ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Tesis:

“Sistema de Control de Inventarios Mediante el Uso de Tecnologías de Internet y Comunicación Inalámbrica”

Previo a la obtención del Título de:

Ingenieros en Computación

Especialización:

Sistemas de Información

Autores:

Hugo Camilo Robayo Ayala

Gerardo Rodrigo Romero Romero

Director de Tesis:

Ing. Galo Valverde

Guayaquil – Ecuador

2005

AGRADECIMIENTO

A Dios y a mis padres

Hugo Camilo Robayo Ayala

AGRADECIMIENTO

A Dios y a mis padres

Gerardo Rodrigo Romero Romero

DEDICATORIA

A mis padres,

Hugo Camilo Robayo Ayala

DEDICATORIA

A mis padres

Gerardo Rodrigo Romero Romero

TRIBUNAL DE GRADUACION

Ing. Hernán Gutiérrez	Ing. Galo Valverde.
Sub-Decano (E) de la FIEC	Director de Tesis
Ing. Guido Caicedo R.	Ing. Otilia Alejandro
Vocal Principal	Vocal Principal

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Proyecto de Tesis, nos corresponden exclusivamente; y el patrimonio intelectual de la misma a la ESCUELA SUPERIOR POLITÉCNIA DEL LITORAL".

(Reglamento de Graduación de la ESPOL)

Hugo Camilo Robayo Ayala

Gerardo Rodrigo Romero Romero

RESUMEN

La necesidad de reducir costos y aumentar la productividad en el trabajo de los personas que realizan los ingresos de inventarios en la bodega de la ESPOL, junto con la aparición de dispositivos móviles inalámbricos con mayores capacidades en poder de procesamiento, tamaño de la memoria y vías de comunicación son la base para el desarrollo de un sistema automatizado que ayude en el trabajo del personal de bodega que se lo ha llamado: Sistema de Control de Inventarios (SCI).

Actualmente, la ESPOL dispone de un sistema de cómputo para el área financiera, el cual también maneja la información de inventarios llamado SAF. El SAF está desarrollado sobre la plataforma de Visual Age de IBM y bajo el lenguaje de programación SmallTalk, debido a que este ambiente no permite la integración con dispositivos móviles el SCI esta desarrollado sobre las Tecnologías Java. El SCI se basa en una arquitectura multicapa y aprovecha las ventajas que ofrecen las especificaciones Java 2 Edición Empresarial (J2EE) y Java 2 Edición Mínima (J2ME). La integración con la plataforma actual de la ESPOL es hecha a través del acceso a la base de datos, es decir, los dos sistemas trabajan con los mismos datos.

El SCI está dividido en una aplicación cliente basada en J2ME y la aplicación que se ejecuta en el servidor de aplicaciones compatible con las

especificaciones J2EE. La aplicación cliente puede ejecutarse en cualquier dispositivo móvil que soporte las especificaciones J2ME.

ÍNDICE GENERAL

ÍNDICE DE ANEXOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Con el desarrollo de nuevas Tecnologías de Información y Comunicación (TIC) y el adelanto en las comunicaciones inalámbricas, se ha creado un creciente campo para el desarrollo de nuevas soluciones a diferentes problemas.

Un problema que se puede resolver con estas tecnologías es la falta de eficiencia en la recolección de datos durante el proceso de ingreso de inventarios en la ESPOL.

Por tal motivo, se ha procedido a desarrollar una herramienta para automatizar el proceso de ingreso y verificación, llamado Sistema de Control de Inventarios (SCI), haciendo uso de tecnologías inalámbricas y asistentes digitales personales (PDA).

CAPÍTULO 1

1. INTRODUCCIÓN Y PRESENTACIÓN

En este capítulo se detallan los antecedentes y se plantea el problema a solucionar. Además, se listan los objetivos y se definen los alcances en el desarrollo del proyecto.

Adicionalmente, se describe en el plan de proyecto las tareas ejecutadas en las etapas de investigación, análisis e implementación del sistema.

Por último, se proponen los criterios de evaluación y se especifican los resultados que se esperan obtener.

1.1 Antecedentes y Definición.

Con el desarrollo de nuevas Tecnologías de Información y Comunicación (TIC) y el adelanto en las comunicaciones inalámbricas, se ha creado un creciente campo para el desarrollo de nuevas soluciones a diferentes problemas.

Un problema que se puede resolver con estas tecnologías es la falta de eficiencia en la recolección de datos durante el proceso de ingreso de inventarios en la ESPOL. Algunos factores [1] que determinan la importancia de un buen control de inventarios son:

- Mantener a los empleados encargados del inventario lejos de los registros contables.
- Prevenir situaciones de déficit para mantener suficiente inventario disponible.
- Desperdiciar tiempo en la localización de mercadería y en la recolección de información durante el proceso de ingreso y verificación de inventarios.

Por tal motivo, se ha procedido a desarrollar una herramienta para automatizar el proceso de ingreso y verificación, llamado Sistema de Control de Inventarios (SCI)[1], haciendo uso de tecnologías inalámbricas y asistentes digitales personales (PDA)[2].

1.2 Objetivos

Los objetivos generales que se plantearon para el desarrollo del SCI estuvieron basados en las tareas que realizan

[1] SCI.- Sistema de Control de Inventarios, nombre corto del proyecto de Tesis: "Sistema de Control de Inventarios mediante el uso de tecnología inalámbrica y computación móvil"
[2] PDA: Personal Digital Assistant, dispositivo de tamaño muy pequeño que combina computación, teléfono / fax, Internet y características de redes de computadoras

los encargados del departamento de bodega de la ESPOL y son los siguientes:

- Mejorar la productividad del personal encargado de la bodega (bodegueros[3]).
- Facilitar las tareas que realiza el personal administrativo[4].
- Ayudar a que el ingreso de inventarios se haga de manera más rápida y al menor costo posible.

Los objetivos particulares que han motivado el desarrollo del SCI son:

- Desarrollar un modelo de software que se aplique tecnologías inalámbricas y computación móvil al sistema actual de inventarios de la ESPOL.
- Investigar y aplicar las tecnologías que ofrezcan mayores beneficios en la implementación el SCI.

1.3 Alcances

Los alcances planteados para la implementación del SCI están basados principalmente en restricciones en la integración de procedimientos contables con los sistemas de la ESPOL:

---

[3] Bodegueros**.-** Personas responsables del ingreso y verificación en la bodega.

[4] Personal de Administración.- Responsables de proporcionar documentos para el ingreso de inventarios.

- El SCI automatiza los procesos de ingreso y verificación de inventarios.
- El SCI no modela, ni implementa procesos contables tendientes a compra o venta de inventarios.
- El SCI utiliza tecnologías inalámbricas y dispositivos móviles para el ingreso y verificación de inventarios.

El SCI es un sistema independiente a los sistemas existentes en la ESPOL, sin embargo, se logra la integración a estos mediante el acceso a las bases de datos de la ESPOL.

1.4 Plan de Proyecto.

En base a los aspectos técnicos del proyecto, las herramientas y tecnologías utilizadas se presenta el plan de proyecto, en el cual se detallan las tareas en el desarrollo del SCI.

1.4.1 Introducción al Proyecto

Los proyectos en los cuales se tiene que integrar sistemas desarrollados en diferentes plataformas, generalmente aparecen complejidades como: administrar conexiones a bases de datos, manejar transacciones, establecer comunicaciones y manejar la integridad de datos con sistemas existentes.

Por este motivo el SCI se basa en una arquitectura de cuatro capas como lo muestra la Figura 1, las cuales establecen niveles de desarrollo que resuelven estas complejidades, las capas se listan a continuación:

- Capa 0: Presentación Cliente.- Se ejecuta en un dispositivo móvil, procesa las entradas del usuario y almacena datos localmente, sobre los ingresos de inventarios.
- Capa 1: Presentación del servidor.- esta capa representa las interfaces a los servicios a los cuales los clientes tienen acceso.
- Capa 2: Lógica del negocio.- esta capa representa el modelo de la aplicación, con clases que abstraen los conceptos como: compromisos de compra, activos fijos, pedidos y las operaciones del proceso de ingreso de inventarios.
- Capa 3: Lógica de los datos.- esta capa representa los subsistemas externos que manejan la persistencia de los datos.

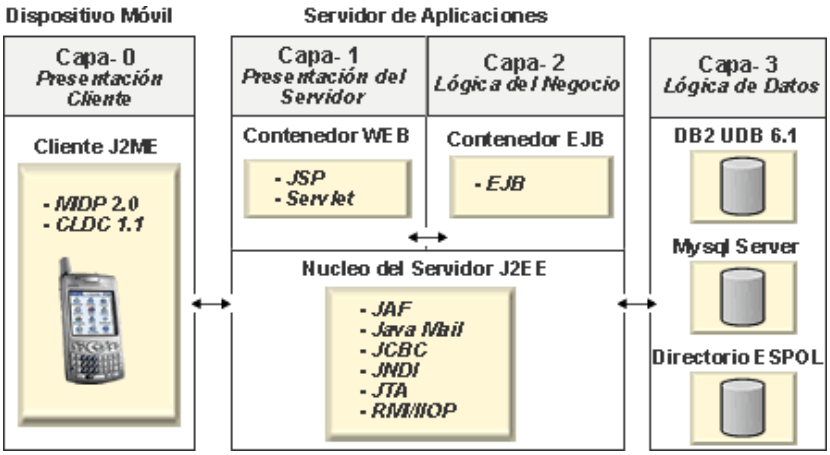


Figura 1.- Diagrama de bloques del Sistema de Control de Inventarios

En la capa 1 y 2 se ejecutan sobre una plataforma denominada Servidor de Aplicaciones, el cual ofrece un marco de trabajo y acceso a tecnologías especializadas que sirven de capa media entre el cliente y los datos.

Adicionalmente, el desarrollo de la aplicación cliente, la cual se ejecuta en el dispositivo móvil, representa enfrentarse a un nuevo paradigma de programación, debido a los limitados recursos que los dispositivos móviles ofrecen. En este sentido, para manejar eficientemente el modelo de la aplicación cliente, el diseño y arquitectura se basa en el patrón de diseño “Modelo, Vista, Controlador” (MVC) el cual es explicado en mayor detalle en el Capítulo 3.

1.4.2 Descripción de Herramientas y Tecnologías

Las herramientas usadas en el análisis, diseño e implementación del SCI fueron:

- Para el diseño de diagramas UML se utilizó *Poseidon for UML* 3.0[5].
- El Entorno de Desarrollo Integrado (IDE[6]) fue Elipse 3.1M4[7]. También fue usado el IDE *WebSphere Device Developer* 5.6[8] de IBM basado en Tecnología Eclipse.
- *Wireless Toolkit 2.1* de *Sun Microsystems Inc.* fue usado para las pruebas de desarrollo.

Los equipos usados para las pruebas de rendimiento fueron.

- Un dispositivo móvil *Palm Tungsten C* con sistema operativo Palm OS 5.2[9].

---

[5] Poseidon For UML.- www.gentleware.com
[6] IDE.- Entorno de Desarrollo Integrado (IDE, Integrated Development Environment)
[7] Eclipse 3.1M4.- Eclipse es un tipo de herramienta universal, un IDE abierto y extensible para todo y nada en particular., www.eclipse.org
[8] WebSphere Device Developer 6.1.- es el IDE de desarrollo de IBM, para sistemas basados en Java 2 Micro Edition, www.ibm.com
[9] Palm OS 5.2.- Sistema Operativo desarrollado por la Corporación Palm para dispositivos móviles, www.palmsource.com

- Un ruteador inalámbrico WIFi[10] D-Link DW-614+ que permite establecer una red inalámbrica compatible con estándar IEEE 802.11b.

Los sistemas que manejan la persistencia de datos del SCI son:

- Base de datos relacional DB2 UDB 6.1[11], la cual utiliza la ESPOL.
- Base de datos relacional MySql 3.5[12]
- Base de datos jerárquica con la información del Directorio de la ESPOL.

Además, el SCI usa las tecnologías Java 2 Enterprise Edition (J2EE)[13] y Java 2 Micro Edition (J2ME)[14]. Una descripción más detallada de las tecnologías se encuentra en el Capítulo 2.

1.4.3 Plan de Proyecto

Para la elaboración del plan de proyecto se conformó un equipo de 2 desarrolladores, un experto en el sistema

[10] WIFI.- Siglas de Wireless Fidelity, se lo relaciona con el estándar IEEE 802.11b, www.standars.ieee.org
[11] DB2 UDB 6.1.- Base de Datos desarrollada por IBM. www.ibm.com
[12] MySql.- Base de Datos de fuente abierta.
[13] J2EE.- Java 2 Platform, Enterprise Edition (J2EE) define el estándar para desarrollar componentes-basados en aplicaciones empresariales multicapa.
[14] J2ME.- Micro Edition of the Java 2 Platform provee un ambiente de aplicación que esta especialmente orientado las necesidades y comodidades de una vasto y creciente espacio de dispositivos móviles, incluyendo teléfonos móviles, pagers, PDA (Personal Digital Assistants), set-top boxes, y sistemas de telemática de vehículos.

financiero de la ESPOL y un experto en la base de datos de la ESPOL. La tabla 1 muestra la descripción de las tareas necesarias para el desarrollo del SCI.

Paso	Descripción
1	Preliminares Administrativos (aceptación del tema y temario)
1.1.	Adecuación de un lugar de trabajo
1.2.	Conseguir equipos para el desarrollo del Sistema
1.3.	Capacitación necesaria para iniciar el proyecto
2	Inicio de Actividades
2.1.	Recolección de Requerimientos
2.2	Especificación de requerimientos
3	Diseño
3.1.1	Diseño de Objetos
3.1.2	Diseño de la Interfaz de Usuario
3.1.3	Diseño de Modelo de Base de Datos(NN)
3.2	Revisión y Evaluación del Diseño
4	Implementación del Sistema
4.1.	Desarrollo de la aplicación para el dispositivo inalámbrico.
4.2.	Integración de los componentes desarrollados
4.3	Documentación Técnica
4.4	Documentación de usuario
5	Pruebas de Rendimiento
5.1	Pruebas propuestas.
5.2	Realización de las Pruebas (por componentes desarrollados)
5.3.	Resultados de las Pruebas.
6	Entregas
6.1.	Entregas de la aplicación inalámbrica.
6.2.	Entrega de documentación realizada.
7	Conclusiones

Tabla 1.- Tareas en el desarrollo del proyecto. Realizado por los autores.

1.5 Riesgos potenciales en la implementación en un ambiente de producción.

El uso de nuevas tecnologías ayuda a las instituciones en la mejora de procesos, las tareas para la implementación de estas mejoras pueden resultar tediosas y no contar con el apoyo

necesario de parte de los usuarios y administradores de los sistemas actuales.

A menudo varios sistemas en los cuales se ha invertido gran cantidad de tiempo y recursos no logran el objetivo para los cuales fueron creados. Los riegos más importantes en los que se puede incurrir durante la implementación del SCI se muestran en la tabla 2.

Descripción	PR	IM	PO	Responsable	Actividad	Tiempo
Nivel de aceptación del SCI	1	5	4	Líderes de Proyecto y desarrolladores	Capacitar al personal de bodega sobre uso del sistema	Antes de entrar en producción
Poco interés en nuevas tecnologías	1	3	2.1	Líderes de Proyecto	Informar sobre los beneficios de usar computación móvil	Antes de entrar en producción
Costo de la infraestructura	1	3	1.5	Líderes de Proyecto	Hacer una investigación en el mercado, sobre dispositivos disponibles para implementar el SCI	Antes de entrar en producción
Fallo de red inalámbrica de bodega	1	5	2,5	Administrador del sistema	Métodos alternativos de conexión de los dispositivos móviles	Sistema en operación

Tabla 2.- Matriz de Riesgos del SCI, PR.- Probabilidad, IM.- Impacto, PO.- Potencial. Realizado por autores.

1.6 Criterios de Evaluación

Los criterios de evaluación fueron utilizados para calificar la usabilidad y rendimiento de la aplicación cliente. Además, se presentan algunos criterios que fueron usados para calificar aspectos generales como los costos de desarrollo y pruebas,

importantes al momento de obtener conclusiones sobre el proyecto.

1.6.1 Criterios relacionados con la interfaz de usuario en un dispositivo móvil

Debido a que la aplicación cliente está implementada en dispositivos móviles, el diseño de la interfaz de usuario se vuelve de vital importancia, considerando las restricciones en cuanto a dispositivos de entrada y salida. A continuación se muestran estos criterios [2]:

- El uso de un lenguaje familiar al personal de bodega en los mensajes que muestra la aplicación y mantener la claridad de estos mensajes, es fundamental debido al reducido tamaño de la pantalla. No es recomendable utilizar mensajes muy largos, sin embargo, se debe proveer la suficiente información para que el usuario tome una acción.
- El correcto uso de los colores, consideración debida a las limitantes de color y tamaño de la pantalla en ciertos dispositivos móviles.

- El mantener la simplicidad, equilibrio y armonía se vuelve de vital importancia en el desarrollo de aplicaciones, si se toma en cuenta que en los dispositivos móviles inalámbricos no disponen de grandes recursos.
- La consistencia en la distribución de los botones, pantallas, distribución de la información en la aplicación.
- El tamaño, tipo de letra y fuente son aspectos importantes al momento de considerar que la aplicación se debe apreciar correctamente en diferentes dispositivos.

Si se siguen los criterios el sistema asegurará un nivel de usabilidad muy alto, no demandará demasiado tiempo en la capacitación y será incorporado rápidamente en la realización de las tareas del personal de bodega.

1.6.2 Criterios de navegación para un dispositivo móvil

De igual manera los criterios de navegación [2] que se tomaron en cuneta son:

- La interacción con el usuario debe ser sencilla y fácil. El sistema debe proveer un rápido aprendizaje.

- El sistema debe proveer control al usuario, es decir, en cada momento el usuario debe saber que esta se está realizando y pueda detener la ejecución del programa cuando él lo desee.
- Las animaciones, ¿Son necesarias?, dependiendo del tipo de tarea que se realice, las animaciones serán de gran ayuda. Un caso típico del uso de animaciones son aquellas que informan al usuario el avance de determinado proceso. Aun así, el sistema debe tomar en cuenta que los usuarios trabajan la mayor parte del tiempo que los dispositivos inalámbricos y su capacidad de procesamiento impide mostrar animaciones complejas.
- El usuario del sistema debe estar en la libertad de navegar por la aplicación y siempre hay que proveer la manera de regresar al punto de inicio.
- El sistema debe proveer la opción de salir del sistema así como la de ayuda.
- La utilización de sonidos es de importancia para alertar al usuario que el estado de determinado

proceso. Aun así su presencia no es del todo crítica y podrían ser omitidos.

- La utilización de imágenes o videos seria de gran utilidad en el SCI para la identificación de determinado producto.

Algunas de las ayudas extras deben ser analizadas e implementadas siempre y cuando representen beneficios claros a la aplicación, caso contrario, se puede incurrir en un consumo excesivo de recursos.

1.6.3 Otros criterios de evaluación

Los criterios relacionados con los costos de desarrollo e implementación del sistema se muestran a continuación.

- Costo de desarrollo e implementación del SCI.- En la implementación del SCI se incurre en costos de hardware y software, los gastos no sólo dependen del número de dispositivos móviles o el costo de la infraestructura inalámbrica, sino de todas las fases del desarrollo: análisis, diseño e implementación. Además, los costos de mantenimiento de equipo y software deben ser tomados en cuenta.

- Capacitación del personal: Aún cuando los usuarios tengan conocimiento sobre software o equipos similares se debe proveer una adecuada capacitación en el sistema.
- Material complementario: Los manuales deben ser claros y con lenguaje sencillo para que el usuario este en la capacidad de revisarlos y solucionar problemas.
- Mantenimiento y puesta al día: La compra de nuevos equipos o actualizaciones al sistema pueden ser requeridas.

Los criterios mencionados pueden ayudar a determinar si los esfuerzos que se realizan en el desarrollo del SCI caen en un rango normal, es decir, no involucran recursos, ni tiempo de manera excesiva.

1.7 Resultados Esperados

A continuación se detallan los resultados que se espera conseguir con la implementación del SCI, éstos pueden verificados en las conclusiones que se muestran el Capítulo 6.

- Mejoramiento en la portabilidad de la información. Datos completos sobre los inventarios se deben mantener en los dispositivos móviles, reduciendo el tiempo de acceso.
- Reducción de costos: actualización automática sobre sistema central, consolidando la información, eliminando errores y costos en el ingreso de datos.
- Perfeccionamiento de la eficiencia de la gestión: distribución electrónica y automatizada de los compromisos a ser ingresados, eliminando los costos de impresión de formularios de ingreso.
- Automatización del proceso: aprovechamiento de la inteligencia del dispositivo, la aplicación cliente debe guiar en el ingreso, reduciendo así los tiempos de capacitación.

Finalmente, se espera que este proyecto sirva de guía para futuros desarrollos en el área de la computación móvil en la ESPOL.

CAPÍTULO 2

2. MARCO TEÓRICO

Este capítulo hace énfasis el marco teórico para la implementación del SCI, además se hace muestra una revisión de conceptos sobre la computación móvil, así como, las aplicaciones, tendencias y consideraciones en este campo.

Adicionalmente, se analizan las tecnologías en las que se basa el estándar IEEE 802.11b para redes de área local inalámbrica y se explica los motivos para la elección de: herramientas, tecnologías y plataforma de desarrollo.

Finalmente, se menciona aspectos relacionados a la autenticación, autorización e integridad de datos manejados por el sistema, además de otros conceptos sobre seguridad en redes inalámbricas.

2.1 Introducción a la Computación Móvil

La computación móvil aparece como el paradigma dominante en el desarrollo de aplicaciones, además, con la

integración de tecnologías de comunicaciones tales como G3[15]; CDMA[16], Bluetooth[17], Wi-Fi[18], GPS[19], se ha logrado que los dispositivos móviles puedan comunicarse con otras aplicaciones y computadores.

Entre los dispositivos móviles se puede mencionar: PDA (tales como Palm[20] o Pocket PC[21]), gameboys[22], teléfonos móviles, reproductores digitales de audio, cámaras inteligentes y muchos más. Ahora, los fabricantes desarrollan dispositivos con mucha más capacidad de almacenamiento, poder de procesamiento, memoria, multimedia que hacen posible su crecimiento y desarrollo.

En conclusión, en el mundo se usan alrededor de 31 millones de PDAs, esta cifra no se compara con los cerca de 1.52 billones de teléfonos celulares vendidos hasta Diciembre del 2004. En estos momentos, los teléfonos celulares ofrecen más capacidades de PDA y la venta de teléfonos inteligentes crece a una taza impresionante, se estima que crecerá de un 8.5% a un 35% entre el 2003 y el 2007; y las ventas de PDA se incrementarán de 6.9

15 G3: Es un término genérico que cubre un rango de tecnologías de redes inalámbricas de tercera generación.
16 CDMA: Code Division Multiple Access, tecnología celular conocida originalmente como IS-95
17 BLUETOOTH: Es la norma que define un estándar de comunicación inalámbrica, que posibilita la transmisión de voz y datos entre diferentes equipos mediante un enlace por radiofrecuencia.
18 Wi-Fi: Wireless Fidelity, es un conjunto de estándares para redes inalámbricas basado en los estándares IEEE 802.11 (Ethernet inalámbrica)
19 GPS: El Sistema de Posicionamiento Global
20 Palm.- dispositivo móvil de la empresa PalmOne, wwww.palmone.com
21 Pocket PC.- dispositivo móvil, de la empresa Compaq, www.hp.com
22 Gameboys..- dispositivos especializados para ejecutar juegos.

millones a 17.1 millones de unidades en el mismo periodo, esto es un 117% [3].

2.1.1 Aplicaciones

En un ambiente en que las circunstancias hacen inapropiado el uso de una computadora portátil, un dispositivo de mano ofrece una alternativa conveniente. Se puede mencionar varios campos [30] en los que no sólo son de utilidad, sino que es de competencia esencial para las empresas, por ejemplo:

Manejo de pacientes.

La computación móvil permite al médico o institución mantener contacto con un paciente cuyo estado requiere continua vigilancia. Lo cual incluye monitoreo constante de signos vitales que pueden anticipar una emergencia.

Ventas directas.

Poder consultar inventarios, precios y realizar pedidos en forma inmediata resulta de particular interés para cualquier empresa que se dedique a la comercialización de productos.

Servicio a clientes.

Asesoría, servicio técnico y consultoría es un área en donde la computación móvil es vital.

Personal móvil en oficinas.

No es raro encontrar a personal que, pese a encontrarse siempre en el mismo edificio, se mudan de lugar con frecuencia, por ejemplo, dar soporte técnico al personal o revisar proyectos.

La computación móvil no sólo les permite ser localizados con facilidad, sino que también le auxilia en la consulta de datos que por lo regular estarían en su oficina.

Profesionales viajeros.

Entre ellos encontramos contadores con los registros de una empresa bajo el brazo, gerentes regionales que integran metas empresariales, dirigentes corporativos que requieren información actualizada y muchos otros.

Grupos de trabajo.

La globalización y expansión ha vuelto necesario atacar proyectos con el personal adecuado, el cual no siempre

trabaja bajo un mismo techo y, en ocasiones, ni siquiera en la misma ciudad o país.

2.1.2 Tendencias

Internet Móvil significa mucho más que sólo "hacer que la Internet se vuelva móvil" [31]. Este concepto permite la personalización de una gran cantidad de servicios para satisfacer las preferencias individuales y de ubicación de cada usuario.

Actualmente, varios sistemas basan su trabajo en el Internet lo cual ayuda en gran medida, pero surgen problemas al no tener la computadora de escritorio a la mano. Es ahí cuando la computación móvil ofrece una variedad de alternativas muy útiles.

Según proyecciones de la empresa multinacional Ericsson[23], para el año 2004, habrá más de 600 millones de usuarios de Internet Móvil en todo el mundo, lo cuál significa que la cantidad de personas que utilicen este medio será superior al volumen actual de usuarios de Internet Fijo, que se ubican aproximadamente en 592 millones. Otra evidencia de esta tendencia se muestra en

[23] Ericsson Internet Móvil: un modo de vida, consulta 02 de Marzo del 2004. http://www.ericsson.com.mx/press/referencias/modo_vida.html>

los 8 mil millones de mensajes SMS que recorren el mundo mensualmente. Por otra parte, un informe de *PriceWaterhouse*[24], destaca que después de dos años Internet Móvil superará al acceso tradicional a la red, pues será más barato, rápido y más accesible que las líneas telefónicas actuales [4].

2.1.3 Consideraciones

Antes de optar por una solución móvil para un sistema de cómputo se deben considerar puntos muy importantes [32] como:

- Dispositivos móviles.- Aquí se encuentra uno de los aspectos más sensibles de la computación móvil, debido a que existen compromisos entre la capacidad de los equipos, el tamaño, peso, facilidad de uso y la alimentación de energía. El equipo debe ser pequeño y liviano para transportarse, pero en el momento de usarlo debe ser suficientemente grande para poder operarlo con facilidad, algo que puede resultar contradictorio.

[24] PriceWaterHouse.-Transnacional para consultoría, http://www.pwcglobal.com/

- Seguridad.- El concepto de seguridad es muy amplio y depende del punto de vista. El empresario desea control, acceso remoto y la confidencialidad de su información, por otro lado, el usuario se centra en la actualización de los datos en su computador.
- Software.- Las aplicaciones deben ser diseñadas para aplicarse bajo esta tecnología. Mucho depende de las comunicaciones y éstas no son tan rápidas como las comunicaciones en redes cableadas. Preferiblemente, debería considerarse una arquitectura cliente/servidor para que el tráfico impuesto sea sólo lo estrictamente indispensable para actualizar información. Generalmente los dispositivos móviles se usan como clientes ligeros de presentación de información.

El uso de esta tecnología es costoso, el usuario debe tener claro lo que puede esperar de la tecnología, debido a que no se puede trasladar todo el paradigma de una computadora de escritorio a un dispositivo móvil.

2.2 Reseña de las Tecnologías Inalámbricas

El desarrollo de las tecnologías inalámbricas se viene dando desde hace mucho tiempo y ahora estas poseen un gran mercado,

de tal manera que el crecimiento de dispositivos y software continúa de manera acelerada. En esta sección se describen varios conceptos acerca de las comunicaciones inalámbricas.

2.2.1 Espectro electromagnético

Las comunicaciones inalámbricas se basan en la emisión ondas electromagnéticas, al conectarse una apropiada antena a un circuito eléctrico, las ondas electromagnéticas se pueden difundir de manera eficiente y captarse por un receptor a cierta distancia [5].

La transmisión electromagnética depende de la frecuencia: a bajas frecuencias, las ondas de radio cruzan bien los obstáculos, pero la potencia se reduce drásticamente con la distancia de la fuente. A frecuencias altas, las ondas de radio tienden a viajar en línea recta y a rebotar en los obstáculos. La lluvia y motores eléctricos también dificultan el paso de estas ondas [5].

La Figura 2 muestra una ilustración sobre el espectro electromagnético y su división en: microondas, infrarrojo, radio y luz visible que pueden servir para transmitir información.

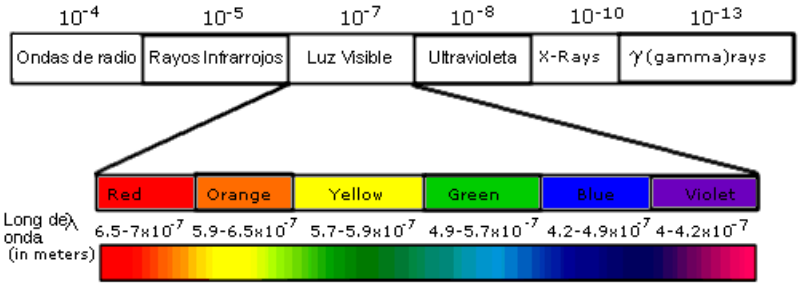


Figura 2.- Espectro electromagnético. Referencia: http://library.thinkquest.org/C003776/espanol/print/spectrum.htm?tqskip1=1

2.2.2 Redes de radio frecuencia

La FCC[25] permitió la operación sin licencia de dispositivos que utilizan 1 Watt de energía o menos, en tres bandas de frecuencia: 902 a 928 MHz, 2,400 a 2,483.5 MHz y 5,725 a 5,850 MHz. Estas bandas de frecuencia, llamadas bandas ISM[26], estaban anteriormente limitadas a instrumentos científicos, médicos e industriales. La idea es tomar una señal de banda convencional y distribuir su energía en un dominio más amplio de frecuencia. Así, la densidad promedio de energía es menor en el espectro equivalente de la señal original [5] [6].

[25] FCC: The Federal Communications Commission
[26] ISM: (Industrial, Scientific and Medical).- Bandas de frecuencias para uso comercial y sin licencia

2.2.3 Redes de área local inalámbricas

Las ventajas de las Redes de Área Local Inalámbricas (LAN) sobre las cableadas son: flexibilidad en la localización de la estación, fácil instalación y menores tiempos en la reconFiguración. Las tecnologías para las LAN inalámbricas son dos: Infrarrojas y Radio Frecuencia [6].

El grupo IEEE[27] 802.11[28] esta desarrollando normas para LAN inalámbricas. La Red Ethernet Inalámbrica, que se basa en el estándar IEEE 802.11, está actualmente disponible en dos versiones [6]: 802.11ª, 8011.b y 8011.g el cual no es parte de esta revisión por no estar disponible al momento de escoger las tecnologías. Se muestra la tabla 3 una revisión más completa.

El estándar original 802.11 y la versión 802.11b de 11 Mb se transmiten a 2,40 GHz. Por otro lado, la versión más rápida 802.11a (con un máximo de 54 Mb por segundo), utiliza la banda UNII[29] (5 GHz). De este modo, resulta evidente que las redes 802.11a y 802.11b no se pueden

[27] IEEE: Instituto de Ingenieros Electricos y Electrónicos
[28] 802.11: Grupo del IEEE dedicado a las redes de área local inalámbricas
[29] UNII.- Banda de 5GHz, Infraestructura de Información Nacional sin Licencia, que en los Estados Unidos está regulada por la FCC.

comunicar entre sí tal cual. No obstante, existen puntos de acceso de modo dual que admiten tanto 2,40 y 5 GHz [6].

El grupo IEEE 802.11 planea introducir una nueva sub-capa de Control De Acceso al Medio (MAC) que tenga capacidad de acceder varios medios de transmisión y que tenga un rango aceptable para los requerimientos del usuario. No es fácil para el grupo tratar de volver a usar alguna de las subcapas MAC ya existentes por dos razones principales:

- El rango de requerimientos de usuario impide el soporte simultáneo de estaciones fijas, móviles y estaciones vehiculares.
- El permitir múltiples medios de transmisión, especialmente en la tecnología de radio frecuencia, el cual requiere de complicadas estrategias para cubrir la variación del tiempo en el canal de transmisión.

Descripción	IEEE 802.11b	IEEE 802.11a
Rango de frecuencia	2,40 GHz (banda ISM)	5,15 a 5,35 GHz (banda UNII)
Velocidad de transferencia de datos	De 1 a 11 Mbps	De 6 a 54 Mbps
Rango en espacio libre (según la velocidad de transferencia de datos)	120 metros (11 Mbps) a 460 metros (1 Mbps)	30 metros (54 Mbps) a 300 metros (6 Mbps)

Rango en habitaciones (según la velocidad de datos)	30 metros (11 Mbps) a 90 metros (1 Mbps)	12 metros (54 Mbps) a 90 metros (6 Mbps)
Números de canales independientes	3	8
Número de usuarios admitido por punto de acceso	192	512
Aplicación	Datos	Multimedia
Técnicas de modulación	DSSS	OFDM
Protocolo	TCP/IP	

Tabla 3.- Comparación entre 802.11a y 802.11b, Referencia:

http://www.intel.com/es/home/trends/wireless/info/ethernet.htm.

2.2.4 Topologías de red inalámbrica

Las redes de área local inalámbrica se encuadran básicamente en dos topologías: redes de infraestructura y ad-hoc [21]. Las redes ad-hoc se forman entre dispositivos que se encuentran en su rango de alcance, además, todos los dispositivos conectados tienen exactamente los mismos derechos.

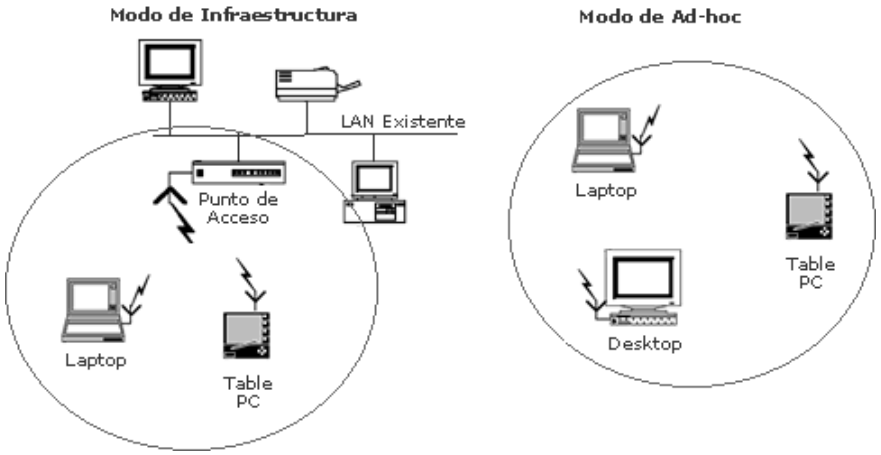


Figura 3.- Modos de Infraestructura y ad-hoc. Referencia: [21]

Las redes de infraestructura están formadas por uno o varios puntos de acceso y uno o varios dispositivos inalámbricos como los muestra la Figura 3.

2.2.5 Estándar IEEE 802.11

Actualmente, los estándares más populares sobre redes de área local inalámbrica son desarrollados por organizaciones internacionales como el IEEE (*Institute of Electrical and Electronic Engineers*) y el ETSI (*European Telecommunications Standards Institute*). Entre los principales estándares tenemos:

- IEEE 802.11: El estándar original de WLAN que soporta velocidades entre 1 y 2 Mbps30.

[30] Mbps.- Mega bits por segundo, medida de ancho de banda en una transmisión.

- IEEE 802.11a: El estándar de alta velocidad que soporta velocidades de hasta 54 Mbps en la banda de 5 GHz.
- IEEE 802.11b: El estándar dominante de WLAN (conocido también como Wi-Fi) que soporta velocidades de hasta 11 Mbps en la banda de 2.4 GHz.
- HiperLAN2: Estándar que compite con IEEE 802.11a al soportar velocidades de hasta 54 Mbps en la banda de 5 GHz.
- HomeRF: Estándar que compite con el IEEE 802.11b que soporta velocidades de hasta 10 Mbps en la banda de 2.4 GHz.

En la tabla 4 se pueden observar las especificaciones de estos tipo de redes.

El gran éxito de las redes inalámbricas (WLAN) es debido a que utilizan frecuencias de uso libre, es decir, no es necesario pedir autorización o algún permiso para utilizarlas. Aunque hay que tener en mente, que la normatividad acerca de la administración del espectro varía de país a país.

Estándar	Velocidad máxima (Mbps)	Interfase de aire	Ancho de banda de canal (MHz)	Frecuencia (GHz)	Disponibilidad
802.11b	11	DSSS[31]	25	2.4	Ahora
802.11a	54	OFDM[32]	25	5.0	Ahora
802.11g	54	OFDM DSSS	25	2.4	Finales 2002
HomeRF 2	10	FHSS[33]	5	2.4	Ahora
HiperLAN2	54	OFDM	25	5.0	2003
5-UP	108	OFDM	50	5.0	2003

Tabla 4.- Principales estándares WLAN. Referencia:

http://www.eveliux.com/articulos/estandareswlan.html

La desventaja de utilizar este tipo de bandas de frecuencias es que las comunicaciones son propensas a interferencias y errores de transmisión. Estos errores ocasionan que los paquetes sean reenviados una y otra vez [6].

Una razón de error del 50% ocasiona que se reduzca el caudal eficaz real (throughput) en dos terceras partes aproximadamente. Por eso la velocidad máxima especificada teóricamente no es tal en la realidad. Si la especificación IEEE 802.11b dice que la velocidad máxima es 11 Mbps, entonces el máximo caudal eficaz será aproximadamente 6 Mbps o menos [6].

[31] DSSS: Direct Sequence Spread Spectrum
[32] OFDM: Orthogonal Frequency Division Multiplexing
[33] FHSS: Frequency Hopping Spread Spectrum 5-UP: 5-GHz Unified Protocol (5-UP), Protocolo Unificado de 5 GHz propuesto por Atheros Communications

La velocidad real en las WLAN está muy debajo de lo especificado por las normas, ya que esta depende de diversos factores tales como el ambiente de interferencia, la distancia o área de cobertura, la potencia de transmisión, el tipo de modulación empleada, etc. La mayoría de las redes 802.11b pueden alcanzar oficialmente distancias hasta 100 metros en interiores y 400 metros en exteriores [6].

Con una mayor potencia se puede extender esa longitud, aunque en interiores puede limitarse la potencia de transmisión debido a paredes y otros objetos que pueden interferir en la señal. En realidad, la comunicación punto a punto en una WLAN en ambientes exteriores puede alcanzar varios kilómetros, mientras exista línea de vista y libre de interferencia. Bajo este esquema se utiliza el método conocido como DSSS (*Direct Sequence Spread Spectrum*) para transmitir datos entre los dos puntos.

La comunicación se establece conectando en un lado un equipo conocido como puente inalámbrico (*Wireless Bridge*) y en el otro extremo un punto de acceso (*Access Point*). La salida de estos equipos hacia la red *Ethernet* local se da a través de la interfaz RJ45, por lo que se

puede conectar directamente un concentrador (*hub)* o un conmutador de paquetes (*switch*), en donde se conectarán las computadoras de nuestra red [6].

2.3 Consideraciones sobre las Tecnologías

Después de revisar conceptos de computación móvil y tecnologías inalámbricas, en esta sección se presenta una descripción de los actores involucrados en el desarrollo del sistema. Además, se revisan algunas consideraciones sobre la plataforma de desarrollo y limitaciones de los dispositivos móviles.

2.3.1 Actores involucarados.

Los actores involucrados en el desarrollo del SCI son:

Los Desarrolladores

Los desarrolladores de software que crean grandes y poderosos sistemas tienen un ambiente de lujo, con memoria y potencia de procesamiento casi ilimitada. Sin embargo, el desarrollo en dispositivos móviles presenta nuevos retos y obstáculos, debido a que los recursos con los que se cuenta son limitados, además, tienen

características tales como: tamaño reducido de la pantalla, poco memoria y poder de procesamiento bajo.

Adicionalmente, los desarrolladores se enfrentan a retos como manejar redes inalámbricas y movilidad de los dispositivos, que típicamente ofrecen bajos anchos de banda y menos confiabilidad que las redes cableadas. Estas limitaciones requieren que los desarrolladores piensen diferente acerca de la interfase de usuario, uso del poder de procesamiento, manejo de memoria y excepciones [27].

Los Usuarios

Los usuarios sean estos altos gerentes corporativos o usuarios de producción de bajo nivel están acostumbrados al poder y flexibilidad de grandes sistemas, con aplicaciones complejas, con mucha funcionalidad y que prácticamente lo hacen todo, se enfrentan a la adopción de equipos limitados, que aparentemente no aportan algún valor agregado a su trabajo y con aplicaciones simples e intuitivas.

Desarrollar aplicaciones para dispositivos con recursos limitados requiere habilidades adicionales de los

programadores, quienes tienen que aprender y comprender como operar en un ambiente inalámbrico y como tomar decisiones inteligentes acerca del diseño de la arquitectura del software y de el buen rendimiento de la aplicación [33].

2.3.2 Limitantes de los dispositivos móviles

El crecimiento dramático que ha tenido el negocio de los dispositivos inalámbricos ha estimulado los esfuerzos por adaptar la mayoría de las tecnologías que han sido desarrollados para computadores de escritorio hacia dispositivos móviles, sin embargo esta tendencia tiene sus limitantes [10]:

- Memoria.- Dispositivos como teléfonos celulares y navegadores de dos vías tienen cantidades limitadas de memoria, esto obliga a considerar una administración de este recurso más cuidadosa cuando se diseñan los objetos.
- Poder de Procesamiento.- Los dispositivos inalámbricos tienen un limitado poder de procesamiento (procesadores típicos son los de 16 bits).

- Medios de entrada de datos.- Las capacidades de entrada son muy limitadas, muchos teléfonos celulares proveen solo un teclado que se maneja con una mano, con 12 botones, 10 numerales, un asterisco y un signo de numeral.
- Pantalla.- La pantalla es pequeña, regularmente 96x54 píxeles y 1 bit de profundidad para el color (blanco o negro). La cantidad de información que se puede mostrar en la pantalla es muy limitada.

Además de enfocarnos en las limitaciones de los dispositivos también se encuentran las limitaciones que el ambiente inalámbrico impone:

- Las redes inalámbricas son poco confiables y los anchos de banda siguen siendo una gran limitación.
- Estas redes tienden a experimentar más errores que las redes cableadas.
- La movilidad de dispositivos inalámbricos incrementa el riesgo de que la conexión se pierda o se degrade.

Los desarrolladores deben tener en cuenta estas limitaciones para que sus aplicaciones sean eficientes y confiables.

2.4 Plataforma de Desarrollo

Actualmente, para sistemas empresariales existen dos tendencias en plataformas de desarrollo: Java de Sun Microsystems Inc. y Microsoft **.**NET. Cada una tiene su estrategia para el desarrollo de sistemas. Aunque Microsoft Punto NET ofrece varias herramientas para desarrollar en dispositivos móviles, sólo es compatible con dispositivos móviles que ejecutan sobre el sistema operativo *Windows CE* y que la mayoría de dispositivos móviles se ejecutan en otros tipos de sistemas operativos. Lo cual se convierte a Microsoft Punto NET en una limitante crítica.

Para la selección de la plataforma del sistema se tiene que considerar las restricciones tecnológicas y administrativas. Las restricciones administrativas están encaminadas a la adaptación del SCI para el uso en el Departamento de Suministros de la ESPOL, las restricciones tecnológicas se presentan en la tabla 5.

Restricción	Descripción	Investigación y Desarrollo
Base de Datos	La integración del modelo conceptual de un sistema de inventario móvil al esquema y arquitectura de la Base de Datos utilizada en la ESPOL en este caso DB2 versión 6.1 sobre Sun Solaris OS.5.8 , es uno de los factores críticos para su desarrollo.	Muy Alto
Servicio de Autenticación y Autorización	En Septiembre del 2003, el Centro de Servicios Informáticos implemento el Servicio de Directorio, para la autenticación y autorización de la red de computadores habilitados con el sistema operativo Windows 95/98/XP de Microsoft y para la autenticación del servicio de correo de la ESPOL. El servicio se implemento con el Directorio Activo de Windows 2000 Server de Microsoft, en la actualidad el directorio tiene registradas al 12 de Junio: 16.096 usuarios.	Muy Bajo
Plataforma Actual del Sistema	El sistema actual está desarrollado en el lenguaje orientado a objetos SmallTalk y sobre el ambiente de desarrollo "Visual Age for Enterprise" versión 5.0 de IBM	Alto

Tabla 5.- Restricciones tecnológicas en la selección de la plataforma de desarrollo, Referencia: Camilo Robayo, desarrollador Web del CSI 2003-2004.

En el diseño del SCI se aplican muchos modelos para dar soluciones aceptables a los problemas que presentan las restricciones tecnológicas, sin embargo, estos modelos no tienen relación con los objetivos técnicos de este proyecto los cuales son la Computación Móvil y Tecnologías Inalámbricas por lo cual no serán presentados con detalle en este documento.

Las restricciones administrativas que se detallan en la tabla 6 van encaminadas a la compleja tarea de acceder, entender e integrar el SCI con los sistemas actuales.

Restricción	Descripción	Investigación y Desarrollo
Acceso y uso de la información sobre la arquitectura del sistema	Debido a que el sistema se desarrollo y se puso en marcha en 1997 en la actualidad no existe documentación actualizada de cambios y actualizaciones sobre el sistema y sobre el esquema de la base de datos. Además, el acceso a esta información es restringida y actualmente sólo se cuenta con el conocimiento de las personas que mantienen el sistema.	Muy Alto
Acceso a datos del sistema	De la misma manera el acceso a los datos del sistema es muy restringido, y sólo se tiene acceso de consulta sobre ciertas tablas. Además, el acceso a la base de datos del directorio también es restringido, sobre todo obteniendo permisos y roles para la autorización de usuarios.	Muy Alto
Acceso a expertos locales en desarrollo de sistemas móviles	Debido a la poca incursión de la ESPOL en el campo del diseño de arquitecturas de sistemas móviles y las escasas iniciativas en la actualización de tecnologías de los sistemas actuales, esto da como consecuencia la ausencia de expertos en tecnologías móviles e integración de datos.	Medio

Tabla 6.- Restricciones administrativas y su impacto en la investigación y desarrollo del sistema propuesto. Referencia: Camilo Robayo, desarrollador Web del CSI 2003-2004.

Como consecuencia de las restricciones administrativas y tecnológicas la elección de plataforma de desarrollo se ve limitada a la plataforma que ofrezca mayores ventajas y que se integre de mejor manera a los sistemas actuales de la ESPOL.

Después de haber considerado diversas restricciones para escoger la plataforma, se ha elegido la Tecnología J2ME para

desarrollar la aplicación cliente, debido principalmente a que esta plataforma ofrece beneficios en el modelo de programación de los dispositivos y que este grupo de especificaciones estándar para desarrollar en dispositivos móviles están siendo desarrolladas bajo el patrocinio de *Java Comunita Process* (JCP[34]), lo cual garantiza una mayor compatibilidad y portabilidad de la aplicación clientes. Por otro lado, no se puede hacer una elección de la base datos, debido a que se debe usar la base de datos DB2 6.1 que usa la ESPOL.

Debido a que los sistemas de la ESPOL, no soportan integración directa con la aplicación cliente, se escogió la Tecnología J2EE para desarrollar una aplicación de servidor que sirva de intermediario entre la aplicación cliente y los recursos de datos. Para comprender mejor la Tecnología J2EE, se hará una breve revisión de las arquitecturas para diseñar sistemas.

2.4.1 Arquitectura de Sistemas

El valor de la arquitectura de sistemas ha sido recientemente reconocido en la industria del software. Arquitectura se puede definir como el espacio en donde los objetos operan. Esta define los contratos a través de

[34] JCP.- Java Community Process.- Comunidad que se dedica al desarrollo de el lenguaje Java, <www.jcp.org>

los cuales ellos interactúan con componentes externos y con cada uno de ellos [24].

Cliente/Servidor es una arquitectura de software distribuido en el cual los sistemas son divididos en procesos autónomos, donde un cliente envía las peticiones al servidor y este envía una respuesta como resultado de dicha petición al cliente [24].

Comprender la lógica de Cliente/Servidor es muy importante para comprender que el software puede estar dividido en capas lógicas, donde todas ellas forman la aplicación. Cabe destacar que no es necesario que todas las capas lógicas se encuentren juntas en la misma capa física. La teoría distingue 6 capas de software [35], en la práctica estas divisiones no son tan claras.

- Manejador de Presentación, muestra la interfaz al usuario.
- Presentación, es responsable por que se muestra al usuario.
- Aplicación, contiene la lógica de la aplicación.
- Capa de Negocios, define las reglas del negocio o dominio.

- Base de datos, consiste en el diccionario de datos de la aplicación.
- Manejo de Base de datos, es responsable por el almacenamiento de los datos.

Cuando se diseña un sistema Cliente/Servidor hay varias estrategias que se pueden usar para dividir las capas lógicas sobre los procesos. Esto varía, desde que todas las capas se encuentran en un único proceso (el cual por supuesto no es distribuido), hasta el otro extremo, en el cual esta la implementación de cada capa lógica en múltiples procesos. Una capa física es conocida como tier. Un tier es para propósitos prácticos y consideraciones lo mismo que un proceso.

2-Capas

Una aplicación de dos capas es una aplicación en la cual las capas lógicas están divididas sobre 2 procesos [24]. Se puede ver tres de las variantes más comunes. El primer esquema (Figura 4 a la izquierda) es llamado 2-Capas con cliente liviano (2-Tier/Thin-Client), porque el cliente sólo maneja la capa de presentación, todo lo demás es llevado a cabo por la capa de servidor. Se

puede sacar las respectivas conclusiones acerca de los otros dos modelos.

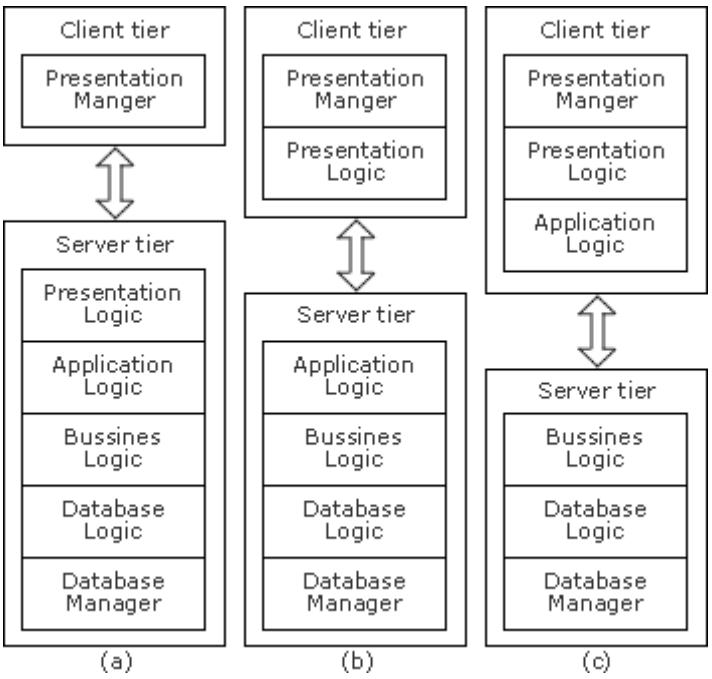


Figura 4.-Arquitectura Cliente/Servidor

Dos-Capas es la forma más común de sistemas distribuidos. La mayoría de las aplicaciones de los años 80 basadas en RDBMS son de tipo de 2-Capas.

3-Capas

Las aplicaciones en tres capas han madurado desde los años 90 y dividen las 6 capas en 3 procesos. Este modelo reconoce la importancia de separar las reglas de negocios del RDBMS. Además los típicos clientes en este modelo utilizan CORBA o DCOM.

N-Capas (n-Tier)

Como su nombre lo implica es un sistema en el cual las 6 capas son divididas de acuerdo a la estrategia de la aplicación en múltiples capas como lo muestra la Figura. 5 [34] Este modelo empezó a finales de los 90. Ejemplos típicos de aplicaciones de n-capas son basados en componentes con tecnologías DCOM y J2EE.

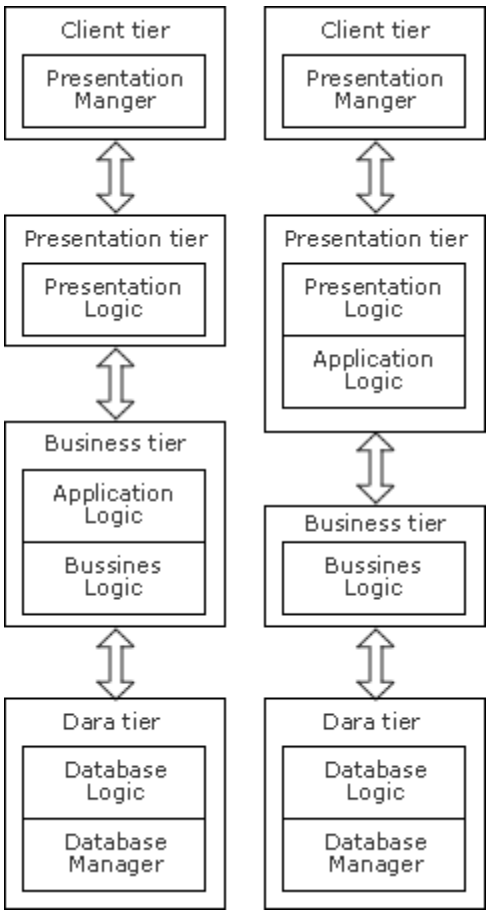


Figura 5.- Arquitectura n-capas

2.4.2 La Tecnología Java™

Desde su introducción en el mundo en 1994 a la fecha actual, Java[35] ha revolucionado la industria del software. Java ha sido usado en una amplia gama de formas para implementar muchos tipos de sistemas

La tecnología Java es un portafolio de productos basados en el poder de las redes y la idea de que un mismo software debería ejecutarse en diferentes tipos de sistemas y dispositivos[36]. El SCI se basa en dos plataformas de Java, La Plataforma en su Edición Empresarial (J2EE) y la Plataforma en su Edición Micro (J2ME.).

Desde la introducción en 1998 de un proceso abierto y participativo para desarrollar y revisar las especificaciones de: La Tecnología Java, implementaciones de referencia y suites de prueba. El programa *Java Community Process* ha fomentado la evolución de la plataforma Java con la cooperación internacional de más de 700 miembros corporativos e individuales. Adicionalmente, las

[35] Designing Wireless Clients for Enterprise Applications [Draft] 2, Contents © 1999-2003 by Sun Microsystems
[36] Java Technology, http://java.sun.com

especificaciones se desarrollan a través los JSR[37] (*Java Specification Request*), este proceso de desarrollo de la Tecnología Java se muestra en la Figura 6.

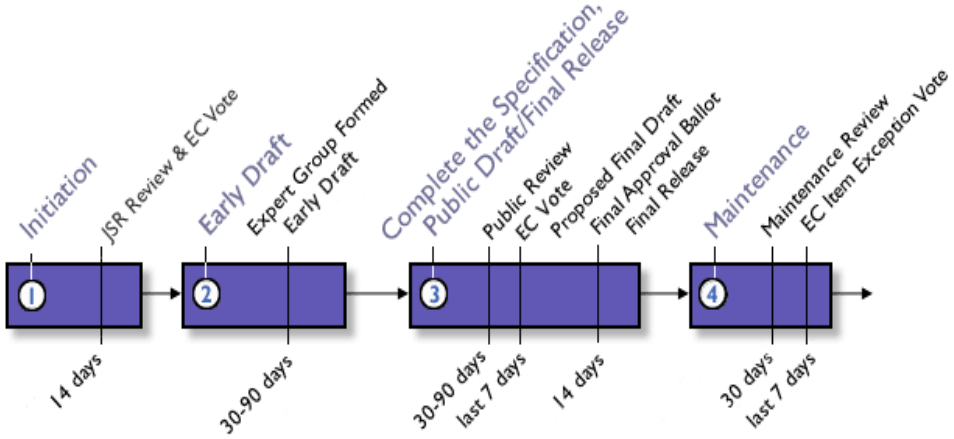


Figura 6.- Línea de tiempo para el desarrollo de una especificación del lenguaje de programación Java. Referencia: <www.jcp.org>.

La aplicación cliente del SCI está desarrollada sobre la plataforma J2ME que se basa en las especificaciones definidas entre otros en los JSR: 37, 118, 30, 139 (http://www.jcp.org/). Estas especificaciones definen el ambiente de programación al cual los desarrolladores e industrias se tienen que apegar al momento de desarrollar aplicaciones y máquinas virtuales respectivamente.

El SCI está desarrollado sobre las plataformas J2EE y J2ME, las cuales son desarrolladas bajo las

[37] JSR.- Java Specification Request, documento que contiene la especificación de las tecnologías Java <www.jcp.org>

especificaciones propuestas por JCP. En este documento no se trata a fondo sobre las especificaciones de J2EE al no ser un objetivo del proyecto, sin embargo en la sección 2.4.2 se detallan motivos por los cuales se usa J2EE.

Hay muchas formas de crear avanzadas aplicaciones distribuidas. La plataforma J2EE es una de tales tecnologías. Otras tecnologías que hacen lo mismo son: CORBA, DCOM y .NET. (Fuera del alcance de este documento)

2.4.3 Plataforma Java 2 Enterprise Edition (J2EE)

La plataforma J2EE es un conjunto de especificaciones de un ambiente distribuido en n-capas. La primera parte, contiene APIs y descripciones completas de todos los componentes y servicios que una aplicación J2EE puede usar. De hecho, la característica más importante de esta especificación es la clara distinción entre las APIs que juntas forman el modelo de programación para las aplicaciones J2EE, y el ambiente de ejecución del lado del servidor formado por los componentes y servicios.

La segunda parte de la plataforma J2EE es la implementación de referencia, la cual es una completa implementación de la especificación proveída por Sun Microsystems. Cuyos códigos binarios y fuentes están disponibles. Esta implementación puede ser usada para determinar la portabilidad de una aplicación J2EE.

La tercera parte de la plataforma J2EE es la Suite de Pruebas de Compatibilidad de J2EE. (J2EE Compatibility Test Suite). Esta suite valida que una especificación particular cumpla las especificaciones.

La cuarta parte es el J2EE SDK, que incluye la implementación de referencia, una aplicación de verificación y herramientas de despliegue y documentación.

J2EE ha sobresalido en la estandarización de muchos conceptos importantes acerca de la capa intermedia para aplicaciones (m*iddleware*) [23]. Por ejemplo, J2EE provee una interfaz para administración de transacciones distribuidas, servicio de directorio y mensajes, en adición, Java 2 Standard Edition (J2SE) el cual sostiene J2EE, provee de un estándar exitoso para la interacción entre Java y bases de datos relacionales.

Motivos de uso de el *Framework* J2EE

Un *Framework* de software es un conjunto de clases que sirven de diseño reusable para una aplicación o capa de una aplicación. Mientras que el código de la una aplicación llama a una librería de clases para pedir sus servicios, un *framework* llama al código de la aplicación y así administra el control de flujo.

El concepto de *framework* esta bien adaptado a las complejidades que presenta el desarrollo de aplicaciones J2EE porque puede proveer un modelo de aplicación simple y fácil de usar para los desarrolladores. Un *framework* de fuente abierta ofrece muchas ventajas [27]:

- Con un buen *framework* los desarrolladores escriben sólo el código que necesitan escribir, no tienen que trabajar directamente con infraestructura y API de bajo nivel.
- Un *framework* bien diseñado puede proveer consistencia y estructura a una aplicación. La estructura sería clara de tal manera que desarrolladores adicionales se puedan añadir al proyecto.

- Un *framework* fácil de seguir puede promover las buenas prácticas de programación a través de ejemplos y documentación.
- Generalmente los f*rameworks* de fuente abierta exitosos son mejor probados que los hechos en casa.

J2EE por si sólo provee muchos *frameworks*. Por ejemplo, los *Enterprise Java Beans* (EJB), el contenedor o máquina de Servlets. Con J2EE administrando la instanciación e invocación de objetos. *Frameworks* de fuente abierta como *Struts* añaden su propio *framework* sobre el *framework* estándar de los *Servlets*. Lo importante de mencionar es que los *frameworks* arriba de J2EE ofrecen un modelo de programación simple y otros beneficios adicionales.

El *framework* J2EE que usa la aplicación de servidor del SCI esta dado por el Servidor de Aplicaciones Java Sun System Server 8 Update 1.

2.4.4 Plataforma Java 2 Micro Edition (J2ME)

La Plataforma *Java 2 Micro Edition* (J2ME) [8] permite a los desarrolladores crear aplicaciones Java para

productos de consumo tales como teléfonos móviles y PDA

Los parámetros en los cuales se clasifican las aplicaciones inalámbricas son [36]:

- La conectividad de red requerida
- El procesamiento local requerido
- La parte del modelo de datos que la aplicación, necesita que este disponible localmente.

Generalmente se puede clasificar las aplicaciones inalámbricas en 3 categorías [36]:

- Aplicaciones Clientes Inalámbricas Independientes.- son las más autosuficientes en todo el espectro, realizan la mayoría, si no es todo, el procesamiento de datos. Ocasionalmente se conectan a la red para sincronizar o archivar información que esta en el repositorio local y usualmente esta operación es requerida por el usuario.
- Aplicaciones Clientes Inalámbricas livianas.- son el otro extremo del espectro, son típicamente navegadores genéricos o aplicaciones de navegación. A más de la presentación estas también

tienen limitaciones en el procesamiento, si hay alguno. A pesar de estar conectadas a la red proveen almacenamiento temporal y funciones limitadas en modo desconectado.

- Aplicaciones Clientes Inalámbricas densas.- este tipo de aplicaciones cliente caen en el espectro de la especificación MIDP. Aplicaciones Clientes densas hacen gran parte del trabajo conectadas a la red, sin embargo tienen una gran cantidad de funciones en modo desconectado, dependiendo de cuan bien el modelo de datos pueda ser manejado localmente.

La aplicación cliente se basa en la especificación J2ME API[38] y es sus especificaciones: *Mobile Information Device Profile* (MIDP) y *Connected Limited Device ConFiguration* (CLDC) los cuales proveen una base sólida para el desarrollo en dispositivos móviles.

Entre los API de J2ME se encuentran paquetes especializados para:

- Programar interfases de usuario usando el Limited Connected Device User Interface (LCDUI) API

[38] API, Interfaz del Programa de la Aplicación.

- Grabar datos persistentemente en el dispositivo usando Record Management Store (RMS) API
- Crear conexiones de Red a un servidor u otro dispositivo a través del Generic Connection Framework (GCF)

Con el uso de J2ME y J2EE se puede crear potentes y muy flexibles sistemas como lo muestra la Figura 7.

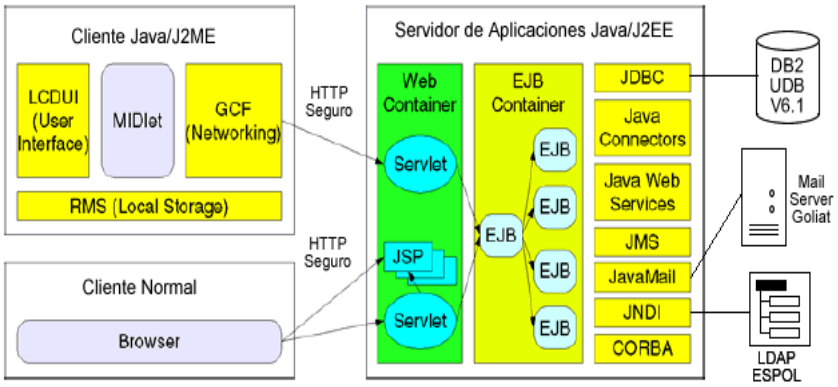


Figura 7.- Interconexión de Plataformas con tecnologías Java.

Referencia Designing wireless clients for enterprise applications

[DRAFT] 2, Contents © 1999-2003 by Sun Microsystems,:

http://java.sun.com

Consideraciones sobre *Websphere Micro Enviroment J2ME Runtime*

La implementación de la máquina virtual de Java *Websphere Micro Enviroment J2ME Runtime* de IBM

(J9VM) [11] para dispositivos Palm se basa en los siguientes estándares de *Java Community Process* (http://jcp.org/en/home/index):

- MIDP 2.0 <http://jcp.org/en/jsr/detail?id=118>
- CLDC 1.1, <http://jcp.org/en/jsr/detail?id=139>

Adicionalmente, soporta operaciones de punto flotante de la especificación CLDC1.1 la cual se basa en el estándar IEEE 754 <http://grouper.ieee.org/groups/754/>.

J9VM es soportada por el Sistema Operativo Palm O.S. 5.2 disponible en los dispositivos: Tungsten C, Tungsten T2, Tungsten T3, Tungsten E, Zire 71, y Treo 600. La versión original Tungsten T oficialmente no soporta la especificación de la máquina virtual de IBM, debido a que tiene una muy limitada cantidad de memoria dinámica (*dynamic heap*)[39], pero este modelo puede ser usado para ambientes de desarrollo. El modelo Zire 21 no soporta esta especificación [11].

Tungsten W (Palm OS 4.1) soporta esta especificación previa de *WebSphere Micro Environment* 5.6 *68k/PACE*

[39] Dynamic heap.- la memoria dinámica es el espacio donde la máquina virtual puede cargar clases, crear objetos, posicionar imágenes, etc.

Runtime, la cual soporta MIDP 1.0 y CLDC 1.0 (http://jcp.org/en/jsr/detail?id=30)

Limitaciones y Disponibilidad de Memoria

WebSphere Micro Environment J2ME Runtime tiene acceso a toda la memoria dinámica disponible en el dispositivo móvil, no se tiene memoria extra para las aplicaciones Java, la tabla 5 muestra las especificaciones de memoria dinámica y capacidad de almacenamiento de los dispositivos Palm [11].

Dispositivo	Memoria Dinámica (MB)	Capacidad de Almacenamiento (MB)
Treo 600	3	32
Tungsten T3	4	51
Tungsten C	4	51
Tungsten E	1	32
Tungsten T2	1	32
Zire 71	1	16
Tungsten W	1	16

Tabla 7.- Limitaciones de memoria dinámica y capacidad de almacenamiento de dispositivos Palm, MIDP 2.0 Porting Guide, Copyright PalmOne 2004, Referencia: www.palmone.com/java.

Adicionalmente, *WebSphere Micro Environment* puede dinámicamente gestionar más memoria, si la aplicación lo requiere, hasta el máximo disponible de memoria dinámica, además, la memoria es compartida entre otras aplicaciones o servicios del sistema que se encuentren

corriendo. Casos típicos de uso excesivo de memoria son las pilas TCP/IP, *WiFi, Bluetooth, CDMA*, etc.

Implementaciones de la Máquina virtual

WebSphere Micro Environment J2ME Runtime para Palm OS 5, tiene dos implementaciones de la máquina virtual que tienen diferencias aun cuando se ejecuten en un mismo dispositivo como la administración de memoria. Además, ambas soportan diferentes conFiguraciones y versiones. La implementación de la máquina virtual 68k/PACE soporta CLDC 1.0/MIDP 1.0 y la implementación para los procesadores *ARM* soporta CLDC 1.1/MIDP 2.0 [11]. A continuación se detallan los dos tipos de implementaciones:

- Implementación 68k/PACE.- El factor crítico que determina que tipo de aplicaciones pueden ejecutarse sobre un dispositivo habilitado con Palm O.S. es la memoria dinámica. La implementación de 68k/PACE hace uso de la memoria de almacenamiento. Esto permite que las aplicaciones Java hagan uso de todos los tipos disponibles de almacenamiento. Este tipo de implementación

perjudica la interoperabilidad con otras aplicaciones tales como reproductores de MP3 que se ejecutan en segundo plano.

- Implementación ARM.- esta implementación hace uso sólo de la memora dinámica, regularmente en dispositivos habilitados con ARM la memoria dinámica disponible es considerablemente más grande. Esto hace que el uso de este tipo de máquinas virtuales no tengan otra restricción más que el del tamaño de la memoria dinámica.

Observaciones del uso de la máquina virtual ARM

Mientras más complejas se vuelven las aplicaciones el uso de la memoria dinámica aumenta, por este motivo al momento de desarrollar aplicaciones para dispositivos móviles se debe tener en cuenta los siguientes aspectos [11]:

- Número y tamaño de clases a ser cargadas.
- Número de instancias a ser creadas
- Número y tamaño de las imágenes que van a ser creadas y dibujadas en la pantalla.

Como consecuencia de no tomar en cuenta estas consideraciones, es que una aplicación muy pequeña puede muy fácilmente desbordar la memoria dinámica.

Según la especificación CLDC 1.1 la implementación de la máquina virtual para procesadores ARM sólo requiere 440 KB de memoria dinámica para iniciarse y correr una aplicación simple, sin embargo, 440 KB no es suficiente para correr aplicaciones grandes.

La cantidad de memoria disponible puede variar dependiendo del estado de las aplicaciones y los servicios. Sobre los teléfonos Treo 600 la memoria dinámica varía dependiendo de la actividad del módulo de radiofrecuencia, la cual decrece cuando se activa este módulo y decrece muy rápidamente cuando se transmiten datos [11]. Si el dispositivo no puede proveer de la suficiente cantidad de memoria dinámica simplemente la aplicación no puede correr, dando los siguientes resultados, los cuales aparecieron en la fase de pruebas antes de tomar medidas correctivas mencionadas en la sección 5.2.3 “Mejorando el Rendimiento”:

- Se inhibe la aplicación y se reporta una excepción “*java.lang.ErrorOutofMemoryError”.*

- Una caja de dialogo con la mensaje: “Unable to create offscreen window” aparece en la pantalla.

En la tabla 8 se detallan las especificaciones de memoria dinámica y observaciones sobre el tipo de aplicaciones que puede ejecutar en dispositivos *Palm*.

Memoria Dinámica	Dispositivo	Observaciones
680 Kb	Tungsten T2	Una pequeña cantidad de Midlets pueden ser ejecutados, tales como Midlets con gráficos simples y basados en pequeños formularios
930 Kb	Zire 71	La mayoría de los Midlets pueden ser ejecutados. Midlets que cargan muchas y grandes imágenes pueden fallar.
1960 Kb	Tingsten E	Casi la totalidad de Midlets pueden ser ejecutados. Sin embargo, esto no incluye algunos Midlets que ejecutan operaciones sobre la red.
2900 Kb	Treo 600	Todos los Midlets comunes pueden ser ejecutados incluso los que usan conexiones de red

Tabla 8.- Pruebas de rendimiento sobre dispositivos Palm tomando como factor comparativo la capacidad de memoria dinámica, *MIDP 2.0 Porting Guide*, Copyright PalmOne 2004. Referencia: www.palmone.com/java.

Limitaciones de Memoria sobre RMS

La especificación MIDP provee mecanismos para almacenar los datos, este es llamado Record Management System (RMS)40; es un API que permite el acceso a las

[40] Record Management System (RMS), MIDP especificación 1.0, http://java.sun.com

bases de datos de los dispositivos móviles, que generalmente son simples bases de datos en las cuales se pueden crear registros simples de hasta 64 KB [11]. Cada registro es almacenado y recuperado en un arreglo de bytes. El tamaño de los registros puede variar y el formato de los datos no esta limitado por RMS.

Los *Midlets* pueden añadir, recuperar y remover registros desde un repositorio de registros RMS. El RMS permite métodos para comparar, enumerar, filtrar, monitorear y almacenar registros [11].

Adicionalmente, la especificación de la máquina virtual no puede acceder a espacio de memoria extra como memorias flash.

Multimedia

Las imagines PNG son soportadas con 8 hasta 24 bits de profundidad con transparencias. Sin embargo, esto varía dependiendo de la profundidad de colores soportada por el dispositivo. Capacidades de audio WAV son soportadas desde los 8 KHz mono hasta 44.1 KHz estéreo [11].

Resolución y color de la pantalla

La tabla 7 detalla especificaciones sobre resolución y profundidad de colores de la pantalla en dispositivos *Palm*.

Dispositivo	Resolución(s)	Profundidad del color
Treo 600	160x160	12 bits
Tungsten T3	320x320, 320x480, 480x320	16 bits
Tungsten C	320x320, 160x160	16 bits.
Tungsten E	320x320, 160x160	16 bits
Tungsten T3	320x320, 160x160	16 bits
Zire 71	320x320, 160x160	16 bits
Tungsten W	160x160	16 bits

Tabla 9.- Limitaciones de color y resolución en dispositivos Palm, MIDP 2.0 Porting Guide, Copyright PalmOne 2004. Referencia: www.palmone.com/java.

La Palm Tungsten T3 soporta el cambio dinámico del tamaño y rotación para Midlets de MIDP 2.0 a través del uso de una “Área de Ingreso Dinámica”, expandiendo el espacio entre columnas y rotando el control de movimiento [11].

Optimización

La programación con tecnología J2ME es diferente a la programación en la mayoría de lenguajes de escritorio [27] [28], esto debido las limitaciones de los dispositivos. Adicionalmente, el ambiente de desarrollo de aplicaciones J2ME generalmente es hecho en emuladores. MIDP es

una plataforma pequeña, el procesador de los dispositivos móviles probablemente sea mucho menos poderoso que los procesadores para computadoras de escritorio y la memoria seguramente es mucho menor. El objetivo de la optimización es hacer que la aplicación se ejecute rápidamente, manteniendo el nivel de usabilidad y la estructura del código. Una regla importante es optimizar sólo donde se lo necesite, primero hay que tomar en cuenta los aspectos como la limpieza y manutención y si hay problemas de rendimiento, identificarlos y optimizarlos, esto es “Escriba primero, entonces pruebe y luego optimice” [27].

Rendimiento

En las especificaciones J2SE y J2EE podemos encontrar una variedad de herramientas para examinar el rendimiento del código.

Las respuestas a los problemas no son fáciles, hay que tomar diferentes decisiones para dispositivos específicos. Y se necesita mucha más experiencia con el mundo real.

Entre algunos de los aspectos que se deben tomar en cuenta, tenemos [28]:

- Usar algoritmos que sean efectivos en el tiempo, se puede compensar con tamaño de código.
- No se puede usar el CPU del dispositivo para trabajos de cómputo intensos, se pueden usar aproximaciones.
- Poner atención en el nivel de programación de micro Java, hay que aprender Java muy bien.
- Usar algoritmos que sean efectivos en el tamaño, se puede compensar con tiempo de ejecución.
- El dispositivo no puede almacenar grandes cantidades de información. Almacene temporalmente la información y trate de anticipar futuros requerimientos.
- Recolección de Basura, cada carga de objetos potencialmente puede causar una Recolección de Basura, hay que estar pendiente por la falta de memoria.
- La mejor aplicación inalámbrica es aquella que usa la conexión al mínimo. La conexión de radio consume potencia de la batería.

- Hacer consultas al servidor para ver si hay datos no es una buena estrategia para conservar la potencia de la batería, la mayoría del tiempo no hay nueva información en el servidor y hay que subir datos al servidor sólo cuando este disponible.

Herramientas para medir el rendimiento de la aplicación Cliente

Para poder medir el rendimiento de la aplicación se ha utilizada varias herramientas, a continuación se revisan estas y su campo de acción.

El API de J2ME

El API de J2ME ofrece métodos muy útiles al momento de examinar el uso de memoria y duración de los procesos, el código de ejemplo 1 muestra un método para medir el tiempo que se demora un proceso:

```
long inicio, fin;
inicio = System.currentTimeMillis();
someMethod();
fin = System.currentTimeMillis();
long duracion = fin - inicio;
```

Código de ejemplo 1.- Examinar el tiempo de ejecución de un proceso. Realizado por autores.

El código de ejemplo 2 muestra un método para medir el consumo de memoria dinámica de un proceso.

```
Runtime runtime = Runtime.getRuntime();
long antes, despues;
System.gc();
antes = runtime.freeMemory();
Object newObject = new String();
depues = runtime.freeMemory();
long memoria_usado = antes - despues;
```

Código de ejemplo 2.- Verificación de la memoria usada. Realizado por autores. Wireless ToolKit 2.4 de Sun (WTK2.1)[41]

Otro método más automático para examinar él rendimiento del código son las herramientas de monitoreo del Micro Edition Wireless Toolkit para desarrollar aplicaciones J2ME de Sun. Las opciones que se pueden acceder son:

- Monitor de Memoria

[41] J2ME Wireless Tolkit, es un conjunto de herramientas que provee Sun para ejecutar tareas como compilación, ejecución, empaquetamiento, monitoreo y otras para los desarrolladores sobre la plataforma tecnológica J2ME, http://java.sun.com/mobility

- Monitor de Red
- Trazar las Excepciones
- Trazar la carga de clases
- Trazar el acceso a métodos
- Trazar los colectores de basura

En las siguientes secciones revisaremos los puntos críticos a ser revisados.

Optimización de los recursos de un dispositivo móvil inalámbrico

Es fácil para programadores J2SE desentenderse acerca del manejo de memoria. Después de todo, tener un colector de basura significa que no se debe preocupar acerca de liberar memoria y además este se ejecuta en hilo de baja prioridad.

En el universo J2ME, sin embargo, el uso de la memoria debe ser manejado con respeto, mucho más si el poner un colector de basura puede afectar a la velocidad de la aplicación.

Crear y Descartar Objetos

Si se está creando nuevos objetos dentro de un lazo, esto debe de ser una alarma, cada vez que se crea un objeto (***new Object***), nueva memoria es posicionada y peor, cada objeto que es creado al inicio del lazo hace a que el runtime ejecute una recolección de basura al final de lazo, el siguiente ejemplo de código muestra el problema.

```
// ConFigurar las entradas y salidas en
arreglos.
Object[] entradas = new Object[0];
int[] resultados = new int[0];
// Procesa cada entrada en un resultado.
for (int i = 0; i < inputs.length; i++) {
  Procesador p = new Procesador(entradas[i]);
  resultados[i] = p.calculeResultado();
}
```

Código de ejemplo 3.- Mala programación sobre un lazo.

Realizado por autores.

En esta sección típica de código encontramos algunas penalidades:

- En lugar de crear un nuevo objeto para cada proceso, se puede crear el mismo objeto fuera del lazo y actualizar las entradas.

- No revalúe las expresiones de terminación en el lazo. El código de ejemplo 4 muestra la optimización del código de ejemplo 3. Esto se puede hacer siempre y cuando el contenedor no vaya a cambiar de tamaño. Asegúrese de que múltiples hilos no se encuentres involucrados ya **que el orden transversal no es importante.**

```
// ConFigurar las entradas y salidas en
arreglos.
Object[] entradas = new Object[0];
int[] resultados = new int[0];
// Procesa cada entrada en un resultado.
Procesador p = new Procesador();
for (int i = inputs.length-1; i >= 0; --i) {
  Procesador p = p.setEntradas(entradas[i]);
  resultados[i] = p.calculeResultado();
}
```

Código de ejemplo 4.- Optimización del Ejemplo 4.7. Realizado por autores.

Strings y StringBuffers

Las cadenas son los únicos objetos donde el operado **+** es sobrecargado. Cada vez que se concatena cadenas con el operador + nuevos objetos son creados. Es una mejor estrategia usar *StringBuffers* en vez de *Strings* cuando se hacen concatenaciones.

Trate de eliminar el uso de *Strings* Literales del código, *String x = new String ("hola"),* crea 2 objetos, es mucho mejor: `String x = "hola"`. Además, no cree y sobre cargue Strings dentro de lazos, use *StringBuffer*.

Arreglos u Objetos

El uso de *Vector* o *Hashtable* es simple y conveniente, pero extremadamente costoso. Usar Arreglos correctamente dimensionados ahorra memoria y poder de procesamiento.

Ser Limpios

Un consejo muy útil es ser limpios en la programación. Liberar recursos tan pronto hayan hecho el trabajo puede incrementar el rendimiento de la aplicación. Si se usan estructuras de datos y arreglos, se debería referenciar a ***null*** los objetos y ejecutar una recolección de basura.

Las conexiones de red también debería ser liberadas tan pronto hayan hecho el trabajo esto se puede hacer con la cláusula ***finally***.

Use un Ofuscador de Código

Un ofuscador es una herramienta que hace difícil la reingeniería inversa a las clases compiladas. Aunque no es su propósito principal reduce el tamaño de los nombres que se usan para las variables y de los métodos, esto reduce en gran medida el tamaño del código.

Sumario

Corrija las falencias en el código antes de optimizar.

Haga las mediciones de rendimiento y después optimice

Use algoritmos eficientes

Minimice la creación y retención de objetos

Use el lenguaje de programación Java Eficientemente

2.4.5 Consideraciones sobre el diseño de las Interfaces

Simplificar las interfaces de usuario es un objetivo de todos los diseñadores, hay que encontrar el equilibrio entre la funcionalidad y la simplicidad.

Mientras más funcional es el producto, por lo general, es más difícil de aprender y usar. Por el contrario, si el

producto no es ampliamente funcional es probable que no cubra las necesidades del cliente.

Los diseñadores de aplicaciones usan la regla “80/20”: Identifican el 20% de la funcionalidad de modo que reúna el 80% de las necesidades del consumidor y optimizan su diseño en base a esto. Después que se ha dado soporte a este 20%, se puede considerar si incluir o no algunas características que se encuentren en el 80% de la funcionalidad restante. Además, si se incluyen características adicionales al 20% básico no deberá cambiarse el diseño para proporcionar el mismo nivel de acceso que al porcentaje crítico.

En las siguientes secciones se detallan algunas consideraciones para desarrollar aplicaciones para dispositivos móviles.

Hacer las aplicaciones simples

Es preferible reducir la abundancia por características que se usen ampliamente y remover opciones que no sean esenciales. Por ejemplo Un MIDlet[42] siute puede contener uno o más MIDlets. Un MIDlet suite conteniendo multiple

[42] Midlet.- Clase perteneciente al API del J2ME, que representa las aplicaciones J2ME, http://java.sun.com/j2me

MIDlets esta forzando al consumidor a elegir cual MIDlet iniciar. Si un MIDlet suite contiene un MIDlet sencillo, la implementación de este MIDlet podría dar inicio al suite, liberando al cliente de tener que elegir de entre varios MIDlets.

La implementación de un MIDlet que remueve esos puntos de elección al consumidor tiene un diseño más sencillo y reúne necesidades de los consumidores.

Hacer la aplicación predecible

Los consumidores deberían ser capaces de predecir que sucederá cuando ellos realizan una acción, es decir, raramente se recurrirá a la documentación del producto.

Por ejemplo, asuma una aplicación de correo sobre un teléfono móvil y una aplicación de libreta de direcciones. La primera tiene menús que presentan las operaciones disponibles, y la libreta de direcciones tiene una lista de elementos llamado: “New Entry for adding record”. La pregunta para la aplicación de correo debería ser: Write or New Message. Usando “New Message” tiene sentido paralelo a la opción “New Entry” en la libreta de

direcciones, esto hace que las dos aplicaciones tengan un mayor grado de consistencia.

Un consumidor no piensa en crear un nuevo mensaje, el simplemente desea escribirle a alguien, por ende, la palabra “*Write*” sería una opción más predecible para la acción de escribir un mensaje.

Si se debe elegir entre predecible y eficiente, se elige predecible.

Tareas simples para lograr eficiencia

Minimizar la cantidad de navegación e interacciones de usuario que se requieran para completar tareas frecuentes o cruciales. Estas deberían requerir el menor costo posible.

Cuando se considere esta recomendación recuerde que ser predecible es más importante que ser eficiente en dispositivos móviles.

Una manera de minimizar la navegación es el diseñar aplicaciones con un nivel jerárquico no muy profundo de modo que el consumidor no vaya de pantalla en pantalla para encontrar lo que busca para que no tenga que

regresar al principio de la navegación recorriendo todas las pantallas visitadas.

Hacer que el sistema responda

Los consumidores esperan que los sistemas respondan inmediatamente a sus acciones o entradas, cuando una respuesta no es inmediata, ellos pueden tornarse irritados, repetidamente presionan los botones y asumen que el dispositivo esta dañado.

Por ejemplo: el dispositivo puede pedir confirmación sobre acciones como presionar un botón o con un sonido al hacer un clic y en algunas circunstancias una aplicación puede usar retroalimentación visual para decir al consumidor que el dispositivo capto sus entradas.

Proveer retroalimentación constante y que no obstruya las tareas que se están realizando

Siempre se debe tener en cuenta que siempre sobre la pantalla se debe tener algo que muestre que el producto está encendido, el indicador no debería se frustrante o que distraiga, este podría ser parecido al indicador de señal o

de poder de los teléfonos móviles como lo indica la Figura 8(a).

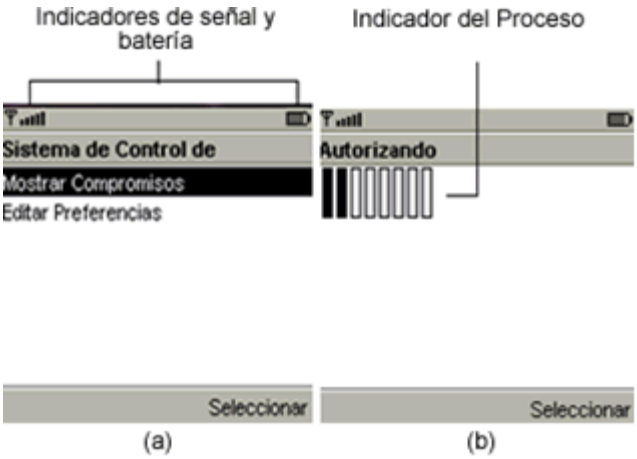


Figura 8.- (a) Indicador de señal en teléfonos móviles, (b) Indicador de conexión a Internet. Realizado por autores

Además, estos indicadores deben estar posicionados en lugares donde no interfieran con cualquier otra información. Animaciones y sonidos pueden expresar retroalimentación al consumidor. En algunos casos, animaciones pequeñas pueden ser menos obstructivas que otros tipos de retroalimentación. Por ejemplo una animación que indique la conexión a Internet no debe cambiar rápidamente, es más efectivo que lo haga despacio, como lo indica la 15(b).

Todo proceso debe ser interrumpibles

Se debe habilitar al consumidor para interrumpir los procesos sin importar en que estado se encuentre el sistema. Siempre se debe poder cancelar, apagar o simplemente interrumpir los procesos que se estén ejecutando. En todos los casos el consumidor debería poder reiniciar sus actividades sin ninguna penalidad.

Por ejemplo: cuando un PDA muestra una alerta, el consumidor debería poder se capaz de cerrarla no sólo presionando el botón requerido sino también presionando en cualquier otra parte de la pantalla. De la misma manera si se escribe un correo electrónico y se va a otra aplicación el sistema debería guardar automáticamente el mensaje como borrador, sin mostrar mensajes u otra advertencia.

Minimizar las interrupciones desde la aplicación y la máquina virtual

Restringir las interrupciones a la información crítica. Los consumidores no deberían se interrumpidos y requerir una respuesta por parte de ellos a menos de que sea absolutamente necesario. Los Mensajes de confirmación,

retroalimentación innecesaria y mensajes de error que requieran respuesta, disminuyen una buena experiencia del usuario ante al sistema, se debe usar mecanismos menos obstructivos, tales como alertas temporizadas, para proveer retroalimentación.

2.4.6 Consideraciones sobre la Base de Datos de la ESPOL

Actualmente, la base de datos que usa la ESPOL es DB2 UDB (Universal Data Base) V6.1 que se actualizó en el año 1999, comercialmente la compañía IBM provee la versión 8.1. La integración de la tecnología Java con DB2 V6.1 fue un factor de incompatibilidad grave del sistema, debido a que la versión 6.1 de DB2 incorpora en sus librerías el lenguaje de programación Java. DB2 V6.1 tiene incorporado su propia máquina virtual basada en las especificaciones de Java 1.1.7 y desarrollada por IBM, esta especificación de Java no implementa JDBC[43] 2.0, lo que da como consecuencia que la versión 6.1 de DB2 no implementa la interfaz *javax.sql.DataSource* que representa una fábrica de conexiones a la fuente física de datos, la cual requerida por el servidor de aplicaciones

43 JDBC.- Java Data Base Connection, JDBCTM API provee un acceso universal a datos desde el lenguaje de programación Java , *J2SE v1.4.2,* Java Programing Lenguaje Documentation, <www.java.sun>

escogido para desarrollo, *Sun Application Server 8.0 Update 1* para crear los recursos de persistencia que necesita la aplicación de servidor del SCI

En la búsqueda de una solución para este inconveniente se probó una infinidad de manejadores de terceras compañías que tenían la implementación de la interfaz *Datasource* para DB2, pero todos dieron el mismo resultado: "*error por incompatibilidad de la versión*".

De la misma manera la nueva versión de DB2 distribuido por IBM tiene la siguiente restricción: En DB2 Versión 8 los manejadores tipo 2 y tipo 3 continúan usando la interfaz propietaria de DB2 para comunicarse con los servidores DB2 UDB: OS/390® y z/OS™, UNIX®, Windows®, Linux, e iSeries™. Con lo que se determina que la compatibilidad para DB2 UDB V6.1 sobre Sun Solaris no está dada en la versión actual de DB2 [34].

Después de un proceso de pruebas se encontró que sólo la versión 7.2 de DB2 provee compatibilidad para la versión 6.1 de DB2. Además, la versión 7.2 si implementa la interfaz *DataSource* necesitada.

2.4.7 Consideración sobre el servicio de Directorio

El servicio de directorio de la ESPOL está basado en el estándar X500[44] para servicios de directorio siendo parte del modelo *Open Systems Interconect* (*OSI Suite of Services*), el directorio de la ESPOL contiene información de estudiantes, personal administrativo y docente.

Este servicio se implementó en Septiembre del 2003 sobre un servidor de Windows 2000 Server SP4, y permite mecanismos para la organización de usuarios, autenticación de red que ofrece Windows en base a *Kerberos*[45]*,* y por último, ofrece el servicio de acceso al directorio a través de *Lightweight Directory Access Protocol* (LDAP)[46]

El SCI usa el directorio de la ESPOL para autenticar a los usuarios del sistema, a través del servicio de autenticación

---

44 X500.- El servicio de directorio es un servicio global, que se compone de información de administración global como países, organizaciones, gente, máquinas y cosas de de alcance global. La información mantenida en un directorio de base de información (DIB). Estas entradas son mantenidas en una estructura tipo árbol. Cada entrada es un objeto y contiene una serie de atributos con uno o más valores. El esquema del directorio define obligatoriamente y opcionalmente los atributos para cada clase de objeto.
El espacio de nombres de X500 es jerárquico, y una entrada es definido por un único *distinguished name* (DN). Un *distinguished name* es una concentración de atributos que califican a un objeto en particular. Para el caso de la ESPOL tenemos: cn=hrobayo,ou=users,ou=fiec,dc=espol,dc=edu,dc=ec, The Java Tutorial, Sun Microsystems Inc., CCITT X.500 (1988/1993)/ISO Directory

45 Kerberos.-Estándar de Ingeniería para la autenticación, autorización e integridad de datos en una red, Windows 2000 Kerberos Authentication, "The Kerberos Network Authentication Service (V5)," RFC 1510, September 1993, http://www.microsoft.com/technet/prodtechnol/windows2000serv/deploy/confeat/kerberos.mspx

46 LDAP.- EL Protocolo Ligero de Acceso al Directorio (LDAP) v3 (RFC 2251), permite el acceso al servicio de directorio.

de Red de *Windows 2000 Kerberos* y para obtener datos del usuario a través de *LDAP*.

2.5 Análisis de las Herramientas

La elección de las herramientas de desarrollo se ha basado en el soporte y flexibilidad que brindan, además, de esto dependió el éxito del proyecto.

2.5.1 Tendencias en el desarrollo de software

En los últimos años, las tendencias en el desarrollo de aplicaciones se orientan a usar varias herramientas en un solo proyecto. Esto permite a los programadores decidir entre varias herramientas y escoger las más convenientes.

En años recientes, uno de las falencias en el área del desarrollo de software ha sido la aparición de software comercial de baja calidad que es poco compatible con la mayoría de las TIC actuales. El Instituto Nacional de Estándares y Tecnología de los Estados Unidos (www.nist.gov), estima que en el 2001, los errores en el código costaron a las compañías estadounidenses $59.5 billones de dólares. El Consorcio para la Computación Sustentable (SCC, www.sustaintablecomputing.org), en

una iniciativa académica, gubernamental, en negocios de las TI y mejoramiento de software estima que anualmente el costo de software defectuoso es de $200 billones de dólares. El Grupo Standish, una firma de investigación de mercado, reporta que en el 2002 sólo el 34% de los proyectos de software fueron completados en el presupuesto convenido con todas las características iniciales y que alrededor del 17% fallaron completamente.

2.5.2 Herramientas de desarrollo

Las tres súper-compañías IBM, Oracle y Sun Microsystems comenzaron proyectos independientes con diferentes enfoques pero con la misma meta: Mejorar la calidad del software con mejores herramientas.

El objetivo de este análisis es disminuir costos y eliminar el trabajo de hacer depuración. La Figura 9 muestra una relación de la cantidad de errores y el tiempo en que se hacen las pruebas y depuraciones y como se afecta esta relación cuando las pruebas se realizan en tempranas etapas del desarrollo.

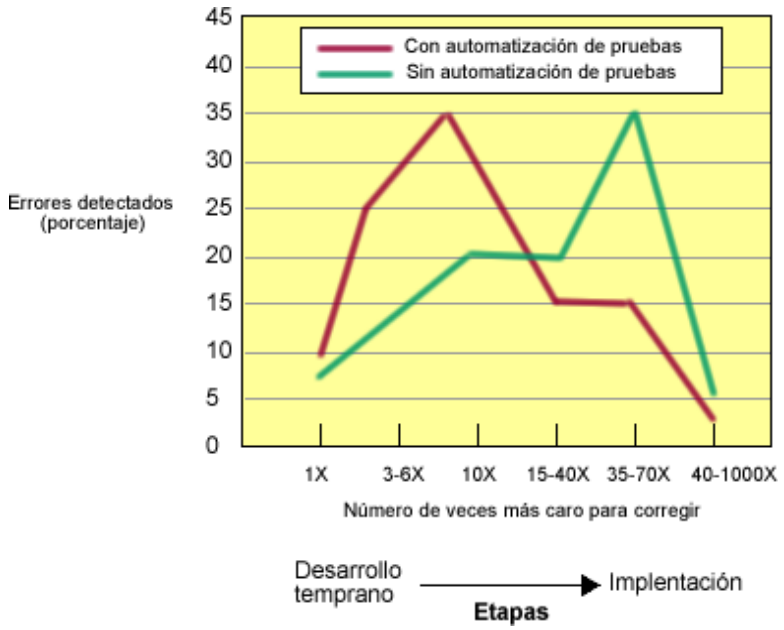


Figura 9.- Usando una herramienta de diagnostico se puede ahorrar dinero detectando errores en etapas tempranas del desarrollo.

Referencia: Mercury Interactive; Siebel Systems

En este sentido, en el mercado se encuentran varias herramientas de desarrollo de aplicaciones de software de uso libre entre las que se puede nombrar:

Eclipse

La Fundación Eclipse (www.eclipse.org) fue originalmente creada por un grupo de industrias líderes del mercado en el 2001.

Eclipse es una plataforma abierta para la integración y construcción de herramientas, totalmente desarrollado sobre la plataforma Java.

Una particularidad de Eclipse es que su interfaz gráfica está basado en el API SWT, el cual provee mecanismos para crear interfases graficas directamente desde el sistema operativo, lo cual desemboca en una aplicación más rápida, como lo muestra la Figura 10. Sin embargo, esto disminuye la portabilidad del IDE.

Además, Eclipse tiene integradas herramientas de reconstrucción (*refactoring*), detección (*debuging*) y rastreo (*trace*) de errores.

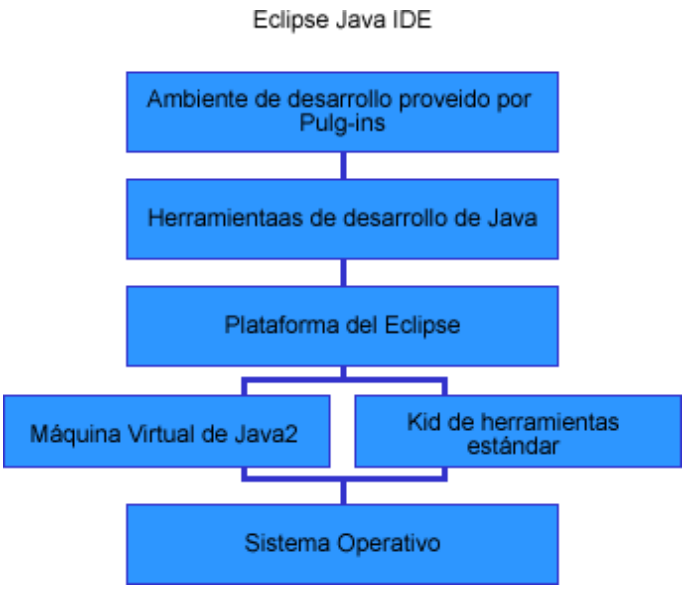


Figura 10.-.Diagrama de bloques del Eclipse IDE Referencia: Lee Graber, IEEE Computer Magazine página 22, The battle over the universal Java IDE.

Netbeans

Originalmente *NetBeans* (www.netbeans.org) fue desarrollado como un proyecto de Roman Stanek (estudiante en la República Checa) a mediados de los 90's, en. En 1999 Sun Microsystems compró *NetBeans* y en Junio 2000 Sun Microsystems lo hizo de fuente abierta como parte de un objetivo estratégico para popularizar el uso de Java.

La característica principal que diferencia *NetBeans* de Eclipse es el API de interfaz gráfica con el cual están desarrollados, mientras Eclipse usa SWT, NetBeans usa Swing como lo muestra la Figura 11, que es un API independiente de la plataforma y crea su propia interfaz gráfica, haciendo de *NetBeans* muy portable, sin embargo, esto disminuye drásticamente su rendimiento.

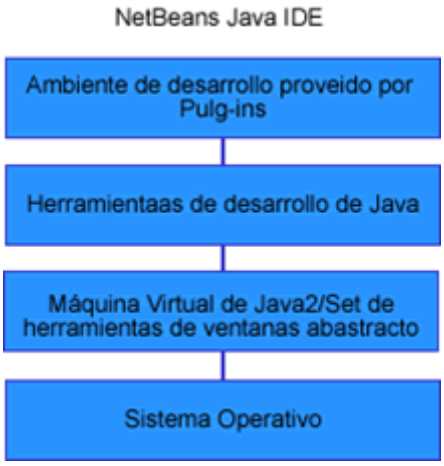


Figura 11.-Diagrama de bloque de NetBeans, Referencia: Lee Graber, IEEE Computer Magazine página 22, The battle over the universal Java IDE.

2.5.3 Elección de las herramientas

Para el desarrollo del SCI, se escogió Eclipse y por Websphere Studio Device Developer Versión 5.6 de IBM (basado en Eclipse). Una ventaja clave sobre *NetBeans* es debido a que la Corporación IBM y la empresa *PalmOne* que fabrica y comercializa los productos Palm, firmaron un convenio en el cual IBM es acreditada como el proveedor oficial de la máquina virtual de Java para los dispositivos Palm.

2.6 Aspectos Generales de Seguridad

La seguridad del SCI esta basada en 3 aspectos: autentificación, autorización e integridad de datos, debido al uso de computación móvil y tecnologías inalámbricas el desarrollo se enfrenta el reto de mantener un buen esquema de seguridad, en esta sección se exponen algunos aspectos relacionados a la seguridad en el SCI.

2.6.1 Seguridad en redes inalámbricas

El acceso sin necesidad de cables es el motivo por el cual las redes inalámbricas son populares, este es a la vez el problema más grande en cuanto a seguridad se refiere. Cualquier equipo que se encuentre en el rango de un AP[47], podría tener acceso a la red inalámbrica.

Por ejemplo, en un mismo edificio varias redes pueden estar en funcionamiento, una situación altamente riesgosa [14] como lo muestra la Figura 12, los controles a tener en cuenta en estas situaciones corren por cuenta del administrador de redes para que de esta manera personas ajenas a la empresa no usen esta indebidamente (colocar virus, robar información, etc.).

[47] AP.- Punto de Acceso Inalámbrico, para redes con infraestructura inalámbrica.

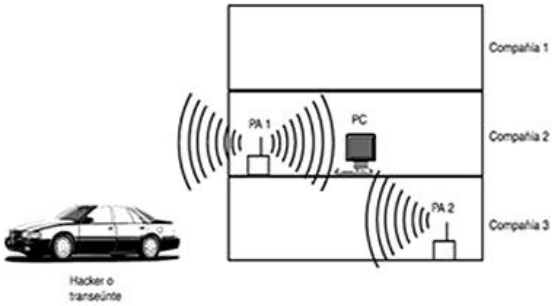


Figura 12.- Acceso a la red por personas no autorizadas, Referencia [14]

Un estudio publicado en 2003 por RSA *Security Inc*.[48] encontró que de 328 puntos de acceso inalámbricos que se detectaron en el centro de Londres, casi las dos terceras partes no tenían habilitado el cifrado mediante WEP (*Wired Equivalent Protocol*). Además, cien de estos puntos de acceso estaban divulgando información que permitía identificar la empresa a la que pertenecían, y 208 tenían la conFiguración con la que vienen de fábrica.

Existen dos prácticas bien conocidas para localizar redes inalámbricas [14]:

- **El warchalking**,[49] consiste en caminar por la calle con un computador portátil dotado de una tarjeta

[48] Dennis Fisher. *Study Exposes WLAN Security Risks*. Marzo 12 de 2003. <http://www.eweek.com/print_article/0,3048,a=38444,00.asp>

[49] *Warchalking*. <http://www.warchalking.org>

WLAN, de esta manera la computadora detectara las señales que están presentes y luego se marca el lugar, es posible colocar información adicional para que otras personas hagan uso de este "descubrimiento", como lo muestra la Figura 13.

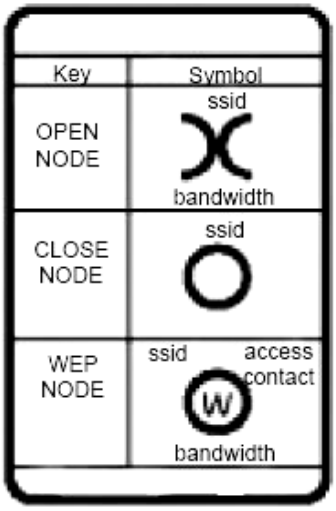


Figura 13.- Warchalking y su simbología, Ref.:

http://www.warchalking.org

- **El wardriving**, similar al anterior solo que esta vez el proceso se lleva a cabo desde un automóvil. Se emplean una computador portátil con una tarjeta WLAN, una antena adecuada, un GPS para localizar los puntos de acceso en un mapa, y software para detección de redes inalámbricas, gratis en Internet.

Una vez se obtenga acceso a la red se puede hacer lo siguiente:

- Uso no autorizado de recursos.
- Configurando un punto de acceso propio es decir engañar a las maquinas legítimamente conectadas a conectarse con la maquina que esta atacando la red, de ese modo el atacante puede proceder a robar información

Además de estos métodos simples de acceso no autorizado a redes inalámbricas, existen mecanismos más complejos que no van a ser tratados en este proyecto.

La red 100% segura es un ideal pero lo que principalmente se debe tener en cuenta lo siguiente aspectos de seguridad [15]:

- Las ondas de radio deben confinarse tanto como sea posible. Las antenas direccionales son una buena opción así como la buena regulación de la potencia en la transmisión de los puntos de acceso
- Autenticación en doble vía, que permita al cliente verificar que se está conectando a la red correcta, y a

la red constatar que el cliente está autorizado para acceder a ella.

- Cifrado de Datos, para evitar que equipos ajenos a la red puedan capturar datos mediante escucha pasiva.

Existen varios métodos para intentar una red inalámbrica segura pero presentan ventajas y desventajas. Algunos de estos métodos son:

Filtrado de direcciones MAC

Consiste en tener una tabla con las direcciones MAC [16] [18] autorizadas para un determinado punto de acceso. Como toda tarjeta de red posee una dirección MAC única, se logra autenticar el equipo.

Es un método sencillo, y aplicado en pequeñas empresas seria ideal. Su sencillez es a la vez su gran desventaja al momento de ponerlo en práctica en grandes empresas, como se hace notar a continuación:

- La actualización manual de las tablas puede llegar a ser engorrosa.
- El manejo de direcciones MAC (en formato hexadecimal puede dar cabida a errores.

- Las direcciones MAC viajan sin cifrar por el aire. Un atacante podría capturar direcciones MAC de tarjetas matriculadas en la red empleando un *sniffer*, y luego asignarle una de estas direcciones capturadas a la tarjeta de su computador, empleando programas tales como *AirJack*[50] o *WellenReiter*[51], entre otros. De este modo, el atacante puede hacerse pasar por un cliente válido.

- De darse el robo de un equipo inalámbrico, el ladrón dispondrá de un dispositivo que la red reconoce como válido. En caso de que el elemento robado sea un punto de acceso el problema es más serio, porque el punto de acceso contiene toda la tabla de direcciones válidas en su memoria de conFiguración.

Este método no garantiza la confidencialidad de la información transmitida, ya que no prevé ningún mecanismo de cifrado.

---

[50] *Wellenreiter* – WLAN Hacking. http://www.wellenreiter.net/

[51] *WEPCrack Project Info.* http://sourceforge.net/projects/webcrack

Wired Equivalent Privacy (WEP)

El algoritmo WEP [16] [17] [18] forma parte de la especificación 802.11, y de diseñó con el fin de proteger los datos que se transmiten en una conexión inalámbrica mediante cifrado. WEP opera a nivel 2 del modelo OSI y es soportado por la gran mayoría de fabricantes de soluciones inalámbricas. La Figura 14 muestra el método de cifrado del algoritmo WEP.

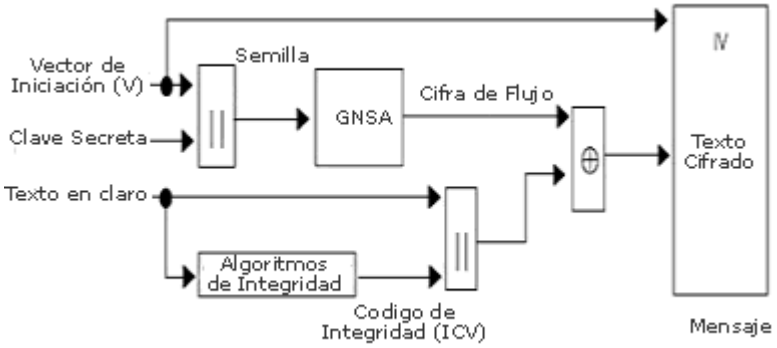


Figura 14.- Algoritmo WEP en modalidad de cifrado, Referencia [14]

A la trama en claro se le computa un código de integridad (*Integrity Check Value*, ICV) mediante el algoritmo CRC-32. Dicho ICV se concatena con la trama, y es empleado más tarde por el receptor para comprobar si la trama fue alterada durante el transporte.

Se escoge una clave secreta compartida entre emisor y receptor. Esta clave puede poseer 40 ó 128 bits.

Si se empleara siempre la misma clave secreta para cifrar todas las tramas, dos tramas en claro iguales producirían tramas cifradas similares. Para evitar esta eventualidad, se concatena la clave secreta con un número aleatorio llamado vector de inicialización (IV) de 24 bits. El IV cambia con cada trama.

La concatenación de la clave secreta y el IV (conocida como semilla) se emplea como entrada de un generador RC4 de números seudo- aleatorios. El generador RC4 es capaz de generar una secuencia seudo-aleatoria (o cifra de flujo) tan larga como se desee a partir de la semilla.

El generador RC4 genera una cifra de flujo, del mismo tamaño de la trama a cifrar más 32 bits (para cubrir la longitud de la trama y el ICV). Se hace un XOR bit por bit de la trama con la secuencia de clave, obteniéndose como resultado la trama cifrada y por último el IV y la trama se transmiten juntos

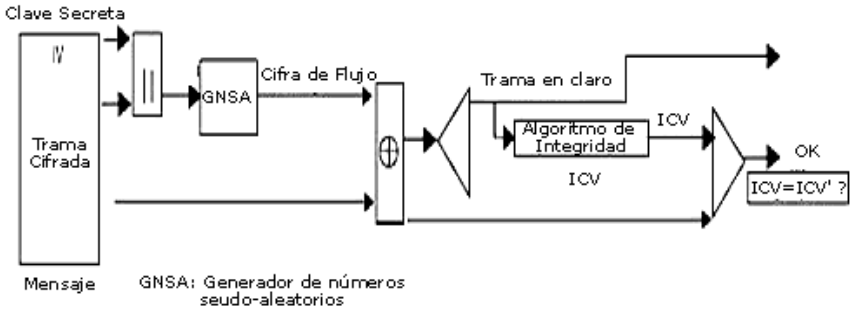


Figura 15.- Algoritmo WEP en modalidad de cifrado, Referencia [14]

La Figura 15 muestra el proceso de descifrado se lleva a cabo en el receptor de descifrado, este proceso se detalla a continuación:

- Se emplean el IV recibido y la clave secreta compartida para generar la semilla que se utilizó en el transmisor.
- Un generador RC4 produce la cifra de flujo a partir de la semilla. Si la semilla coincide con la empleada en la transmisión, la cifra de flujo también será idéntica a la usada en la transmisión.
- Se efectúa un XOR bit por bit de la cifra de flujo y la trama cifrado, obteniéndose de esta manera la trama en claro y el ICV.
- A la trama en claro se le aplica el algoritmo CRC-32 para obtener un segundo ICV, que se compara con el recibido.

- Si los dos ICV son iguales, la trama se acepta; en caso contrario se rechaza.

El algoritmo WEP resuelve aparentemente el problema del cifrado de datos entre emisor y receptor. Sin embargo, existen tres situaciones que hacen que WEP no sea seguro en la manera que es empleado en la mayoría de aplicaciones:

- La mayoría de instalaciones emplea WEP con claves de cifrado estáticas (se conFigura una clave en el punto de acceso y no se la cambia nunca, o muy de vez en cuando). Esto hace posible que un atacante acumule grandes cantidades de texto cifrado con la misma clave y pueda intentar un ataque por fuerza bruta.
- El IV que se utiliza es de longitud insuficiente (24 bits). Los IV para cada trama son diferentes por tanto solamente es cuestión de tiempo para que se agote el espacio de 224 IV distintos. En redes caseras no se advierte el problema, pero en una red que posea alto tráfico se puede agotar el espacio de los IV en más o menos 5 horas. Si el atacante logra conseguir dos tramas con IV idéntico, puede efectuar un XOR

entre ellas y obtener los textos en claro de ambas tramas mediante un ataque estadístico. Con el texto en claro de una trama y su respectivo texto cifrado se puede obtener la cifra de flujo; conociendo el funcionamiento del algoritmo RC4 es posible entonces obtener la clave secreta y descifrar toda la conversación.

- WEP no ofrece servicio de autenticación. El cliente no puede autenticar a la red, ni al contrario; basta con que el equipo móvil y el punto de acceso compartan la clave WEP para que la comunicación pueda llevarse a cabo. Actualmente existen programas para romper claves que utilizar WAP. WEPCrack,[52] que consiste en una serie de scripts escritos en lenguaje Perl diseñados para analizar un archivo de captura de paquetes de un sniffer. La herramienta AirSnort[53] hace lo mismo, pero integra las funciones de sniffer y rompedor de claves, y por lo tanto es más fácil de usar. Airsnort captura paquetes pasivamente, y rompe la clave WEP cuando ha capturado suficientes datos.

52 WEPCrack Project Info. <http://sourceforge.net/projects/wepcrack>
53 AirSnort Homepage. <http://airsnort.shmoo.com/>

WPA (WI-FI Protected Access)

WPA[54], es un estándar propuesto por los miembros de la Wi-Fi Alliance (que reúne a los grandes fabricantes de dispositivos para WLAN) en colaboración con la IEEE. Este estándar busca subsanar los problemas de WEP, mejorando el cifrado de los datos y ofreciendo un mecanismo de autenticación.

Para solucionar el problema de cifrado de los datos, WPA propone un nuevo protocolo para cifrado, conocido como TKIP (Temporary Key Integrity Protocol). Este protocolo se encarga de cambiar la clave compartida entre punto de acceso y cliente cada cierto tiempo, para evitar ataques que permitan revelar la clave. Igualmente se mejoraron los algoritmos de cifrado de trama y de generación de los IVs, con respecto a WEP.

El mecanismo de autenticación usado en WPA emplea 802.1x y EAP, que fueron discutidos en la sección anterior.

Según la complejidad de la red, un punto de acceso compatible con WPA puede operar en dos modalidades:

---

[54] Wi-Fi Alliance. *Overview: Wi-Fi Protected Access*. Octubre 31 de 2002. <http://www.weca.net/OpenSection/pdf/Wi-Fi_Protected_Access_Overview.pdf>

- Modalidad de red empresarial: Para operar en esta modalidad se requiere de la existencia de un servidor RADIUS en la red. El punto de acceso emplea entonces 802.1x y EAP para la autenticación, y el servidor RADIUS suministra las claves compartidas que se usarán para cifrar los datos.

- Modalidad de red casera, o PSK (Pre-Shared Key): WPA opera en esta modalidad cuando no se dispone de un servidor RADIUS en la red. Se requiere entonces introducir una contraseña compartida en el punto de acceso y en los dispositivos móviles. Solamente podrán acceder al punto de acceso los dispositivos móviles cuya contraseña coincida con la del punto de acceso. Una vez logrado el acceso, TKIP entra en funcionamiento para garantizar la seguridad del acceso. Se recomienda que las contraseñas empleadas sean largas (20 o más caracteres), porque ya se ha comprobado que WPA es vulnerable a ataques de diccionario si se utiliza una contraseña corta.[55]

[55] WPA's Little Secret. Noviembre 4 de 2003. <http://www.stargeek.com/item/20270.html>

La norma WPA data de abril de 2003, y es de obligatorio cumplimiento para todos los miembros de la Wi-Fi Alliance a partir de finales de 2003. Según la Wi-Fi Alliance, todo equipo de red inalámbrica que posea el sello "*Wi- Fi Certified*" podrá ser actualizado por software para que cumpla con la especificación WPA.

2.6.2 Manejo de la seguridad en el SCI

La seguridad en el SCI se maneja mediante la autenticación de los usuarios de la aplicación a través del servicio de directorio de la ESPOL como se detalla a continuación.

Mecanismo de autentificación

La autentificación de los usuarios se lo realiza contra el servicio de directorio de la ESPOL, en el directorio se encuentran registrados todos los estudiantes, profesores y personal administrativo de la ESPOL, además, este está implementado sobre el servicio del Directorio Activo de Windows.

La autentificación se realiza mediante el servicio de autentificación *Kerberos* del Windows 2000 Server [19].

Ingresando usuario y la clave asignado a cada miembro del directorio, el sistema cliente envía las credenciales (*username y password*) del usuario hacia el servidor de aplicaciones, después la aplicación que se ejecuta en el servidor de aplicaciones crea una sesión usando el servicio *Kerberos*, si el usuario es autenticado, se ejecuta una tarea privilegiada en nombre del usuario para obtener datos como: nombre, dirección y otros, mediante el Protocolo Liviano de Acceso al Directorio (LDAP) [20].

Autorización de los usuarios

La autorización de los usuarios para los sistemas de la ESPOL, esta dada mediante permisos que asigna el Administrador de Bases de Datos a cada usuario del Directorio, además, no hay sincronización entre roles y permisos de la Base de Datos y el Servicio de Directorio de la ESPOL. Actualmente el SCI no tiene un mecanismo de autorización de los usuarios y únicamente autentifica la validez de las credenciales del usuario.

Debido a esta restricción el SCI no puede hacer una personalización de la aplicación, es decir todos los usuarios tiene el mismo perfil.

Integridad de Datos

El SCI utiliza varios mecanismos para asegurar la Integridad de datos.

El primer método son los mecanismos de encriptación que soporta el estándar IEEE 802.11b, estos métodos se los detalla más ampliamente en la sección 2.6.1 “Seguridad en redes inalámbricas”.

El segundo método es a través de software que provee la tecnología J2ME a través del protocolo HTTPS[56], que es la versión segura del protocolo HTTP.

[56] HTTPS. - Secure Hyper Text Transfer Protocol.

CAPÍTULO 3

3. RECOLECCIÓN DE DATOS Y REQUERIMIENTOS

En este capítulo se analiza el proceso actual de ingreso de inventarios en la bodega de la ESPOL y las especificaciones iniciales del sistema, que están basadas en los requerimientos del personal de bodega y en recomendaciones del personal del Centro de Servicios Informáticos de la ESPOL.

3.1 Análisis de Sistemas Actuales

Actualmente, se dispone de un sistema que administra los inventarios en la bodega, pero aun se llevan varios controles escribiendo los datos en documentos. Un ejemplo del proceso actual de ingreso de inventarios se describe a continuación:

- El personal administrativo recibe la documentación sobre la compra de inventarios.
- Se entrega al personal de bodega hojas con las listas con los datos de los inventarios a ser ingresados.

- El personal de bodega revisa cada ítem y escribe los datos como: marca, modelo, cantidad, etc. de cada inventario en las hojas.
- Este listado es remitido al personal administrativo e ingresado al sistema de la ESPOL.

El flujo de datos del proceso es muy corto, sin embargo, dada la extensa cantidad de información y el método de recolección es muy probable el surgimiento de errores.

3.2 Especificaciones iniciales del Sistema

Como especificaciones iniciales del sistema se mencionan las premisas de desarrollo del proyecto, así como, los documentos de los cuales se ha dispuesto.

La primera premisa es automatizar los procesos de ingreso y verificación de inventarios de la ESPOL. Los artículos (activos fijos o bienes consumibles) son ingresados (mediante ordenes de ingreso) por el personal de administración. Esta información debe ser transmitida al personal de bodega encargado del proceso de ingreso, despacho y verificación de los distintos artículos que llegan a bodega. Los requerimientos no funcionales para el SCI son:

- El uso de computación distribuida.
- La administración de datos persistentes.
- El sistema debe ser auto descriptivo y fácil de usar.
- El sistema deberá correr independiente de la plataforma.

La meta final del SCI es hacer más eficiente el proceso de ingreso de inventarios que se realizan en bodegas de la ESPOL con la ayuda un PDA.

3.3 Recolección de Requerimientos

Para establecer los requerimientos del personal de bodega se realizaron entrevistas al personal de bodega, de las cuales se puede concluir lo siguiente:

- Según el Ing. Fernando Escobar, jefe del departamento de Suministros de la ESPOL, manifiesta que siendo el departamento de suministros parte fundamental para el desarrollo de la ESPOL, no ha habido mejoras sustanciales de los procesos desde su implementación inicial,.
- Según el personal administrativo de las bodegas, en ciertas temporadas aumenta el trabajo significativamente debido al aumento de compra de bienes, lo cual conlleva a retrasos en las tareas y por ende demora los ingresos.

- En general se aprecia que el proceso actual de ingreso a bodega es lento y poco eficiente.

Según lo manifiesta el personal de bodega, se pueden obtener las siguientes percepciones personales sobre el proceso de ingreso de inventarios:

- El proceso actual de ingreso de inventarios es lento y susceptible de errores.
- Debido al aumento de trabajo, los ingresos a bodega se pueden demorar mucho tiempo.

3.4 Visión del Sistema

El sistema de Control de Inventarios debe de servir de guía en futuras investigaciones sobre aplicaciones de la computación móvil en la ESPOL y convertirse en una ayuda fundamental en el trabajo del personal de bodega de la ESPOL.

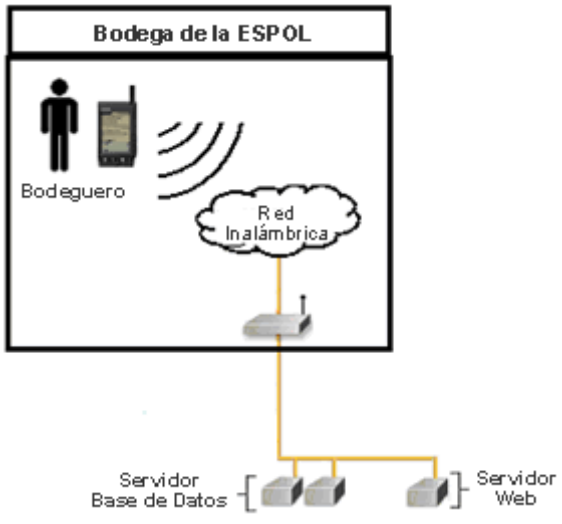


Figura 16.- Ilustración de los actores y tecnologías del sistema

La Figura 16 muestra una ilustración de los actores y tecnologías del sistema. Se puede observar que el bodeguero tendrá acceso a un dispositivo móvil, el cual estará conectado a una red inalámbrica

Los datos que ingrese el bodeguero serán directamente actualizados en las Bases de datos de la ESPOL a través de la red. Sin la intervención del personal administrativo.

CAPÍTULO 4

4. ANÁLISIS DEL SISTEMA

En este capítulo se analiza los requerimientos del personal de bodega y las recomendaciones del personal del Centro de Servicios Informáticos de la ESPOL. Además, se detallan también en este capítulo algunos diagramas estáticos de diseño del sistema como son los diagramas de caso de uso. En esta sección se expone teoría acerca de los modos de operación que debe soportar el sistema cliente.

Finalmente, se hace un análisis sobre el diseño de la interfaz de usuario que debe mostrar la aplicación cliente.

4.1 Análisis de Requerimientos

"Durante el análisis de requerimientos y especificaciones del sistema; un entendimiento común de la funcionalidad del sistema será establecida entre el usuario y los desarrolladores. La descripción de las técnicas debe ser simple y entendible por personas sin experiencia en modelado orientado a objetos" [22].

Según lo presenta las secciones 3.2 las especificaciones no funcionales son cubiertas mediante la introducción de un modelo de distribución que será explicado en la sección 5.2 Diseño de la Distribución, de igual manera, el manejo de la persistencia tanto en el cliente y en el servidor es explicado con detalle en sección 5.3 Decisiones de Diseño.

Para asegurar que el sistema sea independiente de la plataforma se ha escogido el lenguaje de programación Java, que es el que ofrece esta ventaja. Además, los requerimientos del personal de bodega son parte de los objetivos del desarrollo del SCI.

4.2 Análisis de los casos de uso

El modelo de casos de uso muestra a los usuarios y usos del sistema, muestran información no trivial sobre el comportamiento dinámico y la secuencia de transacciones del sistema [22]. Las limitaciones de los casos de uso son:

Los casos de uso no modelan la comunicación entre cada instancia de estos, aún cuando dicha comunicación existe.

No muestran los conflictos entre las instancias entre casos de uso.

No muestran concurrencia.

Los casos de uso modelan gran parte de los requerimientos funcionales del sistema, pero para obtener un beneficio mayor se

lo debe contrastar con los modelos dinámicos, diagramas de secuencia y de actividad.

4.2.1 Diagrama de casos de Uso

El SCI se divide en 2 partes, la aplicación cliente que se ejecuta en un dispositivo móvil y la aplicación de servidor, los casos de uso del SCI se muestran la Figura 17.

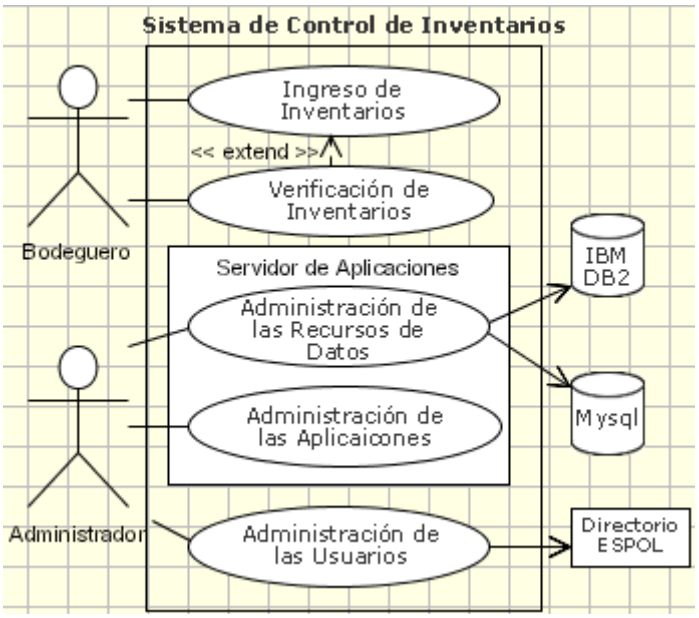


Figura 17.- Diagrama de casos de uso del SCI realizado por autores.

Los actores que interactúan con el SCI son:

- Bodeguero.- Guarda almacén que está encargado de recibir los bienes de los proveedores y hacer los ingresos y verificaciones.

- Administrador.- Es la persona encargada de administrar la aplicación en el Servidor de Aplicaciones y de crear los usuarios en el Servicio de Directorio

Además, el SCI hace uso del Directorio de la ESPOL e interactúa con 2 bases de datos: IBM DB2 y Mysql Server.

4.2.2 Descripción de los casos de uso.

Los casos de uso del Sistema de Control de Inventarios son:

- Ingreso de Inventarios.
- Verificación de inventarios.
- Administración de Recursos
- Administración de Usuarios
- Administración de Aplicaciones

Los casos de uso se dividen en 2 grupos, casos de uso de administración del sistema y casos que tienen relación con el desarrollo en si de la aplicación.

Casos de uso de administración del sistema

Debido al uso del la especificación J2EE para el SCI, la aplicación del lado del servidor se ejecuta sobre un servidor de aplicaciones J2EE, motivo por el cual se presenta la tarea de investigar y conFigurar el servidor de aplicaciones para que pueda acceder a los recursos de persistencia de datos como son las bases de datos relacionales IBM DB2 y Mysql, así como, al servicio de Directorio de la ESPOL.

De este modo los casos de uso de administración del sistema se los puede definir de la siguiente manera:

Administración de Recursos de Datos

Los recursos de a los que se refiere este caso de uso son las Bases de Datos como lo muestra la Figura 13, el rol del actor de este caso de uso es el de un Administrador de Servidor de Aplicaciones-

El caso de uso en si no es tratado en detalle en este documento, pues la conFiguración de los recursos de persistencia son implementados de diferente manera en cada servidor de aplicaciones, sin embargo, la correcta conFiguración de la capa de datos es fundamental para

el correcto funcionamiento de la aplicación que se ejecuta en el servidor del SCI.

Administración de Usuarios

La administración de usuarios no es parte del SCI, este caso de uso es implementado y administrado en aplicaciones propietarias del Centro de Servicios Informáticos quienes son los que administran este servicio. Se debe mencionar que el SCI utiliza este servicio y necesita de la buena administración de éste para su funcionamiento.

Administración de Aplicaciones

La administración de aplicaciones es la tarea primaria en el SCI. El actor en este caso de uso es el responsable de la correcta instalación de las aplicaciones en el servidor de su monitoreo y buen rendimiento.

Para más detalles sobre los casos de administración ver el Anexo 2 Manual de Instalación del SCI.

Ingreso de Inventarios

Ingreso a bodega es el proceso mediante el cual los artículos son ingresados a Bodega y donde se recoge datos relevantes para su posterior identificación.

Resumen:	El bodeguero se encarga de ingresar los datos (números de serie, modelo. Cantidades, etc.) de los artículos que lleguen a bodega
Frecuencia de Uso:	Diariamente, Semanal, mensual
Actores Directos:	Personal de Bodega: El Bodeguero que ingrese al sistema con su user y password.
Escenario Principal:	El Bodeguero ingresa a la aplicación.
	Del Menú principal de la aplicación elige obtener la orden de ingreso.
	Ingresa los datos que sean necesarios para cada artículo y se actualiza los inventarios.
	Confirma los cambios realizados y termina el proceso.
Escenarios Alternativos	El Bodeguero no podrá realizar ninguna operación si no ingresa al sistema.
	No se pueden ingresar artículos a la bodega si antes no se ha generado la respectiva orden de ingreso.

Tabla 10.- Ingreso a Bodega. Realizado por autores.

Verificación de Inventarios

Este proceso se realiza para poder verificar la existencia de inventarios en cualquier momento y estará disponible para los administradores o en su defecto para el personal autorizado.

Resumen:	El bodeguero requiere realizar la verificación de inventarios en bodega ya sea para emitir algún reporte o simplemente como una verificación de rutina.
Frecuencia de Uso:	Mensual, anual.
Actores Directos:	Personal de Bodega: El Bodeguero que ingrese al sistema con su user y password.
Escenario Principal:	El Bodeguero ingresa a la aplicación.
	Del Menú principal elige verificar inventarios.
	Para cada articulo que el necesite verificar el podrá buscarlo o ingresar su código y proceder a la búsqueda.
	Una vez verificados los artículos que sean necesarios se sale de la aplicación.
Escenarios Alternativos	El Bodeguero no podrá realizar ninguna operación si no ingresa al sistema.

Tabla 11.- Verificación de Inventarios. Realizado por autores.

4.2.3 Casos de uso en modo conectado

Los casos de uso que reflejan las operaciones en modo conectado se muestran en la Figura 18 y se detallan a continuación:

- Obtener los Compromisos de Compra
- Obtener los Pedidos del Compromiso.
- Autentificar y Autorizar a un usuario.
- ConFigurar las preferencias de un usuario.
- Actualizar los datos de un Compromiso de Compra así como de los bienes que este posea.

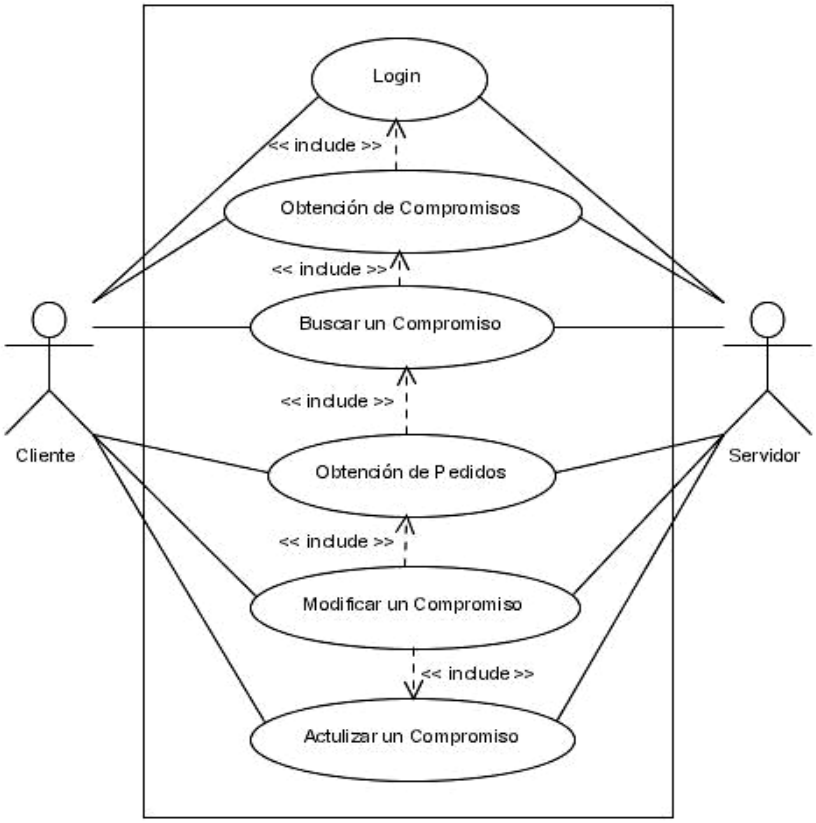


Figura 18.- Casos de Uso para las interacciones entre el cliente y el servidor en modo conectado. Realizado por autores.

Las operaciones en modo conectado se dan cuando el dispositivo móvil puede acceder a los servicios disponibles que están en la aplicación del lado del servidor

4.2.4 Casos de uso en modo desconectado

Soportar el modo de operación desconectado se da por dos razones: la calidad de la red puede ser muy pobre (comparada con redes cableadas) y el paradigma de los dispositivos móviles inalámbricos lo justifica. Esto hace que la combinación de estos factores haga atractivo el uso de operaciones en modo desconectado.

La pobre calidad del servicio de conexión puede darse por uno o más de las siguientes razones:

- Alta latencia de la red.- el tiempo extra necesario para que un paquete vaya de un punto a otro, debido al procesamiento de enrutamiento.
- Ancho de banda limitado.- la limitación en el ancho de banda restringe las tasas de transferencia y es un importante factor cuando se transmiten grandes cantidades de datos.
- Conectividad intermitente.- Los dispositivos inalámbricos generalmente no mantienen una

conexión dedicada a la red, pero se conectan solo intermitentemente debido a las imperfecciones de la red y la pobre calidad de la red (sobre todo cuando nos encontramos en sectores donde la red no puede ser alcanzada).

Es fácil ver que la mala calidad del servicio de red puede causar una experiencia desagradable y frustrante para un usuario que esta ejecutando una aplicación inalámbrica.

Cuando una aplicación esta en modo conectado los usuarios pueden obtener las listas completas de todos los compromisos y los pedidos por compromiso, además pueden buscar cualquier compromiso por su identificador. Si permanece conectado también puede actualizar un compromiso modificado en el servidor.

Los casos de uso para el modo desconectado son algo diferente de los casos online. Mientras que en modo desconectado, los usuarios pueden buscar y modificar localmente compromisos y pedidos directamente además de autentificarse localmente, estas operaciones son hechas con rapidez sin los retardos de la conexión.

Cuando se selecciona un Compromiso en particular la aplicación se dirige al servidor para obtener los pedidos de ese compromiso, modificarlo y posteriormente acuatizarlo.

Mientras los usuarios saben cuando están conectados o desconectados ellos experimentan lo mismo en ambos casos. Para acelerar el rendimiento de la Aplicación, se implementa un almacenamiento temporal los siguientes objetos:

- El compromiso que actualmente esta revisando
- Los pedidos de un compromiso que se estén revisando

El Anexo 1.9 muestra las interacciones entre el cliente y el servidor en modo desconectado.

4.3 Análisis y Prototipo de Interfaz de Usuario

La interfaz que se presenta al usuario es parte esencial en el desarrollo del SCI.

4.3.1 Diseño del prototipo

El diseño de la aplicación cliente y la apariencia que toma está dado por la implementación de MIDP de J2ME en cada dispositivo. MIDP sólo provee una manera

estándar para dibujar las pantallas, es la implementación de la máquina virtual la que brinda la apariencia final a las aplicaciones.

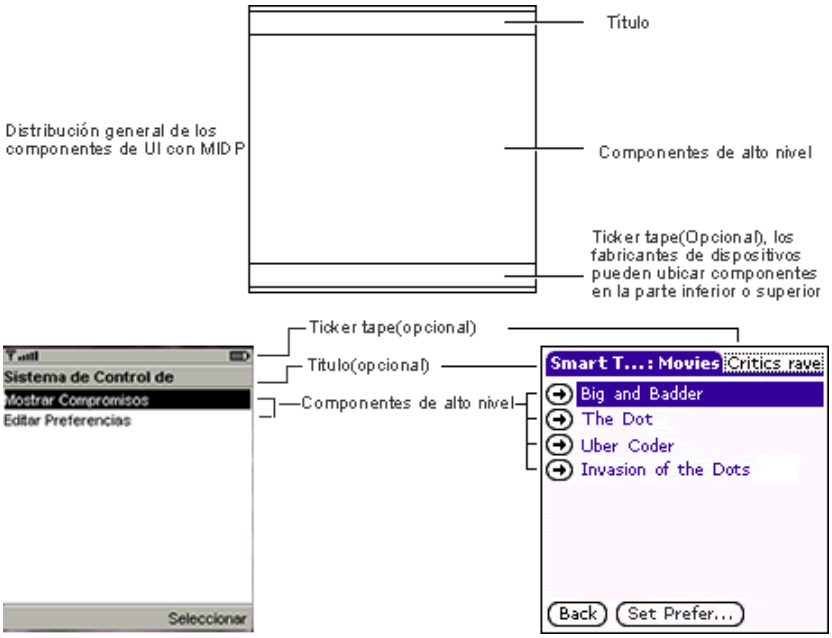


Figura 19.- Distribución general de los componentes de las interfaces de usuario con MIDP, [33]

En general, las aplicaciones que se diseñan con MIDP, tienen características similares en cuanto a la distribución de los elementos en la pantalla de los dispositivos, como lo muestra la Figura 19.

De igual manera, la interacción de los usuarios con cada interfaz depende de las restricciones y limitaciones de los dispositivos. Si el dispositivo se ejecuta sobre un teléfono celular que dispone de una pantalla táctil, el usuario debe

navegar con los botones, esto limita la cantidad de información que se puede poner en cada pantalla y el número de interacciones de que se disponene, como se muestra en la Figura 20.

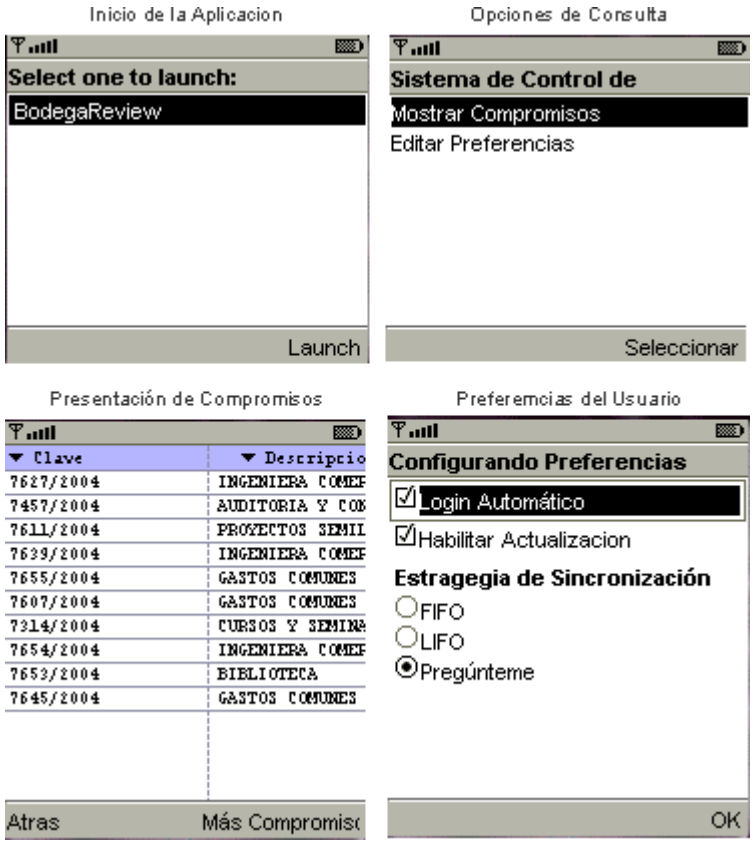


Figura 20.- Visualización de Pantallas en un teléfono celular, que soporte J2ME. Realizado por autores.

En la Figura 21 se muestra la aplicación pero visualizadas en un emulador del Sistema Operativo Palm OS 5.2. Entre las ventajas de usar dispostivos especializados para aplicaciones de este tipo, encontramos que la

implementación de MIDP en dispositivos Palm es mucho más amigable que en telefonos celulares, esto en gran medida gracias al sistema operativo en sí que permite una más amplia variedad de componentes gráficos en la pantalla que el sistema operativo de los teléfonos celulares.

Figura 21.- Visualización de Pantallas en una Palm, que soporte J2ME. Realizado por los autores

Un seguimiento de pantallas puede ayudar a entender mejor la interacción entre el usuario y la aplicación, con

esto se lograr captar la mayoría de situaciones críticas que encontrará el usuario y tratar de darle una mayor facilidad de uso en cada pantalla, así como proveer la ayuda en caso de que la tarea a realizarse así lo requiera.

4.3.2 Flujo de las Interfaces

En la Figura 22 se detalla el flujo de interfaces de la aplicación cliente.

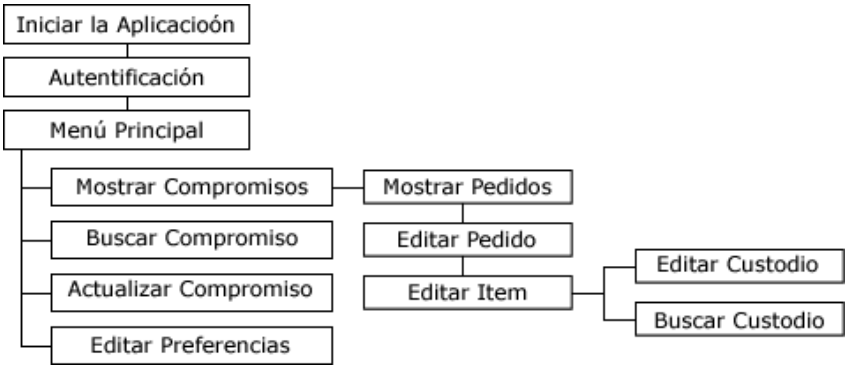


Figura 22.- Flujo de interfaces de la aplicación cliente, realizado por los autores.

Este flujo de interfaces tiene una profundidad de 6 niveles, que se detallan a continuación:

- Nivel 1.- En este nivel se encuentran las interfaces de inicio de la aplicación, como son los de la autenticación y la del menú principal

- Nivel 2.- Están las interfaces que corresponden a la edición de preferencia y a las 3 acciones principales: mostrar, buscar y actualizar compromisos.
- Nivel 3.- corresponde a la interfaz que muestra los pedidos por compromiso.
- Nivel 4.- corresponde a la interfaz que permite editar los pedidos.
- Nivel 5.- corresponde a la interfaz que permite editar los ítems del pedido
- Nivel 6.- corresponde a las interfaces que permite editar o buscar custodios del ítem.

Para este tipo de aplicaciones no se recomienda mantener muchas interfaces en memoria debido a las limitaciones antes mencionadas.

CAPÍTULO 5

5. DISEÑO E IMPLEMENTACIÓN

Este capítulo detalla varias etapas de diseño hasta llegar a las definiciones de los diagramas de clase. También, se detallarán algunas de las decisiones de diseño del SCI que tienen que ver con la arquitectura y distribución del sistema, los métodos para transferir objetos a través de la red mediante el protocolo HTTP, los mecanismos de persistencia tanto en el lado del cliente (dispositivo móvil) como en el lado del servidor (bases de datos).

Otro aspecto que se expone en este capítulo son las estrategias de diseño para la aplicación cliente. Estrategias diferentes a las tradicionales debido a las restricciones y limitaciones que se encuentran en el entorno de desarrollo para dispositivos móviles como: PDA, teléfonos celulares y otros.

Finalmente, se explican algunos de los diagramas de clases y diagramas de secuencia de diseño más relevantes.

5.1 Diseño de Distribución

El diseño de la distribución en el SCI, se basa en los patrones de diseño J2EE (http://java.sun.com/blueprints/corej2eepatterns/), los cuales ofrecen soluciones bien probadas para resolver problemas en el diseño de aplicaciones, también se basa en los planos de aplicaciones (http://java.sun.com/blueprints), que ofrece modelos completos, de aplicaciones ejemplo desarrollados con Tecnologías Java.

5.1.1 Arquitectura de la Aplicación de Servidor

La arquitectura de la aplicación del servidor está basada en una arquitectura n-capas como lo muestra la Figura 1, esto debido al uso del modelo de aplicaciones que ofrece J2EE, las capas de la aplicación de servidor son:

- Capa 0: Presentación Cliente.- Se ejecuta en un dispositivo móvil, procesa las entradas del usuario y almacena datos localmente, sobre los ingresos de inventarios.
- Capa 1: Presentación del servidor.- esta capa representa las interfaces a los servicios a los cuales los clientes tienen acceso. Esta capa es

implementada mediante la tecnología Servlet de Java, y el servidor Web que proporciona el servidor de aplicaciones.

- Capa 2: Lógica del negocio.- esta capa representa el modelo de la aplicación, con clases que abstraen los conceptos como: compromisos de compra, activos fijos, pedidos y las operaciones del proceso de ingreso de inventarios. Los cuales se implementan mediante la especificación J2EE Enterprise Java Beans (EJB) 2.0.

- Capa 3: Lógica de los datos.- esta capa representa los subsistemas externos que manejan la persistencia de los datos, esta capa es manejada por el *framework* de administración de persistencia que de la especificación J2EE y que es proporcionada por el servidor de aplicaciones. El servidor de aplicaciones utiliza implementa las especificaciones Container-Manager Persistence (CMP) de J2EE, en la cual el contenedor administra y maneja la persistencia de los objetos.

5.1.2 Arquitectura de la Aplicación Cliente

La aplicación cliente del SCI se la puede definir como un cliente denso, debido a que el modelo implementa dos modos de operación. El hecho de soportar el modo de operación desconectado implica diseñar un modelo que se adapte a cubrir este requerimiento, por lo cual, la arquitectura de la aplicación cliente se basa en el patrón de diseño Modelo, Vista y Controlador (MVC) [26] como se muestra en la Figura 23, y ayuda a desacoplar las tareas. En la aplicación cliente se puede diferenciar 4 capas:

- Capa 0 Presentación del Cliente.- esta capa representa los interfaces de usuario entre el dispositivo móvil y el usuario
- Capa 1 Control del Cliente.- esta capa representa la lógica de la aplicación y administra los requerimientos del usuario.
- Capa 2.- Modelo la Aplicación.- en esta capa es la representa las reglas de la lógica del negocio, también se agrupan aquí las clases que implementan algunos patrones de diseño indispensables para

implementar los modos de operación conectado y desconectado

- Capa 3 Lógica de datos, en esta capa se agrupan las clases que administran los recursos de persistencia de la aplicación cliente, así como de las clases necesarias para realizar las conexiones HTTP.

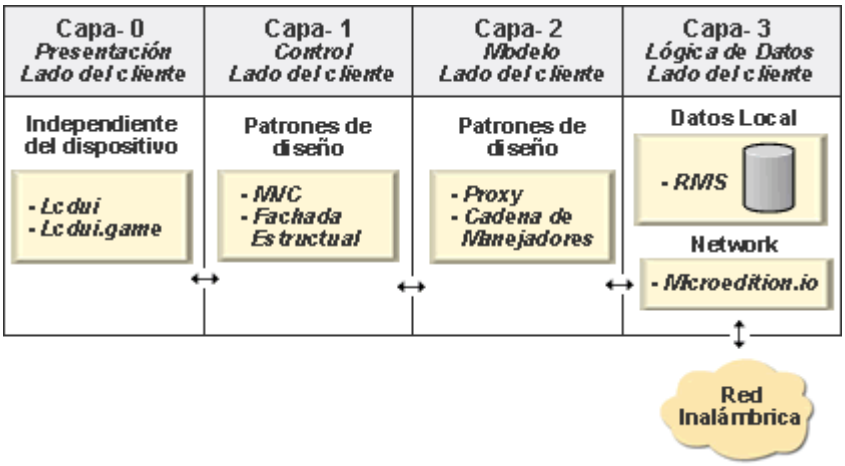


Figura 23.- Arquitectura de la Aplicación Cliente

De esta manera, se reparte las tareas de implementar el modo desconectado y se las separa de la capa de presentación.

5.1.3 Modelo de Datos Transferible

Al modelo de datos común entre la aplicación cliente y la aplicación del servidor se la denomina modelo transferible, la cual es un conjunto serializable de clases

que implementan el patrón de diseño ValueObject [39] y TransferObject [40]. Las clases del modelo transferible son:

- JCompromiso.- corresponde al EJB Compromiso_Compra en la aplicación del servidor.
- JPedidoBien.- corresponde al EJB Pedido_Bien en la aplicación del servidor.
- JItem.- corresponde al EJB Activo_Fijo en la aplicación del servidor.
- JAccount.- corresponde a la información del usuario del directorio activo de la ESPOL.
- JBlob.- Es un objeto transferible de tipo general, puede contener imágenes, texto entre otros datos.

Este grupo de clases mantiene un conjunto relacionado de atributos que forman un valor compuesto. Como lo muestra la Figura 24.

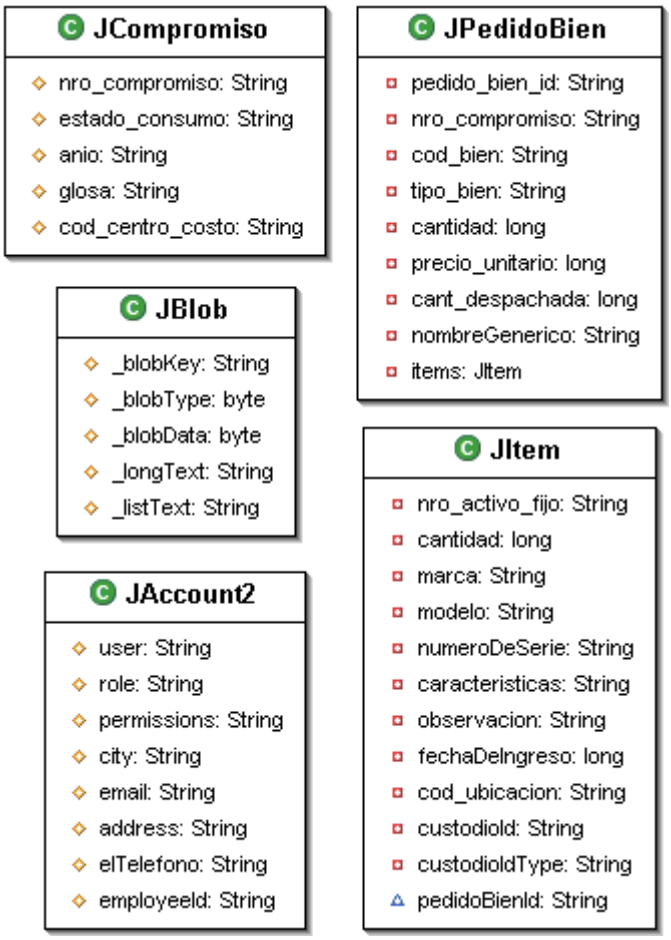


Figura 24.- Modelo transferable entre la aplicación cliente y la aplicación del servidor.

Estas clases son usadas como tipo de retorno de los métodos de negocios remotos de la aplicación del servidor como obtener compromisos, obtener pedidos, etc.

La aplicación cliente recibe las instancias de estas clases, invocando los métodos de negocios de la aplicación del servidor y localmente obtiene los valores de los atributos de estos objetos transferibles. Esta estrategia disminuye

el tráfico en la red y minimiza la latencia producida por el uso de los recursos del servidor.

5.1.4 Modelo Remoto

Para que puedan coexistir la aplicación cliente y la aplicación del servidor debe haber un modelo común de datos y operaciones entre las dos aplicaciones. Esto se logra mediante la implementación de una interfaz que provea de las operaciones que la aplicación del servidor brinda como lo muestra la Figura 25. Esta interfaz que brinda el modelo remoto para la aplicación cliente se denomina RemoteModel.

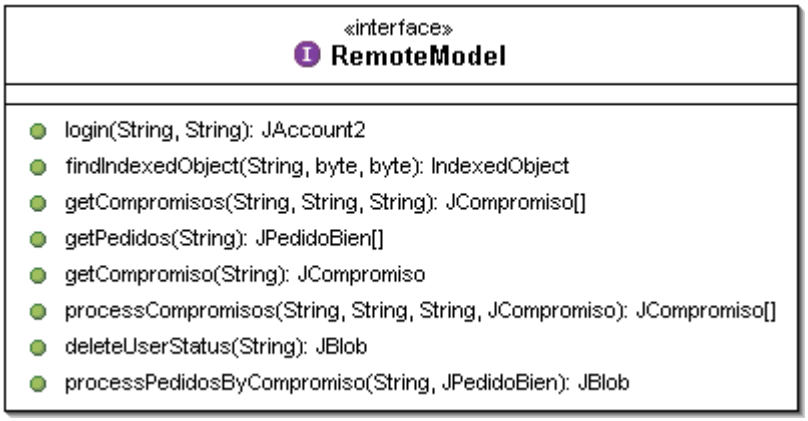


Figura 25.- Modelo Remoto de la aplicación de servidor del SCI

La interfaz RemoteModel utiliza sólo clases del modelo transferible de datos y clases comunes a los API de J2EE Y J2ME.

5.2 Decisiones de Diseño

Esta sección esta dedicada a la explicación de las decisiones que fueron tomadas para diseñar la aplicación cliente y el servidor de aplicaciones antes de la implementación.

5.2.1 Estrategias aplicadas en el diseño de la aplicación del servidor.

La aplicación del servidor implementa el patrón de diseño Delegación de Negocios [38], la estrategia de Adaptador Delegado que es una variante de este patrón es la que se ajusta de mejor manera al SCI, al integrar los servicios que ofrece la plataforma J2EE y los requerimientos del dispositivo móvil al integrar una clase que funcione de adaptador, como lo muestra el Anexo 1.1.

Con esta implementación se resuelve el problema de, traducir los requerimientos de un flujo de datos a objetos, y aparte oculta las complejidades de buscar los objetos de negocios que manejen los procesos de ingreso a bodega.

El proceso de ingreso de inventarios envuelve el uso de algunos EJB y también envuelve otros procesos propios de hacer un ingreso, como acceder a la base de datos hacer verificaciones y obtener numeradores de registros

para hacer las actualizaciones, esto resulta en la perdida de la flexibilidad y claridad del modelo. Esto se lo puede resolver implementando el patrón de diseño Fachada de Sesión [41], el cual define componentes de negocios de alto nivel que resuelven complejas interacciones entre los EJBs, la base de datos. La Figura 26 muestra la implementación de las operación del modelo remoto: getCompromisos() y updateCompromiso() con el patrón de diseño Fachada de Sesión.

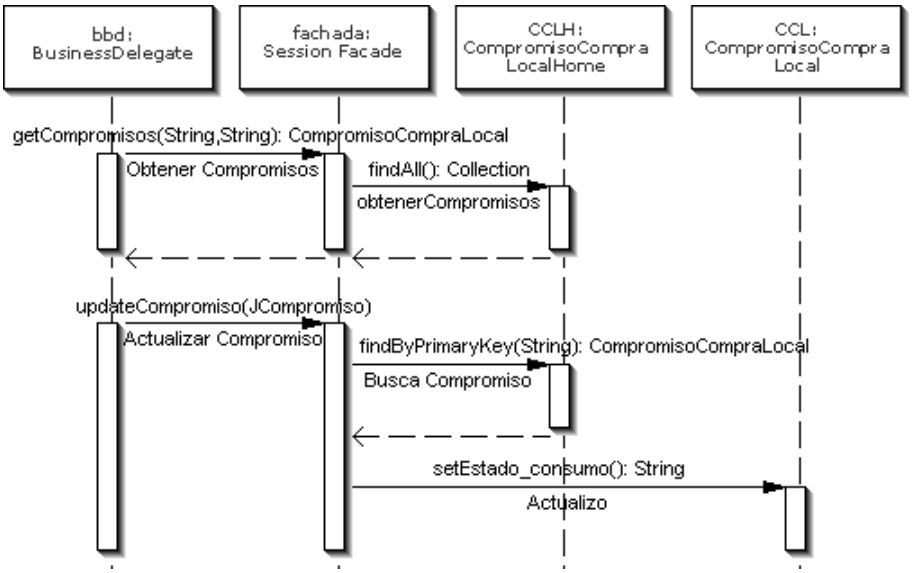


Figura 26.- Implementación del patrón de diseño Fachada de Sesión en la aplicación del servidor

La implementación del patrón de diseño Delegación de Negocios y el patrón Fachada de Sesión en la aplicación del servidor permite una disminución en la cantidad de

mensajes que se transmiten a través de la red y una notoria clarificación del modelo.

5.2.2 Estrategias aplicadas en el diseño de la aplicación cliente.

El diseño efectivo de la aplicación cliente del SCI envuelve estrategias para manejar las: interacciones entre el cliente y el servidor, almacenamiento temporal de datos y sincronización de entre los datos del cliente y el servidor entre otras. En esta sección serán analizadas las más relevantes.

Particionar el Procesamiento

Balancear el procesamiento entre el cliente y el servidor es fundamental para aplicaciones Cliente/Servidor. Para lograr la optimización de la carga de procesamiento se debe pasar todos los procesos fuertes al lado del servidor dejando del lado del cliente móvil, tareas propias de su ambiente

De esta manera se logra alcanzar los siguientes objetivos:

- Disminuir el tiempo que el dispositivo necesita estar conectado a la red, difiriendo el tiempo en pequeños

conexiones dependiendo de las necesidades del usuario.

- Mejorar el balance del procesamiento entre el cliente y el servidor, dándole a este último las tareas de actualización, verificación y procesamiento.

El diseño de la aplicación cliente del requiere un diseño más complejo que si sólo trabajara en modo conectado. De esta manera, el modo desconectado disminuye los requerimientos de conexión, pero añade requerimientos a memoria, almacenamiento y procesamiento extra.

La sincronización es el proceso en la aplicación cliente en donde el procesamiento y el uso de memoria son críticos, esto se debe a que se tienen que hacer verificaciones sobre la fecha de actualización y el estado de los objetos.

El costo en tiempo de procesamiento, acceso a la memoria de almacenamiento y uso de la memoria dinámica de este proceso de verificar fechas de actualización y estado del objeto puede llegar a ser muy alto, llegando a la inhibición de la aplicación o a dar errores por falta de memoria dinámica.

El particionamiento se da al identificar los procesos críticos y transferirlos para que se realicen en el servidor.

Esto se lo implementa haciendo que todos los objetos transferibles entre el cliente y el servidor implementen la Interfaz *Synchronizable* como lo muestra la Figura 24. De esta manera, se puede acceder a la fecha de modificación del objeto y al estado actual. Entonces únicamente se transfieren los objetos y se realiza las verificaciones en el servidor, una vez realizado el proceso se devuelven los mismos objetos con su nuevo estado, y en el cliente se verifica el estado y se toma la acción.

Los estados que pueden tomar los objetos son:

- Sin cambio (STATUS_UNCHANGED).- Es el estado inicial con el cual vienen los objetos desde la aplicación del servidor.
- Insertado (STATUS_INSERTED).- El objeto ha sido insertado en la base de datos.
- Modificado (STATUS_MODIFIED).- El objeto ha sido modificado en la aplicación cliente o servidor.
- Borrado (STATUS_DELETED).- el objeto ha sido borrado del servidor.

- Completado (STATUS_COMPLETED).- Los datos del objeto ha sido totalmente ingresados.

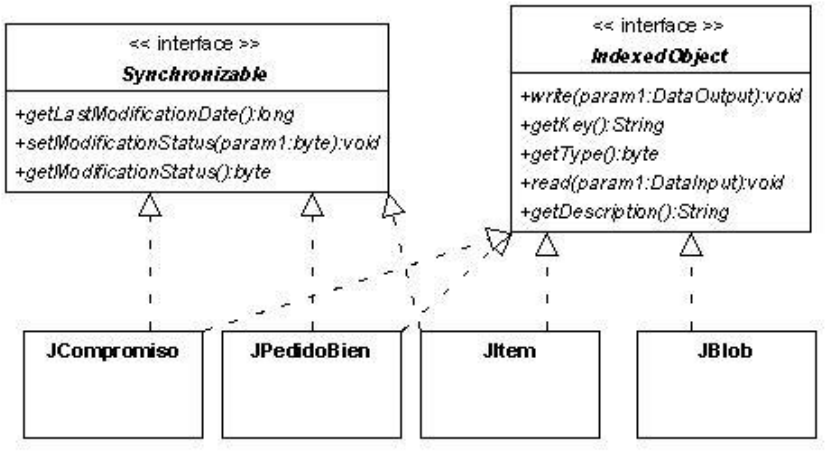


Figura 27.- Implementación de la estrategia de sincronización e indexación para los objetos que se distribuyen a través de la aplicación cliente y el servidor.

No todos los objetos que se transmiten desde el cliente al servidor necesitan ser sincronizados, un ejemplo es el objeto JBlob que solo implementa la interfaz *IndexedObject* la cual proporciona las operaciones necesarias para realizar los procesos de serialización usados en la transmisión HTTP y en la persistencia de objetos en el dispositivo móvil.

Procesamiento y envió de datos a través de la Red

Si no se tiene ninguna restricción a la cantidad de objetos que se recupera a través de la red vía HTTP y si el número de objetos que se transmiten es demasiado grande, eventualmente esto puede ocasionar una sobrecarga de memoria y procesamiento en la aplicación cliente.

Entonces, para disminuir el tamaño de los mensajes que se trasmite en la red vía HTTP, se implementó un mecanismo de envío por partes, esto es, se transmiten 10 objetos cada vez que se realiza un requerimiento de compromisos de compra en el servidor.

La administración de los objetos que se envían desde servidor hacia el cliente ofrece las siguientes ventajas:

- Reduce la latencia de la red.- al ser menos objetos por vez, disminuye el tiempo necesario para que viajen a través de la red.
- Reduce el procesamiento en el dispositivo móvil.- se reduce el procesamiento para convertir bytes en objetos.

Para implementar el envío de objetos por partes, se creó un EJB especializado para que maneje el estado del usuario y la información de los compromisos que este usuario tiene en el dispositivo móvil.

De esta manera al hacer un requerimiento de compromisos, se envía el identificador del usuario en cada requerimiento. Los procesos que se manejan con esta estrategia son:

- Manejo del estado del usuario.
- Almacenamiento temporal de los Compromisos que actualmente están en el dispositivo móvil.
- Verificación y proceso de selección de los nuevos 10 compromisos que serán transmitidos hacia el cliente.

Al mantener el estado del usuario en el servidor, se puede implementar un modelo multiusuario más visible. En la aplicación cliente si un usuario inicia la sesión y luego en el mismo dispositivo móvil otro usuario inicia sesión, el estado, las preferencias y los datos del usuario anterior se borra, tanto en el dispositivo móvil como en el servidor.

Modelo de Datos

Para soportar ambos modos de operación (Modo Conectado y Desconectado), la aplicación Cliente debe mantener su propio modelo local de datos, el cual es una parcial replicación del modelo de datos del servidor. Además, se debe desacoplar el modelo de datos de las interfaces de usuario y de las clases que controlan la aplicación, esto se logra utilizando el patrón de diseño MVC.

En la Figura 25 se puede identificar una de las consecuencias de usar el patrón de diseño MVC, que es separar la interfaces gráficas (UIs) del modelo y del controlador. Las UIs mantienen la referencia del objeto controlador que es UIController, el cual administra las UIs y los requerimientos del usuario.

Cada UI sabe como presentar los datos al usuario, cuando el usuario hace un requerimiento en el UI este es redireccionado hacia el UIController con un identificador de operación. La clase EventIds mantiene como referencias estáticas los identificadores de operación los cuales son obtenidos a través de UIController.

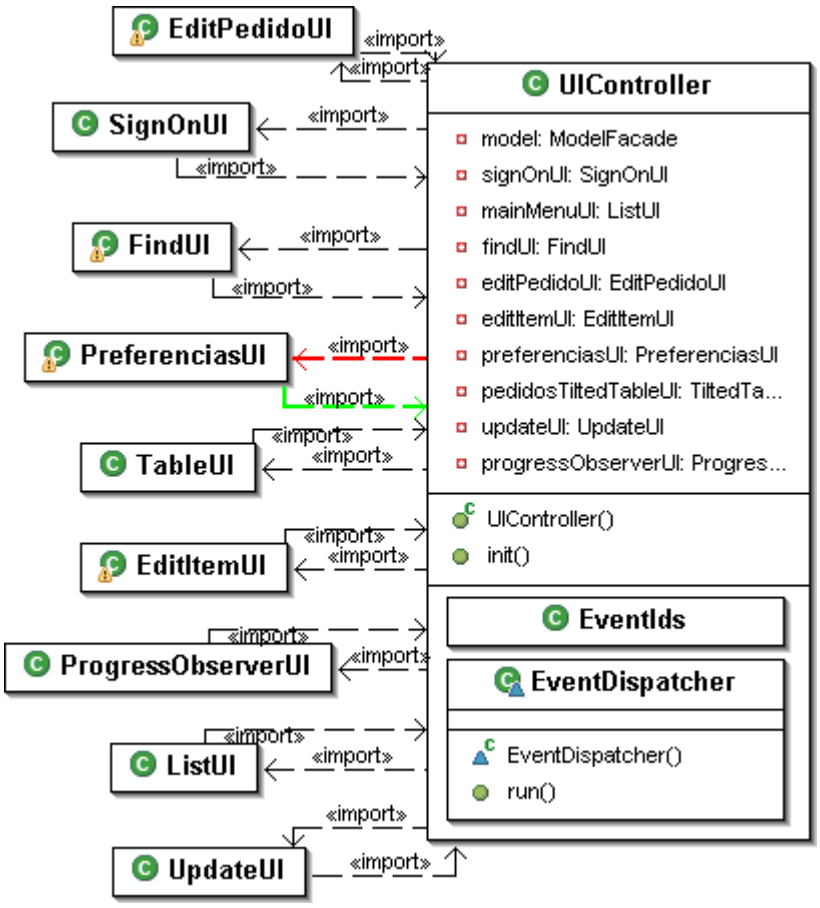


Figura 28.- Implementación de la relación entre las vistas y el controlador en la aplicación cliente del SCI

Entre UIController y el modelo de la aplicación se implementa una estrategia derivada del patrón de diseño de Fachada [37], para ocultar las complejidades de mantener un modelo local de datos y un modelo remoto de datos. La clase ModelFacade sirve de fachada y ofrece una interfaz simple para acceder a los datos, en la Figura 26 se muestra las clases involucradas en el modelo y del control de la aplicación cliente.

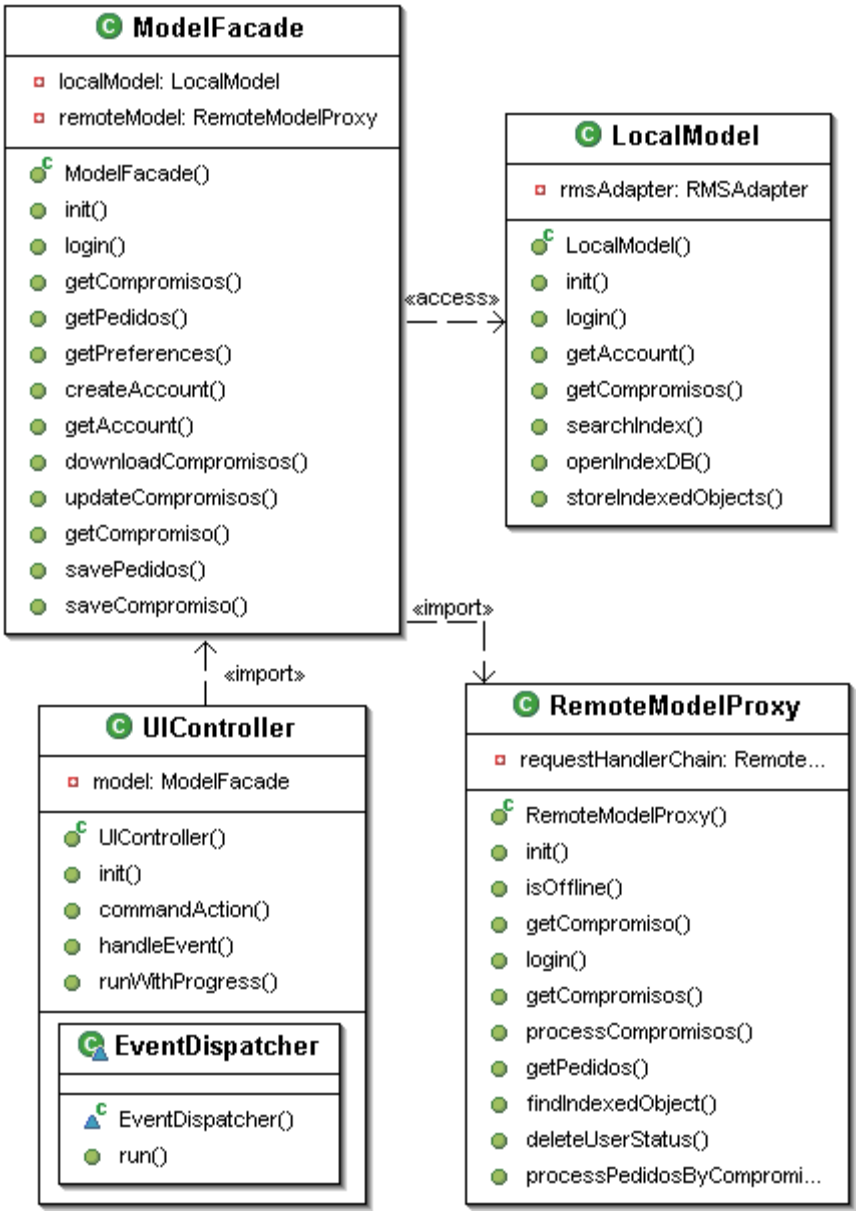


Figura 29.- Implementación de la relación entre el controlador y el modelo en la aplicación cliente.

Replicación de los Datos

El modelo de datos del cliente replica la parte más relevante del modelo de datos del servidor, mediante un

proceso de filtrado de datos. La Figura 24 los objetos que se transfieren desde el modelo de datos del servidor hacia el cliente son: JCompromiso, JPedido, JItem, JAccount y JBlob.

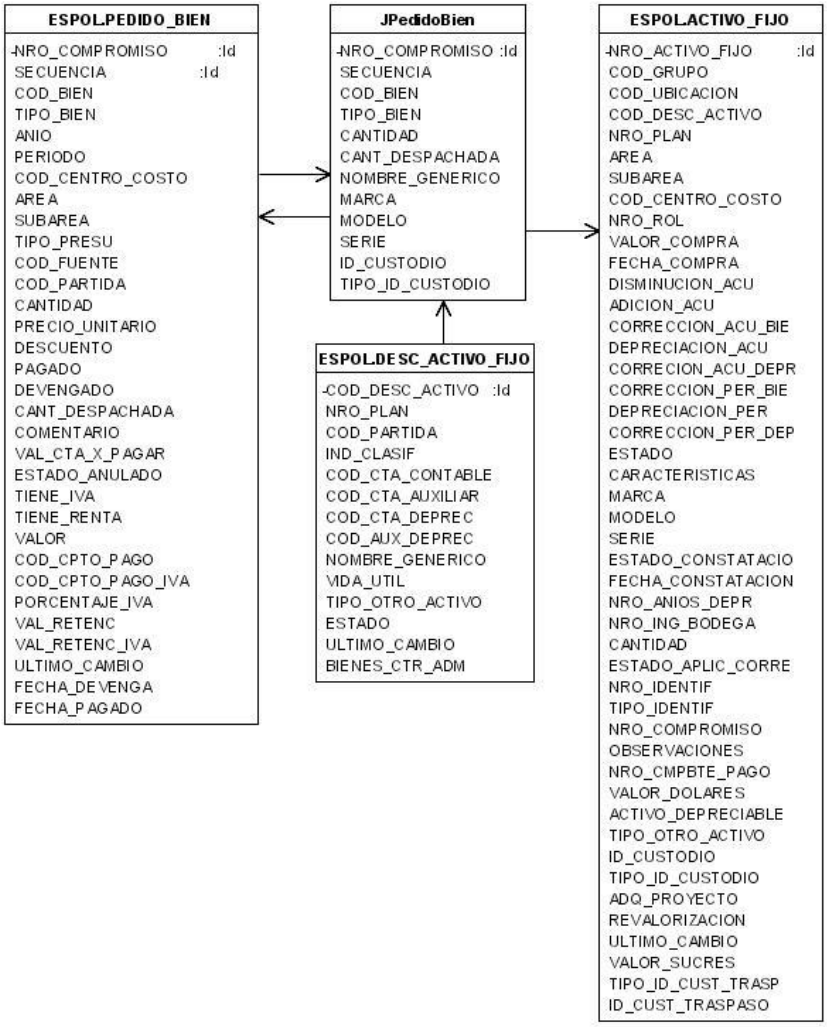


Figura 30.- Filtrado seleccionado del modelo de datos de los Sistemas de la ESPOL hacia el cliente del dispositivo móvil en el objeto transmisible JPedidoBien. Realizado por autores.

En la Figura 30 se aprecia este proceso sobre el objeto JPedido, el cual obtiene sus datos de las tablas ESPOL.PEDIDO_BIEN y ESPOL.DESC_ACTIVO_FIJO. Con estos datos llega a la aplicación cliente, después de ejecutar el proceso de la aplicación cliente, este objeto JPedidoBien regresa al servidor e interactúa con las tablas ESPOL.PEDIDO_BIEN y ESPOL.ACTIVO_FIJO.

Algunas consideraciones [26] que fueron tomadas en cuenta durante el análisis de la replicación de datos hacia la aplicación cliente fueron:

- La aplicación cliente no debe hacer una replicación total del modelo de datos del servidor debido al alto costo en procesamiento, memoria y uso de la red, que implica replicar el modelo de datos.
- Los objetos JCompromiso, JPedido, JItem se transmiten en un esquema de lectura/escritura y los objetos JAccount y JBlob son transmiten en esquema de sólo lectura.
- La modificación concurrente del objeto JCompromiso es permitida, debido a la implementación de un esquema multiusuario. Para evitar problemas de

integridad de datos, se maneja un esquema de syscronización por fecha de actualización, esto es, se actualiza la información si la fecha de modificación del objeto que quiero actualizar es menor a la fecha de modificación del objeto con el que se está actualizando.

- La clase JAccount, la cual mantiene la información del usuario, es sensible a fechas de expiración, teniendo un límite de 24 horas para ser verificado y actualizado en el dispositivo móvil, caso contrario no se aceptará las credenciales de autenticación del usuario como válidas.

Persistencia de Datos.

Para proveer las ventajas de mantener el modo desconectado el sistema requiere que los datos sean almacenados localmente en el dispositivo móvil.

Debido a que RMS almacena y recupera datos como arreglos de *bytes*, MIDP no soporta los estándares de Serialización[57] de Java. Los objetos que se necesitan almacenar localmente como los objetos transferibles

[57] Serialización.- Mecanismo de persistencia de datos de Java.

implementan la interfaz *IndexedObject* como se muestra en la Figura 24. Este método de serialización se lo utiliza para los siguientes dos procesos:

- Almacenamiento de los objetos en el dispositivo móvil.- de esta manera se almacena y recupera en forma binaria los datos en las bases de datos del dispositivo móvil.
- Transmisión de objetos en los mensajes HTTP.- el mismo método de serialización que se usa para almacenar los objetos en las bases locales, es usado para escribir los mensajes HTTP que se transmiten a través de la red.

Para implementar un mecanismo de persistencia que sea válido para los dos procesos arriba mencionados, los objetos que se transfieren a través de la red y los que se almacenan localmente deben implementar la interfaz IndexedObject como se muestra en la Figura 24, esta interfaz provee de las operaciones para identificar un objeto, escribirlo y leerlo desde un flujo de datos binarios.

Localmente, los objetos se almacenan en dos bases de datos como lo muestra la Figura 31

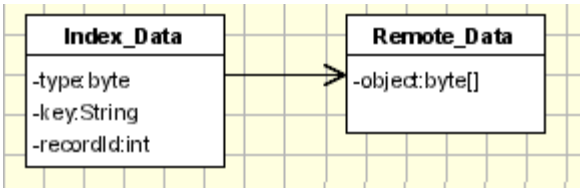


Figura 31.- Modelo de Base de Datos para el dispositivo inalámbrico móvil del cliente. Realizado por autores.

La base para datos *Remote_Data* mantiene los registros de los objetos y la base de datos (*Index_Data*) mantiene los índices, esta estrategia se implementa para optimizar las búsquedas de registros en el dispositivo móvil. Cada registro de *Index_Data* está compuesto por los siguientes datos:

- Primer byte.- el de objeto
- Segundo byte.- clave única del objeto
- Tercer byte.- número de registro en *Remote_Data*

De esta manera cada registro de *Index_Data* tiene una logitud de 3 bytes haciendo muy rápida la búsqueda de los datos.

5.3 Mejorando el Rendimiento

En la aplicación cliente se aplican estrategias para optimizar el código, que según las pruebas realizadas en la sección 6.2 Pruebas de Rendimiento, mejoran notablemente el rendimiento de

la aplicación, al bajar los límites de consumo de memoria y creación de objetos. Las estrategias utilizadas son:

- La aplicación cliente sólo usa arreglos de objetos para implementar asociaciones, agregaciones y demás relaciones. Un ejemplo de esta estrategia es la implementación de la agregación en el modelo de datos transferibles. Entre las clases JPedidoBien y JItem existe una agregación de 1 a muchos, la clase JPedidoBien contiene un arreglo de JItem[] y cada JItem contiene la clave de su JPedidoBien. Esta estrategia elimina el uso de clases especializadas para implementar agregaciones como son: las Colecciones, los Vectores, etc. que ofrecen características importantes, pero su costo en tiempo de procesamiento y memoria es muy alto.
- Se realizan recolecciones de basura, mediante la llamada System.gc(), en las secciones críticas del código. Sin estas recolecciones de basura la aplicación cliente puede llegar a los límites de memoria dinámica y hasta a inhibirse. En la clase UIController se identificaron 14 secciones de código en las cuales se evidenciaron picos de uso de memoria y creación de objetos como lo muestra la Figura 35. Disminuyendo un 44,44% en el pico máximo de uso de memoria y un 40% en el pico máximo de creación de objetos.

- Se mantiene un esquema de almacenamiento temporal en la clase RMSCacheHandler, esto evita costosos accesos a las bases locales del dispositivo móvil y a descargas de información redundante de la red. Por ejemplo:. Si es usuario pide editar un compromiso, la aplicación cliente primero verifica si el compromiso todavía es válido haciendo un requerimiento a la aplicación del servidor de ese compromiso, si es válido hará la petición de los pedidos del compromiso, caso contrario lanzará un excepción de error. Durante la misma sesión. Si durante la misma sesión el usuario pide editar el mismo compromiso la aplicación no volverá ha hacer el requerimiento del compromiso, pues éste ya se encuentra en RMSCacheHandler, evitando así un costoso requerimiento a través de la red.

- Para acelerar la aplicación el controlador (UIController), crea todas las instancias del modelo y de las vista al inicio de la aplicación. Así, la aflicción puede tardar un poco en cargarse, pero disminuye los tiempos de respuesta cuando se la está usando.

- Para manejar adecuadamente el flujo de interfaces cuando se está usando la aplicación cliente, en UIController se implementa un pila, la cual va a contener las instancias de

las interfaces que actualmente se están usando, la Figura 22 muestra la profundidad que puede tener la pila durante la realización de determinado proceso. Por ejemplo, si queremos editar un JItem, primero debemos acceder a ver la lista de los JCompromisos, se añade la instancia de la interfaz que muestra la lista de los compromisos (TableUI), luego se muestra la interfaz con la lista de JPedidoBien, se añade la instancia de la interfaz que muestra la lista de los pedidos (TiltedTableUI), se muestra la interfaz de resumen de los ítems que tiene cada pedido, se añade la instancia de la interfaz que mantiene el resumen de los ítems (EditPedidoUI), y finalmente se añada la instancia de la interfaz de edición de de un pedido (EditItemUI). Si se necesita navegar por la aplicación todas las interfaces mantiene un botón de retroceder, que ejecuta la misma acción, la cual es ejecutar la operación “peek” sobre la pila y mostrar el objeto que se encuentra en la cima de la pila.

5.4 Diagramas de Diseño

En estos diagramas se puede apreciar como esta estructurado el sistema y las interacciones estáticas que se dan entre las clases.

5.4.1 Descripción de las clases de la aplicación en el servidor

En la aplicación que se ejecuta el servidor se tienen algunos objetos específicos que tienen sus tareas claves para el sistema, como se detalla en la siguiente lista:

- CompromisoCompra.- componente que abstrae los compromisos de Compra.
- PedidoBien.- componente que abstrae un pedido de bien, de un compromiso especificado.
- ActivoFijo.- componente que abstrae un activo fijo.
- UserStatus.- componente que abstrae el estatus actual de los clientes móviles del sistema.
- Fachada de Sesión.- Implementa el Patrón de Diseño "*Session Facade*".

Para mayor detalle ver los Anexos 1.7 y 1.8 Diagrama de Clases del Sistema de Control de Inventarios.

Un CompromisoCompra por ejemplo, contiene a su vez al componente Pedido_Bien, y a este objeto se accede por el objeto controlador que implementa el Patrón de Diseño "Session Facade", este objeto lo único que conoce es un CompromisoCompra y los pedidos de compromiso son

obtenidos a través del compromiso y no directamente del componente Pedido_Bien.

Como se puede apreciar en el Anexo 1.8, el modelo de la Aplicación no contempla el diseño de la persistencia ni el de los otros recursos de los cuales el sistema se sirve, estos están ocultos para los objetos externos. En el caso particular de los componentes: CompromisoCompra, Pedido_Bien, Activo_Fijo, que son los que representan datos, su recurso de persistencia esta asociado a la base de datos del sistema Financiero de la ESPOL (SAF) y el recurso de persistencia del Componente UserStatus esta asociado a una base de datos local implementada en Mysql.

Internamente, los objetos de negocios tienen métodos para grabarse y recuperarse desde una base de datos. Información acerca de cómo esto es posible esta disponible fuera del entorno de los objetos, y no afecta a otros a otros.

Además, los objetos de negocios proveen interfaces para que los objetos externos e internos accedan a ellos. En una aplicación de 2 capas, la Interfaz de usuario, la información sobre un compromiso de compra es

directamente mostrada desde la base de datos. Sin embargo, en la aplicación de 3 capas la Interfase de Usuario aprende todo lo que necesita acerca de un compromiso de compra desde los objetos de negocio CompromisoCompra, lo mismo sucede si la Interfase de usuario tiene que cambiar el compromiso de compra, esta hace los cambios sobre CompromisoCompra en lugar de hacerlo sobre la base de datos.

5.4.2 Diagramas de clases

En esta sección se presentaran varios diagramas que explican el diseño de los aspectos más relevantes del cliente. Además, en los Anexos 1.7 y 1.8 contienen los diagramas de clases del SCI.

Paquete com.espol.bodega.comun.model

La implementación del SCI requiere que en las dos partes de la aplicación: cliente y servidor, se encuentren definidas las clases que hacen posible la transferencia de la información.

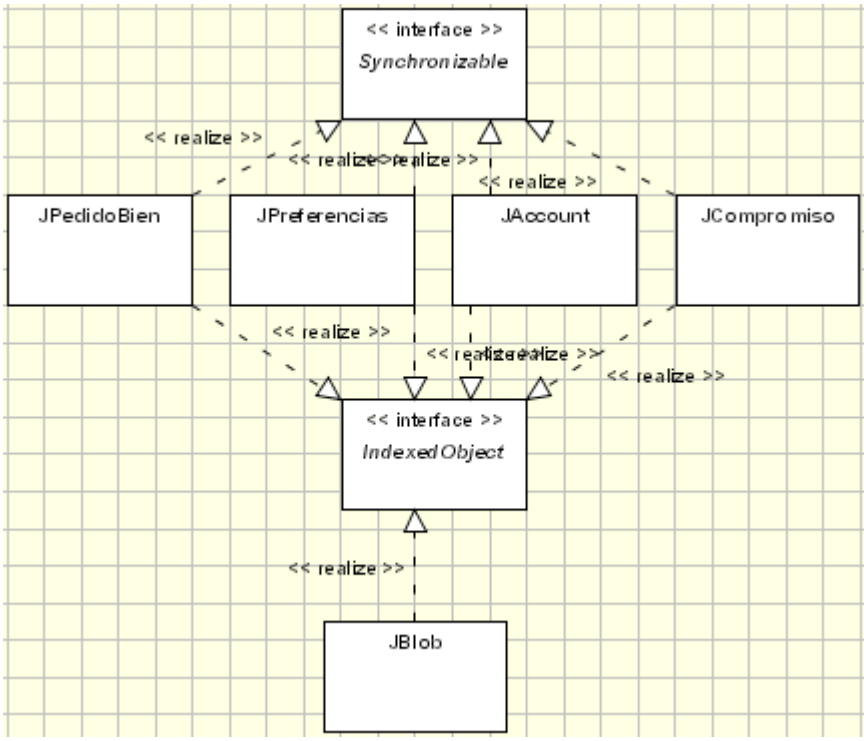


Figura 32.- Diagrama de clases del paquete com.espol.bodega.comun.model.. Realizado por autores.

Este paquete contiene las clases que se utilizan en el servidor y cliente, entre las clases que se encuentran las clases que se implementan para abstraer los datos como son: *JCompromiso*, *JPedidoBien* y *JAccount*. Que implementan la Interfase *IndexedObject*, que les posibilita a ser transferibles y almacenables. Como lo muestra la Figura 29.

Paquete com.espol.bodega.midpCliente.model

Este paquete del Cliente contiene parte del Diagrama de la aplicación, se diferencia claramente la aplicación del Patrón de diseño de ModelProxy, RequestHandlerChain, además se aprecia muy claramente la interacción de las clases del cliente. Como lo indica la Figura 30.

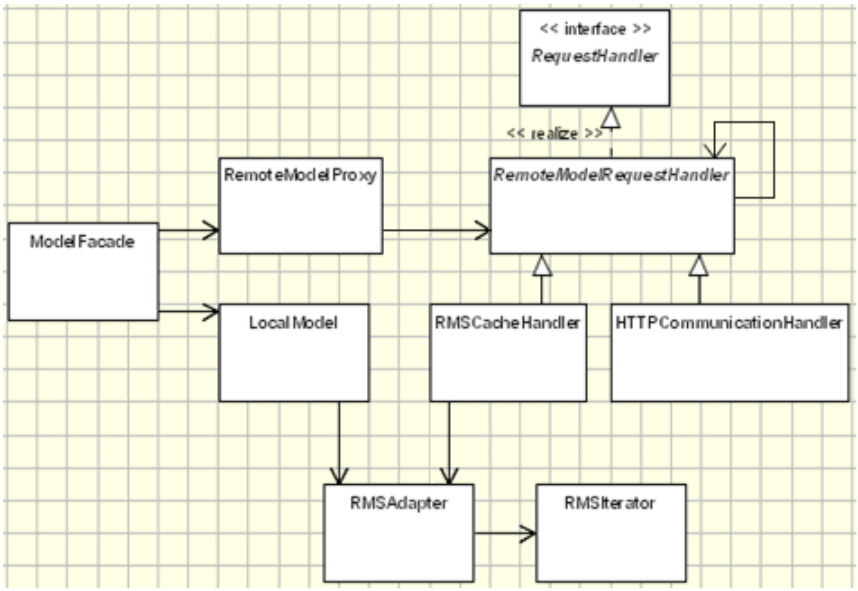


Figura 33.- Diagrama de clases del paquete com.espol.bodega.midpClient.model. Realizado por autores.

5.4.3 Diagramas de secuencia

En esta sección se detallan los diagramas de secuencia más relevantes del SCI.

Recuperación de compromisos (caché), lado del cliente

Para esta secuencia se asume que el usuario ya ha ingresado en la aplicación y posee todos los permisos para trabajar en esta. Para mayor detalle ver Anexo 1.2.-DIO.- Recuperación de compromisos de la memoria temporal.

Se asume además que no se presentan problemas con la red ni con el dispositivo (baterías bajas u otros)

La siguiente secuencia de pasos ocurre cuando el usuario recupera los compromisos de compra y se da por hecho de que existen registros en la memoria caché de la Palm:

1. El usuario elige la opción de mostrar los compromisos de compra.
2. El evento es manejado por el objeto UIController mediante el evento handelEvent. en donde se identifica la operación que el usuario desea se ejecute.
3. Se inicia el proceso de recuperación de los registros con el método runWithProgress

4. En este método(runWithProgress) se crea el hilo de la clase EventDispatcher en la cual ya esta identificada la operación que se debe realizar
5. Desde UIControler se invoca el método run() para que se inicie el hilo del objeto EventDispatcher.
6. En EventDispatcher se identifica la operación que debe realizarse.
7. EventDispatcher invoca al método areThereCachedCompromisos del objeto ModelFacade.
8. El objeto ModelFacade pide la ejecución del método areThereCachedCompromisos al objeto LocalModel.
9. El objeto LocalModel ejecuta el método getNextCompromiso.
10. En el método getNextCompromiso se invoca al método getIterator de RMSAdapter
11. En el método getIterator se procede a llamar al método openPageIterator del objeto RMSIterator, openPageIterator carga el número de registros que se existen en la cache.

12. El objeto RMSIterator retorna un RMSI al objeto RMSAdapter con lo cual se ha cargado la cache que existe.
13. El objeto RMSAdapter pide se recupere el siguiente registro mediante el método getNext del objeto RMIterator
14. RMSIterator devuelve el siguiente registro al objeto RMSAdapter y luego se procede a pedir el getRecordId de este registro al objeto IndexEntry
15. El objeto IndexEntry retorna el id que se solicitó mediante getRecordId.
16. Con las operaciones realizadas se devuelve al objeto LocalModel el siguiente Compromiso.
17. Luego se retorna una respuesta al UIControler sobre la existencia de compromisos en cache, este valor es verdad ya que se asume que existen compromisos en cache.
18. El objeto UIControler invoca al método setPager del objeto ModelFacade.
19. El objeto ModelFacade hace la invocación del método setPager del objeto LocalModel.

20. Luego se invoca al método closePageIterator del objeto RMSIterator donde se cierra la operación realizada por openPageIterator en el paso 11.
21. Luego se invoca al método getIterator del objeto RMSAdapter.
22. El objeto RMSAdapter invoca al método openPageIterator del método RMSIterator y se realiza la operación antes mencionada en el paso 11.
23. El objeto RMSIterator retorna el conjunto de registros sobre los cuales se va a trabajar al objeto LocalModel.
24. El objeto EventDispatcher invoca al método nextPage del objeto ModelFacade.
25. El objeto ModelFacade invoca al método getNextPage del objeto LocalModel.
26. El objeto LocalModel invoca al método nextPage del RMSIterator, este método retorna los un arreglo de los Id’s de los registros que se encuentran en cache.

27. Con el arreglo con los id’s de los registros el objeto LocalModel invoca al método loadIndexedObjects del RMSIterator el cual recibe como parámetro los ID’s recuperados y este método se encarga de formar un arreglo de objetos el cual contiene los registros que corresponden a los Id’s enviados.
28. Estos objetos se devuelven al EventDispatcher que fue el objeto que originalmente los solicito.
29. Estos Objetos son enviados al objeto TableUI el mediante la invocación del método setData por parte del objeto EventDispatcher.
30. Por ultimo el objeto EventDispatcher invoca el método SetCurrentStack el cual se encarga de presentar los datos que se colocaron al TableUI para la presentación al Usuario.

Recuperación de Compromisos (en línea), lado del cliente

Para esta secuencia se asume que el usuario ya ha ingresado en la aplicación y posee todos los permisos para trabajar en esta.

Se asume además que no se presentan problemas con la red ni con el dispositivo (baterías bajas u otros)

.

La siguiente secuencia de pasos ocurre cuando el usuario recupera los compromisos de compra y se muestra en el Anexo 1.3, DIO.- Recuperación de compromisos de en línea (Cliente). Realizado por autores, esta operación se realiza cuando el usuario escoge obtener los compromisos y no existen compromisos en la cache o cuando el usuario pide obtener estos compromisos directamente de la base de datos.

1. El Usuario ha elegido la opción de descargar los compromisos en línea (o se precede a la invocación de esta operación debido a que no existen elementos en cache)
2. El evento es manejado por el objeto UIController mediante el evento handelEvent. en donde se identifica la operación que el usuario desea se ejecute.
3. Se inicia el proceso de recuperación de los registros con el método runWithProgress

4. En este método(runWithProgress) se invoca al método init del objeto ProgressObserver este método algunos parámetros que se usaran luego en la presentación de una barra de progreso.
5. En este método(runWithProgress) se da inicio al hilo de la clase EventDispatcher en la cual ya esta identificada la operación que se debe realizar
6. Desde UIControler se invoca el método run() para que se inicie el hilo del objeto EventDispatcher.
7. En EventDispatcher se identifica la operación que debe realizarse.
8. El objeto EventDispatcher invoca al método getCompromisos del objeto ModelFacade.
9. El objeto ModelFacade invoca al método getCompromisos del objeto RemoteModelProxy
10. El objeto RemoteModelProxy invoca al método getCompromisos del objeto RMSCacheHandler.
11. El objeto RMSCacheHandler invoca al método getCompromisos de objeto HTTPCommunicationHandler

12. El método getComrpmisos del objeto HTTPCommunicationHandler se encarga de enviar un flujo de datos el cual contiene el tipo de operación que el usuario ha solicitado (en este caso la recuperación de compromisos de la base de datos), y otros parámetros más necesarios para esta operación.
13. Este flujo de datos es procesado lo recibe el servidor mediante el servlet BodegaProcessRequestServlet(Este proceso se explicara en otro diagrama ya que abarca varios pasos)
14. El servidor retorna al objeto HTTPCommunicationHandler retorna un flujo de datos el cual contiene los compromisos que se han recuperado de acuerdo a los parámetros que se han enviado.
15. El objeto HTTPCommunicationHandler forma un arreglo de compromisos para proceder a retornarlo al objeto RMSCacheHandler.

16. El objeto RMSCacheHandler con los compromisos recibidos procede a comparar-actualizar la cache con los compromisos obtenidos.
17. El objeto RMSCacheHandler envía un arreglo de compromisos al objeto EventDispatcher
18. En el objeto EventDispatcher estos compromisos son enviados al objeto TableUI mediante el objeto setData para proceder a presentarlos al usuario.
19. El objeto EventDispatcher procede a presentar los compromisos al usuario para esto invoca al método SetCurrentStack de objeto UIControler.

Recuperación de Compromisos (en línea), lado del servidor

Anteriormente se explico la recuperación de compromisos online pero se obvio los procesos que realiza el servidor para retornar estos ya que involucra varios pasos y se explica en este diagrama. Para mayor detalle ver Anexo 1.4.- DIO.- Recuperación de compromisos de en línea (Servidor). Realizado por autores.

Como se explico en el anterior diagrama el objeto HTTPCommunicationHandler se encarga de enviar un flujo

de datos al servidor, en este flujo de datos consta el tipo de operación que se va a realizar.

1. Este flujo es recibido por el servidor mediante el servlet BodegaProcessRequestServlet, el método que administra los pedidos de los clientes es doPost
2. Se identifica la operación que el usuario esta solicitando mediante el método processRequest.
3. Una vez que se ha identificado la operación que el usuario ha solicitado (en este caso recuperación de compromisos) se procede a invocar el método getCompromisos del objeto BodegaBusinessDelegate, este método recibe los parámetros de búsqueda (año y estado de consumo en el caso de todos los compromisos y número de compromiso en caso de que solo sea uno en particular).
4. El método anterior invoca al método getCompromisos del objeto BodegaFacadeLocal
5. Este método invoca al método findByEstadoConsumoAnio del objeto

CompromisoCompraLocalHome, este a su vez retorna un arreglo de compromisos de compra al objeto BodegaBusinessDelegate.

6. El objeto BodegaBusinessDelegate se encarga de armar un arreglo de JCompromisos donde constan los campos que se recupera de un compromiso.
7. Este arreglo se retorna al objeto BodegaProcessRequestServlet el cual se encarga de enviar estos compromisos como un flujo de datos los cuales son recibidos por el HTTPComunicationHandler, de aquí en adelante se sigue con los pasos descritos para el manejo de los compromisos.

Login del usuario, lado del cliente

Se asume que no se presentan problemas con la red ni con el dispositivo (baterías bajas u otros)

Se asume además que estamos ingresando por primera vez al sistema con lo cual se creara el usuario en la Palm. Para mayor detalle ver Anexo 1.5- DIO Login de Usuario (Cliente)

1. Cuando el usuario inicia la aplicación se invoca el método startApp de la clase BodegaReview.
2. En este método se crea el objeto ModelFacade mediante la llamada a su constructor.
3. En el constructor del ModelFacade se crean los objetos LocalModel y RemoteModelProxy.
4. El objeto Local Model Crea a su vez el objeto RMSAdaptor.
5. El Objeto RemoteModelProxy se encarga de crear al objeto RMSCacheHandler y este a su vez crea el objeto HTTPComunicationHandler,
6. Luego de credos todos estos objetos en el método startApp se procede a crear al objeto UIController.
7. Luego desde startApp se invoca al método init del objeto UIController
8. En el método init de UIController se crea al objeto SignOnUI, al crearse este objeto es donde se le presentan al usuario las opciones para que ingrese al sistema(ingreso de user, password)
9. Se invoca al método getAccounts del objeto ModelFacade.

10. El objeto ModelFacade procede a invocar a getAccounst del LocalModel
11. Luego de recuperadas las cuentas de usuario se procede a retornar estas cuentas tanto al objeto ModelFacade como al UIController.
12. El objeto UIController pregunta revisa si el ultimo usuario que ingreso al sistema guardo en sus preferencias el hacer un login silencioso es decir el inicia la aplicación sin necesidad de ingresar user y password.
13. El objeto UIController invoca al método init del objeto SignOnUI al cual se le envían el usuario y la cadena de cuentas recuperadas. Aquí se inicia la pantalla de ingreso al sistema con las cuentas recuperadas presentadas en un choice group(si estas existieran) de lo contrario una pantalla para el ingreso de user y password.
14. Si no se realiza un login silencioso se presenta la pantalla que anteriormente se armo en la cual se presenta las cuentas de usuarios recuperadas para que se elija con que usuario se va a ingresar al

sistema (si es el primer ingreso al sistema se presentan una pantalla para que se ingrese user y password.

15. Cuando el usuario ha ingresado al sistema el objeto SignOnUI delega el manejo de este evento al UIController mediante la llamada al método handleEvent al cual se le envía como argumento el evento que se esta manejando (en este caso el evento del login del usuario) y la cuenta con la cual se pretende ingresar al sistema.
16. El método event del objeto UIController se encarga de realizar la operación solicitada ya sea esta el logonear a un usuario que ya lo ha hecho previamente o a un nuevo usuario.
17. Desde el objeto handleEvent se invoca al método signOn enviándole el objeto EventDispatcher(el cual se instancia), user y password del usuario.
18. El objeto UIController además inicia un hilo que será el encargado de realizar la operación de validar al usuario para su ingreso al sistema

19. Luego de esto se invoca al método runWithProgress el cual corre el hilo creado en el paso anterior, este hilo esta conFigurado con los parámetros que permiten manejar el requerimiento de ingreso al sistema que el usuario solicitó
20. Al correr este hilo se invoca al método login del objeto modelFacade al cual le son enviados user y password
21. Este método invoca al método login del objeto RemoteModelProxy al cual le son enviados user y password.
22. Este método invoca al método login del objeto RMSCacheHandler y este a su vez al método login del objeto HTTPCommunicationHandler.
23. El método login del objeto HTTPCommunicationHandler es quien envía un flujo de datos la cual contiene la operación que se desea realizar así como también el user y password del usuario.
24. Esta petición es manejada por el servidor mediante BodegaProcessRequestServlet el cual devuelve un

flujo de datos conteniendo los datos de la cuenta de un usuario.

25. En el objeto HTTPCommunicationHandler se arma un objeto del tipo JAccount con el flujo de datos recibidos y se devuelve este JAccount al hilo que inicio el proceso.
26. Cuando los datos se retornan se da inicio a otro hilo(el cual se envió como parámetro), este hilo corresponde al objeto EventDispatcher.
27. En este caso se asume que es el primer usuario que ingresa a la Palm por lo que la operación que realiza el EventDispatcher es la de EVENT_ID_LOGIN_USER_CREATE.
28. En este hilo se procede a invocar al método createAccount del objeto ModelFacade al cual se le envía la datos de la cuenta del usuario(estos datos fueron recuperados del servidor).
29. Por ultimo se presenta la pantalla que contiene las opciones de preferencias del usuario.

Login del Usuario, lado del servidor

Se ha explicado que el servidor retorna un flujo de datos el cual contiene información del usuario que se recupera con el user y password de este. Se explicara como el servidor retorna estos datos. Para mayor detalle ver Anexo 1.6.- DIO Login de Usuario (Servidor)

Como se explico en el anterior diagrama el objeto HTTPCommunicationHandler se encarga de enviar un flujo de datos al servidor, en este flujo de datos consta el tipo de operación que se va a realizar.

1. Este flujo es recibido por el servidor mediante el servlet BodegaProcessRequestServlet, el método que administra los pedidos de los clientes es doPost
2. Se identifica la operación que el usuario esta solicitando mediante el método processRequest.
3. Una vez que se ha identificada la operación que el usuario ha solicitado (en este caso el ingreso al sistema) se procede a invocar el método login del objeto BodegaBusinessDelegate, este método

recibe los parámetros de user y password del usuario.

4. Se procede a invocar al método loginKerberos del objeto LoginUtility .
5. Luego este método invoca al método login del objeto KerberosAutenticationUtility, este método recibe user y password, valida que el user y password existan en el LDAP.
6. Luego se invoca al método getLdapAttributes del objeto KerberosAutenticationUtility para solicitar los atributos del usuario.
7. Hecho esto se invoca al método logout para cerrar la conexión que se creo por invocación del método login.
8. Con los atributos recuperados el objeto LoginUtility retorna un JAccount al objeto BodegaProcessRequestServlet y este se encarga de enviar esto al HTTPCommunicationHandler como un flujo de datos para que este se encargue de su manejo en lo posterior.

5.5 Implementación

La implementación de la aplicación cliente se hizo en una Palm Tungsten C, con las siguientes especificaciones:

Tamaño y Peso	4.8' x 3.07' x .65'; 6.3 oz.
Memoria	64MB (51MB capacidad de actual)
Sistema Operativo	Palm OS® 5.2.1
Procesador	400MHz Intel® PXA255, Intel® XScale Technology
Pantalla	16-bit, 320x320 pantalla de color *TFT. transflective*
Batería	1500mAh Rechargeable Lithium Ion/Polymer
Ranura de Expansión	Tarjetas adicionales no incluidas
Conector Universal	Hardware adicional no disponible
Teclado incrustado	Teclado QWERTY
Navegador de 5 Botones	Navegación con 1 mano
Radio Móvil	WI-FI, IEEE 802.11b, antena interna
Notificación	Notificación vía: Vibración, Audio, LED

Tabla 12.- Características de una Palm Tungsten C. Realizado por los autores

Adicionalmente, fueron utilizados el servidor de aplicaciones Sun Java System Server 8 Update 1 y MySql Server 3.6, que se ejecutaron en una computadora con las siguientes características:

Memoria	632MB RAM
Sistema Operativo	Windows XP SP 1
Procesador	2.4GHz Intel® Pentium 4
Tarjeta de Red	Intel Pro 1000/100/10

Tabla 13.- Características de la computadora usada como servidor. Realizado por autores

Finalmente, se utilizó un AP con certificación 802.11b, marca D-Link DW-614+.

Se adjunta a esta documentación un CD que contiene los instaladores del servidor de aplicaciones Sun System Server 8, el servidor de base de datos Mysql y los archivos de la aplicación cliente y de la aplicación del servidor. Detalles sobre la instalación del SCI se pueden encontrar en el Anexo 2 y la información necesaria para utilizar la aplicación se encuentra en el Anexo 6 “Manual de Usuario”.

CAPÍTULO 6

6. DEFINICIÓN Y RESULTADOS DE PRUEBAS

En este capitulo se presentan pruebas sobre la instalación y pruebas de rendimiento sobre: el consumo de memoria, creación de objetos y uso de la red.

6.1 Eficiencia en la instalación de la aplicación cliente

Se realizó dos tipos de pruebas de instalación, estas pruebas analizan el tipo de archivos, su peso en bytes y el tiempo que demora la instalación.

6.1.1 Instalación vía OTA

OTA[58] que significa aprovisionamiento sobre el aire [29], es un método de instalación de aplicaciones sobre Internet, que permite descargar listas de aplicaciones y aplicaciones como lo indica la Figura 34.

[58] OTA.- Over the air provisioning, método para transferir las aplicaciones hacia los dispositivos móviles.

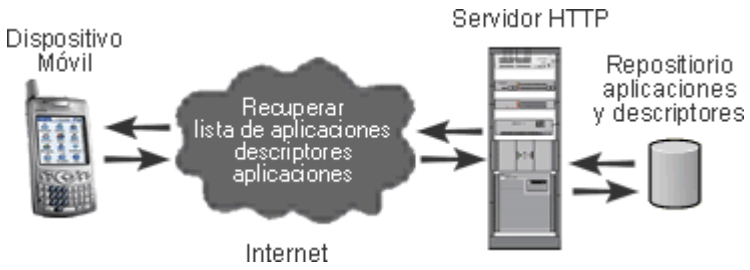


Figura 34.- Modelo de OTA. Referencia: http://wireless.java.sun.com/

Las pruebas se realizaron con los datos mostrados en las 2 primeras columnas de la tabla 15, donde se indican el tipo de los archivos que se necesitan para un aplicación móvil según J2ME que son el archivo de descripción de la aplicación (JAD) y el archivo de recursos de la aplicación (JAR).

Descripción	Tamaño PC (bytes)		Tamaño-Palm (bytes)	Tiempo de Descarga
	JAD	JAR		
Aplicación normal	468	111.219	213.596	12 seg.
Aplicación Ofuscada	250	82.138	155.585	5 seg.

Tabla 14.- Pruebas de Instalación vía OTA. Realizado por autores.

Cuando los archivos son instalados en la Palm toman el nombre del archivo descriptor. De la tabla 16 se concluye que el archivo sea o no ofuscado aumenta a casi el doble de su tamaño original y que los archivos aumentan aproximadamente un 73% en tamaño más en un archivo sin ofuscar y demoran un 41% en instalarse en un archivo sin ofuscar.

6.1.2 Instalación vía HotSync

La Palm Tungsten C, soporta hacer sincronizaciones locales conectando el dispositivo directamente al computador o a través de la red.

Al conectar el dispositivo directamente a la computadora se pueden obtener velocidades de transmisión de hasta 115200 bits por segundo. El tiempo que demora la sincronización a través de una red puede variar dependiendo del estado de la conexión. En ambos casos el PRC[59] ocupa un tamaño de 160.142 *bytes*.

6.2 Pruebas de rendimiento.

Las pruebas de control de rendimiento en el Sistema se basan en los planos de diseño de *Sun Microsystems* para aplicaciones J2ME (*Java Blueprints for Mobility*[60]), este análisis se lo realizó con las herramientas de monitoreo del *J2ME Wireless Toolkit*.

Las pruebas de rendimiento sobre la conectividad fueron realizadas simulando un proceso real de ingreso y registro del sistema.

---

[59] PRC, Palm Resource Component, archivos de aplicaciones del sistema operativo Palm OS, www.palmsoirce.com

[60] Java Blueprints for mobility, http://java.sun.com/blueprints

6.2.1 Análisis del rendimiento en el uso de la Memoria[61]

Los aspectos que se tomaron en cuenta en este análisis fueron el uso de memoria y la creación de objetos, los parámetros con los cuales se configuraron en el emulador para realizar las pruebas de rendimiento fueron:

- Capacidad de Almacenamiento: 50000 KB
- Tamaño de *heap*: 4000 KB

Estos parámetros obedecen a las especificaciones de la Palm Tungsten C con la cual se probó el sistema, la tabla 16 muestra el comportamiento del sistema sin tomar medidas correctivas de optimización.

[61] Performance Tuning and Monitoring Applications
http://java.sun.com/j2me/docs/wtk2.1/user_html/perftuning.html

Punto de Revisión (Puntos críticos con altos consumos de memoria)	Número de Objetos en el *Heap*	Memoria (4096000 bytes)	
		Usada	Libre
Hasta Iniciar la Aplicación	2736	(A)78652	4017348
Lanzo la aplicación, antes de HTTP	1520	(B)64380	4031620
Hasta mostrar el menú principal	2663	(C)107216	3988784
Muestro el UI de Editar Preferencias	2755	(D)111912	3984088
Cambio *isSilentlyLogin* y lo guardo las Preferencias	3020	(E)123012	3972988
Después de Navegar con los botones de TableUI, 10 veces cambiando la página, y recorriendo 20 compromisos	6162	(F)206528	3889472
Haciendo un requerimiento HTTP, de pedidos por compromiso	6910	(G)233254	3862746
Mostrando el Pedido 8430/2004 que tiene 16 JpedidoBien	8299	(H)296928	3799072
AL editar 1 pedido	8745	(I)312540	3781212
Hasta escribir el número de cédula para buscar un compromiso	13537	(J)479368	3616632
Hasta crear un custodio	17558	(K) 626532	3469468
Guardar un pedido	23231	(L)826420	3269580
Regresar al Menú Principal	24378	(M)871668	3636587
Al buscar el compromiso 8030/2004	1520	(N)59682	4036318
La Maquina Virtual ejecuto System.gc()			

Tabla 15.- Análisis de creación de objetos y consumo de memoria, antes de la recolección de basura, realizado por los autores.

Del análisis de la tabla 15 se establecen algunas consideraciones:

- No se puede predecir cuando la máquina virtual ejecutará una recolección de basura.
- En cada cambio de página aumentan 10 Objetos IndexEntry, se crean 10 JCompromiso, y el aumento de Strings y String[] es exagerado.
- El uso de un Ticker[62] aumenta el costo en memoria y creación de objetos interrumpidamente en el tiempo, aumentan a una tasa de 400 objetos por minuto y 7400 bytes por minuto.

Además, verificando el código de la aplicación se puede establecer, empíricamente, secciones de código en las cuales el uso de memoria y creación de objetos se incrementan rápidamente sin liberar recursos. En aquellas secciones se ejecutó una recolección explícita, las variaciones se pueden apreciar en la Figura 35.

[62] Ticket.- Clase de MIDP que permite inser29tar un componente animado en formularios.

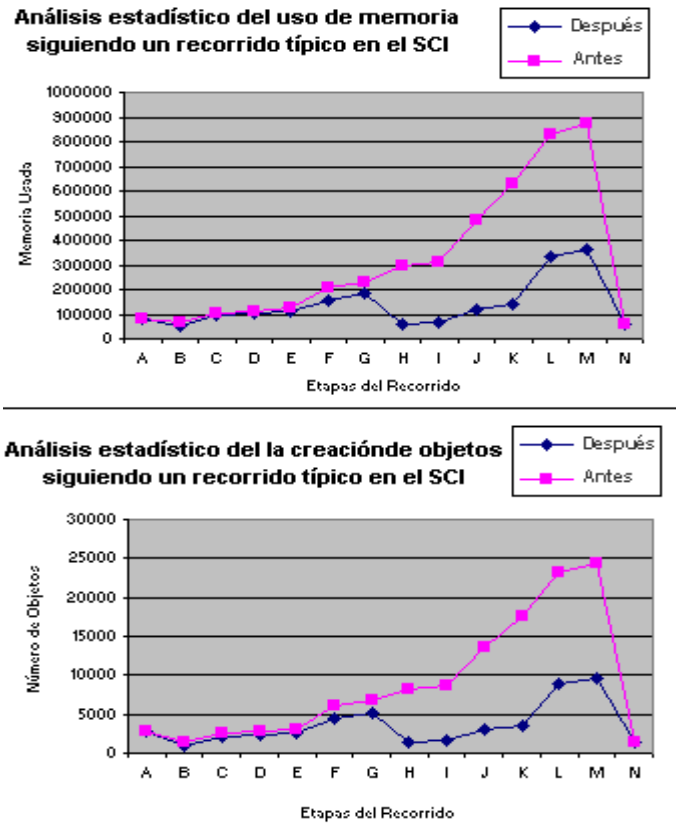


Figura 35.- Figura obtenida de la tabla 14. Realizado por autores.

En la Figura 35, el color fucsia muestra el uso de memoria y la creación de objetos cuando las recolecciones de basura son administradas por el sistema, el color azul denota el efecto sobre los recursos del sistema al ejecutar varias recolecciones de basura de manera explícita, mediante la llamada al sistema *System.gc()*.

Claramente se aprecia, una disminución considerable en los picos en la creación de objetos y uso de memoria.

6.2.2 Comportamiento de la red en el envío y recepción de mensajes via HTTP.

El comportamiento de la red puede ser monitoreado a través de las herramientas que ofrece WTK2.1, esta herramienta permite configurar el *throughput* de la red con valores que van desde los 1200 (*bit/sec*) hasta los 112000 (*bit/sec*) como lo muestra el Anexo 3. Los datos que provee la herramienta de monitoreo de red son los siguientes:

- Monitoreo de los Mensajes HTTT/HTTPS, SMS/CBS, *Socket*, *Secure Socket Layer (SSL)*, *Datagramas* y Puertos *Comm*.
- Ordenar los mensajes por URL, tiempo y conexión.
- Característica especial para filtrar los mensajes HTTP.

La tabla 18 muestra el tamaño de los mensajes HTTP enviados y recibidos entre la aplicación cliente y el servidor.

Punto de Revisión	*Bytes* enviados	*Bytes* recibidos
Login	19	204
GetCompromisos	23	754
GetCompromisos (10 siguientes)	23	747
GetPedidos (el primer envío se verifica la validez del compromiso y en el segundo obtengo los pedidos)	12	82
	12	110
GetPedidos (Esta vez sólo verifico la validez del compromiso, los pedidos ya los obtuve)	12	82

Tabla 16.- Muestreo de los tamaños de los mensajes HTTP.

Realizado por autores.

Se puede observar en a tabla 16 lo económico del uso de la heurística usada para serializar los objetos, además, la estrategia de enviar información por lotes disminuye significativamente el tamaño de los mensajes haciendo la transferencia más rápida y menos costosa.

Pruebas de conectividad

En este aspecto se han presentado algunos inconvenientes los cuales con el respectivo desarrollo de la aplicación han sido superados con éxito de tal modo que se ha probado la aplicación ubicando los puntos de acceso en distinto lugares y se ha obtenido en cada una de estas pruebas resultados satisfactorios.

Colocando el punto de acceso en diferentes lugares y variando la distancia entre este y el dispositivo móvil, se

ha probado a instalar la aplicación por distintos medios, se han conseguido tiempos similares tanto en la instalación como en la recuperación de datos cuando la aplicación entra en funcionamiento.

Pruebas de latencia

Los tiempos de latencia que se obtienen en la aplicación en lo que a operaciones contra el servidor se detallan en la tabla 17:

Operación	Descripción	Tiempo Requerido
Ingreso al Sistema	Solo el personal autorizado puede acceder al sistema (se solicita user y password).	El tiempo requerido en esta operación esta en el rango de 3 – 5 seg.
Recuperación de compromisos.	La recuperación en primera instancia es solo de los primeros 10 compromisos existentes.	El tiempo requerido en esta operación esta en el rango de 10 – 15 seg.
Recuperación de los ítems que forman un compromiso	Los ítems que forman un compromiso pueden ser uno o varios. El tiempo de recuperación de estos dependerá de este número.	El tiempo requerido en esta operación esta en el rango de 5 – 10 seg.
Búsqueda de compromisos	Se solicita el numero de compromiso que se requiere	El tiempo requerido en esta operación es esta en el rango de 3 – 6 seg.

Tabla 17.- Tiempos de Latencia Registrados. Realizado por los autores.

Estos tiempos tienen mucho que ver con la calidad de la red que se esta empleando así como también de obstáculos en la línea de vista entre el punto de acceso y el dispositivo inalámbrico.

6.3 Pruebas con los Usuarios

Las pruebas con los usuarios se dividen en dos categorías:

- Pruebas con los desarrolladores del Centro de Servicios Informáticos.
- Pruebas con los usuarios finales del SCI con el personal del Departamento de Bodega que realizan los ingresos a bodega.

En ambas pruebas se establecieron requerimientos diferentes, enfocados a la total integración del SCI al SAF.

6.3.1 Pruebas desarrolladas con el personal del CSI

Los procesos que el SCI implementa son en parte replicación de procesos actuales y en otra se mejoró la gestión de un proceso actual, los procesos son:

- Replicación del proceso de Afectación Presupuestaria.

- Mejoramiento en la gestión del proceso de Ingreso a Bodega.

Ambos procesos fueron probados y posteriormente revisados por la Coordinadora de Desarrolla de Sistemas del CSI.

En la parte de Ingreso a Bodega se realizaron las siguientes pruebas:

- Se realiza el ingreso a bodega de un bien, mediante el uso del SCI.
- Se revisan e imprimen los reportes de movimientos de activos fijos e ingresos de activos fijos.

Los resultados de estas pruebas fueron exitosos, no habiendo ningún inconveniente en la impresión de los reportes de movimiento e ingreso.

Las pruebas del proceso de Afectación Presupuestario se realizaron de la siguiente manera:

- Se realiza el ingreso a bodega de un bien, mediante el uso del SCI.
- Se contabiliza el movimiento.

La contabilización del movimiento se realiza sin ningún error.

Después del proceso de pruebas correspondiente, la Coordinadora de Desarrollo de Sistemas del CSI, ha emitido un certificación de que el SCI cumple con los requerimientos y objetivos para el cual fue desarrollada.

6.3.2 Pruebas desarrolladas con el personal de Bodega

Estas pruebas se desarrollaron con el personal de bodega que hacen los ingresos de bienes, el usuario manifestó su total acuerdo y respaldo a la implementación de este tipos de iniciativa.

Además, el usuario hizo comentarios sobre futuras mejoras y actualizaciones del SCI, las cuales está fuera del alcance de este proyecto.

CONCLUSIONES Y RECOMENDACIONES

En esta sección presentan las conclusiones obtenidas en el desarrollo e implementación del proyecto. Además, se ofrecen recomendaciones relacionadas a la implementación del sistema en un ambiente de producción y sobre futuros desarrollos sobre computación móvil.

Conclusiones

En base a los objetivos planteados y a los criterios de evaluación mencionados en el Capítulo 1 se puede llegar a las siguientes conclusiones:

- Creemos que el SCI mejorará la productividad de los bodegueros al proveer de una herramienta automatizada que ofrece mejoras el método tradicional de recolección de datos. Con la cual se espera disminuir el tiempo en el que se hace un ingreso de inventarios y eliminar el proceso de trascripción de datos realizado por el personal administrativo de la bodega. La comparación entre el método de ingreso tradicional y el método de ingreso automatizado se detalla en el Anexo 4.

- Se espera una mejora en la forma de manejar la información, eliminando procesos redundantes, como el que se realiza actualmente al escribir los datos 2 veces (en papel y en el sistema) y tratando de evitar posibles errores en un ingreso de inventario al proveer de un método de ingreso guiado e intuitivo en la aplicación cliente del SCI.

- Entre las tareas que se hacen en el proceso actual de ingreso de inventarios, se tiene que, el personal administrativo prepara documentos para que los bodegueros escriban la información que recolectan de los inventarios y después transcriben esta información al sistema de la ESPOL. Mediante la utilización del SCI, estas dos tareas son eliminadas, dando como resultado una disminución en las tareas que tiene el personal administrativo.

- Se ha logrado la integración del SCI y de nuevas tecnologías con la plataforma actual de los sistemas de la ESPOL (*SmallTalk*), mediante la utilización de estrategias de diseño recomendadas por el personal experto del SCI y el acceso a la base de datos del sistema financiero de la ESPOL, la cual es compartida por los dos sistemas.

- El hecho de haber usado las especificaciones MIDP 2.0 y CLCD 1.1 de J2ME asegura la portabilidad del sistema sobre

dispositivos móviles que soporten estas especificaciones y que cumpla con las recomendaciones del capitulo 6.2.

- Creemos que cuando el SCI sea puesto en un ambiente de producción se logrará acceder a la información sobre ingresos a bodega de una manera más rápida y a un menor costo del actual, debido al ahorro en tiempo y materiales al mejorar el proceso de recolección de datos.
- Se espera lograr el mejoramiento de la gestión realizada por el personal de bodega, al proveer de un flujo de información más eficiente, lo cual se logra al permitir que el bodeguero realice consultas y búsquedas de datos de forma automática.
- Se concluye que los mensajes enviados vía HTTP desde el servidor hacia el dispositivo móvil no deben tener un tamaño muy grande, en promedio para este tipo de aplicaciones se recomienda un tamaño no superior a los 750 *bytes,* para evitar problemas de alta latencia de la red. En este sentido, el tamaño de los mensajes esta ligado al número de objetos que se transmiten a través de la red, para el SCI, se llego a la conclusión que 10 objetos a la vez permiten un nivel de rendimiento aceptable.

- Se llega a la conclusión de que a pesar de los escasos recursos que proveen los dispositivos móviles, el SCI provee un alto grado de usabilidad mediante la aplicación de metáforas usadas por el sistema de la ESPOL y el uso de términos familiares al personal de bodega.
- Mediante el uso colores y animaciones en la aplicación cliente se logra brindar retroalimentación sobre las acciones realizadas por el usuario sobre determinada pantalla. De esta manera por ejemplo, se puede identificar rápida y fácilmente el compromiso que se quiere verificar al resaltarlo con un color diferente al resto o visualizar el progreso de las tareas que se están ejecutando.
- En cuanto a los costos de desarrollo e implementación se concluye que son relativamente bajos y no implican grandes inversiones en equipo. Sin embargo el tiempo que se le dedica es alto y esto se debe a la ausencia de expertos locales en los sistemas de la ESPOL. Los detalles de los costos se pueden apreciar en el Anexo 5.
- Se encontró durante las pruebas con la base de datos de la ESPOL que sólo la versión 7.2 de DB2 provee compatibilidad para la versión 6.1 de DB2. Además, la versión 7.2

implementa la interfaz *DataSource* requerida por la mayoría de servidores de aplicaciones acutales.

Recomendaciones.

Las siguientes recomendaciones proporcionan pautas para la puesta en producción del SCI. Estas recomendaciones están basadas en las pruebas desarrolladas y en experiencias adquiridas durante el desarrollo.

- Se recomienda hacer las pruebas del SCI con el personal de bodega para que sea incorporado a un ambiente de producción.
- En las pruebas realizadas en el Capítulo 5 se determinó que el sistema debe de disponer de 1 MB para ejecutar la aplicación cliente, es por este motivo, que se recomienda que para ambientes de producción se utilicen dispositivos que dispongan de una cantidad de memoria dinámica igual o superior. Este tamaño sólo se refiere a lo que consume la aplicación cliente y no se toma en cuenta la memoria dinámica empleada por el sistema operativo y otras aplicaciones.
- Debido a que la ESPOL tiene licenciado DB2 UDB versión 6.1 de IBM, que el Servidor de Aplicaciones escogido fue

Sun System Server Personal Edition v.8 Update 1 y que este servidor necesita un pool de conexiones para crear el recurso de persistencia JDO, se recomienda que para ambientes de producción se utilicen las librerías de conexión que provee la versión 7.2 de DB2, la cual es la única que provee compatibilidad a la versión 6.1.

- Se recomienda una capacitación de 10 horas para introducir al personal de bodega en el uso del sistema.

Para finalizar se presentan recomendaciones no funcionales muy útiles para futuros desarrollos:

- Se recomienda fomentar la capacitación de los estudiantes en computación móvil y tecnologías de punta como J2ME y J2EE. Así como, actualizar los pensums académicos para promover la investigación y desarrollo en los estudiantes.
- Se considera que la mejor opción para una futura migración del sistema financiero y académico de la ESPOL, es a plataformas con tecnologías compatibles con las especificaciones J2EE.

REFERENCIAS BIBLIOGRÁFICAS

[1] Charles Hongreen, Contabilidad Básica, (Reading, Massachussets. Addison- Wesley, 1982), pp.88-106

[2] Jun Gong, Peter Tarasewich, Guidelines for Handheld Mobile Device Interface Design, College of Computer and Information Science, Northeastern University

http://www.ccs.neu.edu/home/tarase/GuidelinesGongTarase.pdf

[3] Cellular Online, Diciembre 21004, “Latest Mobile, GSM, Global, Handset, Base Station, & Regional Cellular Statistics”, http://www.cellular.co.za/stats/stats-main.htm

[4] Tendencias Digitales, 2003, Internet móvil marcará el rumbo de las comunicaciones y la nueva forma de hacer negocios, <http://www.tendenciasdigitales.com/td/mundo9.htm>

[5] Paola Reina, 2003, Redes Inalámbricas. http://www.monografias.com/trabajos12/reina/reina.shtml

[6] Intel, 2004, Ethernet Inalámbrica. http://www.intel.com/ es/home/trends/wireless/info/ethernet.htm

[7] Sun Microsystems Inc., Java 2 Enterprise Edition, http://java.sun.com/j2ee

[8] Sun Microsystems Inc., Java 2 Micro Edition, http://java.sun.com/j2me

[9] Documentation Types of JDBC technology drivers, Sun Microsystems Inc., http://java.sun.com/products/jdbc/driverdesc.html

[10] Basic Customization Guide, Java™ 2 Platform, Micro Edition, Wireless Toolkit Version 2.1, Diciembre 2003, Sun Microsystems, Inc.

[11] WebSphere Micro Environment for palmOne Devices, MIDP 2.0 Porting Guide (GM Final), Marzo 2004, PalmOne Inc. www.palmone.com/java

[12] Sun Microsystems Inc, 2002, Connected, Limited Device, ConFiguration, Specification Version 1.0a, Java 2 Platform Micro Edition, http://java.sun.com.

[13] Sun Microsystems Inc, 2002, The CLDC HotSpot™, Implementation Virtual Machine, Java™ 2 Platform, Micro Edition (J2ME) http://java.sun.com

[14] Juan Manuel Madrid Molina, 20-4-2004, Seguridad en redes inalámbricas 802.11, Universidad Icesi

[15] Pau Oliva Fora, Marzo 2003, (In)seguridad en redes 802.11b, http://pof.eslack.org/wireless/

[16] Young Kim, Octubre 2002, 802.11b Wireless LAN Authentication, Encryption, and Security.

[17] Grupo IEEE 802.11, http://grouper.ieee.org/groups/802/11/

[18] Nikita Borisov, Ian Goldberg, David Wagner, Security of the WEP algorithm. <http://www.isaac.cs.berkeley.edu/isaac/ wep-faq.html>

[19] Microsoft, IT Introduction to Windows 2000 Security, Enero 05, 2000, http://www.microsoft.com/windows2000/server/evaluation/busines

s/secintro.asp

[20] Rosanna Lee, Noviembre 1, 2002, Sun Microsystems Inc., the JNDI Tutorial, Building Directory-Enabled Java Applications.

[21] Microsoft, 20 de febrero de 2002, Introducción a las redes LAN inalámbricas, http://www.microsoft.com/latam/windowsxp/pro/biblioteca/

planning/wirelesslan/intro.asp

[22] Klaus Bergner, Andreas Rausch, Marc Sihling, Julio 1997, Using UML for Modeling a Distributed Java Application, Institut für Informatik, Technische Universität München

[23] Rod Jhonson, Enero del 2005, J2EE Development Frameworks, IEEE Computer Magazine, www.computer.org

[24] George Reese, Noviembre del 2000, Database Programming with JDBC and Java, Second Edition.

[25] D. Alur, D. Malks, J. Crupi, 2001 Core J2EETM Patterns, http://java.sun.com/blueprints/corej2eepatterns/index.html

[26] Thierry Violleau and Ray Ortigas, Junio 26, 2003, Supporting Disconnected Operation in Wireless Enterprise Applications, Sun Microsystems, http://java.sun.com/blueprints.

[27] JavaOne y Anthony Scian,Senior Software Developer. Research in Motion, Efficient Java 2 Platform, Micro Edition (J2ME) Programming Tips and Techniques, www.rim.com, http://java.sun.com/javaoneonline/

[28] Jonathan Knudsen July 2001, Wireless Java: Developing with Java 2 Micro Edition , Chapter 10 Performance Tuning.

[29] Enrique Ortiz, November 2002, Introduction to OTA Application Provisioning; http://java.sun.com.

[30] Jóse Pérez, Computación Móvil, http://www.monografias.com/trabajos5/compumo/compumo.shtml

[31] Ericsson**,** **"**Internet Móvil: un modo de vida, http://www.ericsson.com.mx/press/boletines/20001123_2.html.

[32] Carmen Azara, Irasema Molia, Guillermo Sergent, José Pérez Leal, 22-05-1997, "Computación Móvil", Universidad Central de Venezuela, http://www.monografias.com/trabajos5/compumo/compumo.shtml

[33] Sun Microsystems Inc, Midp-Style-Guide 7,. Referencia: http://java.sun.com/j2me

[34] © 2002 International Business Machines Corporation , Developing Enterprise Java Applications Using DB2 Version 8,. All rights reserved.

<http://www-106.ibm.com/developerworks/db2/library/techarticle/0209

hutchison/0209hutchison.html#Header_37>

[35] Robert Chartier, Application Architecture: An N-Tier Approach Part 1, http://www.15seconds.com/issue/011023.htm

[36] Thierry Violleau and Ray Ortigas, Blueprints of Mobility of Sun Inc, 2002 Supporting Disconnected Operation in Wireless Enterprise Applications.

[37] Sun Microsytems Inc., 2002, Session Facade Design Pattern, http://java.sun.com/blueprints/corej2eepatterns/Patterns/SessionFacade.html

[38] Sun Microsytems Inc., 2002, Business Delegate Design Pattern, http://java.sun.com/blueprints/corej2eepatterns/Patterns/BusinessDelegate.html

[39] Sun Microsytems Inc., 2002, Value Object Design Pattern, http://java.sun.com/blueprints/corej2eepatterns/Patterns/ValueObject.html

[40] Sun Microsytems Inc., 2002, Transfer Object Design Pattern, http://java.sun.com/blueprints/corej2eepatterns/Patterns/ValueObject.html

[40] Sun Microsytems Inc., 2002, Session Facade Pattern, http://java.sun.com/blueprints/corej2eepatterns/Patterns/SessionFacade.html

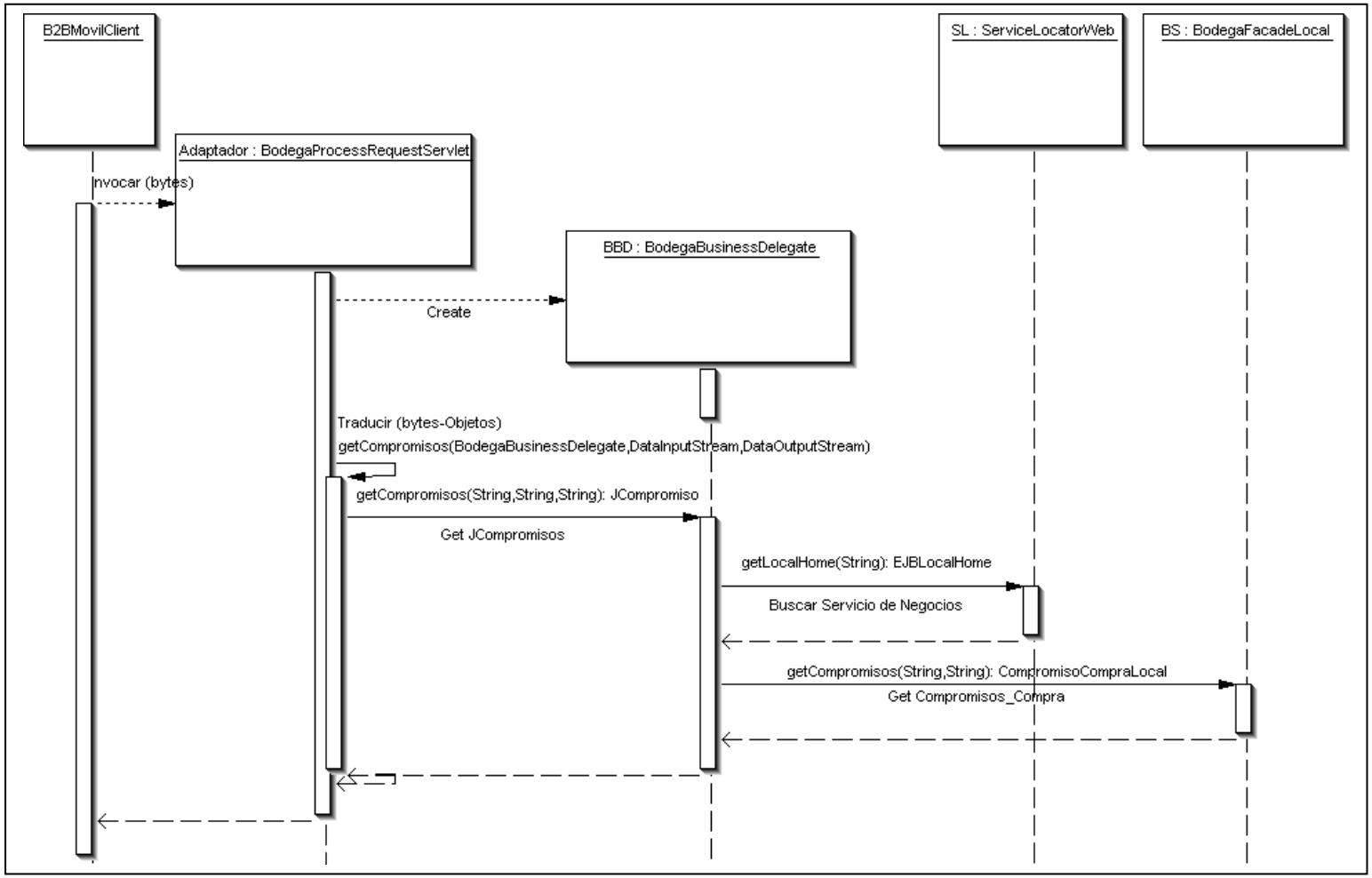
ANEXOS

ANEXO 1

Diagramas UML del Sistema de Control de Inventarios

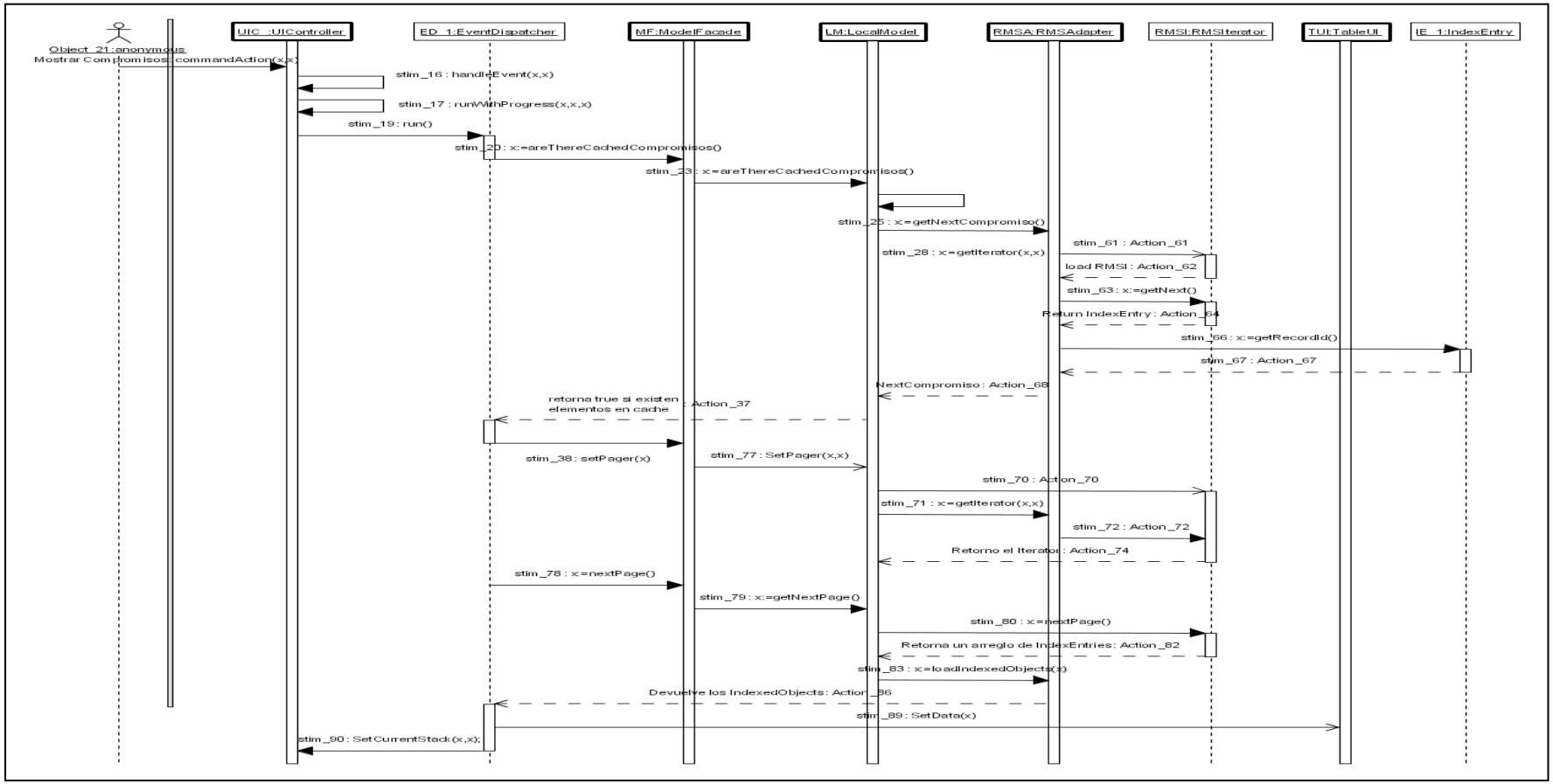
Anexo 1.1

DIO.- Implementación del patrón de diseño Delegación de Negocios en la aplicación de servidor



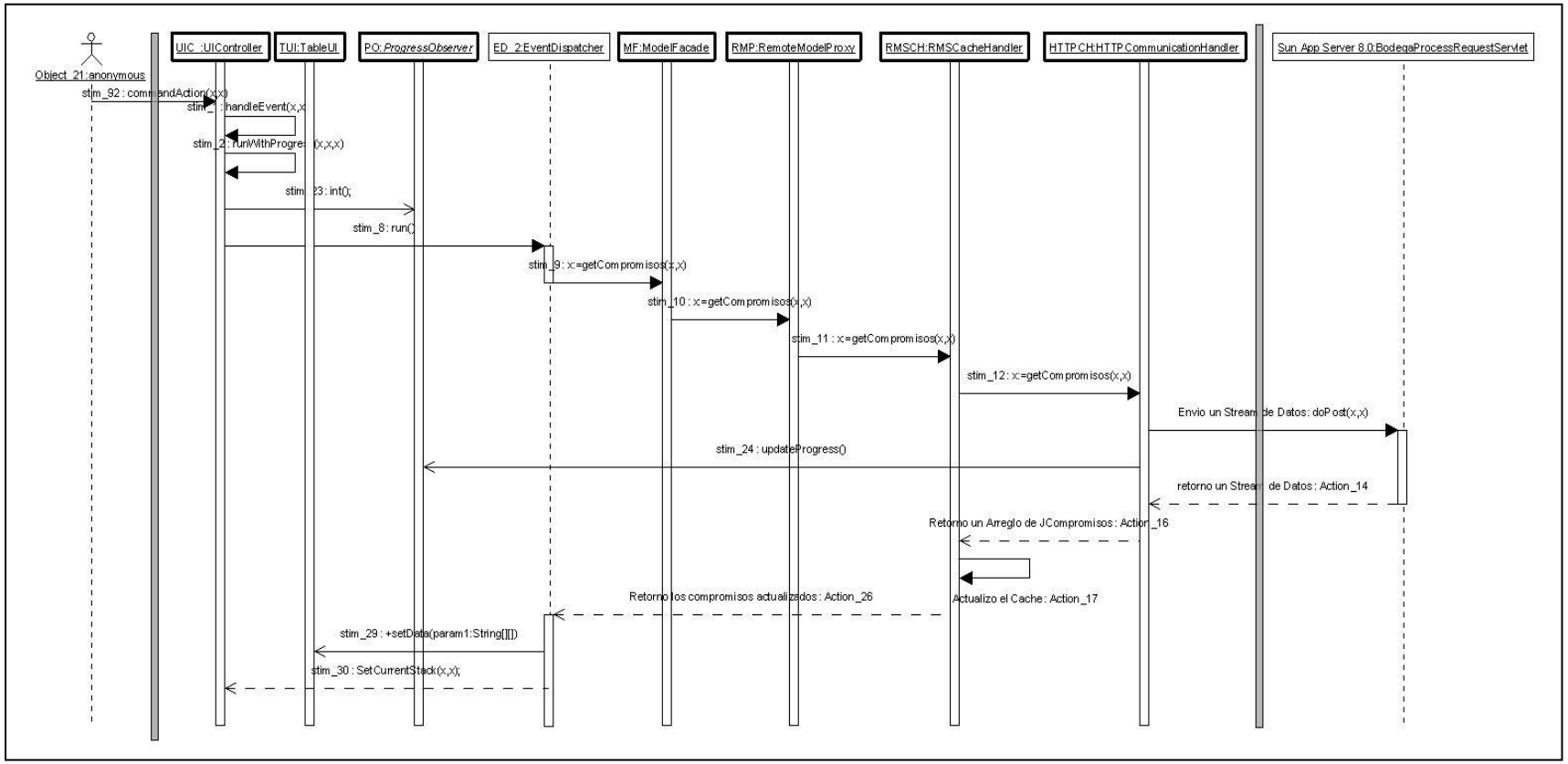
Anexo 1.2

Diagrama de interacción de objetos del escenario recuperación de Compromisos de la memoria temporal



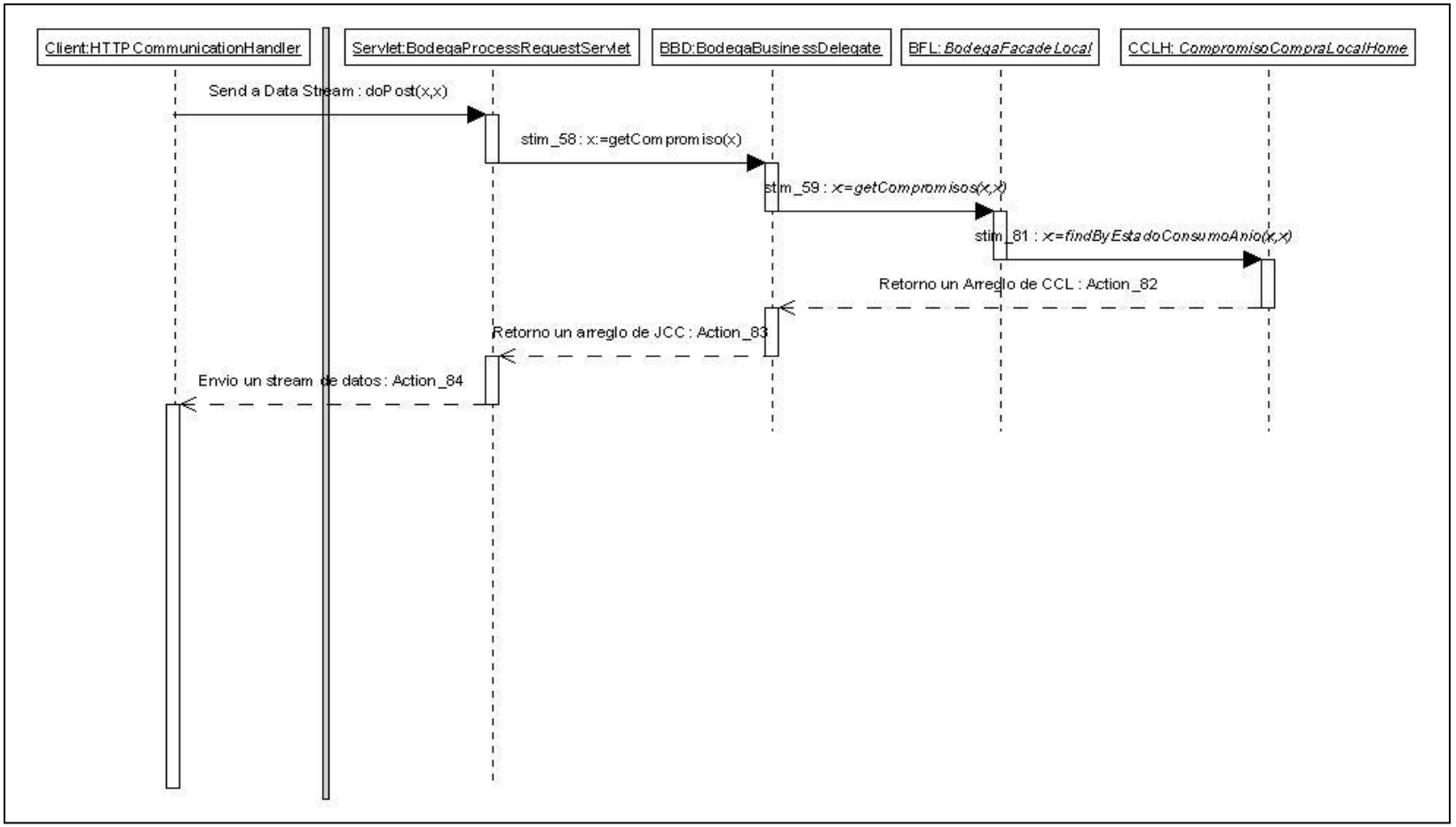
Anexo 1.3

Diagrama de interacción de objetos del escenario recuperación de compromisos en línea (Cliente).



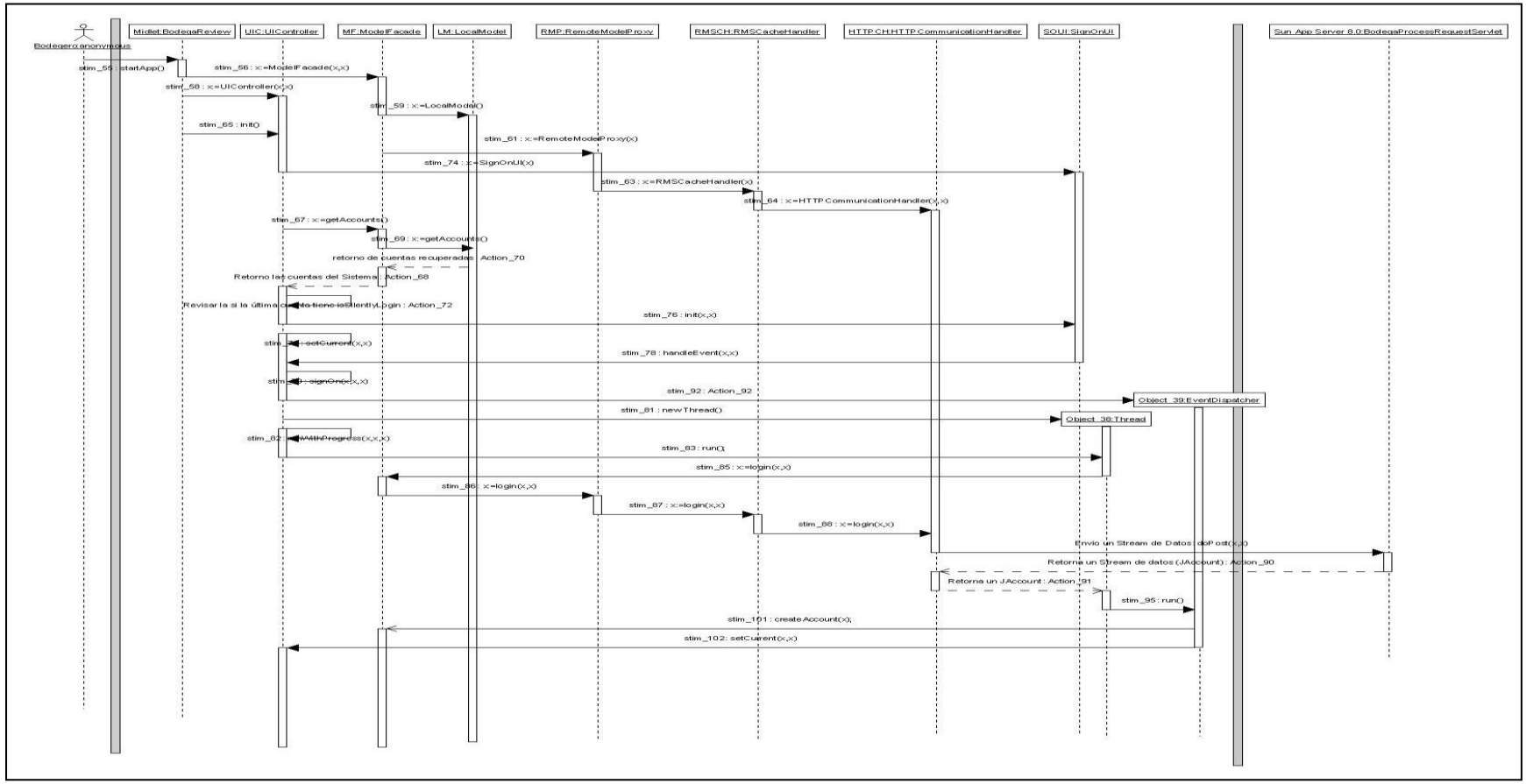
Anexo 1.4

Diagrama de interacción de objetos del escenario recuperación de compromisos en linea exitoso (Servidor).



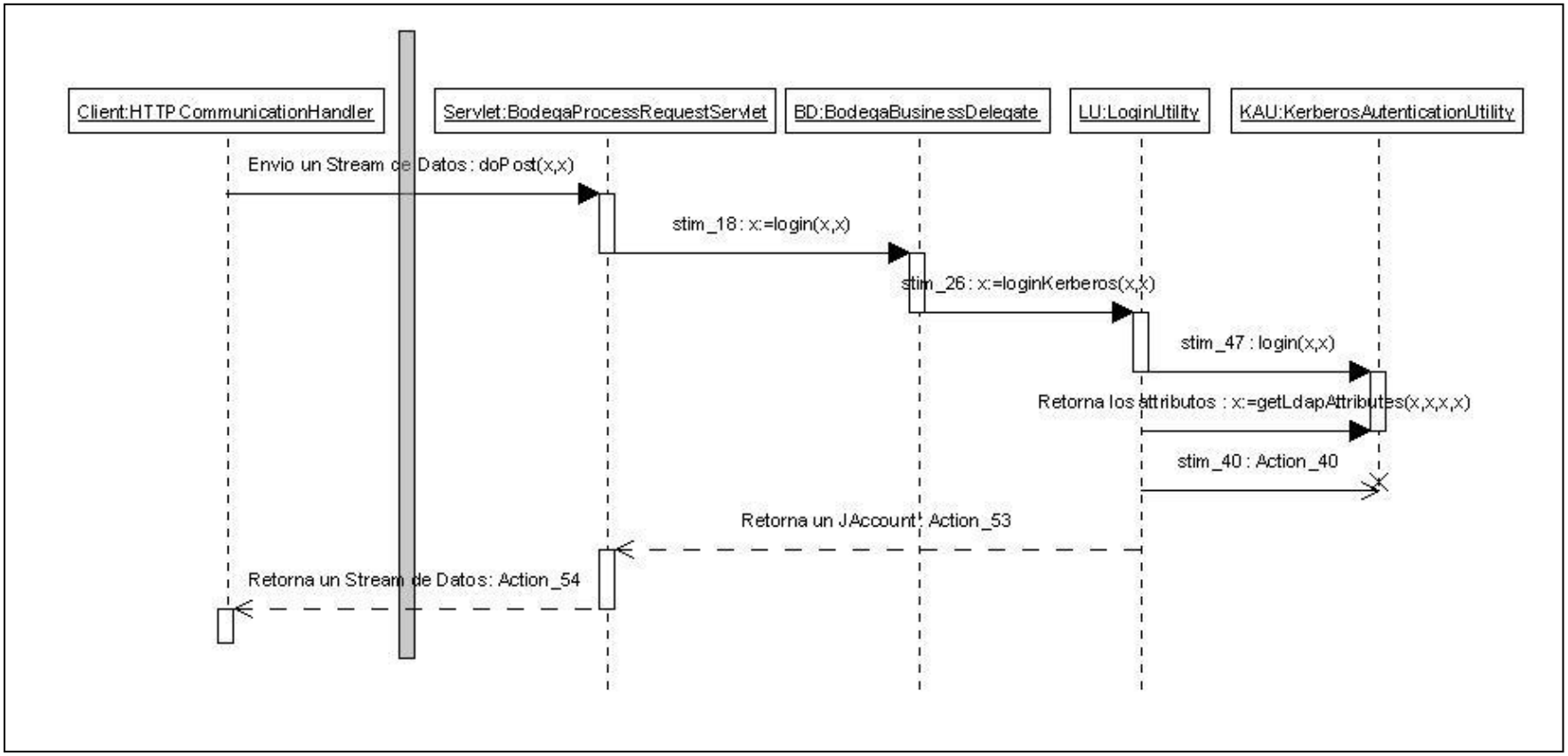
Anexo 1.5

Diagrama de interacción de objetos del escenario login de usuario exitoso (Cliente).



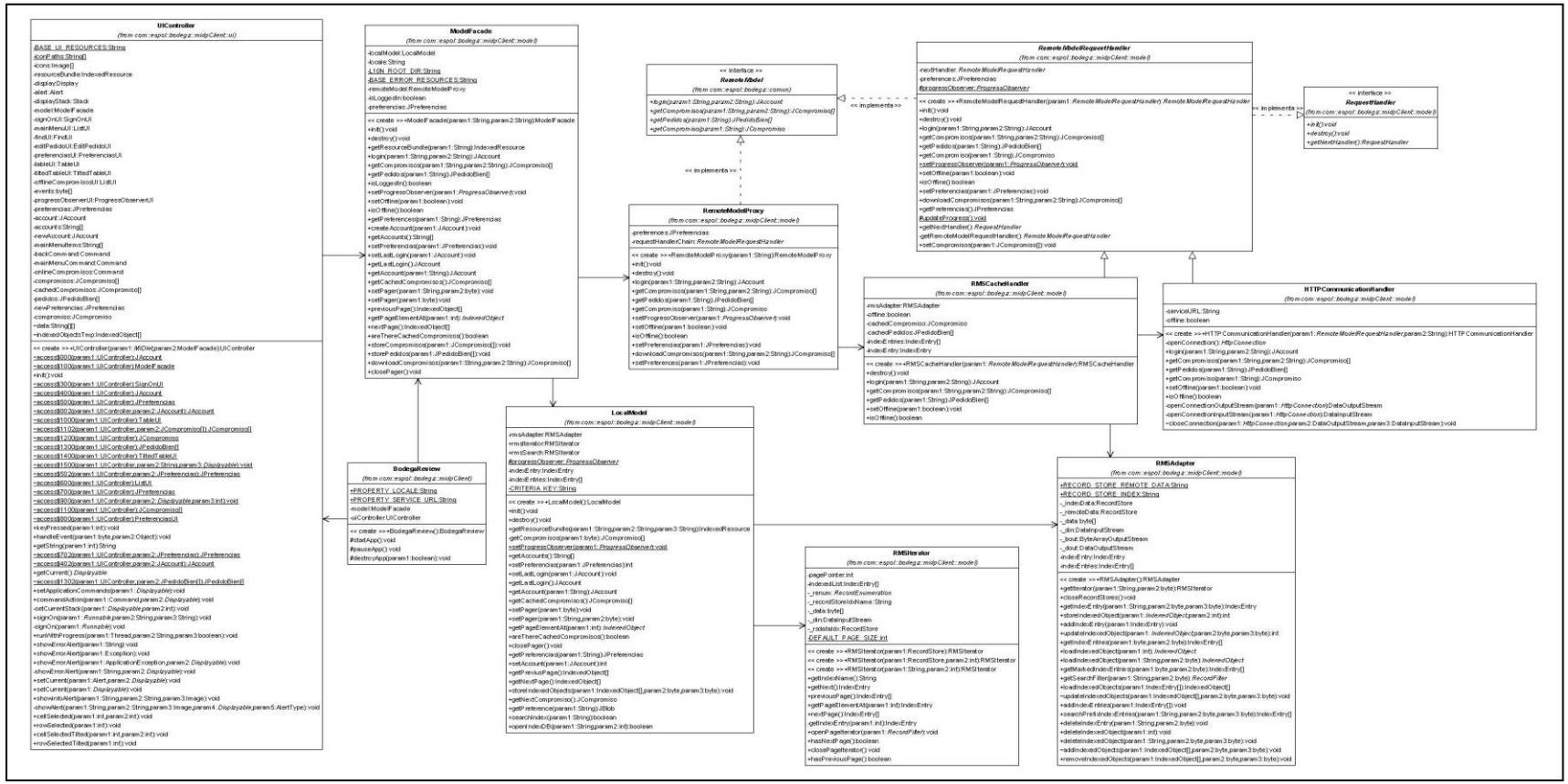
Anexo 1.6

Diagrama de interacción de objetos del escenario login de usuario exitoso (Servidor).



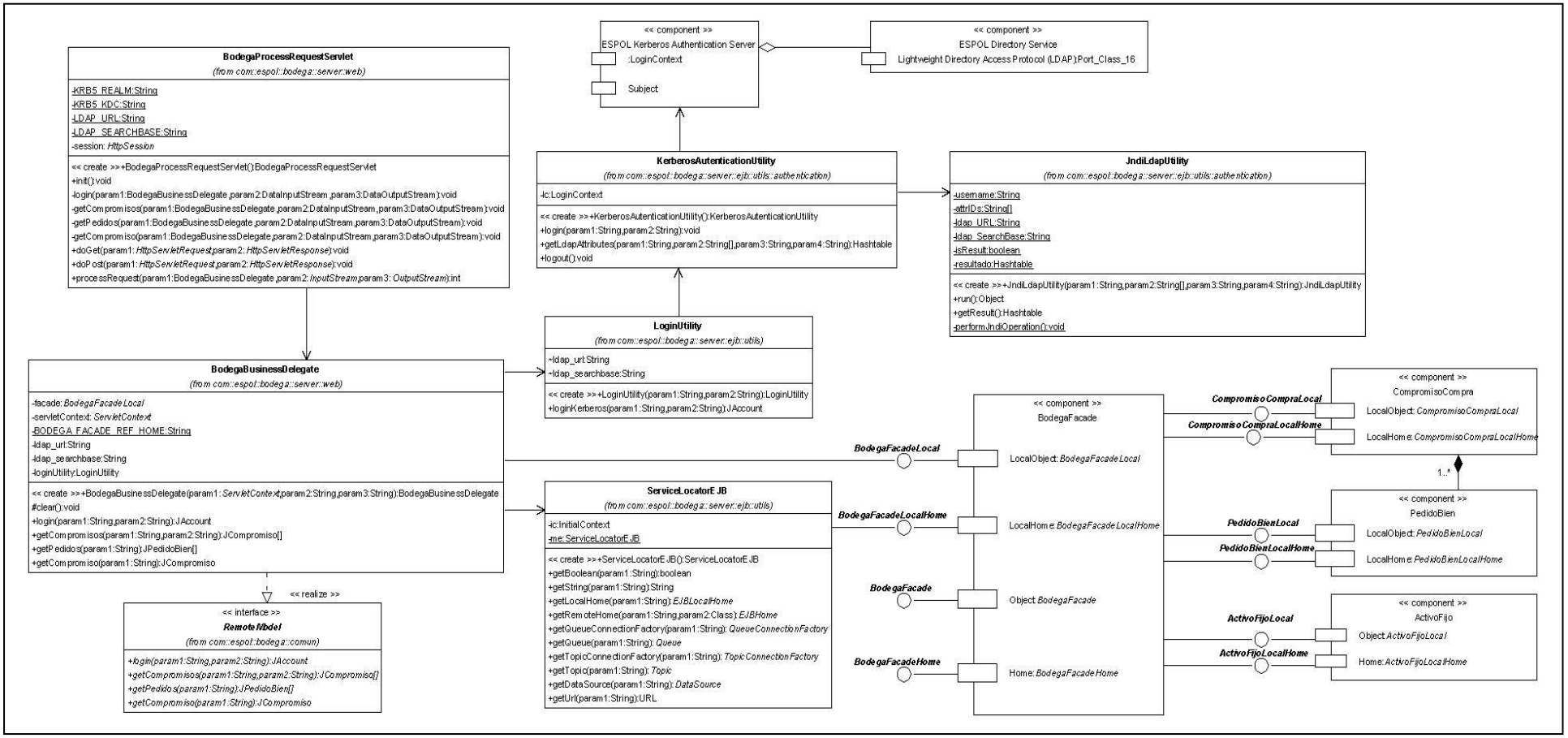
Anexo 1.7

Diagrama de Clases del Sistema de Control de Inventarios (Aplicación Cliente).



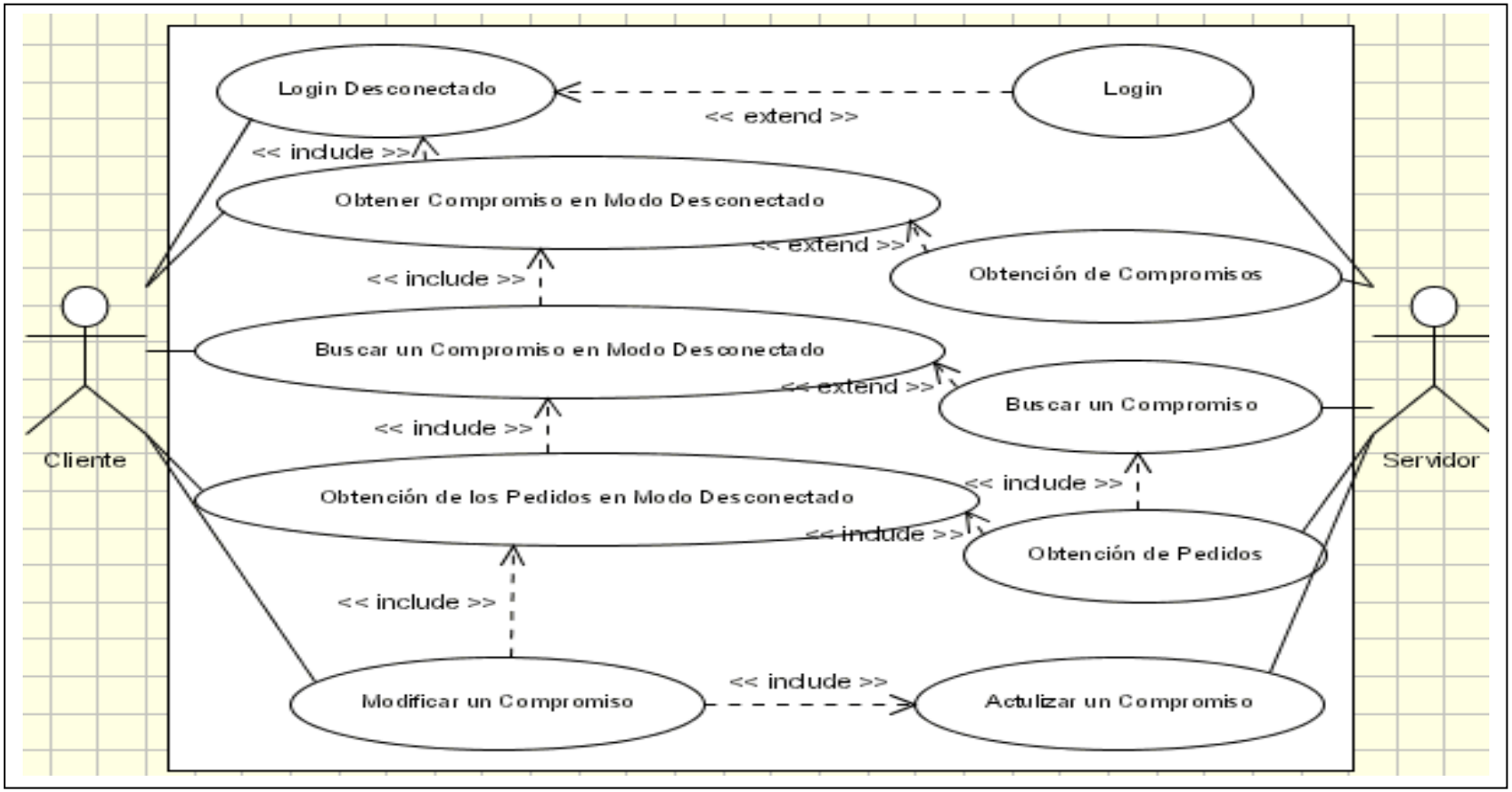
Anexo 1.8

Diagrama de Clases del Sistema de Control de Inventarios (Aplicación del Servidor)



Anexo 1.9

Casos de Uso – interacciones entre el cliente y el servidor en modo desconectado.



ANEXO 2

Manual de Instalación del SCI

Dentro del CD que se provee en esta documentación se encuentran los archivos, que contienen la aplicación cliente y de servidor del SCI:

- Aplicación de servidor.- Es un archivo llamado “bodega.ear”.
- Aplicación cliente.- Son dos archivos “BodegaPalm.jad” y “BodegaPalm.jar”. Adicionalmente, se encuentra un archivo llamado “ClienteBodega.prc” el cual contiene la aplicación cliente lista para instalarse en un dispositivo móvil Palm.

Requerimientos

Antes de instalar el SCI, se debe instalar y configurar lo siguiente:

- El servidor de aplicaciones “Sun System Server 8”, el cual puede ser descargado de http://java.sun.com, contenido en el CD adjunto.
- La base de datos relacional MySql, que puede ser descargada de http://www.mysql.com, contenido en el CD adjunto.
- El software de administración de clientes de la base de datos relacional IBM DB2 versión 7.2, el cual debe ser requerido al Centro de Servicios Informáticos de la ESPOL.
- El emulador del sistema operativo Palm OS 5.0, el cual puede ser descargado de “http://www.palmsource.com”, contenido en el CD adjunto.

Configuración e Instalación

Antes de instalar la aplicación del servidor y del cliente se debe configurar el servidor de aplicaciones y el dispositivo móvil respectivamente.

Para configurar el servidor de aplicaciones se deben de seguir los sientes pasos:

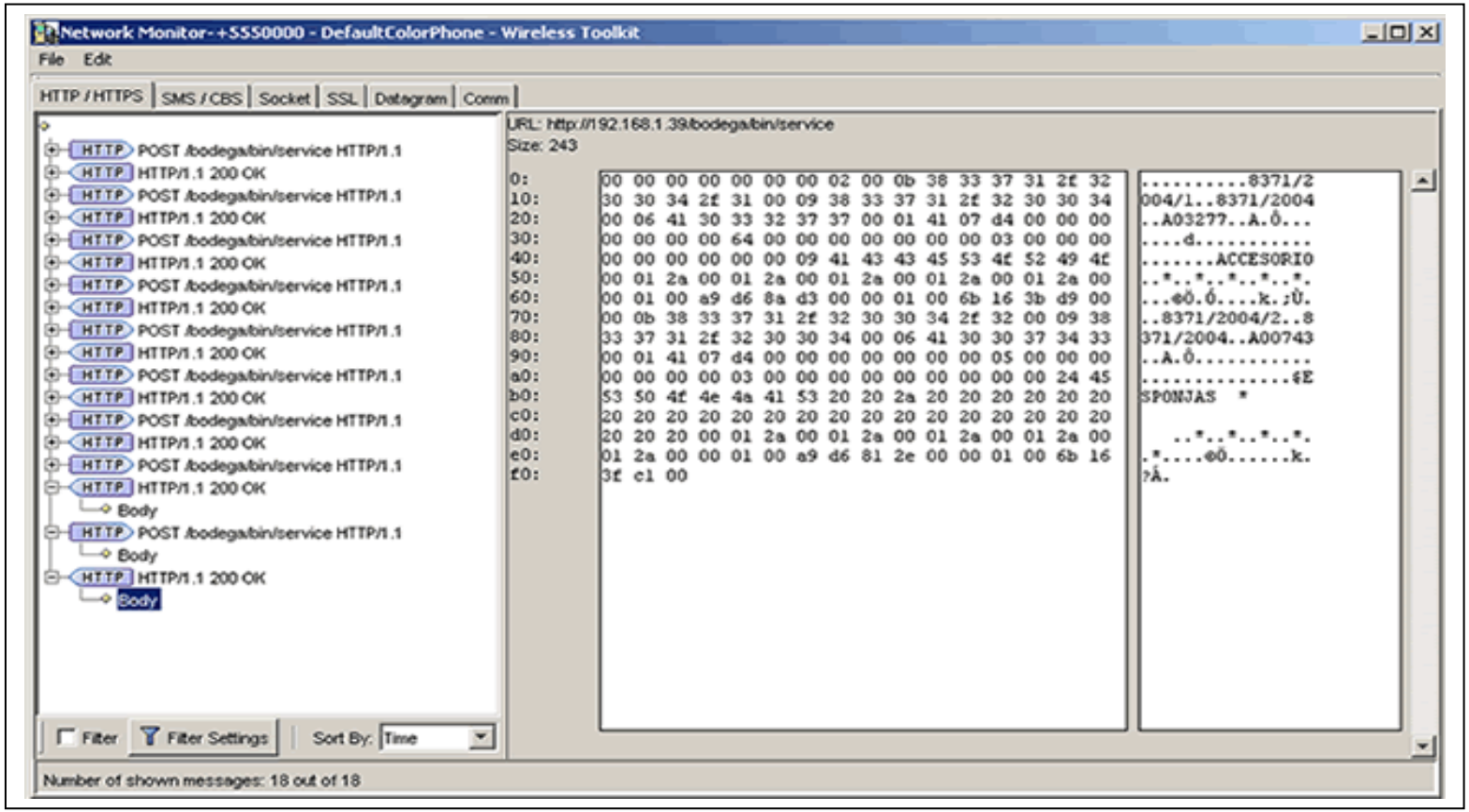
- Instalar el servidor de aplicaciones Sun System Server 8.
- Instale el servidor de base de datos relacional Mysql. Para iniciarlo tiene que ejecutar: %MYSQL_HOME%\bin\mysqld-nt.exe –standalone. Copie el archivo “mysql-connector-java-3.0.6-stable-bin.jar”, que está contenido en el CD a %J2EE_HOME%\domains\domain1\lib\ext
- Instalar y configurar, el software cliente de DB2, además configurar la conexión a la base de datos de financieros de la ESPOL (Este paso requiere autorización y soporte del Centro de Servicios Informáticos de la ESPOL).
- En la ruta de instalación de DB2 7.2 que se llamará %DB2_HOME%, ejecutar la aplicación: %DB2_HOME%\java12\jdbc2.0.exe, el cual creará algunos archivos en esta mismas carpeta, entre los cuales estará: “db2java.zip”, renómbrelo a “db2java.jar” y ejecute la aplicación :” usejdbc2.bat”.
- Copie “db2java.jar” a la carpeta %J2EE_HOME%\domains\domain1\lib\ext
- Inicie el servidor de aplicaciones.

- Coloque el path %J2EE_HOME%\bin, como variable de entorno del sistema.
- Ejecute : %BODEGA_SRC_DIR%\asant

Después de instalar la aplicación del servidor, ejecute el emulador de Palm Os 5.2. y comience a usar la aplicación ClienteBodega.

ANEXO 3

Herramienta para monitoreo de mensajes de red.



ANEXO 4

Comparación entre el método de ingreso tradicional y el método de ingreso automatizado

Descripción	Método tradicional	Método automatizado	Observación
Información sobre la orden de ingreso a bodega	Una o varias hojas para tomar datos	Un lista siempre actualizada con los compromisos que se deben ingresar	Hay un ahorro en el tiempo de búsqueda de la hoja que corresponde al compromiso que se quiere ingresar
Visualización de Datos	Cada registro representa una línea muy estrecha en cada hoja	Formularios personalizados para cada tipo de registro	El método automatizado presenta una forma más natural para visualizar los datos
Manejo de correcciones	Correcciones con tachones o líquidos correctores	Funciones de edición incorporadas	Las correcciones manuales disminuyen legibilidad y pueden inducir a errores en el proceso de trascripción
Almacenamiento de la información	Carpetas expuestas al ambiente	Bases de datos del dispositivo móvil	Las hojas tienden a deteriorarse y pueden darse una pérdida en la legibilidad de los datos
Actualización de los datos en los sistemas de la ESPOL	Las hojas pasan a mano de otra persona que posteriormente los transcribe a el sistema de la ESPOL	Los datos son enviados vía HTTP hacia la aplicación en el servidor y estos automáticamente son actualizados en la base de datos de la ESPOL	Este paso intermedio de digitación es obviado, eliminando posibles errores y demoras en la trascripción de la información

ANEXO 5

Detalle de los costos de desarrollo e implementación

Descripción	Duración meses	Costo USD	Observación
Investigación de las nuevas tecnologías*	3	$350	En este periodo se investigó y consiguió la gran mayoría de materiales (HW/SW) para el desarrollo del sistema.
Pago de los investigadores y desarrolladores	8	$3840	El costo de los 2 desarrolladores por 8 meses a un costo de $12 USD por día. Estos costos incluyen el análisis, diseño e implementación.
Equipos computacionales	N/A	$3700	(2 estaciones de trabajo aproximadamente $1500 cada una), un punto de acceso inalámbrico, una Palm Tungsten C
Centro de Desarrollo	8	N/A	El sistema se desarrollo en el Centro de Servicios Informáticos de la ESPOL
Costos de Publicaciones y Papers especializados	N/A	$219	Computer Society del IEEE, Transaction on Software Engineer, Pervasive Computing, Distribute Systems Online, Computer Magazine
Costo de Software	N/A	$0	Eclipse IDE (open source), Sun App Servfer 8 Update 1 (freware for development), J2ME Wireless Toolkit 2.1 (freeware), IBM Websphere Device Developer 6.5 (trial), Mysql Server 3.5 (freeware), IBM UDB DB2 6.1 para Solaris (licenciado para la ESPOL), Poseidon for UML Enterprise Edition (trial)
* El costo de Investigación de las nuevas tecnologías está calculado en base al costo de conexión a Internet, impresiones y otros materiales de apoyo			

ANEXO 6

Manual de Usuario

Este manual esta enfocado para los usuarios de la aplicación cliente y explica los pasos a seguir para hacer un ingreso a bodega.

Inicio de la Aplicación

Para usar la aplicación en el dispositivo móvil el usuario debe tener una cuenta válida en el servicio de Directorio de la ESPOL. Una vez iniciada la aplicación, aparece una pantalla de inicio donde pide que se ingre el usuario y la clave como lo muestra la siguiente figura, al aceptar estos datos serán verificados en el servidor.

Figura 36.- Pantalla de inicio de la aplicación cliente

La casilla de verificación “Recuerdeme” en la pantalla de inicio, hace que los datos del usuario como el password sean almacenados localmente.

La primera vez que se necestie hacer uso de la red, la implementación de la máquina virtual de Java en el dispositivo móvil, preguntará si se está seguro de usar tiempo aire de red hasta que la aplicación BodegaPalm termine. Dependiendo de la configuración de red del dispositivo, la aplicación intentará conectarse cada vez que se necesite usar la red. Si se escoje NO la aplicación pasa a modo desconectado. En este caso presionamos SI como lo muestra la siguiente figura.

Figura 37.- Verificación de uso de la red inalámbrica

Si el usuario y la clave son correctos y si es la primera vez que el usuario inicia la apliación aparece la pantalla de las preferencias como lo muestra la figura 38. Las preferencias del usuario, muestran las configuraciones de la aplicación, que son de 2 tipos:

- Habilitar Actualización.- que permite la actualización direicta de los datos enel servidor

- Login Automático.- permite utilizar los datos del usuario almacenados enel dispositivo móvil, verificarlos en el servidor e ingresar a la aplicación automáticamente.

Si el sistema se encuentra en modo desconectado, se pueden usar las credenciales del usuario almacenadas en el dispositivo móvil, sólo hasta 24 horas despúes de su última actualización, caso contrario el sistema mostrará una excepción de usuario con credenciales expiradas.

Figura 38.- Pantalla de preferenias del usuario

Las preferencias del ususario pueden ser modificadas, posteriormente desde el menú principal de la aplicación. En el menú principal de la aplicación se muestran 5 opciones como lo muestra la figura 39.

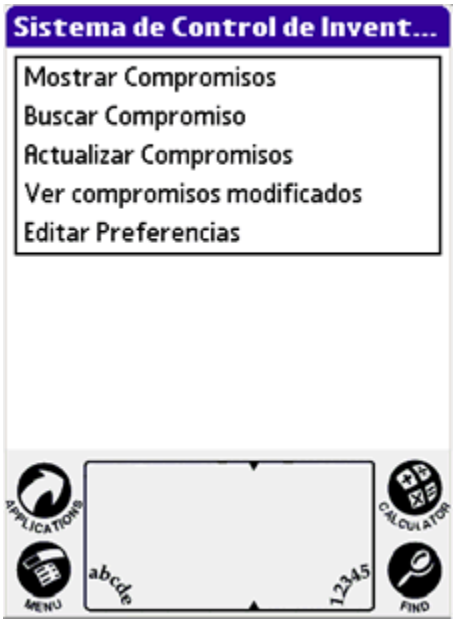


Figura 39.- Menú prinicpal de la aplicación cliente.

El detalle de las funciones del menú principal son:

- Mostrar Compromisos.- presenta la lista de compromisos de compra listos para ser ingresados.
- Buscar Compromisos.- permite hacer la busqueda de compromisos de compra por el número de compromiso.
- Actualizar Compromisos.-muestra la lista de compromisos que están totalmente ingresados y están listos para hacer actualizados
- Ver Compromisos Modificados.- presenta los compromisos que están modificados pero todavía no tienen el estado de completamente ingresados.
- Editar Preferencias.- muestra la pantalla de las preferencias, como lo muestra la figura 38.

La pantalla que muestra la lista de compromisos permite 2 tipos de modos de navegación, se puede navegar con los botones de navegación del dispositivo móvil o haciendo tap con el stylus. Para pasar a la siguiente página, hay que ubicarse en la ultima línea y presionar el boton de navegación inferior del dispositivo móvil y para regresar a la página anterior hay que ubicarse en la primera línea y oprimir el botón de navegación superior. La siguiente figura muestra la pantalla con los compromisos.

Figura 40.- Lista de compromisos en la aplicación cliente

Con un doble tap se puede seleccionar un compromiso de la lisa de compromisos, a continuación aparece la lista de pedidos del compromiso como lo muestra la figura 41, la cual tiene los mismos modos de navegación que la lista de compromisos.

Figura 41.- Lista de pedidos de la aplicación cliente

Cuando se selecciona un pedido, aparece la pantalla de edición de pedidos, que consiste en un grid y un contador de la cantidad de items por pedido que se van ingresando. Para ingresar un pedido presionamos el botón “Añadir” que nos llebva a la pantalla de ingreso de Items, la cual tiene los siguientes campos por ingresar:

- Número de Activo Fijo.- código asignado por el personal de bodega.
- Cantidad Despachada.- no puede ser mayor a la cantidad de items del pedido
- Marca.- Marca del item
- Modelo.- Modelo del item.
- Serie.- Serie del Item

- Ubicación- es un grupo de selección de ubicaciones que dependen del centro de costos del compromiso.
- Tipo de Custodio.- puede ser tipo cédula, ruc o pasaporte.
- Identificación del Custodio.- es el número de identificación, depende del tipo de custodio.
- Fecha de Ingreso.- por defecto es la fecha y hora actual.
- Observación.- campo de 40 caracteres para una pequeña observación.
- Características.- campo de 100 caracteres para escribir las características.

La figura 42 (a), muestra la pantalla de edición del pedido y la figura 42 (b muestra) la pantalla de ingreso de un item.

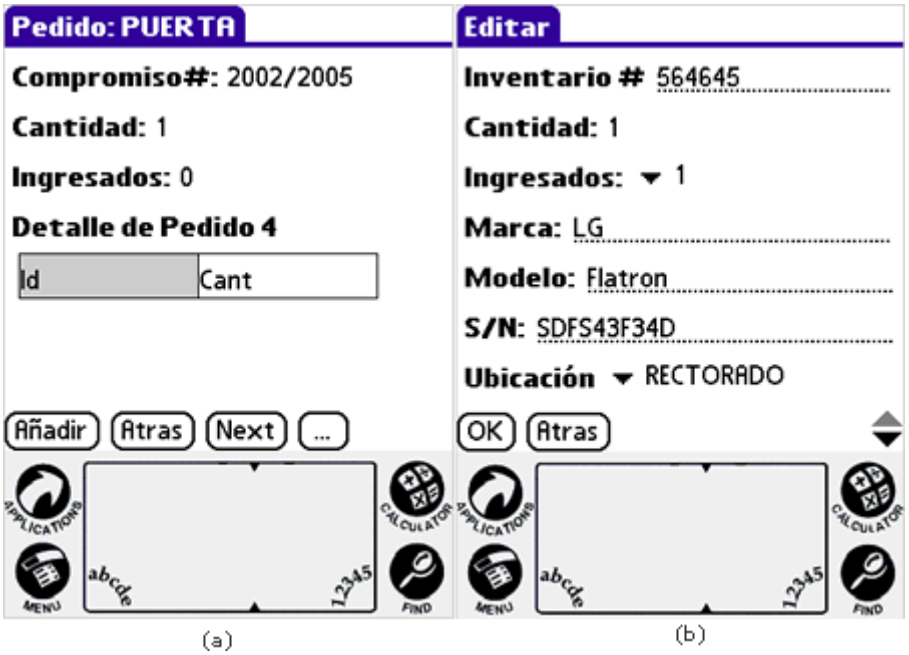


Figura 42.- (a) Pantalla de edición de un pedido. (b) Pantalla de ingreso de items, en la aplicación cliente.

Los cambios que se hagan en los pedidos deben ser guardados explícitamente, para que los compromisos alcancen el estado de completamente ingresados, todos lo pedidos deben estar completamente ingresados.

La pantalla para actualizar los compromisos muestran todos los compromisos que están completamente ingresados, esta es una lista de selección múltiple, se pueden actualizar todos los compromisos o solamente los seleccionados como lo muestra la figura 43.

Figura 43.- Actualización de compromisos en la aplicación cliente.

Después de actualizar los compromisos son borrados de las bases locales y se puede continuar con el ingreso de más compromisos.