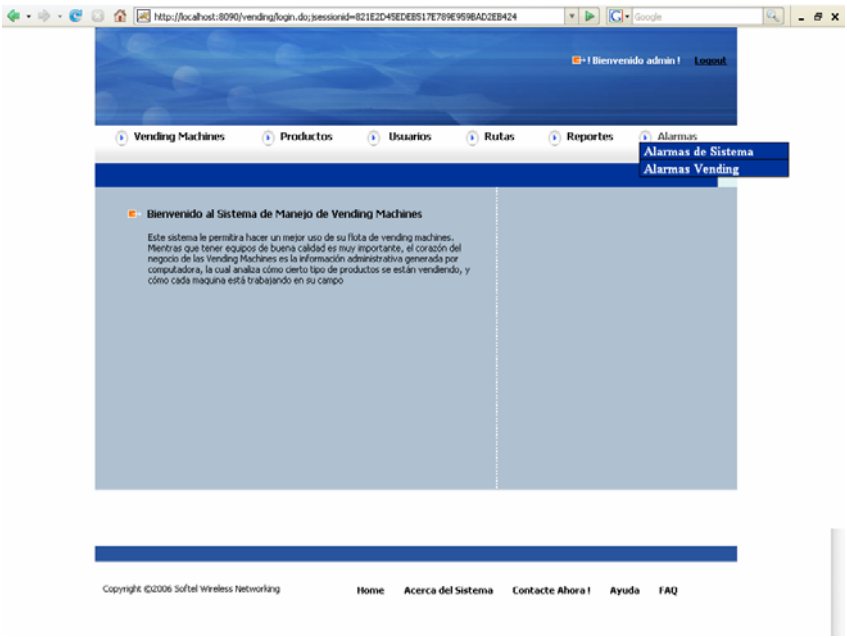


Figura 5.28 Menú Alarmas

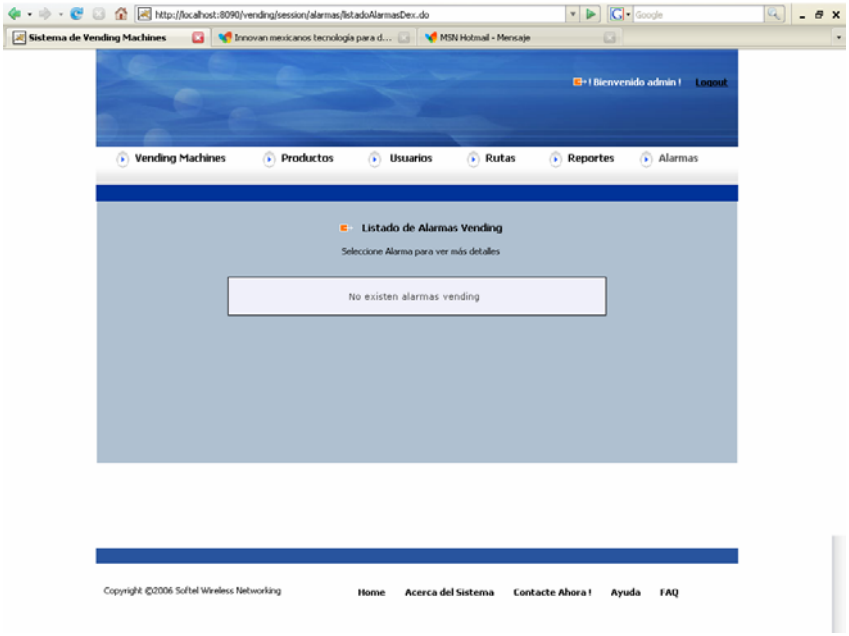


Figura 5.29 Listado de Alarmas Vending

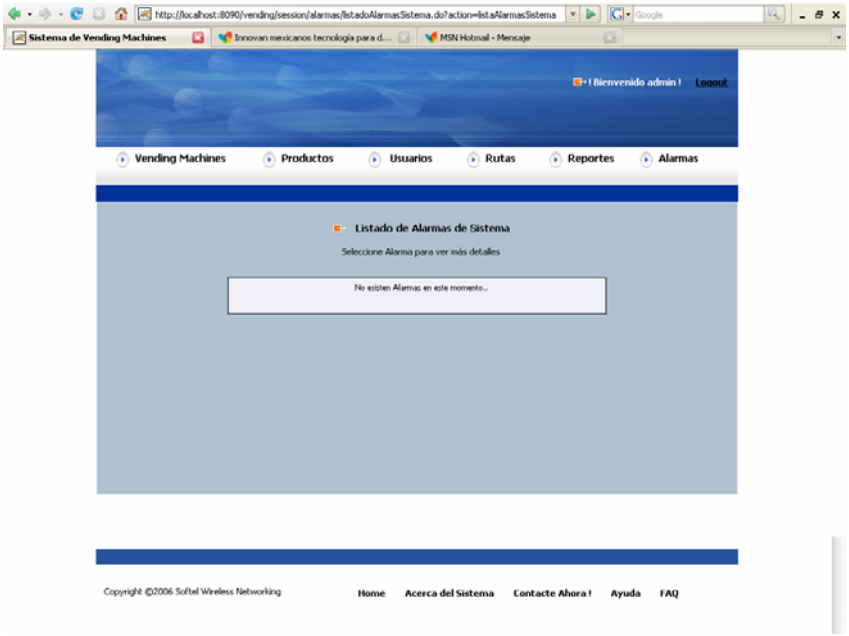


Figura 5.30 Listado de Alarmas del Sistema

5.3 Arquitectura del Servidor Web para Control de Vending Machines

Hemos definido los distintos módulos, niveles de acceso, pantallas y la BD que necesitamos crear para el sistema que necesitamos, ahora necesitamos definir la arquitectura que el mismo tendrá.

En los sistemas Web modernos, se utilizan muchas metodologías y arquitecturas, las cuales son probadas y con el tiempo son modificadas para ser mucho más robustas y reutilizables para que en cualquier momento el sistema pueda crecer sin necesidad de modificar la columna vertebral del mismo.

Uno de estos modelos utilizados en los sistemas basados en web, es el llamado MVC o Modelo Vista Controlador. Este modelo, nos permite dividir el sistema en 3 grandes rubros los cuales son:

- **Modelo**: El modelo del sistema, es aquel donde se encuentra toda nuestra lógica del negocio, es decir, la forma en la que se resolverán las peticiones recibidas del usuario de acuerdo a los requerimientos del sistema que se esté implementando.

- **Controlador**: Esta parte del sistema, es la encargada de enviar las peticiones a donde se requiera, de acuerdo a lo que el usuario necesite, hace uso de las clases de negocio o modelo para satisfacer la petición del cliente.

- **Vista**: Sencillamente es la forma en la que se le presentara la información al usuario, en los sistemas java convencionales, esta es representada por los JSP o también llamados Java Server Pages.

Este modelo, es muy utilizado, debido a que si se requiere modificar algo, se ataca únicamente a la división que se requiera del sistema y no debe afectar a las otras dos partes. Esto es especialmente útil cuando se requiera de modificar para ampliar las capacidades del sistema o sencillamente para brindarle un mantenimiento. Hay que recalcar que sistema que es verdaderamente utilizado, requerirá de constates mantenimientos y el hecho de construir una buena arquitectura radicara en que las personas involucradas en el mantenimiento del mismo, lo haga de una manera optima rápida y sobre todo totalmente funcional.

Para la correcta implementación del sistema basado en J2EE, se hará uso de un Framework. Un framework es la extensión de un lenguaje mediante una o más jerarquías de clases que implementan una funcionalidad y que (opcionalmente) pueden ser extendidas.

En caso de aplicaciones web, un Framework muy utilizado en las aplicaciones web actuales y que se basa en el MVC es Struts.

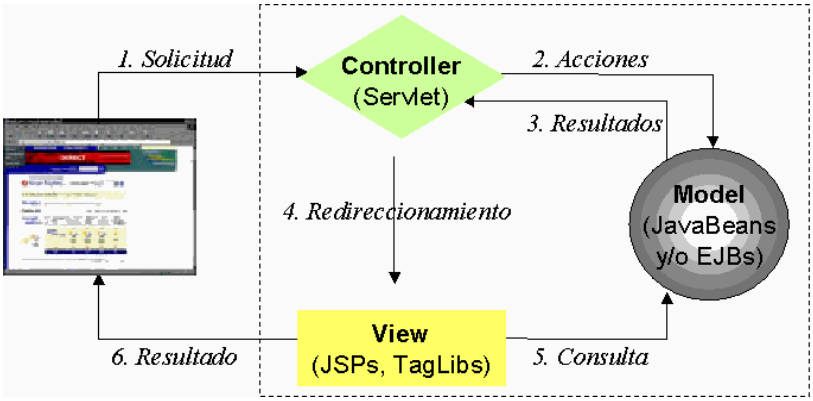


Figura 5.31 Modelo MVC

En la figura anterior podemos ver en términos generales, como se ve una arquitectura de un sistema web implementando Struts. El navegador genera una solicitud que es atendida por el Controller (un Servlet especializado). El mismo se encarga de analizar la solicitud, seguir la configuración que se le ha programado en su XML y llamar al Action correspondiente pasándole los parámetros enviados. El Action instanciará y/o utilizará los objetos de negocio para concretar la tarea. Según el resultado que retorne el Action, el Controller derivará la generación de interfaz a una o más JSPs, las cuales podrán consultar los objetos del Model a fines de realizar su tarea.

De acuerdo al patrón de diseño MVC (Modelo-Vista-Controlador), las aplicaciones Struts tiene tres componentes principales: un Servlet controlador, que está proporcionado por el propio Struts, páginas JSP (la "vista"), y la lógica de negocio de la aplicación (o el "modelo").

El Servlet controlador Struts une y en ruta solicitudes HTTP a otros objetos del marco de trabajo, incluyendo JavaServer Pages y subclases o`org.apache.struts.action.Action` proporcionadas por el desarrollador Struts. Una vez inicializado, el controlador analiza un archivo de configuración de recursos (Struts-config.xml) , La configuración de recursos define (entre otras cosas) los `org.apache.struts.action.ActionMapping` para una aplicación. El controlador usa estos mapeos para convertir las solicitudes HTTP en acciones de aplicación.

Un `ActionMapping` normalmente especificará:

- una path solicitado (o "URI"),
- El tipo objeto (subclase de `Action`) para actuar sobre la solicitud,
- y otras propiedades según se necesite.

El objeto `Action` puede manejar la solicitud y responder al cliente (normalmente un navegador Web), o indicar a que control debería ser reenviado. Por ejemplo, si un logín tiene éxito, una acción logín podría desear reenviar la petición hacia el menú.

Los objetos `Action` tienen acceso al Servlet controlador de la aplicación, y por eso tienen acceso a los métodos del Servlet. Cuando se reenvía un control, un objeto Action puede reenviar indirectamente uno o más objetos compartidos, incluyendo JavaBeans, situándolos en una de las colecciones estándar compartidas por los Servlets Java.

Un objeto acción puede crear un bean de tarjeta de compra, o un ítem de la tarjeta, situando el bean en la colección de sesión, y luego reenviando el control a otro mapeo. Este mapeo podría usar una página JavaServer Page para mostrar los contenidos de la tarjeta del usuario. Como cada cliente tiene su propia sesión, cada uno también tendrá su propia tarjeta de compra. En una aplicación Struts, la mayoría de la lógica del negocio se puede representar usando JavaBeans. Una `Action` puede llamar a las propiedades de un JavaBean sin conocer realmente como funciona. Esto encapsula la lógica del negocio, para que la `Action` pueda enfocarse en el manejo de errores y dónde reenviar el control.

Los JavaBeans también se pueden usar para manejar formularios de entrada. Un problema clave en el diseño de aplicaciones Web es retener y validar lo que el usuario ha introducido entre solicitudes. Con Struts, podemos definir un conjunto de clases bean formulario, extendiendo de la clase `org.apache.struts.action.ActionForm`, y almacenar fácilmente los datos de un formulario de entrada en estos beans. El bean se graba en una de las colecciones estándar o de contextos compartidos, por eso puede ser usado por otros objetos, especialmente un objeto `Action`.

El bean de formulario (o form-bean) puede usarlo una JSP para recoger datos del usuario, por un objeto `Action` para validar los datos introducidos por el usuario, y luego de nuevo por el JSP para rellenar los campos del formulario. En el caso de validación de errores, Struts tiene un mecanismo compartido para lanzar y mostrar mensajes de error.

Un form-bean Struts se declara en la configuración de recursos definida en un archivo fuente Java, y enlazado a un `ActionMapping` usando un nombre de propiedad común. Cuando una solicitud llama a un Action que usa un bean de formulario, el Servlet controlador recupera o crea el bean formulario, y lo pasa el objeto Action. Este objeto entonces puede chequear los contenidos del bean de formulario antes de que su formulario de entrada se muestre, y también la cola de mensajes a manejar por el formulario. Cuando esta listo, el objeto Action puede devolver el control con un reenvío a su formulario de entrada, usando un JSP. El controlador puede responder a la solicitud HTTP y dirigir al cliente a la JavaServer Page.

El marco de trabajo Struts incluye etiquetas personalizadas que pueden rellenar automáticamente los campos de un formulario o un bean de formulario. Lo único que la mayoría de las páginas JSP necesitan saber sobre el resto del marco de trabajo son los nombres de los campos apropiados y dónde enviar el formulario. Los componentes como los mensajes "encolados" por el Action pueden salir usando una simple etiqueta personalizada. También se pueden definir otras etiquetas específicas de la aplicación para ocultar detalles de implementación de las páginas JSPs.

Las etiquetas personalizadas en el marco de trabajo Struts están diseñadas para usar las características de internacionalización incluidas en la plataforma Java. Todas las etiquetas de campos y los mensajes pueden recuperarse desde un recurso de mensajes, y Java puede proporcionar automáticamente el recurso correcto para el idioma y país de un cliente. Para proporcionar mensajes para otro idioma, simplemente añadimos otro fichero de recurso.

Junto al internacionalización, otros beneficios de esta aproximación son las etiquetas consistentes entre formularios, y la posibilidad de revisar todas las etiquetas y mensajes desde una localización central.

Para la aplicación más simple, un objeto Action podría algunas veces manejar la lógica de negocio asociada con una solicitud. Sin embargo, en la mayoría de los casos, un objeto Action, debería llamar a otro objeto, normalmente un JavaBean, para realizar la lógica de negocio real. Esto permite al objeto Action enfocarse en el manejo de errores y el control de flujo, en vez de en la lógica del negocio. Para permitir su reutilización en otras plataformas, los JavaBeans de lógica de negocio no deberían referirse a ningún objeto de aplicación Web. El objeto Action debería traducir los detalles necesarios de la solicitud HTTP y pasarlos a los beans de la lógica del negocio como variables normales de Java.

Por ejemplo, en una aplicación de base de datos:

- Un bean de lógica de negocio (o Model) conectaría y consultaría la base de datos,
- El bean de lógica de negocio devolvería el resultado al objeto Action,
- El objeto Action almacenaría el resultado en un bean formulario en la solicitud,
- La JavaServer Page mostraría el resultado en un formulario HTML.

Ni el objeto Action ni la página JSP necesitan saber (o no les importa) de dónde viene el resultado. Sólo necesitan saber cómo empaquetarlo y mostrarlo.

El resto de esta guía de usuario explica varios componentes Struts en gran detalle. La versión Struts también incluye varias Guías de Desarrollo que cubren varios aspectos de los marcos de trabajo, junto con aplicaciones de ejemplo, el API estándar JavaDoc, y, por supuesto, el código fuente completo!

Struts se distribuye bajo la licencia de la Apache Software Foundation. El código tiene copyright pero es gratuito para usarlo en cualquier aplicación.

Después de todo este análisis, podemos concluir que Struts es la herramienta que necesitamos puesto que implementa el patrón de diseño que se busca .

Por otro lado, en las pantallas propuestas se repiten tanto el header (donde se encuentran las distintas opciones con las que cuenta el usuario) y el footer (la zona donde se encuentran los créditos del sistema). Para evitar tal repetición se usará una herramienta llamada tiles, que pertenece a Struts y funciona de la siguiente manera:

Supongamos un portal con el siguiente formato:

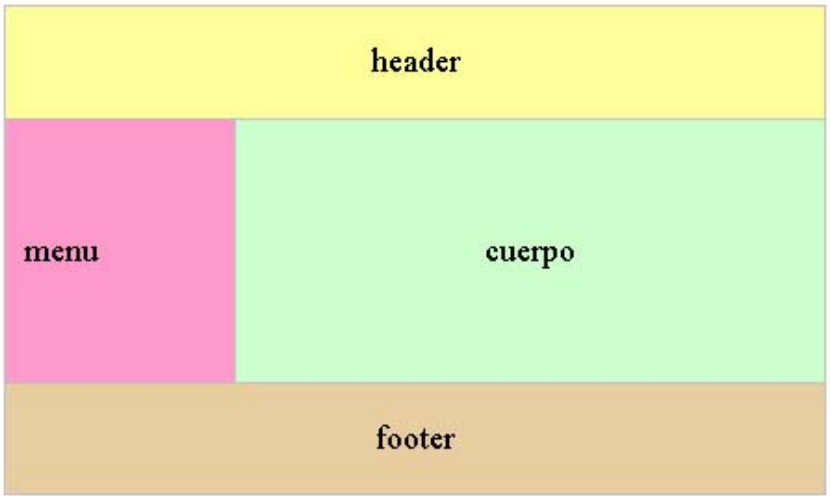


Figura 5.32 Formato de Portales

Probablemente el header y el footer sean estáticos, pero indudablemente el cuerpo va a variar según la opción del menú que sea seleccionada.

Tiles nos permite definir una plantilla para nuestro portal y luego, según corresponda, setear diferentes vistas en cada una de las secciones definidas. Para ello, en este proyecto se pondrán 2 vistas definidas en el archivo tiles-config.xml (la del header y la del footer) y estas se mapearan para ser re direccionadas por Struts cada vez que se necesiten.

5.4 Patrones de Diseño

Este modelo MVC pertenece a una clasificación llamada Patrones de Diseño. Estos modelos de software, son soluciones a problemas muy comunes que se encuentran en el desarrollo de software. En específico, en el desarrollo de software con el lenguaje de programación java, se pueden ver que existen muchísimos, como pueden ser el View Object, Data Transfer Object, Bussiness Delegate, Singleton, etc.

Estos patrones de diseño, son muy utilizados para proveer de mayor robustez al sistema, al ser soluciones a problemas recurrentes en el desarrollo de sistema. Debido a lo anterior, muchos programadores están muy familiarizados con ellos, y gracias a esto al encontrar uno de estos en cualquier sistema, automáticamente saben cómo funcionan y de ahí pueden tener un menor tiempo de familiarización con el sistema, aunado a que el tiempo de desarrollo se minimiza al de cierta manera ya haber estado en contacto con los mismos. El patrón de diseño que fungirá como la columna vertebral de esta aplicación, se conoce como Business Delegate.

Business Delegate

La idea de este patrón es intentar disminuir la complejidad que tiene acceder a componentes de negocios remotos. Los problemas que surgen son de 3 tipos:

1. La interfaz del componente puede cambiar, por lo que el cliente también deberá hacerlo (esto reduce la flexibilidad)

2. Cuando un cliente interactúa directamente con un componente de negocios remoto, esto puede requerir muchas invocaciones. Si a esto le sumamos que las invocaciones son remotas, los problemas son muchos mayores.

3. El 3er problema es toda la infraestructura con la que el cliente tiene que lidiar para acceder directamente a componentes remotos (JNDI, lidiar con los errores, etc.)

El 'business delegate' es una abstracción de la capa de negocios en el cliente que esconde los detalles 'sucios' de la interacción con objetos remotos, es decir se encarga de obtener los EJBs mediante JNDI, se encarga de instanciarlos, se encarga de lidiar con las excepciones remotas que para el cliente no tienen sentido transformándolas en excepciones de aplicación, se encarga de tratar nuevamente en el caso que esto ayude y hasta puede actuar como un cache (para no tener que obtener el EJB handle nuevamente por ejemplo)

De acuerdo a lo anteriormente expuesto, este sería el diagrama de secuencia que tendría que seguir el usuario:

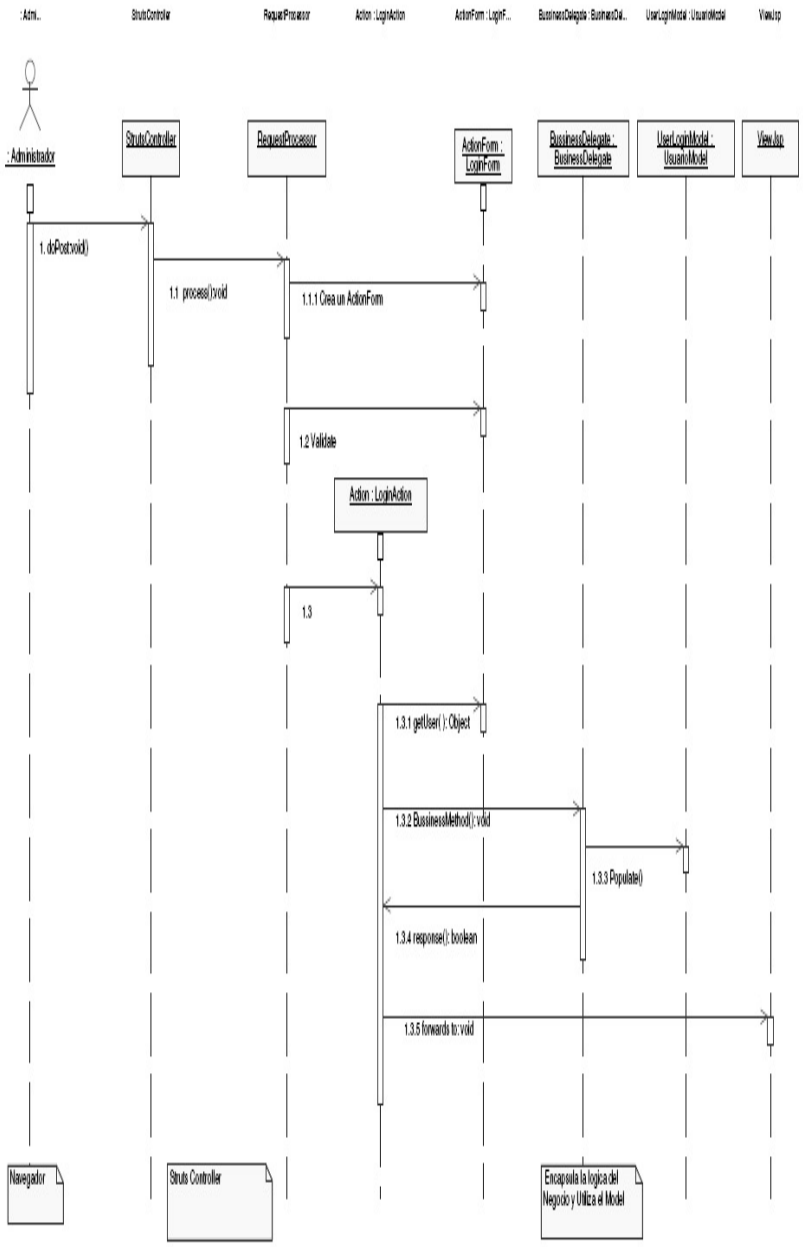


Figura 5.33 Diagrama de Secuencia 1

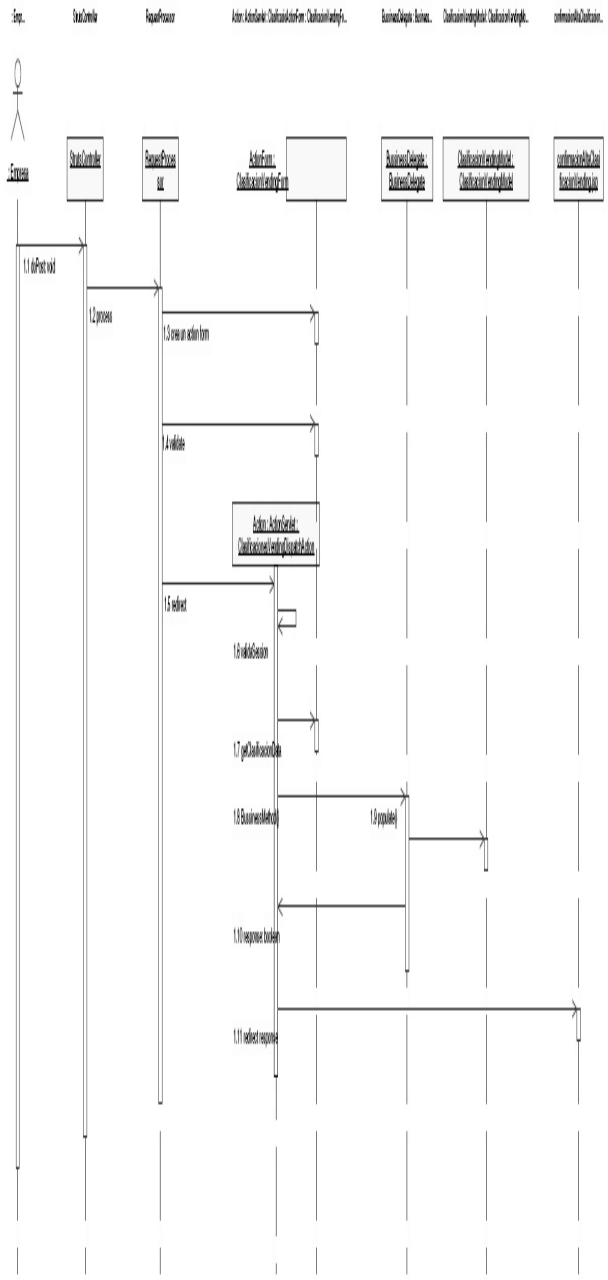


Figura 5.34 Diagrama de Secuencia 2

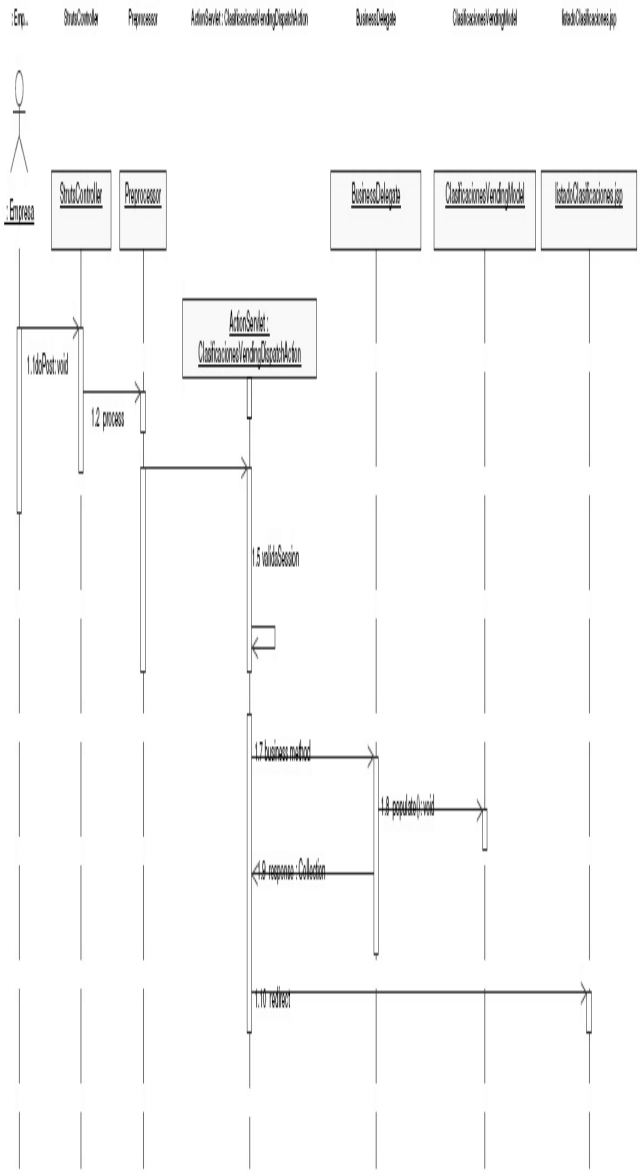


Figura 5.35 Diagrama de Secuencia 3

5.5 Hibernate como motor de persistencia

Trabajar con software orientado a objetos y bases de datos relacionales puede hacernos invertir mucho tiempo en los entornos actuales. Hibernate es una herramienta que realiza el *mapping* entre el mundo orientado a objetos de las aplicaciones y el mundo entidad-relación de las bases de datos en entornos Java. El término utilizado es ORM (object/relational mapping) y consiste en la técnica de realizar la transición de una representación de los datos de un modelo relacional a un modelo orientado a objetos y viceversa.

Hibernate no solo realiza esta transformación sino que nos proporciona capacidades para la obtención y almacenamiento de datos de la base de datos que nos reducen el tiempo de desarrollo.

Hibernate funciona asociando a cada tabla de la base de datos un Plain Old Java Object (POJO, a veces llamado Plain Ordinary Java Object). Un POJO es similar a una Java Bean, con propiedades accesibles mediante métodos setter y getter.

Para poder asociar el POJO a su tabla correspondiente en la base de datos, Hibernate usa los archivos hbm.xml.

Para la clase Usuario se usa el archivo Usuario.hbm.xml para mapearlo con la base de datos. En este archivo se declaran las propiedades del POJO y sus correspondientes nombres de columna en la base de datos, asociación de tipos de datos, referencias, etc.

Hibernate nos proporciona además un lenguaje con el que realizar consultas a la base de datos. Este lenguaje es similar a SQL y es utilizado para obtener objetos de la base de datos según las condiciones especificadas en el HQL. El uso de HQL nos permite usar un lenguaje intermedio que según la base de datos que usemos y el dialecto que especifiquemos será traducido al SQL dependiente de cada base de datos de forma automática y transparente.

Ahora bien, después de haber definido la arquitectura y el tratamiento de los datos , los diagramas de clases de la aplicación serán los presentados a continuación .

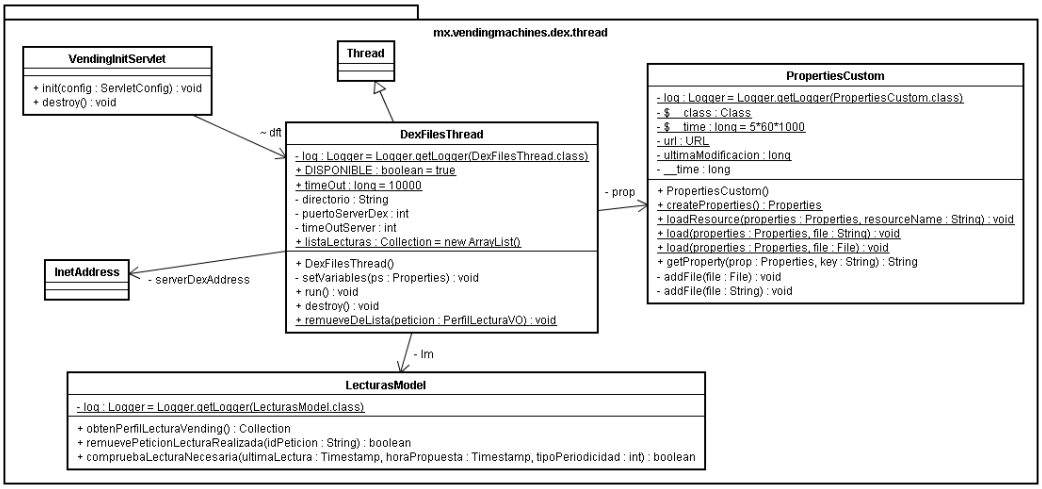


Figura 5.36 Diagrama de Clases 1

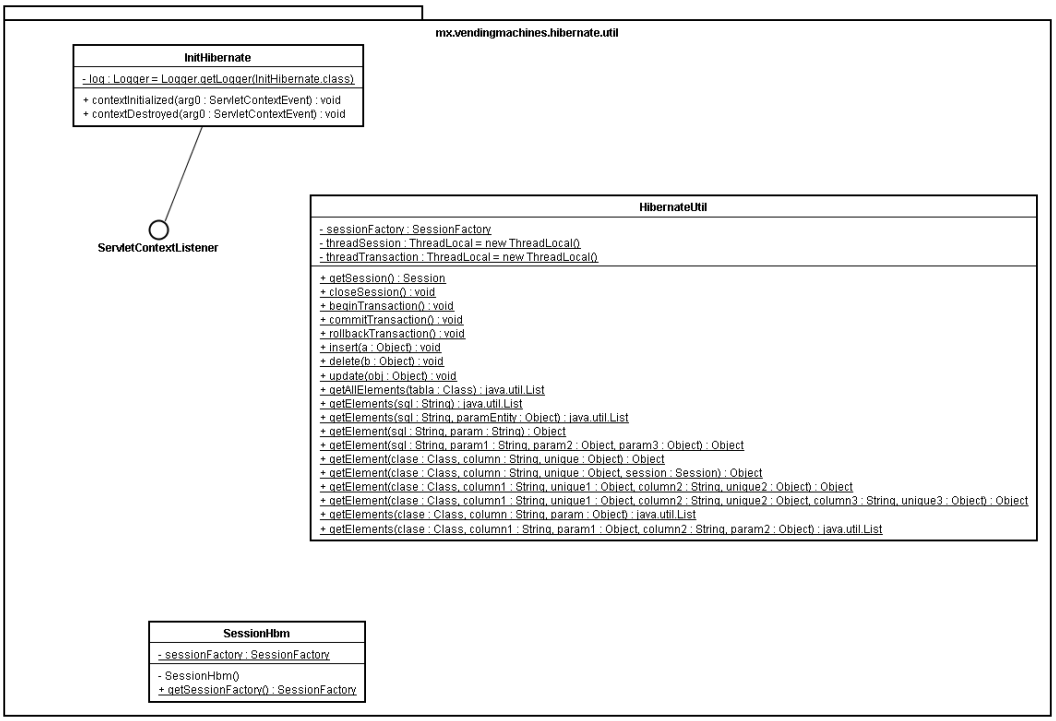


Figura 5.37 Diagrama de Clases 2

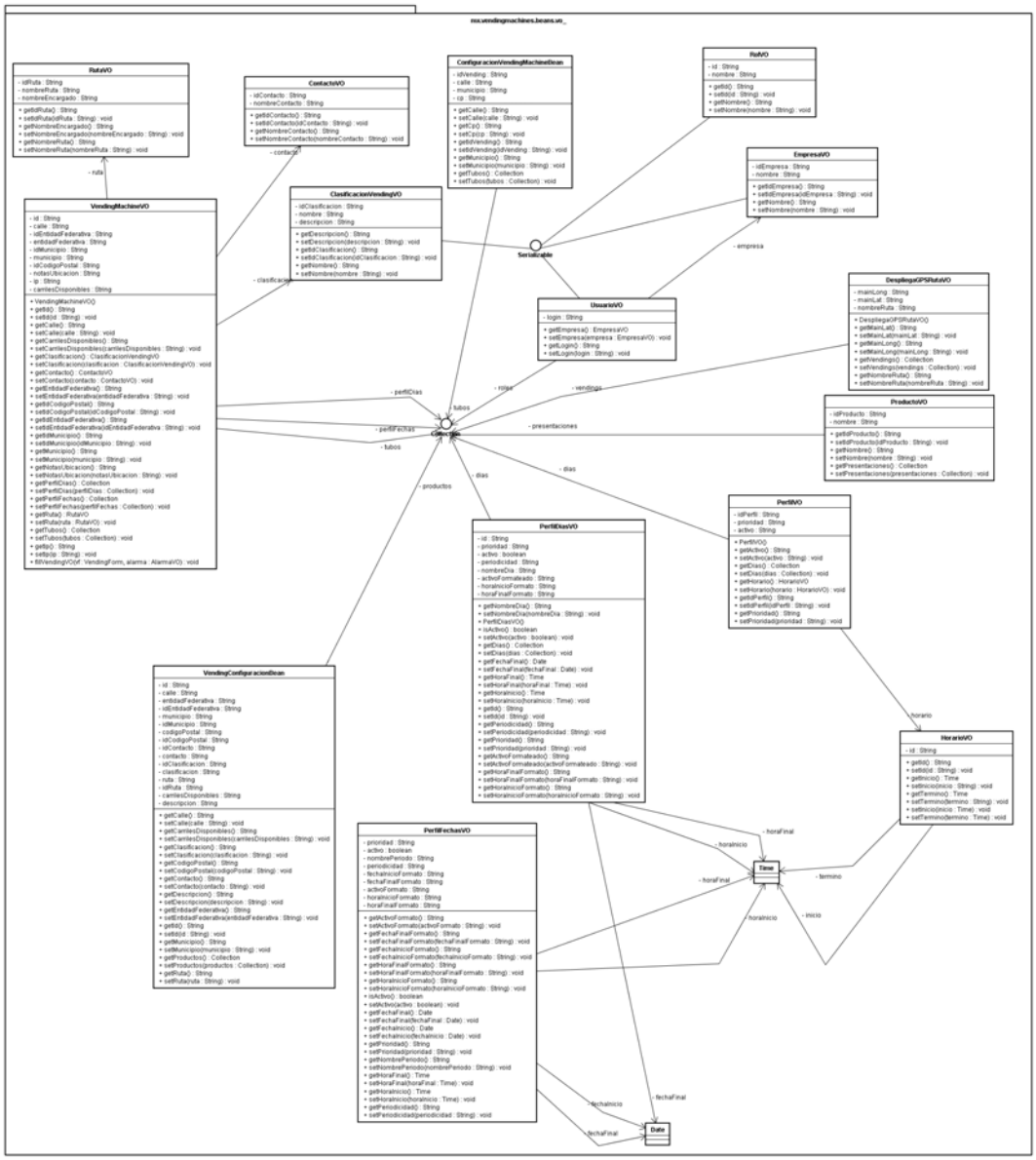


Figura 5.38 Diagrama de Clases 3

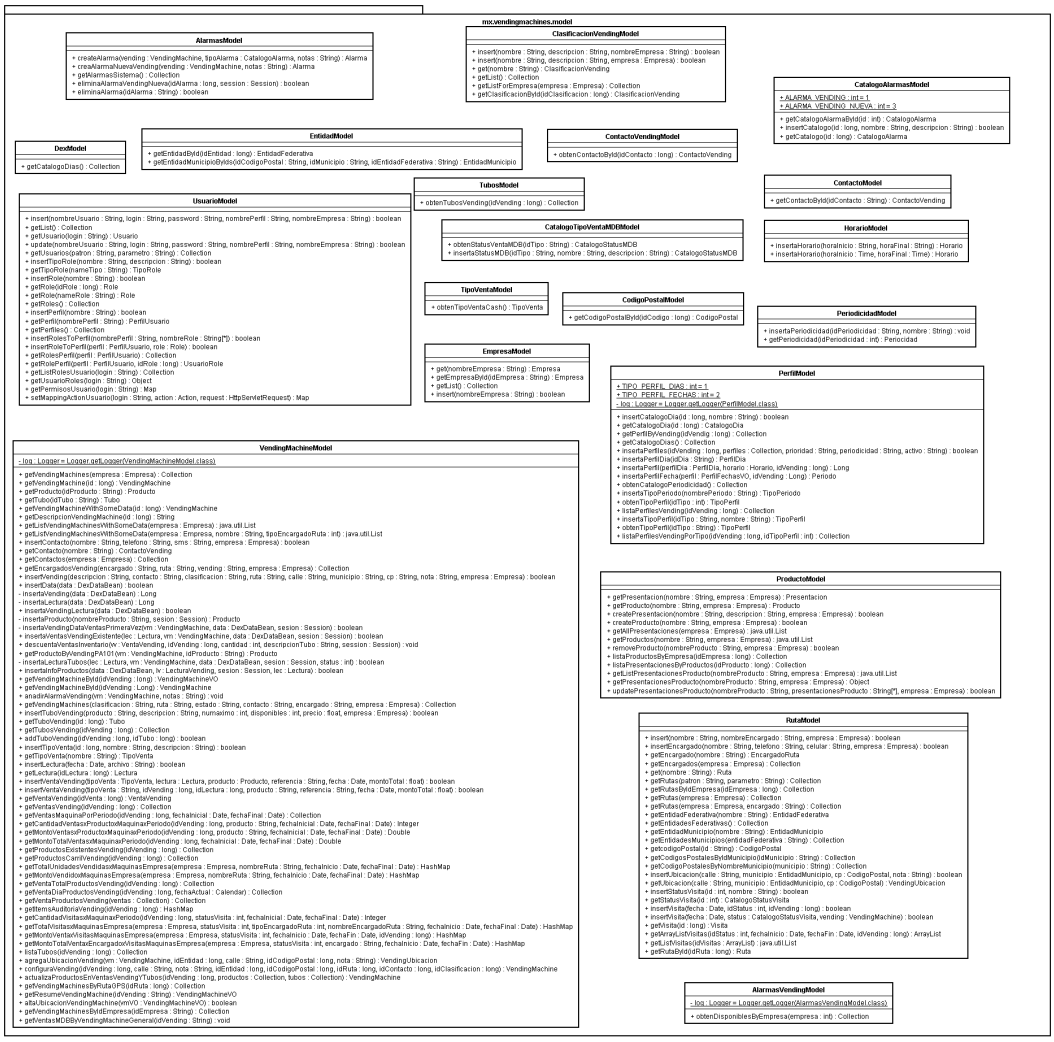


Figura 5.39 Diagrama de Clases 4

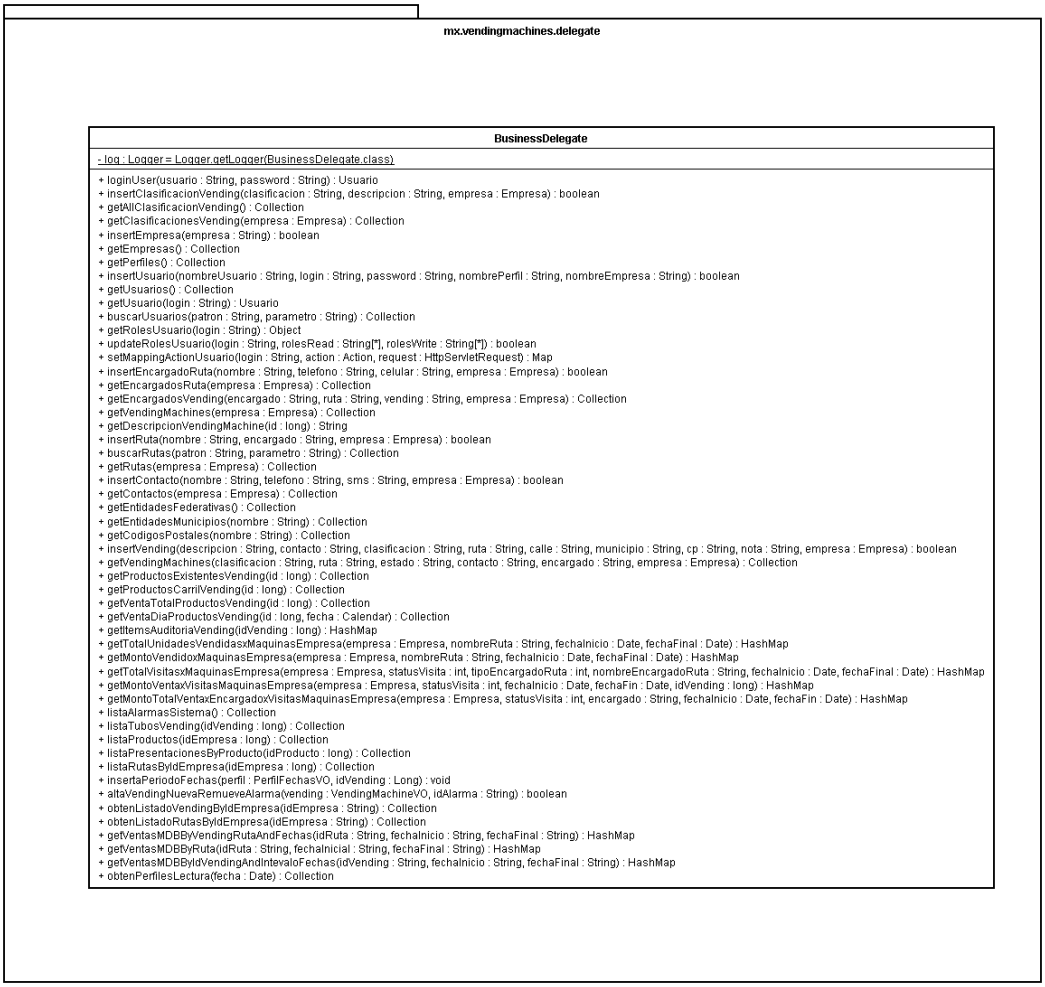


Figura 5.40 Diagrama de Clases 5

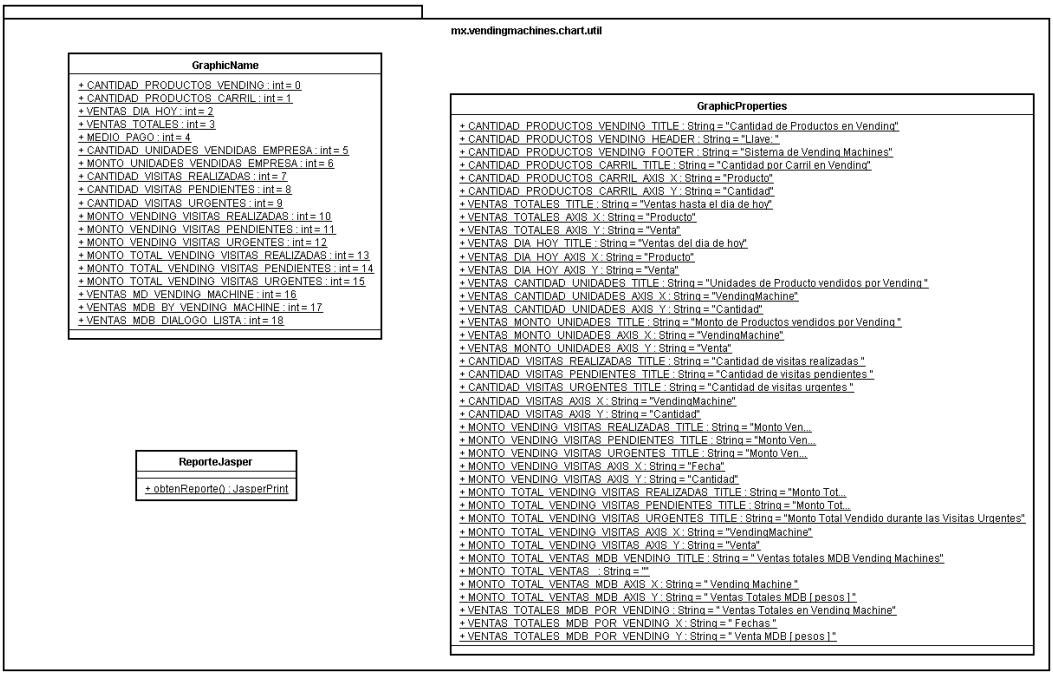


Figura 5.41 Diagrama de Clases 6

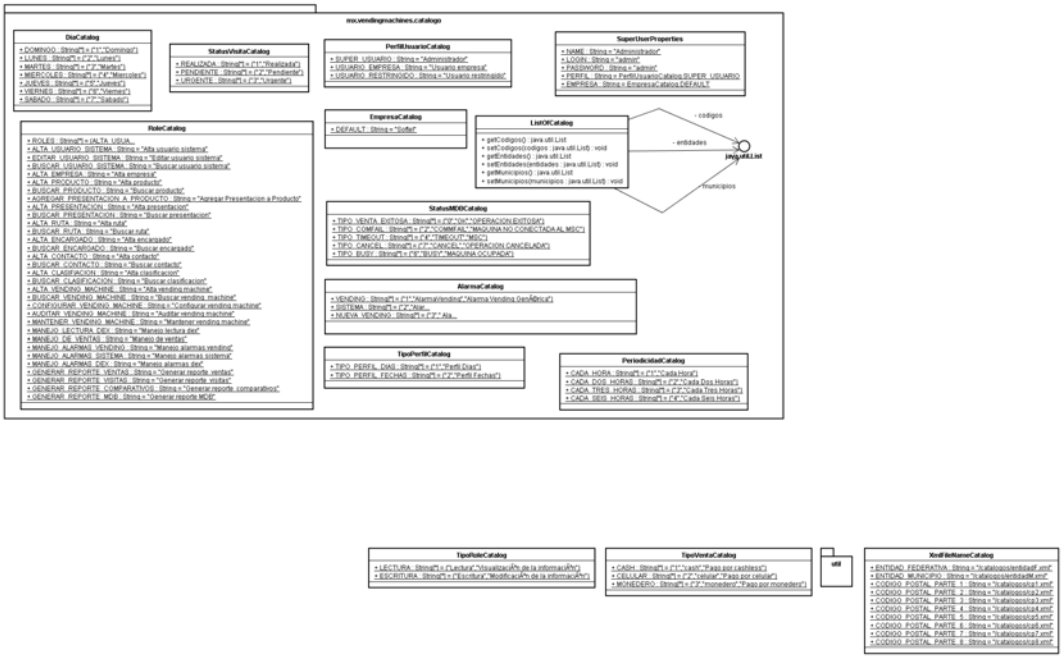
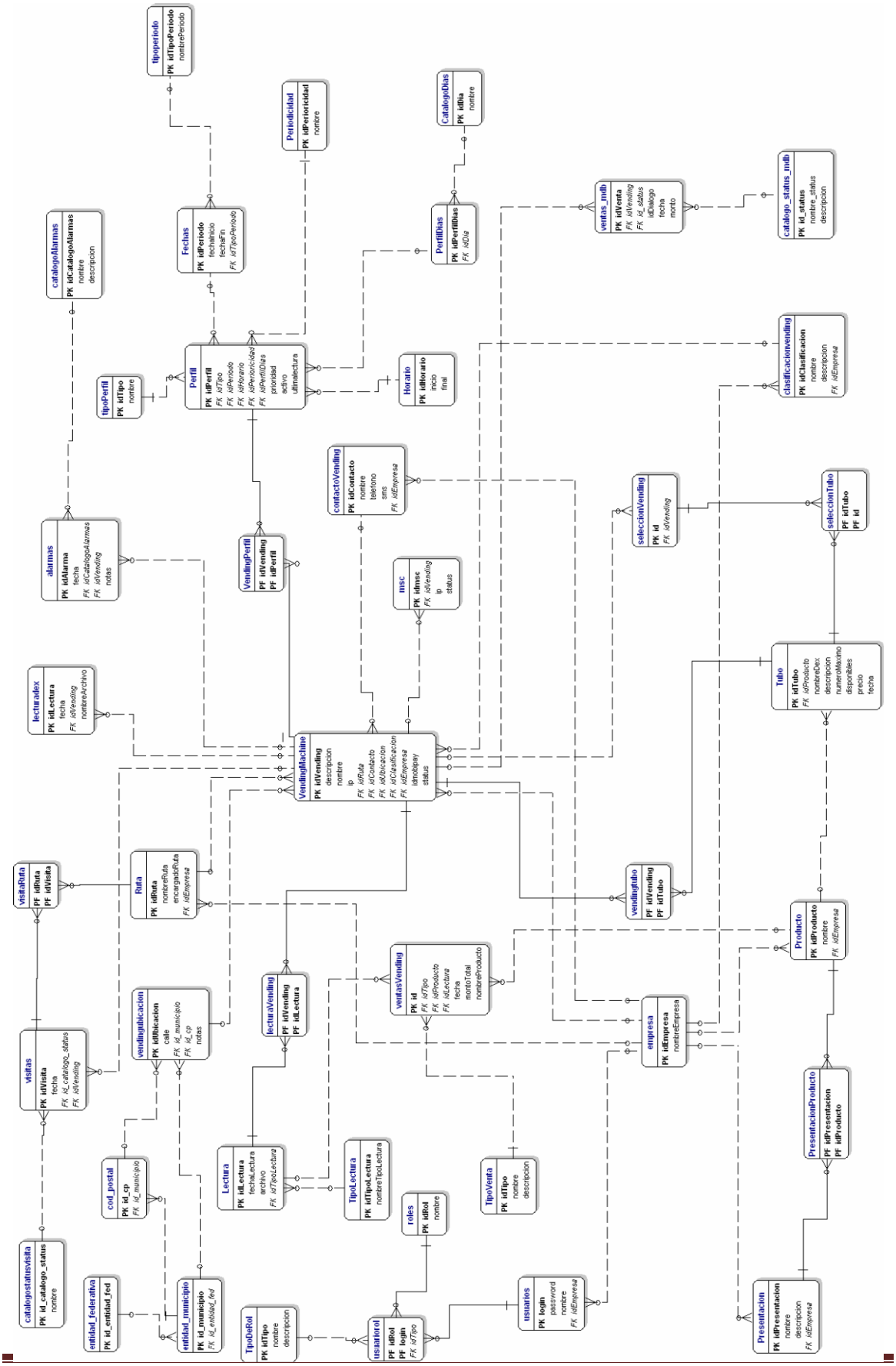


Figura 5.42 Diagrama de Clases 7

Conclusiones finales

La telemetría es una rama que poco a poco será explotada debido a la amplia variedad de ramas en las que puede ser aplicada. La industria de las vending machines es una de ellas , la cual poco a poco se va interesando en dicha tecnología la cual va echando mando de muchas otras alternas como el caso de Java en sus distintas ramas , para que se pueda aprovechar al máximo los servicios que se proveen . Cabe destacar que aunque este trabajo se enfoco más al diseño del sistema como tal, nos sirve para generar un enfoque más preciso de la cantidad de tiempo y personas que se necesitan para desarrollar el sistema propuesto, así como también nos puede ayudar a determinar un costo real, no solo del hardware necesario para implantarlo, sino también del software y el mantenimiento que se requerirá a lo largo de la vida útil del sistema. También podemos concluir que teniendo bases solidas utilizando los patrones de diseño y aplicándolo en herramientas tan poderosas como lo es J2EE , podremos generar aplicaciones lo suficientemente robustas para ser mantenidas a lo largo del tiempo y lo mejor , que al tener los distintos diagramas de UML provistos para la aplicación , podremos extender el mismo en caso de que se requiriera y con esto , aprovechar de una mejor manera el trabajo ya realizado al reutilizar código , en caso de requerirse ayudándose de un análisis de lo ya implantado , o diseñar nuevo código y agregarlo a los diagramas propuestos para luego poder implementarlo de una manera adecuada y sobre todo , en el que el equipo de trabajo sea capaz de entender debido a la familiarización con el sistema propuesto anteriormente . Esto no solo se puede ver reflejado en el servidor implementado con J2EE, sino también en el cliente diseñado, puesto que al surgir nuevas tecnologías en módems GPRS, gracias a los diagramas propuestos tendremos más de la mitad de la tarea hecha puesto que tenemos una idea bastante clara de lo que se busca en los clientes y gracias a la experiencia que nos dé la utilización de los mismos, podremos mejorar los mismos para una mejor calidad en el servicio que se provea.

Apéndice A
Base de Datos y Diccionario de Datos


tipoperiodo
PK idTipoPeriodo
nombrePeriodo
Periodicidad
PK idPeriodicidad
nombre
CatalogoDias
PK idDia
nombre
catalogoAlarmas
PK idCatalogoAlarmas
nombre
descripcion
Fechas
PK idPeriodo
fechaInicio
fechaFin
FK idTipoPeriodo
PerfilDias
PK idPerfilDias
FK idDia
ventas_mdb
PK idVenta
FK idVending
FK id_status
idDialogo
fecha
monto
catalogo_status_mdb
PK id_status
nombre_status
descripcion
Perfil
PK idPerfil
FK idTipo
FK idPeriodo
FK idHorario
FK idPeriodicidad
FK idPerfilDias
prioridad
activo
ultimaLectura
Horario
PK idHorario
inicio
final
TipoPerfil
PK idTipo
nombre
clasificacionvending
PK idClasificacion
nombre
descripcion
FK idEmpresa
contactoVending
PK idContacto
nombre
telefono
sms
FK idEmpresa
alarmas
PK idAlarma
fecha
FK idCatalogoAlarmas
FK idVending
notas
VendingPerfil
PF idVending
PF idPerfil
msc
PK idmsc
FK idVending
ip
status
seleccionVending
PK id
FK idVending
seleccionTubo
PF idTubo
PF id
lecturadex
PK idLectura
fecha
FK idVending
nombreArchivo
VendingMachine
PK idVending
descripcion
nombre
ip
FK idRuta
FK idContacto
FK idUbicacion
FK idClasificacion
FK idEmpresa
idmodpasy
status
Tubo
PK idTubo
FK idProducto
nombreDex
descripcion
numeroMaximo
disponibles
precio
fecha
visitaRuta
PF idRuta
PF idVisita
Ruta
PK idRuta
nombreRuta
cargadaRuta
FK idEmpresa
vendingTubo
PF idVending
PF idTubo
Producto
PK idProducto
nombre
FK idEmpresa
vendingUbicacion
PK idUbicacion
calle
FK id_municipio
FK id_cp
notas
lecturaVending
PF idVending
PF idLectura
ventasVending
PK id
FK idTipo
FK idProducto
FK idLectura
fecha
montoTotal
nombreProducto
empresa
PK idEmpresa
nombreEmpresa
visitas
PK idVisita
fecha
FK id_catalogo_status
FK idVending
PresentacionProducto
PF idPresentacion
PF idProducto
cod_postal
PK id_cp
FK id_municipio
Lectura
PK idLectura
fechaLectura
archivo
FK idTipoLectura
TipoLectura
PK idTipoLectura
nombreTipoLectura
TipoVenta
PK idTipo
nombre
descripcion
roles
PK idRol
nombre
catalogostatusvisita
PK id_catalogo_status
nombre
entidad_federativa
PK id_entidad_fed
entidad_municipio
PK id_municipio
FK id_entidad_fed
TipoDeRol
PK idTipo
nombre
descripcion
usuariorol
PF idRol
PF login
FK idTipo
usuarios
PK login
password
nombre
FK idEmpresa
Presentacion
PK idPresentacion
nombre
descripcion
FK idEmpresa

Lista de Entidades

Entidad	Llave	# Atributos	Descripción
alarmas	idAlarma	5	Tabla que almacena alarmas generadas por vending machines o por eventos del sistema
catalogoAlarmas	idCatalogoAlarmas	3	Tabla que almacena los tipos de alarmas que pueden existir en el sistema de control de vending machines
CatalogoDias	idDia	2	Tabla que almacena el catalogo de dias de la semana , de lunes a sabado para el uso en lecturas
catalogostatusvisita	id_catalogo_status	2	Tabla que almacena los status que puede tener una visita de acuerdo a lo que el encargado de la ruta encuentre cuando la realice
catalogo_status_mdb	id_status	3	Tabla que almacena el catalogo de todos los status de ventas mdb
clasificacionvending	idClasificacion	4	Tabla que almacena el catalogo de clasificaciones de vending , ya que estas pueden estar clasificadas bajo muchos rubros como son : vendings tester , vendings para la industria

			alimentaria , vendings para empresas , etc
cod_postal	id_cp	2	Tabla que almacena el catalogo de todos los codigos postales de Mexico
contactoVending	idContacto	5	Tabla que almacena el contacto que se tiene en el sitio donde se encuentra la vending , esto es para atenderla en caso de que surgiera algun imprevisto
empresa	idEmpresa	2	Tabla que almacena el Catalogo de empresas que estan habilitadas para hacer uso del sistema
entidad_federativa	id_entidad_fed	1	Tabla que almacena el catalogo de entidades federativas
entidad_municipio	id_municipio	2	Catalogo que almacena los municipios con los codigos postales que le corresponden
Fechas	idPeriodo	4	Tabla que almacena intervalos de fechas en las que se leeran las vending machines
Horario	idHorario	3	tabla que almacena los horarios de los perfiles que se

			registran para lecturas de vending machines
Lectura	idLectura	4	Tabla en la que se almacenan todas las lecturas de Dex que se realicen en las Vending Machines
lecturadex	idLectura	4	Tabla que almacena las lecturas dex que se realicen a determinada vending machine
lecturaVending	idVending, idLectura	2	Tabla que liga las lecturas con la vending a la que pertenecen , ya que una vending puede tener muchas lecturas asociadas
msc	idmsc	4	tabla que almacena la informacion del msc que es el dispositivo que esta ligado a una vending machine para realizar las auditorias por medio del protocolo dex
Perfil	idPerfil	9	Tabla que almacena un perfil de lectura dex , estos pueden ser por dias o por intervalos de fechas
PerfilDias	idPerfilDias	2	
Periodicidad	idPerioricidad	2	Tabla que almacena la periodicidad en la que se leeran las vending , se establecen por

			defecto cada hora , cada dos horas y cada tres horas
Presentacion	idPresentacion	4	Tabla que almacena las presentaciones disponibles en el sistema
PresentacionProducto	idPresentacion, idProducto	2	Tabla para relacionar presentaciones con productos , ya que un producto puede tener muchas presentaciones
Producto	idProducto	3	Catalogo de productos disponibles en el sistema , cabe aclarar que un producto pertenece a una sola empresa esto con el objeto de que las empresas no puedan ver los tipos de productos que se manejan
roles	idRol	2	Tabla que almacena los roles que existen para este sistema
Ruta	idRuta	4	Tabla que almacena las rutas que existen para las vending machines
seleccionTubo	idTubo, id	2	Tabla que establece que tubos pertenecen a que seleccion de vending Machine

seleccionVending	id	2	tabla que almacena las selecciones que existen de productos en cada vending
TipoDeRol	idTipo	3	Catalogo que almacena los roles que pueden tener los usuarios del sistema
TipoLectura	idTipoLectura	2	Catalogo de tipo de lecturas existentes en Dex
tipoPerfil	idTipo	2	Catàlogo de Tipo de Perfiles de Lectura Dex
tipoperiodo	idTipoPeriodo	2	Tabla que almacena el tipo de periodo para poder llevar un mejor control , por ejemplo : periodo vacacional , periodo intersemestral , etc
TipoVenta	idTipo	3	Catalogo de Tipos de Venta , esto para poder dar la opción de manejar màs tipos de venta
Tubo	idTubo	8	Tabla que almacena la informacion que nos reporta dex de los tubos de cada una de las vending machines
usuariorol	idRol, login	3	Tabla que romple la relacion de muchos a muchos con la tabla roles
usuarios	login	4	Tabla que almacena la informacion

			relativa a los usuarios
VendingMachine	idVending	11	Tabla en la que se almacena los datos importantes de una vending machine
VendingPerfil	idVending, idPerfil	2	Tabla que almacena todos los perfiles de lectura que pertenecen a una vending machine
vendingtubo	idVending, idTubo	2	Tabla que almacena la relacion de tubos con vending machines
vendingubicacion	idUbicacion	5	Tabla que almacena las ubicaciones de las vending machines
ventasVending	id	7	Tabla que almacena las ventas que lleva a cabo cada vending machine
ventas_mdb	idVenta	6	Tabla que almacena las ventas por mdb registradas por el msc
visitaRuta	idRuta, idVisita	2	Tabla que rompe la relacion mucho a muchos de visitas contra rutas
visitas	idVisita	4	Tabla que almacena las visitas realizadas a la vending asi como tambien el status que se tuvo al finalizar dicha visita

Detalle de Entidades

Entidad: alarmas

Descripción	Tabla que almacena alarmas generadas por vending machines o por eventos del sistema

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?
PK	idAlarma	BIGSERIAL	Si
	fecha	TIMESTAMP	Si
FK	idCatalogoAlarmas	INTEGER	No
FK	idVending	INTEGER	No
	notas	CHARACTER	No

Entidad:catalogoAlarmas

Detalles:

Descripción	Tabla que almacena los tipos de alarmas que pueden existir en el sistema de control de vending machines

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?
PK	idCatalogoAlarmas	SERIAL	Si
	nombre	CHARACTER VARYING	Si
	descripcion	CHARACTER VARYING	No

Entidad:CatalogoDias

Detalles:

Descripción	Tabla que almacena el catalogo de dias de la semana , de lunes a sabado para el uso en lecturas

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?
PK	idDia	SERIAL	Si
	nombre	CHARACTER	No

Entidad:catalogostatusvisita

Detalles:

Descripción	Tabla que almacena los status que puede tener una visita de acuerdo a lo que el encargado de la ruta encuentre cuando la realice

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?
PK	id_catalogo_status	SERIAL	Si
	nombre	CHARACTER VARYING	No

Entidad:catalogo_status_mdb

Detalles:

Descripción	Tabla que almacena el catalogo de todos los status de ventas mdb

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?
PK	id_status	INTEGER	Si
	nombre_status	CHARACTER VARYING	Si
	descripcion	CHARACTER VARYING	Si

Entidad:clasificacionvending

Detalles:

Descripción	Tabla que almacena el catalogo de clasificaciones de vending , ya que estas pueden estar clasificadas bajo muchos rubros como son :

	vendings tester , vendings para la industria alimentaria , vendings para empresas , etc

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?
PK	idClasificacion	SERIAL	Si
	nombre	CHARACTER VARYING	No
	descripcion	CHARACTER VARYING	No
FK	idEmpresa	INTEGER	No

Entidad:cod_postal

Detalles:

Descripción	Tabla que almacena el catalogo de todos los codigos postales de Mexico

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?
PK	id_cp	CHARACTER VARYING	Si
FK	id_municipio	CHARACTER	Si

Entidad:contactoVending

Detalles:

Descripción	Tabla que almacena el contacto que se tiene en el sitio donde se encuentra la vending , esto es para atenderla en caso de que surgiera algun imprevisto

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?
PK	idContacto	SERIAL	Si
	nombre	CHARACTER VARYING	Si
	telefono	CHARACTER VARYING	Si
	sms	CHARACTER	No
FK	idEmpresa	INTEGER	No

Entidad:empresa

Detalles:

Descripción	Tabla que almacena el Catalogo de empresas que estan habilitadas para hacer uso del sistema

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idEmpresa	SERIAL	Si	
	nombreEmpresa	CHARACTER	No	

Entidad:entidad_federativa

Detalles:

Descripción	Tabla que almacena el catalogo de entidades federativas

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	id_entidad_fed	CHARACTER	Si	

Entidad:entidad_municipio

Detalles:

Descripción	Catalogo que almacena los municipios con los codigos postales que le corresponden

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	id_municipio	CHARACTER	Si	
FK	id_entidad_fed	CHARACTER	No	

Entidad:Fechas

Detalles:

Descripción	Tabla que almacena intervalos de fechas en las que se leeran las vending machines

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción

PK	idPeriodo	INTEGER	Si
	fechaInicio	TIME	Si
	fechaFin	TIME	Si
FK	idTipoPeriodo	INTEGER	No

Entidad:Horario

Detalles:

Descripción	tabla que almacena los horarios de los perfiles que se registran para lecturas de vending machines

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idHorario	BIGSERIAL	Si	
	inicio	TIME	Si	
	final	TIME	Si	

Entidad:Lectura

Detalles:

Descripción	Tabla en la que se almacenan todas las lecturas de Dex que se realicen en las Vending Machines

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idLectura	SERIAL	Si	
	fechaLectura	TIMESTAMP	Si	
	archivo	TEXT	Si	
FK	idTipoLectura	INTEGER	No	

Entidad:lecturadex

Detalles:

Descripción	Tabla que almacena las lecturas dex que se realicen a determinada vending machine

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idLectura	SERIAL	Si	
	fecha	TIMESTAMP	Si	

FK	idVending	INTEGER	No
	nombreArchivo	CHARACTER VARYING	No

Entidad:lecturaVending

Detalles:

Descripción	Tabla que liga las lecturas con la vending a la que pertenecen , ya que una vending puede tener muchas lecturas asociadas

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK,FK	idVending	INTEGER	Si	
PK,FK	idLectura	INTEGER	Si	

Entidad:msc

Detalles:

Descripción	tabla que almacena la informacion del msc que es el dispositivo que esta ligado a una vending machine para realizar las auditorias por medio del protocolo dex

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idmsc	INTEGER	Si	
FK	idVending	INTEGER	No	
	ip	CHARACTER VARYING	Si	
	status	CHARACTER	Si	

Entidad:Perfil

Detalles:

Descripción	Tabla que almacena un perfil de lectura dex , estos pueden ser por dias o por intervalos de fechas

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idPerfil	BIGSERIAL	Si	
FK	idTipo	INTEGER	Si	
FK	idPeriodo	INTEGER	No	

FK	idHorario	INTEGER	Si
FK	idPerioricidad	INTEGER	Si
FK	idPerfilDias	INTEGER	No
	prioridad	INTEGER	Si
	activo	BOOLEAN	Si
	ultimalectura	TIMESTAMP	No

Entidad:PerfilDias

Detalles:

Descripción

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idPerfilDias	BIGSERIAL	Si	
FK	idDia	INTEGER	No	

Entidad:Periodicidad

Detalles:

Descripción	Tabla que almacena la periodicidad en la que se leeran las vending , se establecen por defecto cada hora , cada dos horas y cada tres horas

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idPerioricidad	INTEGER	Si	
	nombre	CHARACTER	No	

Entidad:Presentacion

Detalles:

Descripción	Tabla que almacena las presentaciones disponibles en el sistema

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idPresentacion	SERIAL	Si	
	nombre	CHARACTER	No	
	descripcion	CHARACTER	No	
FK	idEmpresa	INTEGER	No	

Entidad:PresentacionProducto

Detalles:

Descripción	Tabla para relacionar presentaciones con productos , ya que un producto puede tener muchas presentaciones

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK,FK	idPresentacion	INTEGER	Si	
PK,FK	idProducto	INTEGER	Si	

Entidad:Producto

Detalles:

Descripción	Catalogo de productos disponibles en el sistema , cabe aclarar que un producto pertenece a una sola empresa esto con el objeto de que las empresas no puedan ver los tipos de productos que se manejan

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idProducto	BIGSERIAL	Si	
	nombre	CHARACTER	No	
FK	idEmpresa	INTEGER	No	

Entidad:roles

Detalles:

Descripción	Tabla que almacena los roles que existen para este sistema

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idRol	SERIAL	Si	
	nombre	CHARACTER VARYING	No	

Entidad:Ruta

Detalles:

Descripción	Tabla que almacena las rutas que existen para las vending machines

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idRuta	SERIAL	Si	
	nombreRuta	CHARACTER	No	
	encargadoRuta	CHARACTER	No	
FK	idEmpresa	INTEGER	No	

Entidad:seleccionTubo

Detalles:

Descripción	Tabla que establece que tubos pertenecen a que seleccion de vending machine

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK,FK	idTubo	INTEGER	Si	
PK,FK	id	INTEGER	Si	

Entidad:seleccionVending

Detalles:

Descripción	tabla que almacena las selecciones que existen de productos en cada vending

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	id	SERIAL	Si	
FK	idVending	INTEGER	No	

Entidad:TipoDeRol

Detalles:

Descripción	Catalogo que almacena los roles que pueden tener los usuarios del sistema

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción

PK	idTipo	SERIAL	Si
	nombre	CHARACTER	No
	descripcion	CHARACTER	No

Entidad:TipoLectura

Detalles:

Descripción	Catalogo de tipo de lecturas existentes en Dex

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idTipoLectura	INTEGER	Si	
	nombreTipoLectura	CHARACTER VARYING	Si	

Entidad:tipoPerfil

Detalles:

Descripción	

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idTipo	INTEGER	Si	
	nombre	CHARACTER VARYING	Si	

Entidad:tipoperiodo

Detalles:

Descripción	Tabla que almacena el tipo de periodo para poder llevar un mejor control , por ejemplo : periodo vacacional , periodo intersemestral , etc

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idTipoPeriodo	INTEGER	Si	
	nombrePeriodo	CHARACTER VARYING	Si	

Entidad:TipoVenta

Detalles:

Descripción	

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idTipo	SERIAL	Si	
	nombre	CHARACTER VARYING	No	
	descripcion	CHARACTER VARYING	No	

Entidad:Tubo

Detalles:

Descripción	Tabla que almacena la informacion que nos reporta dex de los tubos de cada una de las vending machines

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idTubo	BIGSERIAL	Si	
FK	idProducto	INTEGER	No	
	nombreDex	CHARACTER	No	
	descripcion	CHARACTER	No	
	numeroMaximo	INTEGER	No	
	disponibles	CHARACTER	No	
	precio	MONEY	No	
	fecha	TIMESTAMP	Si	

Entidad:usuariorol

Detalles:

Descripción	Tabla que romple la relacion de muchos a muchos con la tabla roles

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?
PK,FK	idRol	INTEGER	Si
PK,FK	login	CHARACTER VARYING	Si
FK	idTipo	INTEGER	No

Entidad:usuarios

Detalles:

Descripción	Tabla que almacena la informacion relativa a los usuarios

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	login	CHARACTER VARYING	Si	
	password	CHARACTER VARYING	No	
	nombre	CHARACTER	No	
FK	idEmpresa	INTEGER	No	

Entidad:VendingMachine

Detalles:

Descripción	Tabla en la que se almacena los datos importantes de una vending machine

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idVending	BIGSERIAL	Si	
	descripcion	CHARACTER	No	
	nombre	CHARACTER	No	
	ip	CHARACTER	No	
FK	idRuta	INTEGER	No	
FK	idContacto	INTEGER	No	
FK	idUbicacion	INTEGER	No	
FK	idClasificacion	INTEGER	No	
FK	idEmpresa	INTEGER	No	
	idmobipay	CHARACTER VARYING	No	
	status	CHARACTER	Si	

Entidad:VendingPerfil

Detalles:

Descripción	Tabla que almacena todos los perfiles de lectura que pertenecen a

una vending machine

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK,FK	idVending	INTEGER	Si	
PK,FK	idPerfil	INTEGER	Si	

Entidad:vendingtubo

Detalles:

Descripción	Tabla que almacena la relacion de tubos con vending machines

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK,FK	idVending	INTEGER	Si	
PK,FK	idTubo	INTEGER	Si	

Entidad:vendingubicacion

Detalles:

Descripción	Tabla que almacena las ubicaciones de las vending machines

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idUbicacion	SERIAL	Si	
	calle	CHARACTER VARYING	Si	
FK	id_municipio	CHARACTER	No	
FK	id_cp	CHARACTER VARYING	No	
	notas	CHARACTER VARYING	No	

Entidad:ventasVending

Detalles:

Descripción	

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción

PK	id	SERIAL	Si
FK	idTipo	INTEGER	Si
FK	idProducto	INTEGER	No
FK	idLectura	INTEGER	No
	fecha	TIMESTAMP	Si
	montoTotal	MONEY	Si
	nombreProducto	CHARACTER VARYING	No

Entidad:ventas_mdb

Detalles:

Descripción	Tabla que almacena las ventas por mdb registradas por el msc

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK	idVenta	BIGINT	Si	
FK	idVending	INTEGER	No	
FK	id_status	INTEGER	No	
	idDialogo	CHARACTER VARYING	Si	
	fecha	TIMESTAMP	Si	
	monto	MONEY	Si	

Entidad:visitaRuta

Detalles:

Descripción	Tabla que rompe la relacion mucho a muchos de visitas contra rutas

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción
PK,FK	idRuta	INTEGER	Si	
PK,FK	idVisita	INTEGER	Si	

Entidad:visitas

Detalles:

Descripción	Tabla que almacena las visitas realizadas a la vending asi como tambien el status que se tuvo al finalizar dicha visita

Atributos:

Tipo Llave	Atributo	Tipo de Dato	NULOS?	Descripción

PK	idVisita	INTEGER	Si
	fecha	DATE	No
FK	id_catalogo_status	INTEGER	No
FK	idVending	INTEGER	No

Apéndice B

Catálogo de etiquetas del archivo DEX (Versión reducida)

ID	*Nombre*	*Descripción*	*Min*	*Max*
CA101	Numero de Serie de Mecanismo de Monedas	Este número solo puede ser puesto por el fabricante	01	20
CA102	Numero de Modelo del Mecanismo de Monedas	Numero de Modelo del Mecanismo de Monedas	01	20
CA103	Mecanismo de Monedas , Numero de Revisión de Software	Mecanismo de Monedas , Numero de Revisión de Software	01	04
CA104	Campo del fabricante	Campo del fabricante	01	12
CA201	Dinero de todas las ventas	no reseteable	01	08
CA202	Numero de ventas realizadas , no reseteable	no reseteable	01	06
CA203	Dinero de todas las ventas desde el ultimo reset	Dinero de todas las ventas desde el ultimo reset	01	06
CA204	Numero de ventas desde el ultimo reset		01	06
CA301	Dinero total desde el ultimo reset	Valor se resetea después de cada auditoria	01	08
CA302	Dinero total enviado a la caja de dinero	Valor reseteado después de cada auditoria	01	08
CA303	Dinero total en tubos desde el ultimo reset		01	08
CA304	Value of Bills In Since Last Reset		01	08
CA305	Dinero en Monedas en total de todas las fuentes	Dinero en Monedas en total de todas las fuentes , no reseteable	01	08
CA306	Dinero total desde la inicialización	No reseteable	01	08

CA401	Dinero total dispensado desde el ultimo reset	Se resetea después de cada auditoria	01	08
CA402	Dinero dispensado desde el ultimo reset	Valor reseteado en cada auditoria	01	08
CA403	Dinero total dispensado desde la inicialización	No reseteable	01	08
CA404	Dinero dispensado manualmente desde la inicialización	No reseteable	01	08
CA1701	Numero de tipo de Moneda	Numero utilizado en la especificación MDB.	01	03
CA1702	Valor de la Moneda		01	08
CA1703	Numero de Monedas en Tubo		01	08
CA1704	Numero de Monedas insertadas en Manual Fill	.	01	08
CA1705	Numero de Monedas dispensadas en Modo Controlado		01	08
EA201	ID Evento	Identificador del evento a reportar	01	20
EA202	Numero de eventos desde el ultimo reset		01	06
EA203	Numero de Eventos desde la Inicialización	Number of times that this event occurred. Non-Resettable.	01	06
EA204	Campo del Fabricante		01	12
EA301	Numero de Lecturas desde Inicialización		01	08

EA302	Fecha de esta lectura		06	06
EA303	Fecha de terminación de esta lectura		04	04
EA304	ID de terminal		01	20
EA305	Fecha de la última lectura		06	06
EA306	Hora de la lectura anterior		04	04
EA308	Campo de Fabricante		01	12
EA701	Numero de fallas de energía desde el ultimo reset	.	01	08
EA702	Numero de fallas de energía desde la inicialización		01	08
ID101	Numero de Serie de la Máquina	Número de identificación de la máquina	01	20
ID102	Número de Modelo de la Máquina	Número de Modelo de la Máquina o descripción	01	20
ID103	Número Estándar De Construcción de la máquina	Estándar de construcción de la máquina	01	04
ID104	Ubicación de la Máquina	Descripción del lugar de ubicación	01	30
ID105	Campo definido por el usuario .	Campo libre , depende del tipo de vending	01	12
ID708	Campo del fabricante	Campo del fabricante	1	12
LC101	Numero de Lista de Precios	Configura campo LA101	01	01
LC102	Numero de producto	Configura campo LA102	01	06
LC103	Precio	Configurar campo LA103	01	08
MA101	Numero de Serie de la Maquina		01	20
MA201	Numero de Serie de Maquina		01	20

MA501	Numero de Bloque		01	12
MA502	Datos de Configuración		01	100
MA503	Campo opcional #2		01	12
MA504	Campo opcional #3		01	12
MA505	Campo opcional #4		01	12
MA506	Campo opcional #5		01	12
MA507	Campo opcional #6		01	12
MA508	Campo opcional #7		01	12
MA509	Campo opcional #8		01	12
MA510	Campo opcional #9		01	12
MA511	Campo opcional #10		01	12
PA101	Numero de Producto	Numero de Selección de Producto	01	06
PA102	Precio de Producto		01	08
PA103	Id de Producto		01	20
PA104	Máxima capacidad de producto		01	04
PA105	Nivel de llenado estándar	.	01	04
PA106	Numero de cantidad a dispensar estándar.		01	04
PA201	Numero de productos vendidos desde la inicialización	.	01	06
PA202	Numero de ventas pagadas desde la inicialización	.	01	08

PA203	Numero de productos vendidos desde el ultimo reset		01	06
PA204	Dinero de productos vendidos desde el ultimo reset		01	08
PA501	Fecha de agotado el producto		06	06
PA502	Hora de agotado el producto		04	04
PA503	Número de veces que el producto agotado fue pedido		01	04
PC101	Numero de Producto	Configura campo PA101	01	08
PC102	Precio de Producto	C Configura campo PA102	01	08
PC103	Id de Producto	Configura campo PA103	01	20
PC104	Numero Máximo de Producto y	Configura campo PA104	01	04
PC105	Numero de productos que deben ser	Configura campo PA105	01	04
PC106	Cantidad estándar que debe ser dispensada	Configura campo PA106	01	04