"Desarrollo de una interfaz entre una base de datos y un dispositivo PDA"

Autor: Juan Antonio Huerta Hernández

Tesis presentada para obtener el título de:

Ing. En sistemas computacionales

Nombre del asesor:

Aldo Israel Sandoval Monroy

UNIVERSIDAD VASCO DE QUIROGA

FACULTAD DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

“DESARROLLO DE UNA INTERFAZ ENTRE UNA BASE DE DATOS Y UN DISPOSITIVO PDA ”

TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN SISTEMAS COMPUTACIONALES

PRESENTA

JUAN ANTONIO HUERTA HERNÁNDEZ

ASESOR

ALDO ISRAEL SANDOVAL MONROY

CLAVE: 16PSU0049F ACUERDO: LIC000808

MORELIA, MICHOACÁN ENERO-2010

Dedicatoria

A quienes siempre me han brindado todo su apoyo

Y confianza para salir adelante, mis padres. Para aquellos que son mi razón

para triunfar y ser un ejemplo, mis hermanos. Y por supuesto a todos mis

familiares que me apoyaron durante mis estudios.

ÍNDICE GENERAL

RESÚMEN

La información es un recurso vital para toda organización, el manejo eficaz de ésta, puede significar la diferencia entre el éxito o el fracaso para los proyectos que se emprendan dentro de un organismo que busca el crecimiento y el éxito. A lo largo de los años hemos encontrado diferentes maneras de almacenar la información, ya sea personal o de una empresa; el método más eficaz de almacenamiento ha sido a través de los sistemas informáticos, las computadoras juegan el papel más importante en este proceso ya que conforme avanza la tecnología obtenemos mayores ventajas al almacenar la información a través de un sistema computacional, algunas de estas ventajas son: rapidez, seguridad, integridad, accesos remotos entre muchas otras.

Las Tecnologías de la Información están cambiando la forma tradicional de hacer las cosas, utilizándolas eficientemente se pueden obtener ventajas competitivas, pero es preciso encontrar procedimientos acertados para mantenerlas en manera constante, así como disponer de cursos y recursos alternativos de acción para adaptarlas a las necesidades del momento, pues las ventajas no siempre son permanentes.

Al día de hoy, la tecnología inalámbrica se ha vuelto más común en el manejo de información, podemos tener el acceso y la manipulación total de los datos además es de gran utilidad para maximizar la eficiencia del tiempo, hoy en día un recurso muy valioso. Gracias a estas ventajas puede aumentar la productividad del personal de cualquier empresa y así lograr ser una organización más competitiva en un mercado en constante evolución.

En el presente proyecto de tesis se realiza el análisis y desarrollo de un sistema de almacenamiento de información para el control de almacén para la empresa Telecable de Tierra Caliente (en los sucesivo TTC), consta de dos partes, un software para computadora de escritorio y otro para dispositivos PDA del cual se podrán realizar transacciones con la base de datos del servidor vía WiFi a través del

uso de Web Services, tecnología reciente por la cual es posible realizar consultas y actualizaciones a bases de datos desde accesos remotos vía Web, entre otros beneficios. En este proyecto se explica y justifica el uso de cada una de las herramientas y tecnologías utilizadas para el desarrollo del sistema.

ABSTRACT

The information is a vital resource for all organization, the effective management of this, can represent the difference between success or failure for the projects that are undertaken within an organism that looks for the growth and the success. Throughout the years we have found different ways to store the information or personal or of a company; the most effective method of storage has been through the computer science systems, the computers play the most important role in this process since in agreement the technology advances we obtain majors advantages when storing the information through a computer system, some of this advantages are: rapidity, security, remote access and many others.

The Information Technologies are changing the traditional way to make the things, using them efficiently competitive advantages can be obtained, but it is essencial to find the correct procedures to maintain them in constant way, as well as to have courses and alternative action resources to adapt them to the needs of the moment, because the advantages not always are permanent.

Today, wireless technology has become more common in the information management, we can have data access and total data manipulation in addition it's very useful to maximize the time efficiency, a very valuable resource nowadays. Thanks to this advantages employees productivity of any company can increase and get to be a more competitive organization in a market on constant evolution.

In the present thesis project it is realised the analysis and development of an information storage system for the warehouse control for Telecable de Tierra Caliente company (hereinafter referred to as TTC), it consists of two parts, a desktop computer software and another one for PDA devices that will be able to make transactions with the server database via WiFi using Web Services, recent technology by which it is possible realise database consultations and updates from remote access via web,

among other benefits. In this project it is explained and justified the use of each tool and technology used for the system development.

PLANTEAMIENTO DEL PROBLEMA

A pesar de que se cuenta con un software para control de inventario, el personal de TTC actualmente no hace uso de él, por tal motivo se observa que existe un descontrol sobre el material que se utiliza para atender las ordenes de servicio así como el que se usa para el mantenimiento de la red de cable de cada ciudad en las que la empresa tiene presencia.

Para los directivos de TTC, la principal fuente de fuga de material está en el que se proporciona a los técnicos, por la razón antes mencionada, no hay buen control de entradas y salidas de éste. Dicha fuga de material significa para la empresa una pérdida económica importante puesto que ese equipo extraviado puede estarse utilizando para conexiones piratas lo cual afecta directamente en los ingresos de TTC.

Por las razones expuestas anteriormente, TTC necesita renovar la manera en que se lleva el control del material, al aplicar el sistema propuesto en este trabajo de tesis, estará elevando el nivel de calidad en los servicios que se ofrecen, pues no cabe duda que al querer ofrecer calidad se deben tomar en cuenta todos los aspectos en lo que a procesos de trabajo se refiere.

Se considera que la implantación de este proyecto provocará consecuencias positivas puesto que ayuda a mantener un mejor control de información que maneje la empresa en cuanto a material se refiere.

Por otra parte, con este sistema de transporte de datos se evita el uso de papelería, lo cual significa un ahorro en los recursos económicos de TTC. Además se iniciará un proceso de cultura por parte de los técnicos, o usuarios de este sistema, pues se tendrá que aprender a utilizar por lo menos de manera básica dispositivos como los PDA’s.

ANTECEDENTES

Antecedentes Telecable de Tierra Caliente

TTC inicia sus operaciones en la ciudad de Huetamo, Mich. en el año de 1977 bajo el nombre de Telecable de Huetamo S.A. de C.V. posteriormente en marzo de 1983 recibe formalmente por parte de la Secretaría de Comunicaciones y Transportes (SCT) el Título de Concesión correspondiente.

Desde 1977 hasta 1983 se luchó con denuedo para lograr una pequeña área de construcción de unos 8 Km de sistema que cubría solo el centro de Huetamo, ofreciendo 4 canales, que recibían y retransmitían, 2 en el cerro El Algodón (canales 2 y 4 de la Cd. De México) y 13 de Tele Azteca en el cerro de Turitzio, donde se utilizaban enormes bancos de baterías, que se cargaban por sistemas conversores de energía eólica y solar a energía eléctrica, el viaje en carro a Turitzio, a pie, o en burro, desde la población hasta la punta del cerro, tenían que hacerse dos veces por día y varias veces a la semana. El cuarto canal era generado localmente.

Debido al buen desempeño de la empresa ante la SCT, recibe por parte de esta, en el año de 1989 la autorización para extender sus servicios a la población de Cd. Altamirano, Gro. Posteriormente en 1992 ocurre lo mismo para extender su área de servicio a la ciudad de Arcelia, también en el estado de Guerrero.

Posteriormente en el año 2001 cambia la razón social de la empresa tomando el nombre con el que se le conoce actualmente; Telecable de Tierra Caliente S.A. de C.V.

Actualmente la empresa se encuentra ofreciendo sus servicios de Televisión por Cable e Internet por Cable en las poblaciones de Huetamo, Mich. así como en Cd. Altamirano, Coyuca, Arcelia, Cutzamala, Tlapehuala y Zirándaro Gro. Cubriendo con ello una zona importante de la región de Tierra Caliente.

Sistema actual de inventario

El sistema de control de los materiales que existen dentro del inventario de la empresa TTC, actualmente se está llevando a cabo mediante la utilización de un programa diseñado por el departamento de informática de dicha empresa.

La forma en que se está utilizando dicho programa es la siguiente:

Existe una persona encargada del almacén que tiene la tarea de mantener actualizada la base de datos del programa, es decir registra en el mismo, la cantidad de materiales que entran y salen. Por otra parte una de las secretarias de la empresa recibe materiales proporcionados directamente por el almacenista, para que ésta a su vez se lo vaya entregando a los técnicos a medida que se va necesitando dicho material.

Así mismo dicha persona lleva a cabo un control manual, anotando la cantidad de materiales que le suministra a cada técnico.

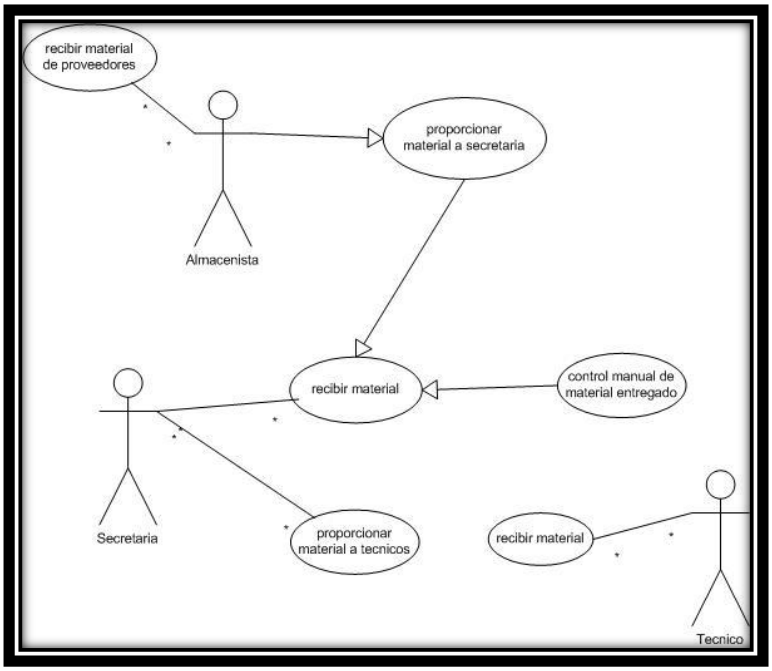


Fig. 1.0, Diagrama del sistema de inventario en TTC

Software actual para control de inventario

Descripción del software

El software ***CABLE-INVENTA*** está diseñado para llevar el control de entrada y salida de los materiales que conforman un inventario, con el uso adecuado del sistema se puede tener información rápida y confiable del estado que guardan todos y cada uno de los materiales introducidos al sistema, con lo cual da oportunidad de realizar a tiempo la compra oportuna de los materiales que se encuentren a punto de agotarse y así garantizar oportunamente la existencia continua de los mismos.

Asimismo el sistema cuenta con la opción para la generación de un reporte, el cual se puede imprimir en el momento que se desee.

Dicho software consiste en una base de datos realizada en Visual FoxPro Versión 9.0 el cual tiene la capacidad de controlar los materiales que ingresan y egresan del almacén.

A continuación se presentan algunas imágenes de las ventanas del programa CABLE-INVENTA.

Fig. 1.1, Ventana de Alta de Productos

Fig. 1.2, Ventana de Búsquedas

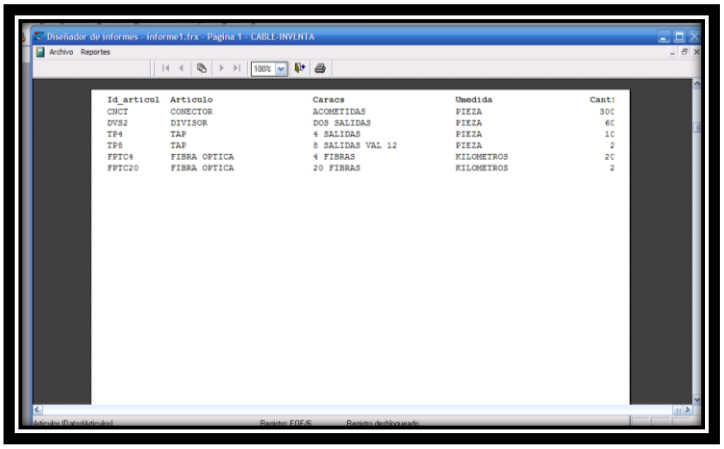


Fig. 1.3, Ventana de Reporte Generado

Experiencias de otras empresas

Abarrotera La Violeta

Empresa dedicada a la venta en mayoreo y menudeo de artículos de abarrotes, inicia operaciones en el año 1968 y hoy en día es uno de los negocios más grandes e importantes del estado.

Actualmente cuenta con una sucursal y una matriz, en donde también se encuentra la bodega, ambas en la Cd. de Morelia, y reparte productos por todo el estado y fuera de él.

Problemática

La empresa cuenta con agentes de venta quienes realizan visitas a todos sus clientes para recoger pedidos de productos, una vez levantado el pedido lo pasaban vía fax a las oficinas matriz, ahí una persona se encargaba de capturar en el sistema todos los pedidos que llegaban de los agentes; una vez capturados, en bodega se realizaba la consulta y se procedía al embarque de los pedidos.

Todo este proceso significaba pérdida de tiempo y en consecuencia de dinero, se requería de un sistema automatizado de tal manera que se ahorrara tiempo y que los embarques pudieran salir más rápido; por lo que se optó por un sistema en donde es posible la consulta y actualización en tiempo real de sus bases de datos a través de un dispositivo PDA desde puntos remotos.

Funcionamiento del Sistema

- Servidor: se encarga de la sincronización con los PDA, este funciona como administrador, este debe tener un puerto abierto para que sea posible tal sincronización.

- Módem -- Ruteador: por medio de este es posible la conexión de los dispositivos móviles; es importante mencionar que este MODEM debe tener un puerto abierto para que se pueda acceder desde un punto remoto a través de las PDA.

- Switch inalámbrico: este va conectado con el módem y es para que las pocket puedan acceder al sistema dentro de las oficinas de La Violeta a través de la tecnología wifi, esto lo hace direccionando al servidor por medio de la dirección ip; esta dirección debe ser fija, de esta manera se evitan errores en la conexión.

Para la conexión al sistema desde un punto remoto los usuarios utilizan la tecnología bluetooth en el PDA y en celular; este último dispositivo debe contar con el servicio de conexión a Internet.

Fig. 1.4, Máquina Servidor

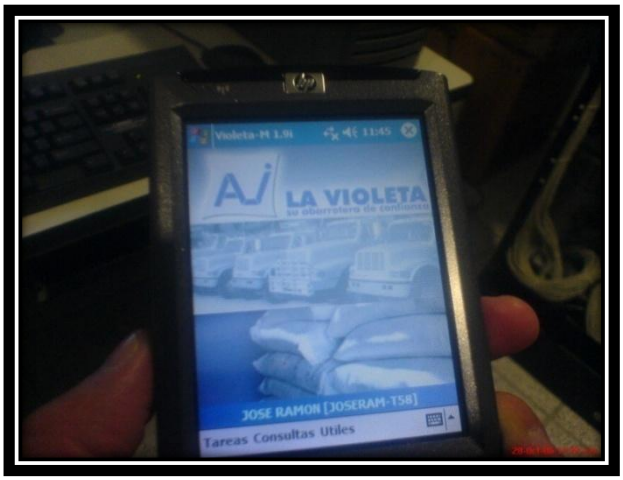


Fig. 1.5, Entrada al sistema desde un PDA

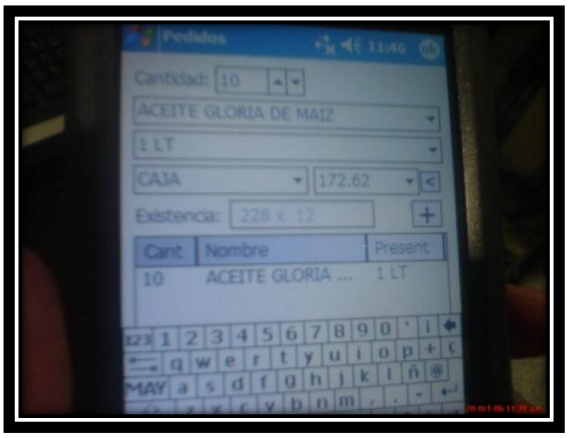


Fig. 1.6, Consulta de producto

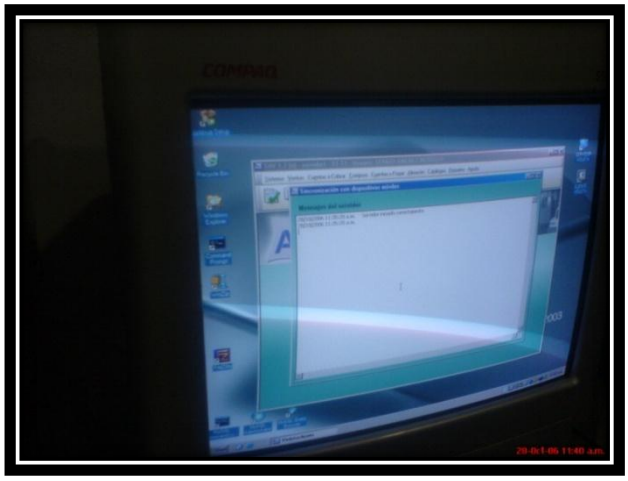


Fig. 1.7, Programa de sincronización en máquina servidor

Los usuarios que utilizan este sistema de actualización de base de datos han observado las siguientes ventajas:

- ***Rapidez***, con este sistema es posible la actualización en tiempo real de la base de datos, por lo tanto los pedidos salen de manera rápida.
- ***Eficacia***, pues ya que se cumplen los objetivos en los tiempos establecidos.
- ***Mejor imagen***, ya que el uso del PDA por parte de los distribuidores de La Violeta significa para el cliente que la empresa está a la vanguardia tecnológica.

- ***Seguridad***, con la implantación de esta tecnología se evita en un gran porcentaje los errores como cuentas del cliente, pedidos, o alguna otra información.
- ***Mejor servicio***, ya que se pueden dar servicios extras como saldos, costos, etcétera.

A continuación se presentan unas preguntas que se formularon a los usuarios el día 4 de julio de 2008:

¿La capacitación recibida fue suficiente para comprender el uso de este sistema?

- *Contestaron que sí, ésta duró 15 días aproximadamente; además expresaron que el uso del sistema es bastante sencillo y no les costó trabajo entender su uso, muchas de las personas no tenían conocimientos básicos de computación y aun así lo comprendieron. Hasta la persona de mayor edad (60 años aprox) comprendió el uso.*

  Cabe recalcar que la capacitación sólo se enfocó en el uso del sistema no en el uso del PDA. La mayoría de los usuarios no saben otras utilidades del dispositivo.

¿Ha habido un cambio significativo con este sistema, es decir, realmente ha sido útil?

- *Expresaron que en definitiva si; si no existiera este sistema seguirían muchos de los problemas que anteriormente tenían como eran los retrasos en los embarques, errores en los totales de las cuentas de los clientes, entre otros.*

Entre el personal de La Violeta existe alguien que se encarga de recibir los pedidos en su sistema y dar la orden para que salga ese embarque y opina lo siguiente con respecto a esta nueva tecnología:

- Con este sistema es posible mandar los embarques de manera pronta, son errores mínimos los que ocurren y se maneja un tiempo favorable.

- Por otro lado y muy importante, el ahorro que existe para la empresa es muy significativo, ya que con la implementación de este sistema se eliminan los "tiempos muertos", es decir, se deja de pagar tiempos extras a los maniobristas, ya que con el sistema anterior ellos tenían que esperar a que los pedidos fueran capturados.

La cantidad diaria de pedidos que se realizan por medio del dispositivo móvil son de 200 aproximadamente, es por esto que significa un gran ahorro en el tiempo para la distribución de los mismos, ya que estos se hacen en tiempo real.

OBJETIVOS

Objetivos Generales

Con el desarrollo de este proyecto se renueva la metodología para llevar el control de almacén de la empresa Telecable de Tierra Caliente, para lograrlo, se desarrolló una interfaz para un dispositivo PDA el cual a través de una red WiFi permita suministrar información a una base de datos. Esto se logrará instalando un switch inalámbrico en las oficinas de la empresa, así la información se podrá trasladar de manera local. Pero en un futuro se puede hacer a través de Internet para que sea la actualización en tiempo real ya que se contará con las bases para hacerlo, solo bastará contratar el servicio de Internet con una compañía de teléfonos celulares.

Objetivos Específicos

- Facilitar la administración de la información de la empresa, pues se eliminaría el manejo exhaustivo de la documentación por medio del uso de papel.
- Permitir un manejo eficiente sobre el equipo de trabajo utilizado, teniendo así un mayor control, de manera que evite, en lo posible, la pérdida de los materiales de trabajo.
- Lograr que la empresa logre mantenerse a la vanguardia tecnológica en cuanto al suministro de información en una base de datos.

ALCANCES Y LIMITACIONES

Alcances

- El sistema permite la transferencia de datos desde una base de datos en un servidor a un dispositivo PDA y viceversa.
- El software para escritorio permite la captura de ingresos y egresos de material así como modificaciones en los datos de éste (descripción, cantidad, ubicación). Además permite la generación de distintos reportes con información detallada de ingresos, egresos y movimientos realizados en el software del dispositivo.
- El software para el dispositivo PDA permite la captura de salidas de material (movimientos) así como consultas del mismo.

Limitaciones

- Para realizar la transferencia de datos entre la PDA y el servidor deberá efectuarse dentro del área de cobertura del ruteador inalámbrico ubicado en las oficinas de la empresa.
- La capacidad de almacenamiento del dispositivo no es muy grande por lo que se tiene que adquirir una tarjeta de almacenamiento externa en caso de ser necesario.

JUSTIFICACIÓN

TTC en su búsqueda de aumentar la calidad en sus servicios, así como aprovechar las ventajas que hoy en día nos ofrecen las tecnologías emergentes que día a día mejoran su desempeño en pro de los usuarios, se desarrolló un sistema que permita a personal técnico de la empresa actualizar la base de datos de inventario con la información que recabaron durante las horas de trabajo.

Al implementar este sistema que de manera sencilla, rápida y segura permita llevar un control de entradas y salidas de material, se podrá contar con un manejo eficiente sobre éste, pues indudablemente toda empresa cuenta con una serie de insumos, que son indispensables para poder ofrecer sus productos o servicios, dichos bienes, constituyen el patrimonio de dicha empresa y por lo tanto necesitan toda la atención para administrarlos y vigilarlos constantemente.

Análisis financiero de la implantación del sistema

Según directivos de TTC, los gastos mensuales por compra de material son aproximadamente $45,000.00, lo cual es suficiente para abastecer a las 3 plazas principales.

Con los datos proporcionados por TTC, se presenta un estudio comparativo sobre los costos de implementación del sistema propuesto en esta tesis contra el que se lleva actualmente en la empresa.

Tabla 1.0, Comparativa de procesos en los que se maneja el material, entre el sistema actual y el propuesto

PROCESO	SISTEMA ACTUAL	SISTEMA PROPUESTO
Control total y actualización del inventario de material	3 Hrs. ⇔ $2,250.00/mes	1 Hr. ⇔ $750.00/mes
Fuga de material del almacén por mal control de inventario	10 % ⇔ $1,500.00/mes	2 % ⇔ $300.00/mes

Con esta primera comparación, podemos realizar un estudio comparativo de la implantación del nuevo sistema con el que actualmente se lleva a cabo; tal como se muestra en la siguiente tabla.

Tabla 1.1 Comparativa de costos de implementación

<table>
<tr><th colspan="2">SISTEMA ACTUAL</th><th colspan="3">SISTEMA PROPUESTO</th></tr>
<tr><td>Costo pérdida de material</td><td>$1,500.00</td><td rowspan="4">Costo implementación</td><td>PDA (2)</td><td>$7,000.00</td></tr>
<tr><td>Pérdida horas de trabajo</td><td>$2,250.00</td><td>Ruteador inalámbrico</td><td>$500.00</td></tr>
<tr><td>Total / mes</td><td>$3,750.00</td><td>Tiempo de desarrollo</td><td>$25,000.00</td></tr>
<tr><td rowspan="5">⇩
12 MESES</td><td rowspan="5"></td><td>Licencias de herramientas de desarrollo</td><td>$3,175.00</td></tr>
<tr><td colspan="2">Inversión inicial</td><td colspan="2">$35,675.00</td></tr>
<tr><td colspan="2">Costo pérdida de material</td><td colspan="2">$300.00</td></tr>
<tr><td colspan="2">Pérdida horas de trabajo</td><td colspan="2">$750.00</td></tr>
<tr><td colspan="2">Total / mes</td><td colspan="2">$1,050.00</td></tr>
<tr><td colspan="4">⇩
12 MESES</td></tr>
<tr><td>Gastos comunes</td><td>$45,000.00</td><td colspan="2">Gastos comunes</td><td>$12,600.00</td></tr>
</table>

Como se puede observar en la tabla, el costo inicial de la implementación del nuevo sistema de inventario es menor a los gastos que se tienen al año por fuga de material y horas de trabajo con el uso del sistema actual, es decir, en menos de 12 meses se podría recuperar la inversión inicial.

En lo que respecta a los gastos comunes, haciendo una comparación de los costos mensuales por el uso del sistema actual contra el propuesto, se podría tener un

ahorro aproximado del 70% al año en cada plaza, por lo que es totalmente justificable la inversión para la adquisición del nuevo software.

El desarrollo del proyecto se considera de suma importancia para TTC, puesto que la implementación de esta tecnología permitirá a esta mantener la vanguardia tecnológica y ser líder en el mercado de las telecomunicaciones no solo en la región de Tierra Caliente sino en toda la República Mexicana.

Capítulo 1

Introducción

Dentro de este capítulo se presenta un pequeño resúmen de qué información se exhibe en cada uno de los capítulos que conforman este proyecto de tesis, desde el marco teórico hasta las conclusiones y trabajo futuro.

Capítulo 1

Introducción

El capítulo 2, el Marco Teórico que conforma al presente proyecto incluye el estudio y análisis de la información que se requiere para empezar con el desarrollo del sistema de almacenamiento. En primer lugar tenemos el modelado del sistema, en el que se presenta información detallada de cada una de as funcionalidades tanto de la aplicación desarrollada para escritorio como las de la aplicación para el dispositivo móvil, así como las especificaciones de casos de uso, diagramas uml, de secuencia y de clases.

Después se expone una breve descripción así como una comparación de las alternativas de dispositivos móviles, las cuales son PALM y PocketPC, principales plataformas de PDA's, fue de suma importancia para este proyecto analizar cada uno de los pros y contras de estos equipos para elegir el adecuado.

Otro análisis presentado en el Marco Teórico es el de las herramientas para desarrollo en PDA's, en el cual se dan a conocer algunas herramientas que hoy en día tenemos al alcance para el desarrollo de aplicaciones para dispositivos PDA. Dichas herramientas fueron las que se tomaron como alternativas para el desarrollo de este proyecto.

En cuanto a bases de datos se refiere, se presenta un análisis de las alternativas de bases de datos; actualmente existen dos principales sistemas de gestión, Oracle y SQL, el primero, muy robusto y seguro pero muy caro, el segundo, con versiones gratuitas, es el más utilizado aunque no presenta la seguridad que ofrece Oracle. En esta parte del capítulo 2 se analizan las características de estos dos gestores así como sus ventajas y desventajas.

Como última parte de ese capítulo, se describen las tecnologías existentes para la conexión de un dispositivo PDA con un servidor o PC, con el objetivo de realizar transferencias de datos. En este tipo de conexiones debemos contar con rapidez y seguridad a la hora de transferir datos, por tal motivo es importante analizar cuidadosamente las ventajas y desventajas que presentan estas alternativas de conexión.

En el capítulo 3, titulado “Herramientas a Utilizar en este Proyecto”, como su nombre lo indica, se presentan todas las herramientas que se utilizaron para el desarrollo del sistema.

Dentro del capítulo 4 se explica la arquitectura del sistema que se desarrolló con el fin de transferir datos del dispositivo móvil, así como una descripción de las capas en las que se divide dicho sistema.

En el capítulo de Resultados se presentan las pruebas a las que fue sometido el sistema una vez desarrollado, las pruebas se realizaron tanto al software de escritorio como el del PDA.

Una vez finalizada la investigación y desarrollo del sistema, en el último capítulo se exponen las conclusiones de dicho trabajo así como una mención de trabajo futuro que es posible realizar partiendo de la base de este proyecto.

Capítulo 2

Marco Teórico

Dentro de este capítulo se presenta toda la información recabada la cual fue base para determinar la metodología a utilizar para el desarrollo de este proyecto. En él se expone el modelado del sistema, análisis de las herramientas, tecnologías y dispositivos disponibles para el desarrollo de la interfaz entre el dispositivo móvil y la base de datos.

Capítulo 2 Marco Teórico

2.0 Modelado del Sistema

2.1 Casos de Uso Sistema Control Material TTC

A continuación se presentan los casos de uso y la especificación de cada uno de los módulos de la aplicación de escritorio.

2.1.0 Catálogo de Proveedores

- Descripción

A través de este módulo se dan de alta los datos generales de los diferentes proveedores de material con los que se surte la empresa, esto con el fin de asignar un Id al proveedor para ir generando un registro de compras o adquisiciones de material. Además se puede eliminar o modificar los datos del proveedor.

- Actores que intervienen

Almacenista: es el usuario del sistema y el que consigue los datos de los proveedores para su captura en el sistema.

- Flujo de Eventos

Flujo Básico

1.- El almacenista ejecuta el sistema, hace login y entra a la opción Catálogo de Proveedores.

2.- El sistema despliega la ventana de proveedores en donde se encuentran las opciones: alta proveedor, modificar y eliminar.

3.- Para dar de alta un proveedor se debe ingresar los siguientes datos: nombre, domicilio, contacto y teléfono y hacer clic en “Agregar” para que los datos queden almacenados.

4.- Para modificar los datos de un proveedor abrir la pestaña “Modificar”, seleccionar de la lista desplegable el nombre del proveedor a modificar y sus datos se mostrarán en pantalla, para modificarlos solo hay que ubicar el cursor en el recuadro correspondiente y realizar las modificaciones necesarias y después hacer clic en “Modificar”.

5.- Para eliminar los datos de un proveedor de la base de datos hay que ir a la pestaña “Eliminar”, seleccionar el proveedor de la lista desplegable y hacer clic en “Eliminar”.

- Precondiciones

El almacenista deberá estar dado de alta como usuario del sistema, este puede fungir como administrador o usuario normal.

- Pos condiciones

Los movimientos que realice el almacenista, ya sea dar de alta, modificar o eliminar datos de un proveedor, quedarán registrados en la base de datos.

- Diagrama

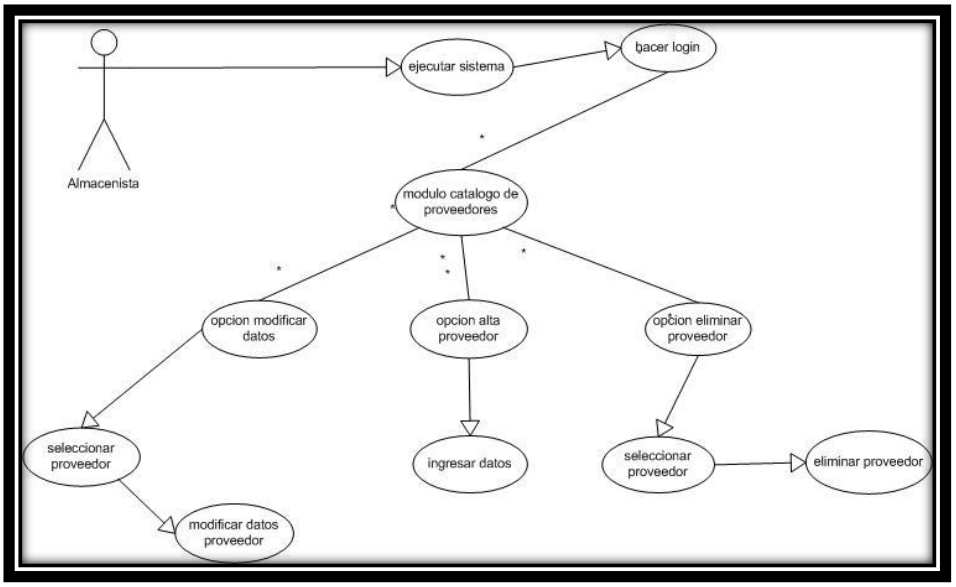


Fig. 2.0, Diagrama UML Catálogo de Proveedores

2.1.1 Catálogo de Artículos

- Descripción

Este caso de uso es para dar de alta los diferentes tipos de artículos (material) que se manejan en la empresa. El material se divide en tipos para mantener mejor organizado su registro a la base de datos, al dar de alta un tipo de artículo se le asigna una clave para que todo material tenga su Id de la clave del tipo de artículo. En este módulo además de dar de alta también podemos modificar la descripción que se introduce de cada tipo de artículo.

- Actores que intervienen

Almacenista: como administrador de sistema y encargado de catalogar los diferentes artículos en “tipos”.

- Flujo de Eventos

Flujo Básico

1.- El almacenista ejecuta el sistema, hace login y entra a la opción Catálogo de Artículos.

2.- El sistema despliega la ventana correspondiente, en donde contiene dos pestañas: Alta y Modificar, la primera se muestra por omisión.

3.- Para dar de alta el tipo de artículo se debe de ingresar el nombre en el recuadro correspondiente a “Nuevo Tipo de Artículo” y hacer clic en el botón Alta.

4.- Si se desea ingresar una breve descripción del tipo de artículo dado de alta, o bien, modificarla tenemos que abrir la pestaña de “Modificar”, seleccionar el tipo de artículo e ingresar su descripción en el recuadro correspondiente y por último hacer clic en Modificar.

- Precondiciones
  - El almacenista deberá estar dado de alta como usuario del sistema.
- Pos condiciones

- Las Altas y Modificaciones realizadas quedarán registradas en la base de datos.
- Con los tipos de artículos registrados se puede realizar el alta de material.

- Diagrama

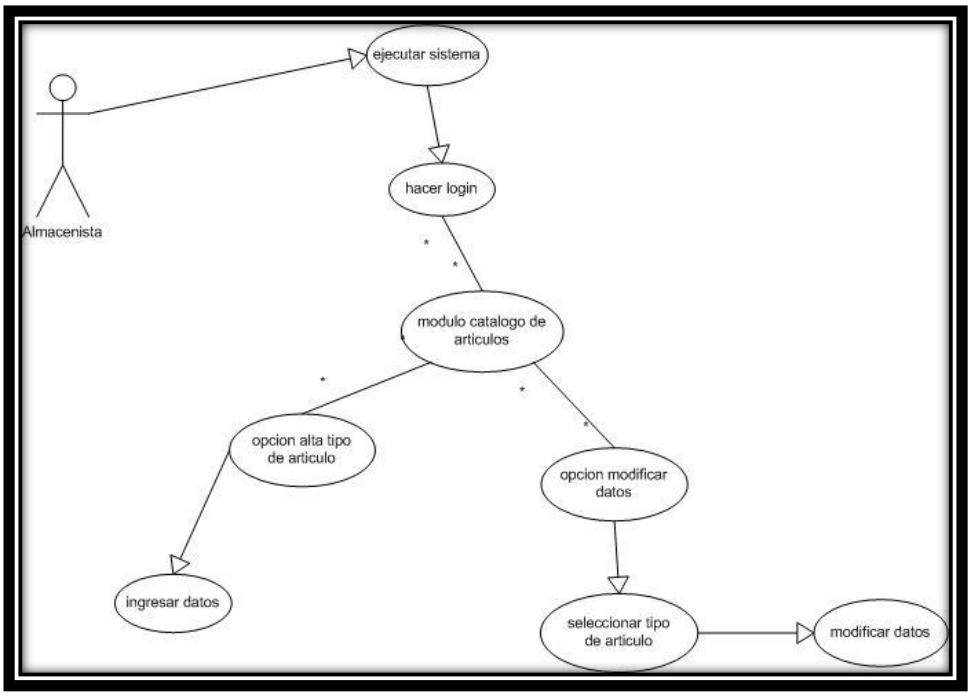


Fig. 2.1, Diagrama UML Catálogo de Artículos

2.1.2 Alta de Artículo

- Descripción

Complemento del catálogo de artículos, en este módulo se ingresa el nombre del material y la unidad (pieza o metro); además de seleccionar el tipo de artículo.

- Actores que intervienen

Almacenista

- Flujo de Eventos

Flujo Básico

1.- El almacenista ejecuta el sistema y del menú principal entra al módulo de "Alta Artículo" y se despliega su ventana correspondiente.

2.- Primero seleccionamos de la lista desplegable el tipo al que pertenece el artículo que se va dar de alta.

3.- Una vez seleccionado el tipo, ingresar el nombre del artículo en el recuadro correspondiente.

4.- Seleccionar de la lista desplegable la unidad que le corresponde al materia, por ejemplo, si se trata de cable la unidad sería metros.

5.- La cantidad será por omisión 0 ya que en este momento sólo damos de alta el artículo, es decir, todavía no hay ingreso a almacén.

- Precondiciones
  - El almacenista deberá estar dado de alta como usuario del sistema para poder ingresar.
  - El tipo que corresponde al artículo que se va dar de alta debe de estar previamente registrado en la base de datos.

- Pos condiciones
- Una vez ingresados el tipo de artículo y el nombre del mismo se podrá proceder a la captura de ingreso de material, la cual se realiza ingresando los datos de las facturas de compra de dicho material.

- Diagrama

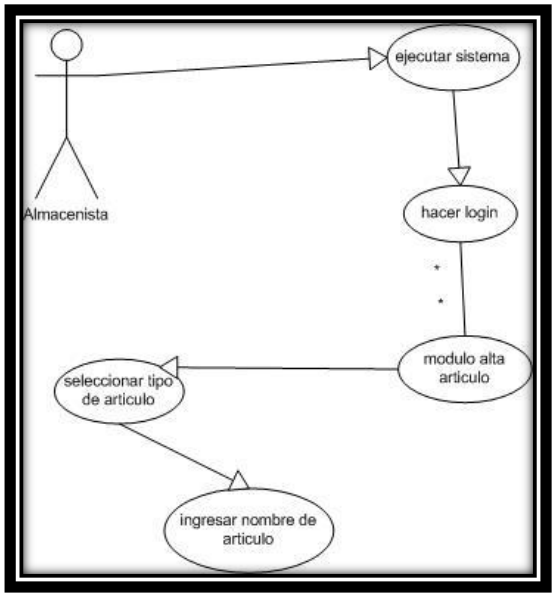


Fig. 2.2, Diagrama UML Alta Artículo

2.1.3 Ingreso Material

- Descripción

Este caso de uso es parte esencial del sistema, a través de este módulo se capturan los datos de las facturas de compra de material, de esta manera, al momento que damos de alta la cantidad de material que entra a almacén, llevamos un registro de las facturas que adquiere la empresa.

- Actores que intervienen
  - Almacenista: recibe el material que envían los proveedores.
  - Personal de Contabilidad: son quienes van a facilitar al almacenista una copia de las facturas de la compra del material.

- Flujo de Eventos

Flujo Básico

1.- El almacenista ejecuta el sistema, hace login y entra al módulo “Ingreso Material”.

2.- Dentro de la ventana correspondiente al módulo, se debe ingresar la fecha de la factura que se va dar de alta, por omisión el sistema despliega la fecha actual, pero es posible modificarla.

3.- Seleccionar el nombre del proveedor de la lista desplegable, es decir, de quien expide la factura.

4.- Ahora se debe seleccionar el tipo de artículo del cual se va a registrar su ingreso.

5.- Una vez seleccionado el tipo de artículo, en una lista desplegable se cargarán los nombres de todos los artículos que tenemos registrados en la base de datos que pertenezcan al tipo de artículo seleccionado, de los cuales seleccionamos el correspondiente.

6.- Al seleccionar el artículo, en el recuadro de “Precio” se carga el precio que tenemos registrado en la base de datos, de ser necesario, esa cantidad la podemos modificar, ingresamos la cantidad de artículo y hacemos clic en el botón Cargar”.

7.- Observaremos que en el recuadro blanco ubicado en la parte inferior de la ventana se carga una fila con los datos del artículo y cantidad que acabamos de ingresar, pero hasta este momento no se ha registrado el movimiento en la base de datos; lo que muestra el recuadro es un resumen meramente informativo sobre lo que va ingresar al almacén.

8.- Si deseamos ingresar más material (cargado en la misma factura), debemos regresar al paso 4, es decir, escoger el tipo de artículo, el artículo y la cantidad. Conforme se vayan cargando los artículos se va ir llenando el recuadro informativo.

9.- Cuando termine de cargar todos los artículos de la factura hacemos clic en “Guardar”, el sistema muestra un recuadro con el total del valor de esa factura, de esta manera los artículos se sumarán a los registros de la base de datos, es decir, quedará actualizado el stock.

Flujo Alternativo

1.- En el paso 7, al cargar los artículos en el recuadro, si hubo algún error, haciendo doble clic sobre la fila no deseada se borrará del recuadro y habrá que seleccionar el artículo o cantidad correctos.

2.- Al terminar de cargar cada artículo, lo que se muestra en el recuadro informativo es lo siguiente: nombre del artículo, cantidad y costo del artículo; lo que se hará en la base de datos es sumar esta cantidad de cada artículo a la ya existente y se actualizará el precio en caso de que haya sido modificado.

- Precondiciones

- El almacenista deberá estar dado de alta como usuario del sistema para poder ingresar.
- El tipo de artículo correspondiente a cada material que va ingresar debe estar previamente cargado en el catálogo de artículos.
- El nombre de los artículos deben estar previamente dados de alta.

- Pos condiciones

- Cuando se realiza un ingreso de material, en la base de datos además de actualizar el stock de material, quedan registrados los datos de la factura como lo es: proveedor, número de factura y fecha.

- Diagrama

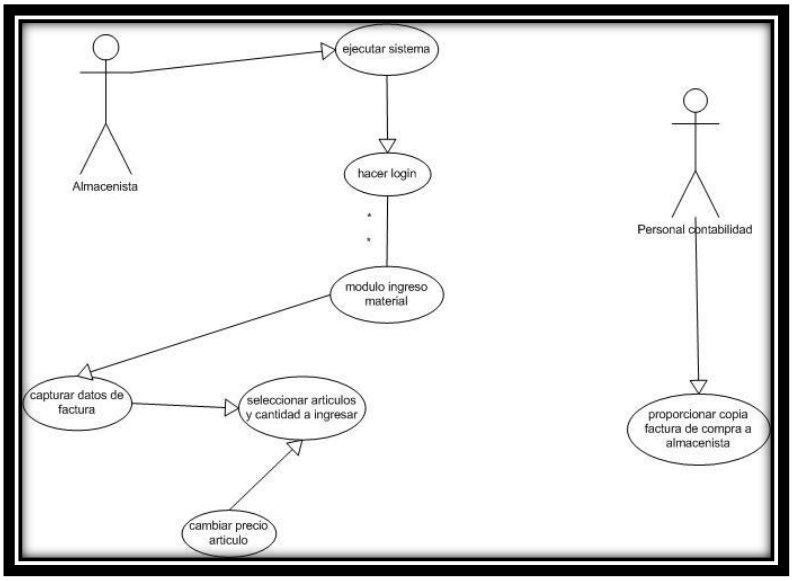


Fig. 2.3, Diagrama UML Ingreso Material

2.1.4 Captura de Salidas

- Descripción

Este caso de uso es para registrar los egresos de material del almacén, los datos capturados nos permiten crear reportes y mantener un historial de salidas.

- Actores que intervienen
  - Almacenista: autorizado para ingresar al sistema y es el que proporciona al técnico de material además de registrarlo en el sistema
  - Personal del Área Técnica: recibe el material de trabajo por parte del almacenista.

- Flujo de Eventos

Flujo Básico

1.- El almacenista ejecuta el sistema, hace login y entra al módulo “Salidas” del menú principal.

2.- Una vez que se despliega la ventana correspondiente debemos ingresar la fecha de salida, por omisión el sistema muestra la fecha actual pero la podemos modificar de ser necesario.

3.- De la lista correspondiente a Empleado, seleccionar el nombre del empleado o técnico que solicita y recibe el material.

4.- A continuación debemos escoger de una lista desplegable el tipo de artículo correspondiente al que será entregado.

5.- Una vez seleccionado el tipo de artículo, en el menú desplegable de Artículo se cargarán todos los artículos que correspondan al tipo elegido, hay que seleccionar el requerido.

6.- Ingresar la cantidad que fue solicitada de ese artículo.

7.- Hacer clic en el botón Cargar, al hacerlo, se cargará en un recuadro blanco ubicado en la parte inferior de la ventana, un resumen informativo del artículo y cantidad que va a egresar del almacén.

8.- En caso de que se haya solicitado más de un solo artículo debemos regresar al paso 4.

Flujo Alternativo

1.- En el paso 3 si el nombre del técnico o empleado que recibe el material no aparece en la lista es porque no ha sido dado de alta como “Empleado” lo cual realizamos a través del módulo “Alta Empleado”.

2.- En el paso 4, si el tipo de artículo no aparece en la lista hay que agregarlo a través del módulo “Catálogo de Artículos”, así mismo ocurre en el paso 5 con los artículos.

3.- En el paso 6, si la cantidad de artículo que fue solicitada es mayor a la existencia registrada en la base de datos, el sistema despliega una ventana de advertencia y nos indica la cantidad existente y deberemos modificar la cantidad solicitada.

4.- En el paso 7, al cargar los artículos, si hubo algún error podemos corregirlo, lo que se tiene que hacer es eliminar la fila del recuadro informativo haciendo doble clic sobre ella y habrá que cargar de nuevo el artículo.

- Precondiciones

- El almacenista deberá estar dado de alta como usuario del sistema para poder ingresar.

- Pos condiciones

- Al registrar una salida, en la base de datos quedan almacenados varios datos: fecha, empleado, artículos solicitados y entregados así como la cantidad de cada uno de ellos que egresan del almacén. Con ello es posible mantener un historial de las salidas y generar reportes.

- Diagrama

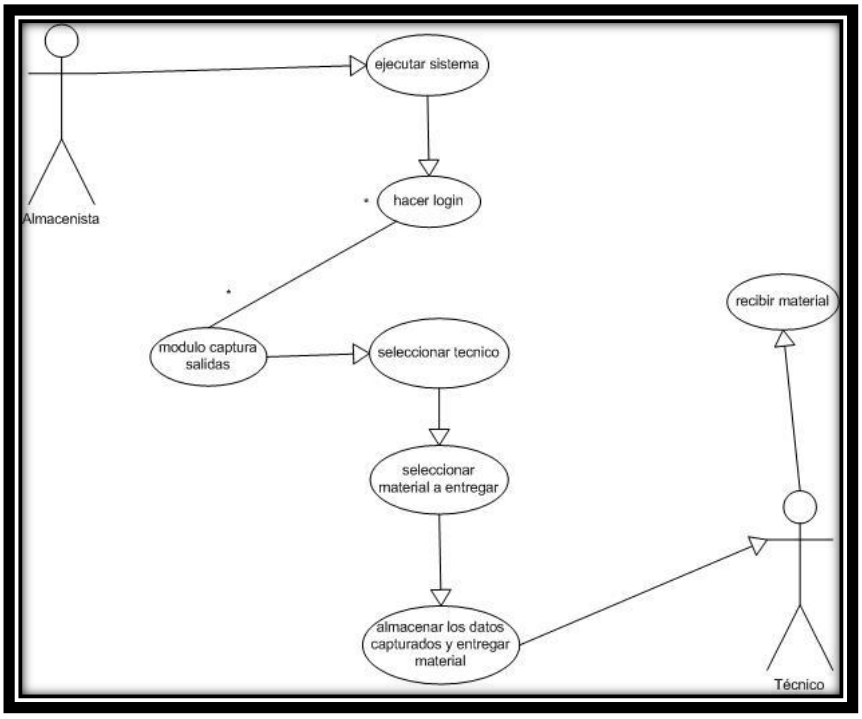


Fig. 2.4, Diagrama UML Captura de Salidas

2.1.5 Modificaciones

- Descripción

Caso de uso para el módulo Modificaciones, como su nombre lo dice, aquí podemos realizar las modificaciones a los datos de los artículos registrados en la

base de datos, lo que podemos modificar es: el nombre del artículo, cantidad y ubicación.

- Actores que intervienen
  - Almacenista: autorizado para ingresar al sistema y realizar las modificaciones.

- Flujo de Eventos

Flujo Básico

1.- El almacenista ejecuta el sistema, hace login y selecciona el botón Modificaciones del menú principal.

2.- En la ventana correspondiente al módulo debemos ingresar el nombre o parte de este y después hacer clic en el botón Buscar.

3.- En el recuadro central se arrojarán los resultados de la búsqueda, en caso de no tener registro alguno con los datos proporcionados el sistema lanza un aviso.

4.- Para realizar algún movimiento en los datos lo que debemos hacer es clic sobre la fila del artículo a modificar en el recuadro en donde se cargan los resultados de la búsqueda.

5.- Una vez que hacemos clic sobre la fila del artículo notaremos que se cargan en cajas de texto los datos correspondientes a dicho artículo, en estas cajas de texto es donde podremos realizar las modificaciones.

6.- Si queremos modificar el nombre del artículo debemos ubicar el cursor en el recuadro correspondiente a Artículo y realizar la modificación; para modificar la cantidad, ubicarnos en el recuadro correspondiente y realizar los cambios, así mismo para la ubicación del artículo.

7.- Una vez finalizados los movimientos realizados del artículo, para que estos tengan efecto en la base de datos hay que hacer clic en el botón Modificar.

Flujo Alternativo

1.- En el paso 2 si no ingresamos ningún nombre y hacemos clic en Buscar se cargan en el recuadro todos los artículos registrados.

- Precondiciones
  - El almacenista deberá estar dado de alta como usuario del sistema para poder ingresar y realizar la modificación.

- Diagrama

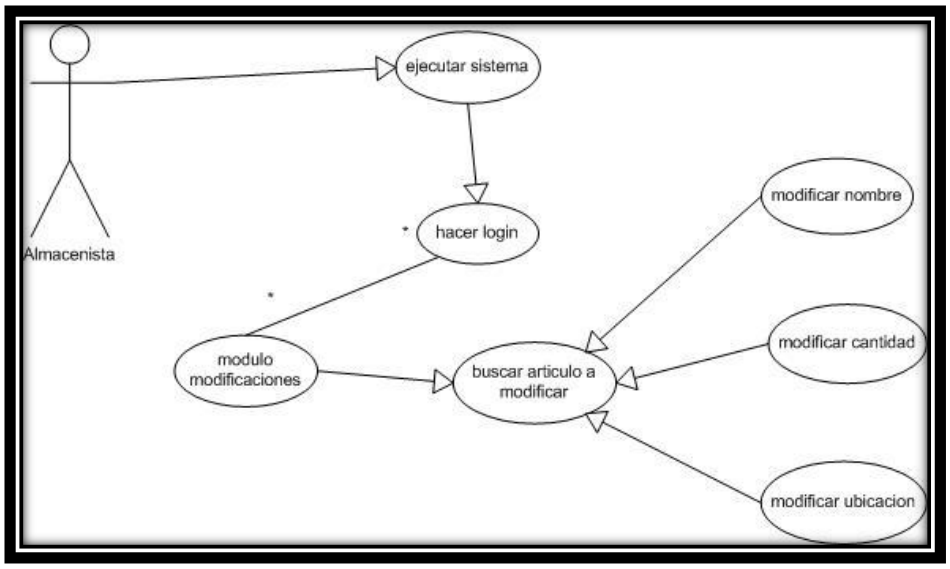


Fig. 2.5, Diagrama UML Modificaciones

2.1.6 Reportes

- Descripción

Como su nombre lo indica, el módulo de Reportes nos servirá para generar listados de informes sobre todos los movimientos que se llevan a cabo en el almacén y que son registrados en el sistema, como lo son: ingresos, salidas y también de existencia de material.

- Actores que intervienen
  - Almacenista: autorizado para ingresar al sistema y genera los reportes para entregar a los directivos o jefe inmediato.

- Flujo de Eventos

Flujo Básico

1.- El almacenista ejecuta el sistema, hace login y entra al módulo de Reportes.

2.- Dentro del módulo de reportes se encuentran cinco pestañas o submódulos que corresponden a distintos tipos de reportes, estos son: entradas almacén, stock, compras detallado, salidas y salidas PDA.

3.- El reporte de entradas almacén nos muestra una lista de los proveedores a quienes se les ha comprado material en un determinado periodo, para crear el reporte lo que debemos hacer ingresar un periodo, es decir, una fecha inicial y final y hacer clic en Generar para que el sistema genere el reporte.

4.- Para generar el reporte de "stock" habrá que abrir la pestaña correspondiente, dentro de este submódulo, tenemos otras opciones de reporte: stock mínimo, agotados, todos o por tipo de artículo.

5.- Para generar el reporte de "Stock Mínimo" solo tenemos que hacer clic en el botón del mismo nombre; en dicho reporte se muestra un listado de todo artículo cuya existencia sea mínima, es decir, se lista todo artículo en el que su existencia sea 20 ó menos.

6.- Si queremos generar el reporte de "Agotados", hacer clic en el botón que le corresponde, aquí se lista todo artículo cuya existencia sea 0.

7.- En el reporte de "Todos" se genera un listado de todo el material registrado en la base de datos, sin importar la cantidad de existencia; para generarlo solo hay que hacer clic en el botón del mismo nombre.

8.- Para generar el reporte por "Tipo de Artículo" hay que seleccionar el tipo de artículo del la lista despegable y hacer clic en el botón Generar; en este listado se muestran los artículos que correspondan al tipo elegido sin importar la cantidad en existencia.

9.- En el submódulo "Compras Detallado" se genera un reporte con un informe detallado de compras; para crearlo hay que ingresar un rango de fechas y hacer clic en el botón Generar.

10.- Si queremos crear un reporte de “Salidas” debemos ingresar al submódulo correspondiente, aquí tenemos dos opciones de generar el reporte, puede ser por fecha y además por área. Para generarlo por fecha solo ingresamos el rango de estas y hacemos clic en Generar, si queremos que el reporte sea por área, además de ingresar el periodo debemos elegir la opción “por área” y seleccionar la de nuestro interés del menú despegable y por último hacer clic en Generar.

11.- El reporte “Salidas PDA” nos muestra un listado de las salidas de material registradas por personal técnico en los dispositivos móviles, para generarlo hay que ingresar al submódulo correspondiente, ingresar un rango de fechas y hacer clic en Generar.

Flujo Alternativo

1.- En el paso 3, los datos que se muestran en el reporte son: proveedor, número de factura, fecha de la factura y el total en pesos de dicha factura.

2.- En el paso 4, la información mostrada en ese tipo de reporte es la siguiente: clave, nombre de artículo, unidad, existencia, descripción y lugar de resguardo.

3.- Paso 9, la información que se despliega en este reporte es: clave y nombre del proveedor, número y fecha de la factura, artículo, cantidad, precio unitario, subtotal y total neto del valor de cada factura.

4.- Paso 10, los datos mostrados en ese reporte son: nombre del empleado a quien se le entrega el material, área a la que pertenece, artículo y cantidad entregada, fecha de entrega y el número de salida.

5.- En el paso 11, la información que se muestra en ese reporte es: id de salida, técnico que utiliza el artículo y por consecuencia sale de almacén, artículo, cantidad, fecha, tipo de movimiento realizado (puede ser salida o traslado), detalle del movimiento y id del contrato (este id es el número de orden de servicio en donde es utilizado el material).

6.- Si el sistema no encuentra datos para generar alguno de los reportes con los criterios ingresados se lanza una ventana de aviso.

- Precondiciones
  - El almacenista deberá estar dado de alta como usuario del sistema para poder ingresar y generar los reportes.
  - Para generar un reporte actualizado de Salidas PDA, se debe de verificar que los movimientos registrados en los dispositivos móviles sean trasladados a la base de datos del servidor a través del web service; este movimiento lo realiza el almacenista.

- Pos condiciones
  - Cada uno de los reportes se puede imprimir haciendo clic en el botón del mismo nombre que corresponda a cada submódulo.

- Diagrama

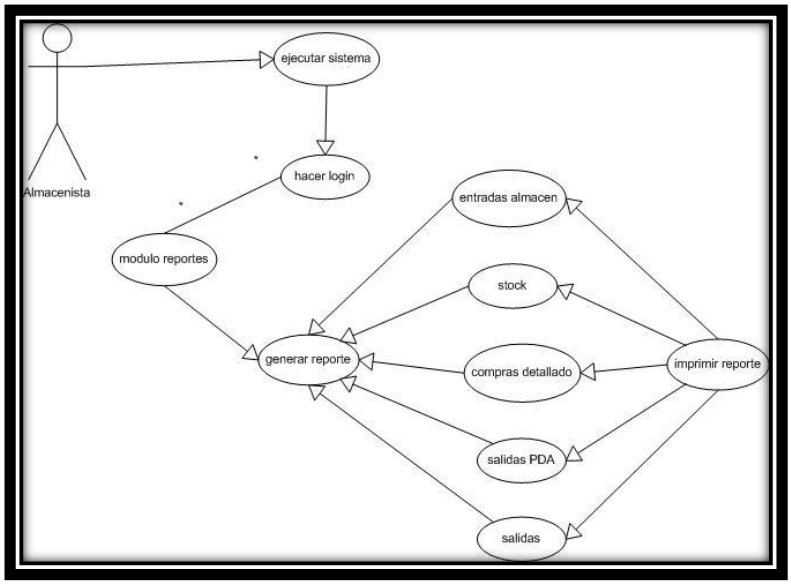


Fig. 2.6, Diagrama UML Reportes

2.2 Diagramas de Secuencia Sistema Control Material TTC

A continuación se presentan diagramas de secuencia correspondientes los procesos de ingreso de material y creación de reportes de la aplicación de escritorio.

En la siguiente figura la secuencia que se sigue en el software para el ingreso de material a la base de datos del servidor.

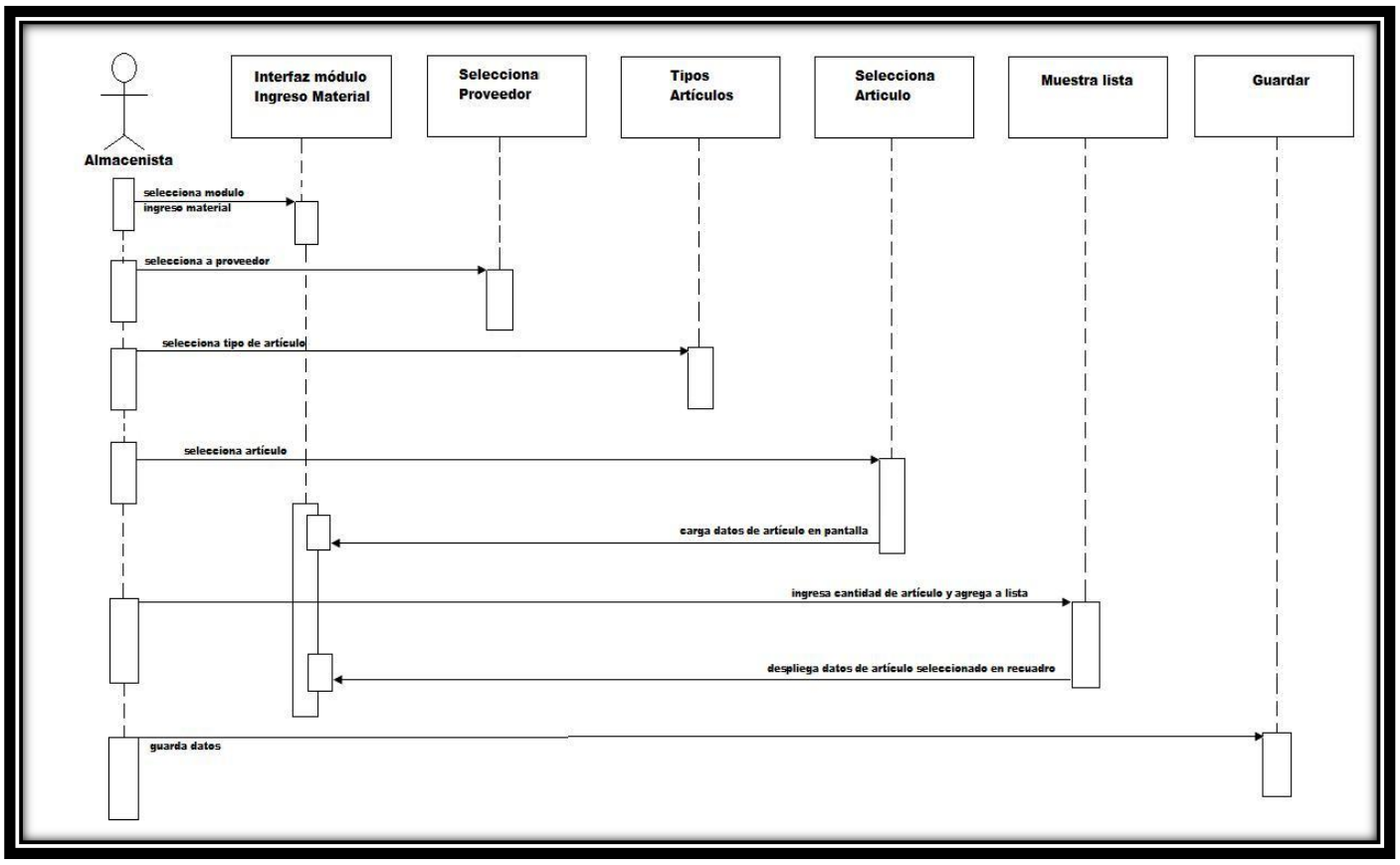


Fig. 2.7, Diagrama de secuencia de módulo Ingreso Material

En dicha figura se puede observar que para este proceso, el actor que participa es el almacenista (o administrador del sistema) además de la función de cada clase que conforma al módulo.

El diagrama de secuencua para mostrar la interacción de los objetos del módulo de Reportes es el siguiente:

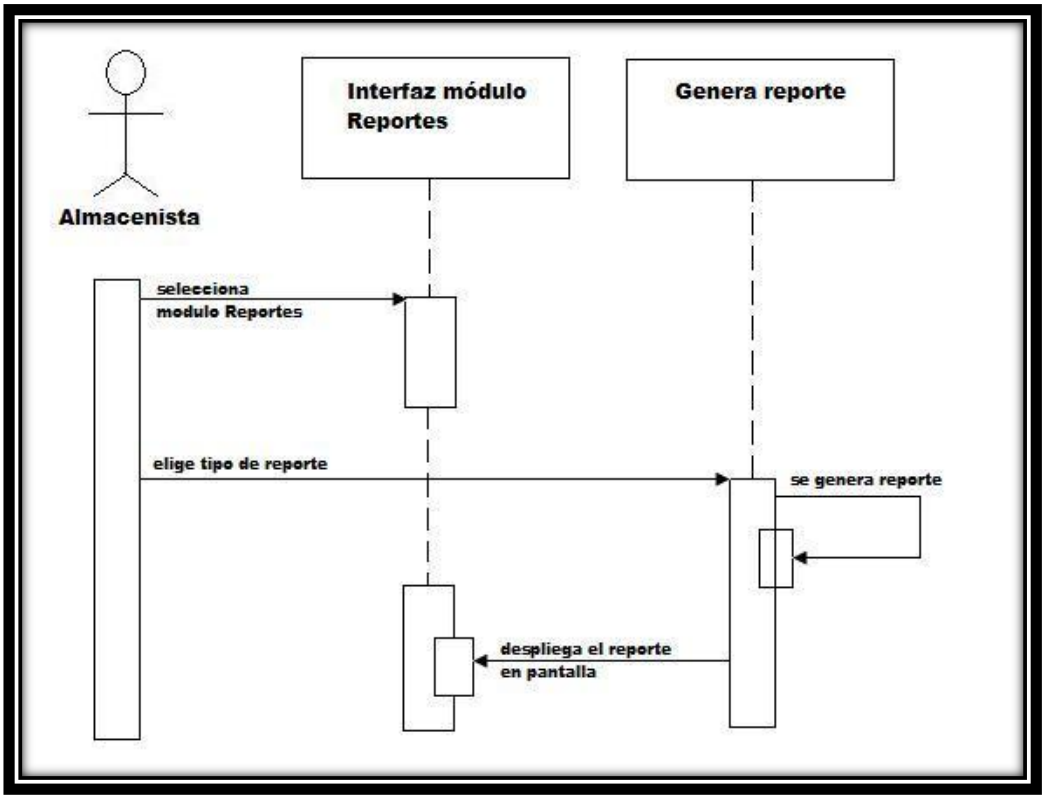


Fig. 2.8, Diagrama de secuencia de módulo Reportes

2.3 Diagrama de Clases Sistema Control Material TTC

En la siguiente figura se presenta el diagrama de clases que se utilizaron para la aplicación de escritorio así como las relaciones entre ellas.

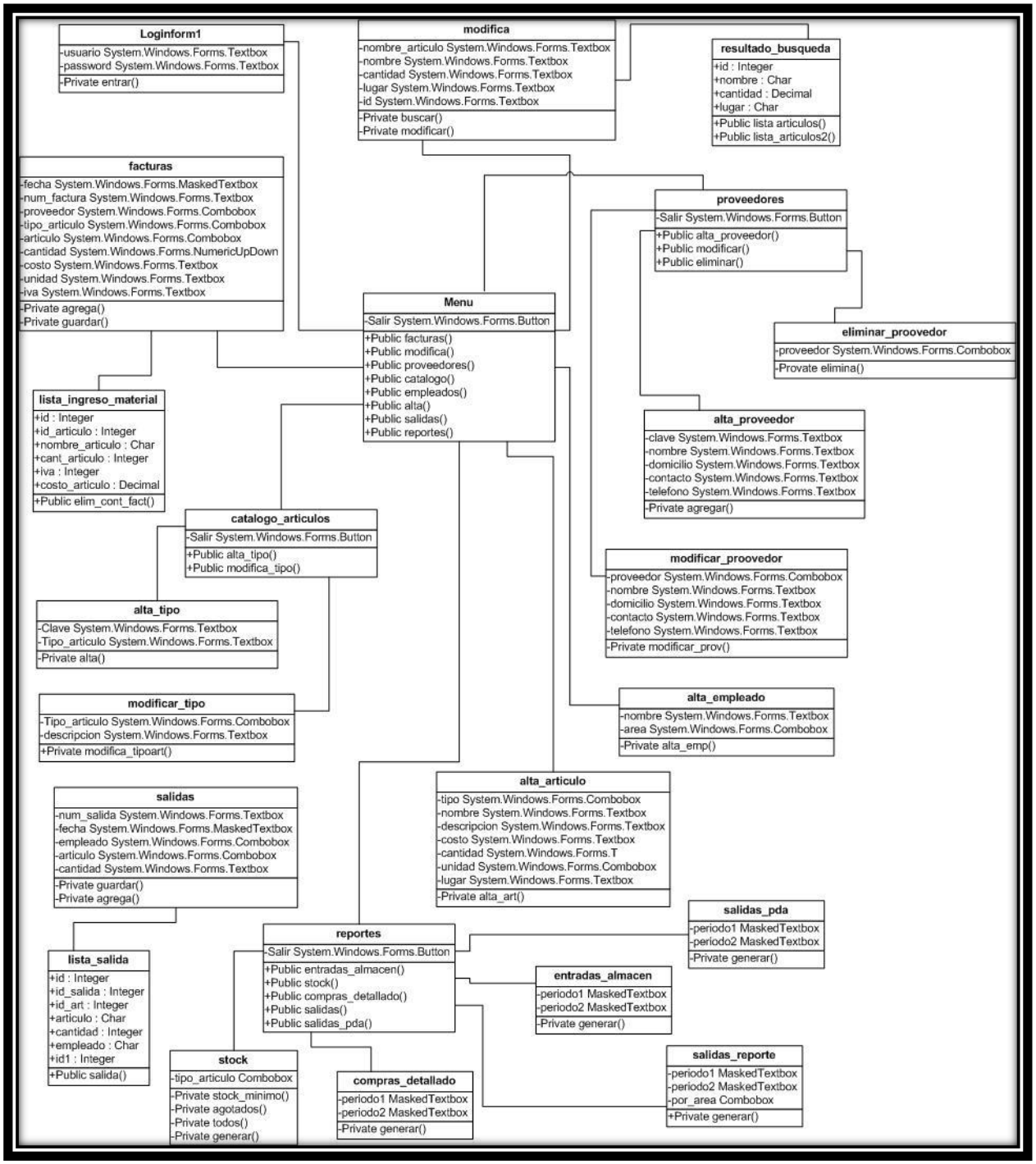


Fig. 2.9, Diagrama de clases Sistema Control Material TTC

2.4 Casos de Uso Sistema Control Material TTC/PDA

Ahora veremos las especificaciones de caso de uso de los módulos de la aplicación instalada en el dispositivo móvil.

2.4.0 Conexión WS

- Descripción

A través de este módulo se hace la conexión con el servicio Web para realizar la carga de artículos a la base de datos de la PDA, este procedimiento se realiza cada vez que se le suministra material al técnico (usuario de la PDA).

- Actores que intervienen

Almacenista: es el que tiene acceso exclusivo a este módulo y autorización para realizar la carga de artículos a la base de datos del dispositivo.
Técnico: es la persona que recibe el material y debe verificar que se carguen correctamente a la PDA los artículos que recibió.

- Flujo de Eventos

Flujo Básico

1.- El almacenista ejecuta el sistema control de material instalado en la PDA, hace login para entrar al menú y accesar al módulo de conexión con el web service.

2.- Dentro del módulo debemos hacer la conexión con el web service haciendo clic en el botón “CARGAR”, en seguida se desplegará en el recuadro central una lista de artículos que están registrados en la base de datos del servidor.

3.- Para cargar un artículo a la base de datos del dispositivo primero debemos ubicarlo en la lista del recuadro y hacer clic sobre la fila correspondiente, notaremos que en las cajas de texto de la parte inferior se cargarán el id y nombre del artículo.

4.- Ingresar en el recuadro correspondiente la cantidad del artículo que se cargará a la base de datos de la PDA (puede ser la solicitada por el técnico).

5.- Seleccionar de la lista desplegable “Empleado” el nombre del técnico usuario del dispositivo o a quien se le entrega el material.

6.- Revisar los datos introducidos y hacer clic en “Aceptar”.

Flujo Alternativo

1.- Si ingresamos una cantidad de artículo mayor a la existente en la base de datos el servidor a la hora de realizar la carga el sistema lanza un aviso por lo cual deberemos modificar la cantidad para proseguir con el procedimiento.

- Precondiciones
  - Solamente el administrador o almacenista podrá ingresar a este módulo.
  - Para poder realizar la conexión con el servidor de la base de datos, la PDA deberá estar conectado vía WiFi con la red establecida para la conexión con el web service.
  - Deberán estar registrados los empleados usuarios de este sistema para que aparezcan en la lista ‘Empleados’.

- Pos condiciones

- Al realizar una carga de artículos, si el nombre de este no estaba registrado en la base de datos del dispositivo lo que pasará es que se registra el nombre y la cantidad ingresada. Si el nombre del artículo ya estaba en la base de datos solamente se suma la cantidad ingresada a la existente.
- De la base de datos del servidor se modificará la existencia del artículo seleccionado, restándole (actualizando) la cantidad que se cargó a la base de datos de la PDA.
- Una vez que finaliza el proceso de carga de un artículo el sistema lanza una ventana de aviso y se podrá realizar la carga de otro solamente habrá que seleccionar el artículo de la lista mostrada en el recuadro central.

- Diagrama

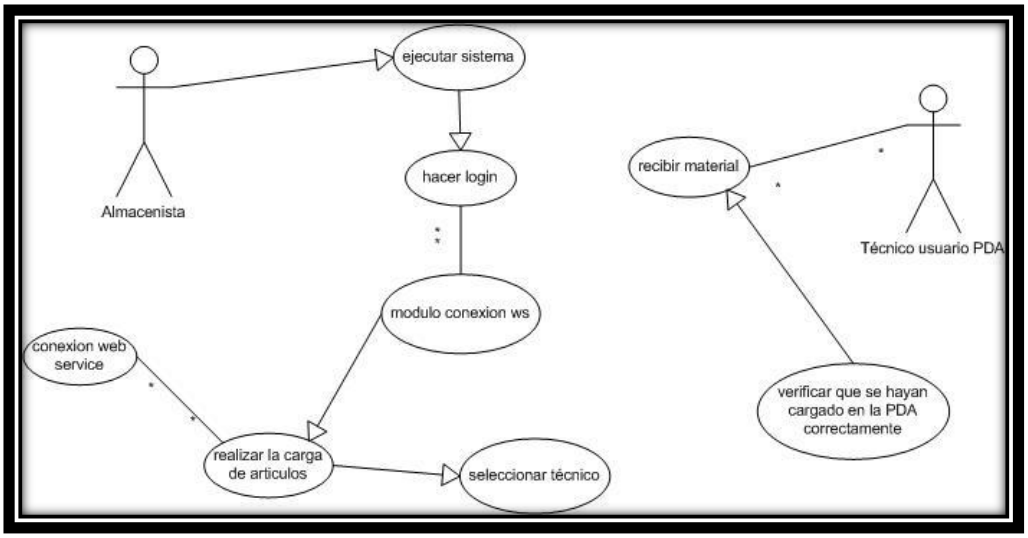


Fig. 2.10, Diagrama UML Conexión WS

2.4.1 Consulta

- Descripción

Como su nombre lo indica, en este módulo consultamos el stock del material que se tiene registrado en el dispositivo móvil.

- Actores que intervienen

Técnico o usuario de la PDA

- Flujo de Eventos

Flujo Básico

1.- El usuario ejecuta el sistema control de material instalado en la PDA, hace login para entrar al menú y accesar al módulo de consulta.

2.- De la lista correspondiente a “Artículo” seleccionar el de nuestro interés para saber su existencia y hacer clic en Consultar.

3.- Los resultados de la consulta aparecen en los cuadros de texto de la parte inferior de la ventana, los datos mostrados son: stock y unidad.

Flujo Alternativo

1.- Si se requiere consultar otro artículo sólo hay que seleccionarlo de la lista correspondiente y hacer clic en Consultar.

- Precondiciones

- Deberá haber artículos cargados en la base de datos del dispositivo para que estos aparezcan en la lista desplegable.

- Pos condiciones

- Este módulo es meramente informativo, solo se mostrarán la existencia y unidad del artículo consultado y no se podrá realizar modificación alguna sobre los datos del mismo.

- Diagrama

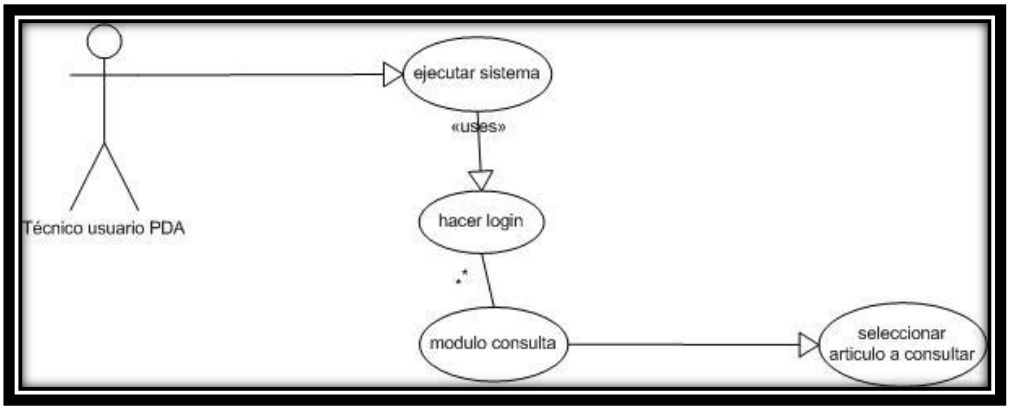


Fig. 2.11, Diagrama UML Consulta

2.4.2 Movimientos

- Descripción

En este módulo se registran todos los movimientos que se hagan con el material (con el que cuenta el técnico o usuario de la PDA). Hay dos tipos de movimientos que se pueden registrar: SALIDA o TRASLADO; además de elegir el “detalle” del movimientos, es decir, si el artículo se utilizó en una acometida (orden de servicio) o reemplazo.

- Actores que intervienen

Técnico o usuario de la PDA

- Flujo de Eventos

Flujo Básico

1.- El usuario ejecuta el sistema control de material instalado en la PDA, hace login para entrar al menú e ingresar al módulo de movimientos.

2.- De la lista de artículos seleccionar el que se utilizó.

3.- Ahora debemos elegir el tipo de movimiento que se realiza, tenemos dos opciones: SALIDA o TRASLADO.

4.- Elegir, de ser necesario, el detalle del movimiento, hay dos opciones: ACOMETIDA y REEMPLAZO, elegir la primera si se trata de una orden de servicio.

5.- Ingresar la cantidad del artículo utilizada, tenemos dos maneras de ingresar la cantidad, puede ser escribiéndola a través del teclado o haciendo clic en las flechas de incremento o decremento del cuadro de texto correspondiente a “cantidad”.

6.- Si el material fue utilizado en una acometida tenemos que ingresar el número de la orden de servicio.

7.- Por último, verificar los datos introducidos y hacer clic en ACEPTAR para almacenar el movimiento.

Flujo Alternativo

1.- Si la cantidad ingresada del artículo es mayor a la existente en la base de datos, el sistema lanza una advertencia para lo cual deberemos modificar la cantidad para que el movimiento pueda ser almacenado.

- Precondiciones

- Deberán existir artículos en la base de datos del dispositivo para que aparezcan en la lista y poder hacer movimientos con ellos.

- Pos condiciones

- Una vez almacenados los movimientos, estos quedan registrados en la base de datos de la PDA listos para ser transferidos al servidor para mantener un historial de los movimientos.
- Se realizan en la base de datos las modificaciones correspondientes a la existencia de cada artículo utilizado en el movimiento.

- Diagrama

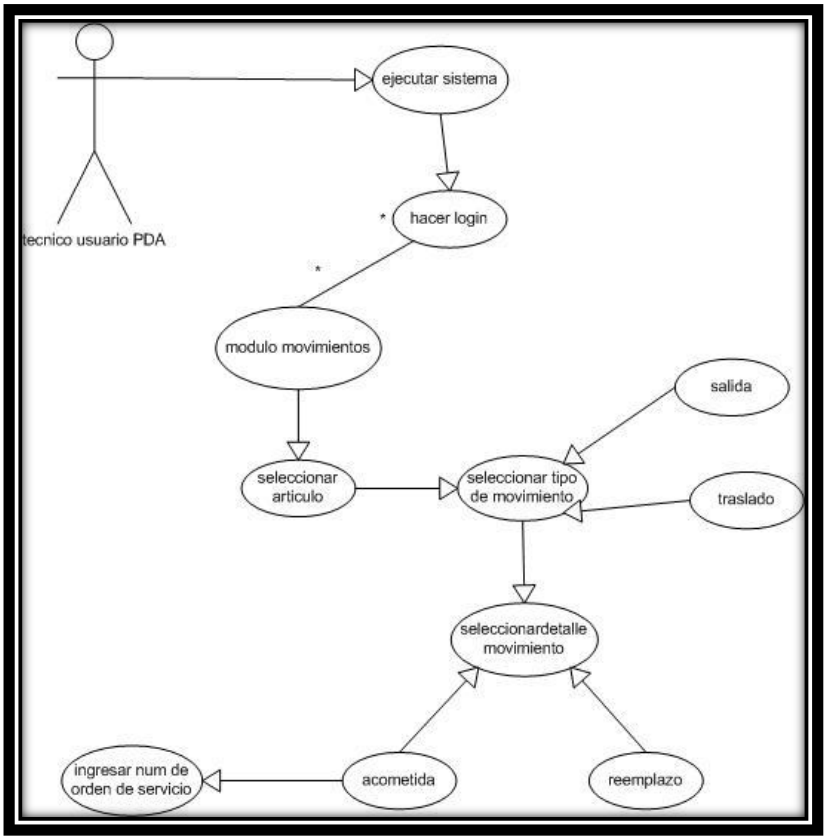


Fig. 2.12, Diagrama UML Movimientos

2.4.3 Conciliar Movimientos

- Descripción

Para mantener un historial de movimientos registrados en el dispositivo móvil y poder crear reportes, es necesario que los datos correspondientes a los movimientos se transfieran a la base de datos del servidor, para esto se debe ingresar a este módulo, en donde se realiza una conciliación de movimientos, es

decir, se hace una búsqueda de los movimientos de material registrados, se conecta al servidor a través del web service y se trasladan dichos movimientos.

- Actores que intervienen

Administrador o almacenista: persona autorizada para acceder a este módulo.

- Flujo de Eventos

Flujo Básico

1.- El administrador ejecuta el sistema control de material instalado en la PDA, hace login para entrar al menú y accesar a este módulo.

2.- Para empezar la conciliación sólo hay que presionar en el botón “Concilia Movimientos”

Flujo Alternativo

1.- Si no hay datos por conciliar el sistema lanza un aviso y no sigue el proceso.

- Precondiciones

- Para realizar la conciliación se debe conectar al web service, por lo tanto la PDA debe estar conectada vía WiFi con la red establecida para la conexión con el web service.
- El acceso al módulo lo tiene solamente el administrador o almacenista de la empresa.

- Pos condiciones

- Al finalizar la conciliación, los movimientos que estaban registrados en la base de datos de la PDA quedan registrados en la base de datos del servidor.
- Una vez transferidos los datos, estos son borrados de la base de datos de la PDA, esto con el fin de no saturar la memoria del dispositivo.

- Diagrama

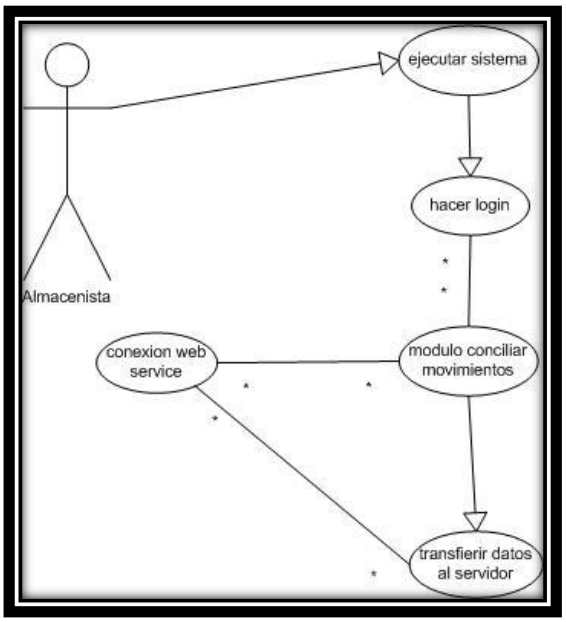


Fig. 2.13, Diagrama UML Conciliar Movimientos

2.5 Diagramas de Secuencia Sistema Control Material TTC/PDA

Los diagramas de secuencia de los procesos que corresponden a los módulos de la aplicación del dispositivo móvil son las siguientes:

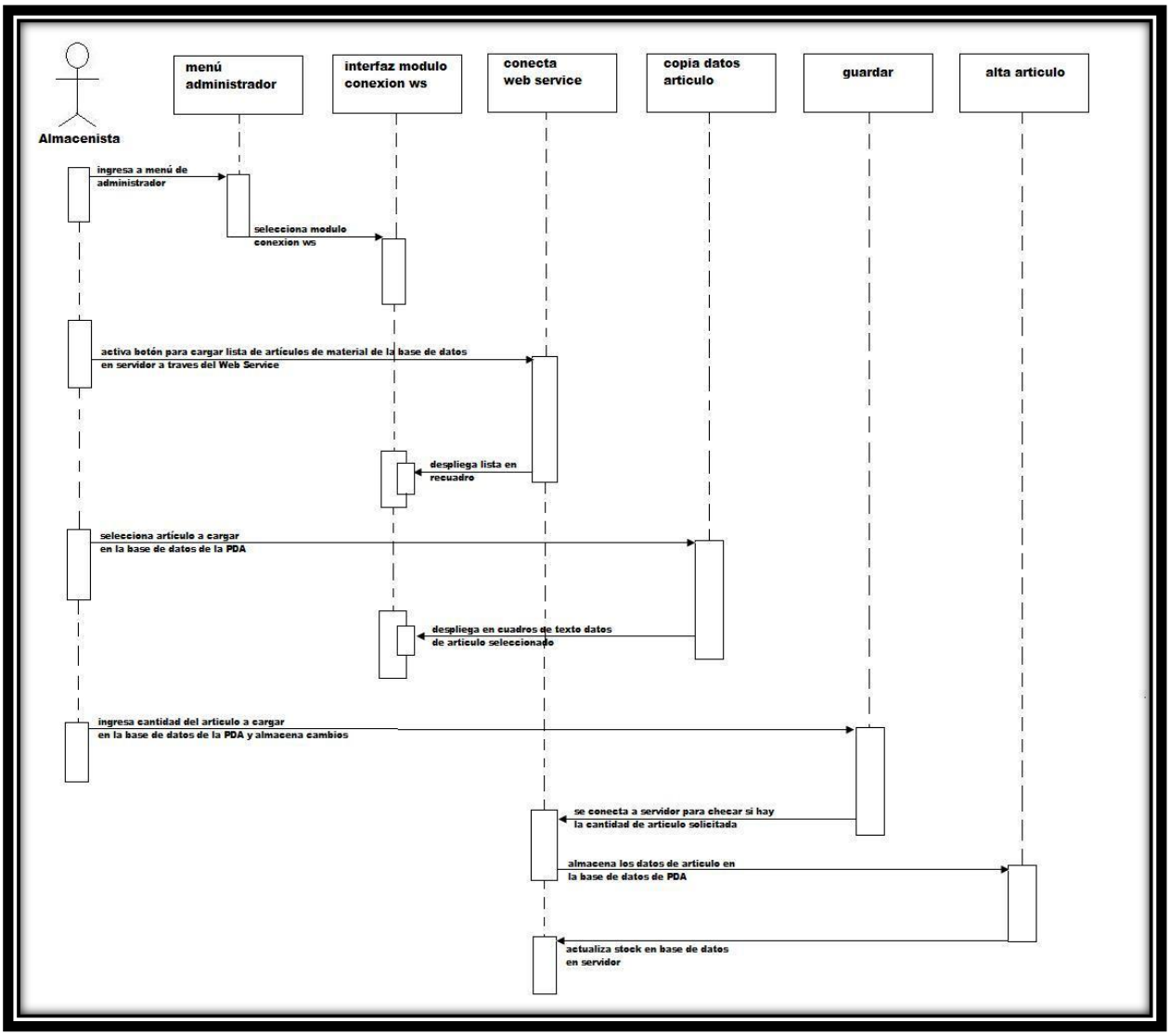


Fig. 2.14, Diagrama de secuencia de módulo Conexión WS

En la figura anterior se ilustra la secuencia del proceso para realizar la carga de artículos en la base de datos del dispositivo móvil, en dicho proceso se hace la conexión con el servidor.

A continuación se presenta el diagrama correspondiente a los movimientos de artículos que se realizan en la aplicación del dispositivo.

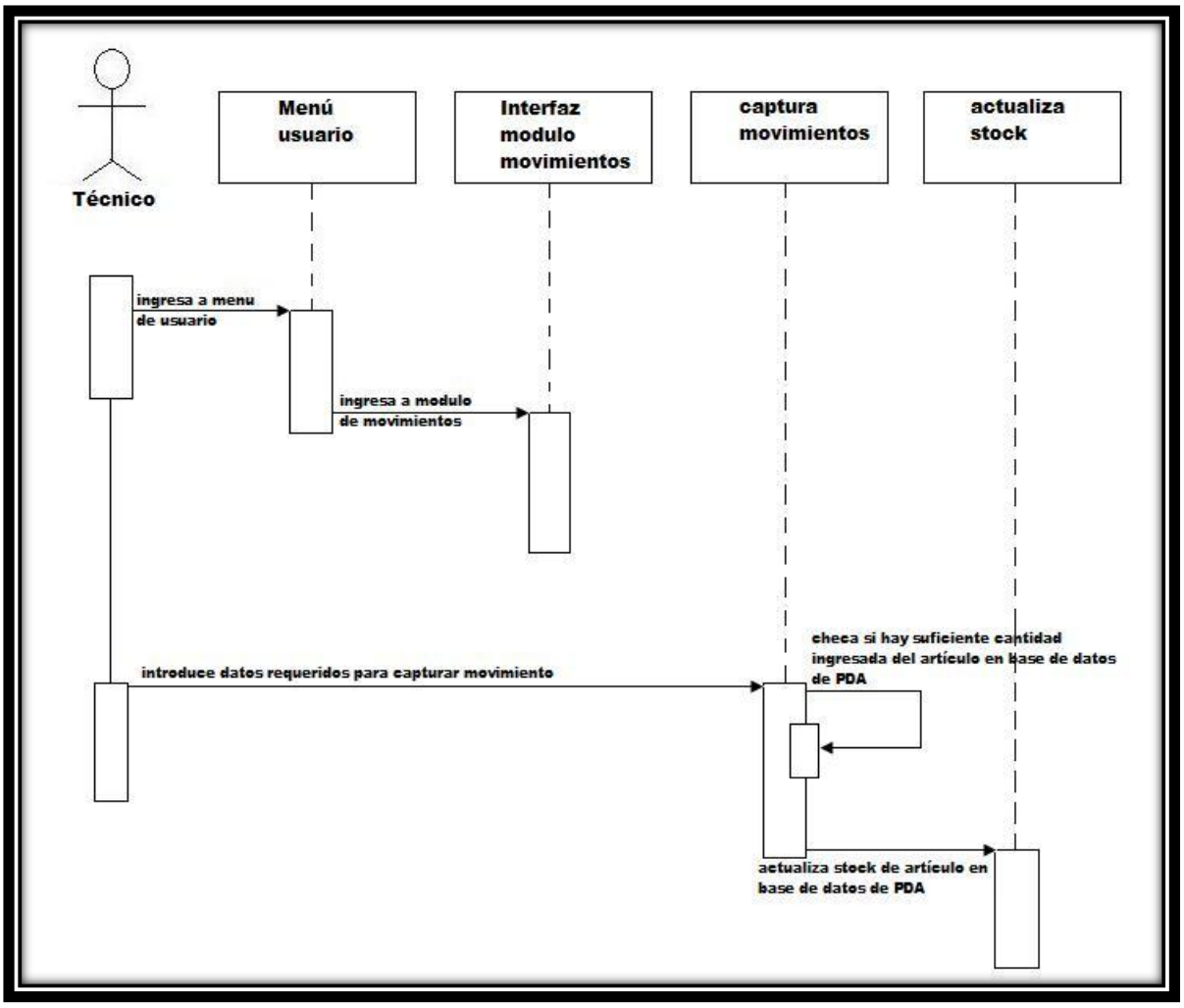


Fig. 2.15, Diagrama de secuencia de módulo Movimientos

En seguida se muestra la secuencia del proceso de conciliación de movimientos, el cual requiere conexión a la base de datos del servidor y que debe ser realizado por el almacenista o administrador del sistema.

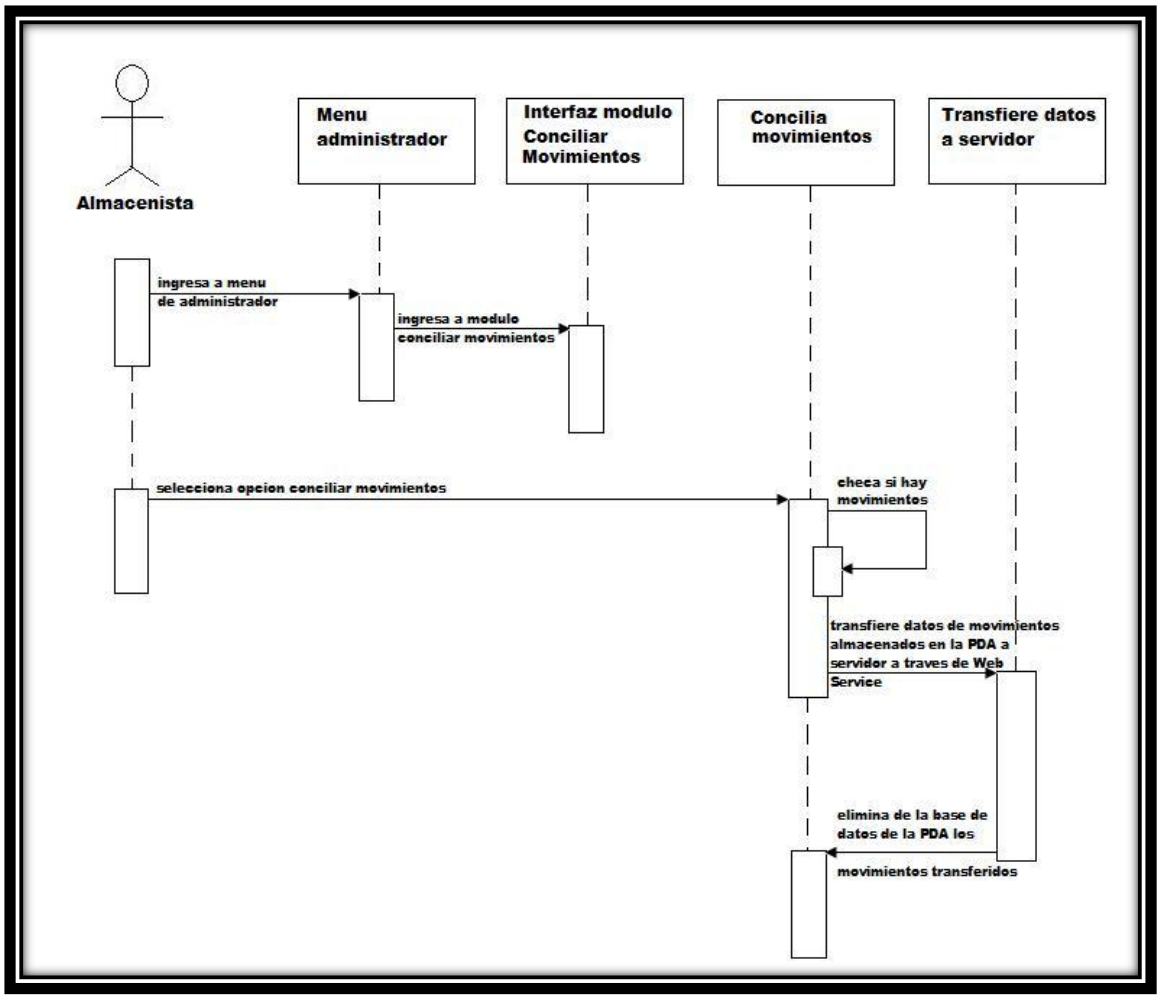


Fig. 2.16, Diagrama de secuencia de módulo Conciliar Movimientos

2.6 Diagrama de Clases Sistema Control Material TTC/PDA

En la siguiente figura se representa el diagrama de clases utilizadas para la aplicación del dispositivo móvil. Cabe mencionar que en este diagrama se incluyen dos clases que se encuentran en el servidor de Web Services las cuales son utilizadas en la aplicación del dispositivo para transferencia de datos con el servidor de base de datos.

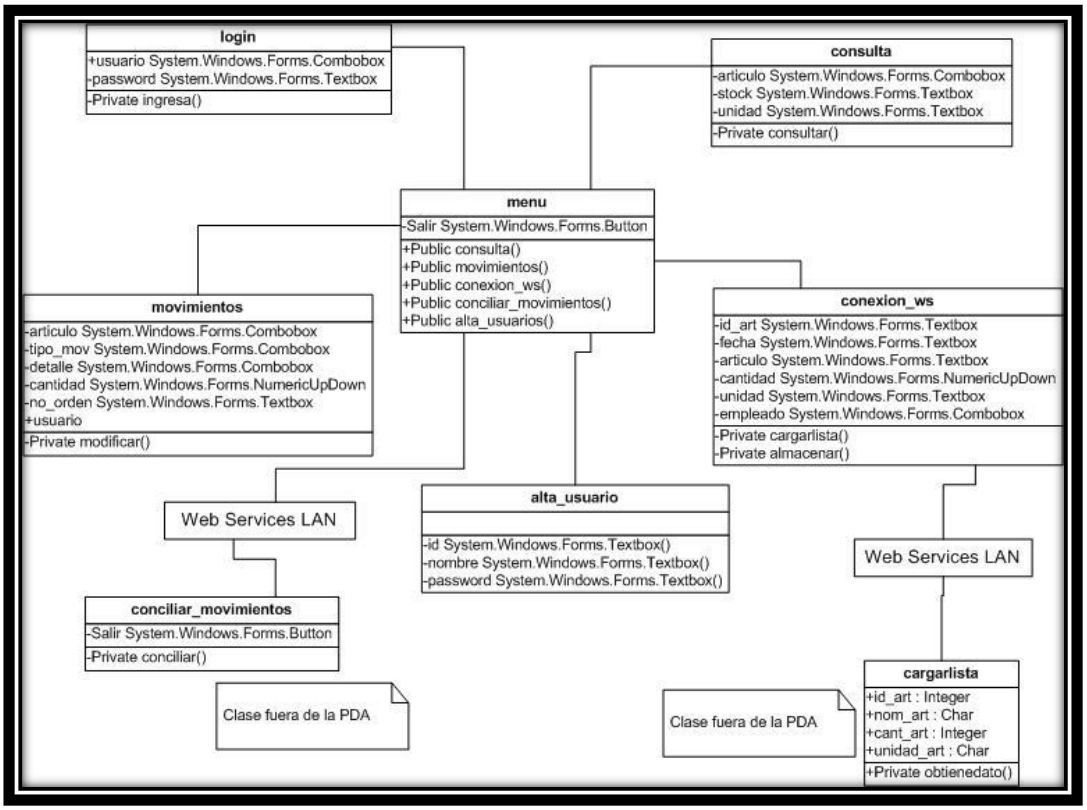


Fig. 2.17, Diagrama de clases Sistema Control Material TTC/PDA

2.7 Alternativas de Dispositivos Móviles

2.7.0 PocketPC

PocketPC, es un tipo de PDA *(Personal Digital Assistant),* con capacidades similares a una computadora de escritorio pero de tamaño pequeño de manera que es fácilmente transportable y cargar en la palma de la mano.

Según Microsoft, un PocketPC es *"un dispositivo de mano que te permite grabar, enviar y recibir e-mails, contactos, citas, mostrar archivos multimedia, juegos, intercambiar mensajes de texto con MSN Messenger, navegar por la web y más".[19]*

Los PocketPC ejecutan el sistema operativo Microsoft Windows CE o el Windows Mobile, además cuenta con un conjunto de aplicaciones en ROM, pantalla a color sensible al tacto (touchscreen) y su dispositivo apuntador conocido como stylus.

Cualquier dispositivo que sea clasificado como un PocketPC debe:

- Ejecutar el sistema operativo Microsoft Windows CE o Windows Mobile (versión PocketPC)
- Tener un conjunto de aplicaciones en ROM
- Incluir una pantalla sensible al tacto
- Incluir un dispositivo apuntador, llamado stylus o estilete
- Incluir un conjunto de botones de hardware para activar aplicaciones
- Estar basado en un procesador compatible con el STRONGARM (los Pocket PCs más antiguos tienen un procesador MIPS o SH3)

Aplicaciones como Internet Explorer, Microsoft Outlook, Word, Excel, Windows Media Player entre otros, van incluidos con el PocketPC pero se pueden desarrollar e instalar muchas otras aplicaciones que sean compatibles con el dispositivo y con fines específicos como pueden ser lectores de código de barras, GPS, juegos, etcétera.

2.7.1 PALM

Se trata de una PDA con características físicas similares a las de una PocketPC; utiliza el sistema operativo Palm OS que fue desarrollado por Jeff Hawkin de la empresa PalmSource Inc. para el primer modelo Palm, la Pilot PDA de la marca US Robotics.

La Palm fue el primer dispositivo móvil de bolsillo que salió al mercado, por lo que se ganó la popularidad de los consumidores, aunque hoy en día están parejos con su principal competidor, la PocketPC, aunque el crecimiento de este último es mayor.

Palm OS fue pensado como un sistema operativo para aplicaciones de agenda, control de lista de contactos, bloc de notas y algunas otras aplicaciones sencillas por lo que Palm se caracterizaba por ser un sistema sencillo pero completo; aunque al día de hoy, con las ventajas que proporcionan los nuevos procesadores, salen al mercado equipos más potentes y con características cada vez más similares a las de la PocketPC[6].

2.7.2 Diferencias entre PocketPC y PALM

A continuación se presenta una tabla comparativa de los dispositivos PDA previamente mencionados, algunos datos presentados son una recopilación de varias comparativas que se encuentran en la Internet y cuya referencia se encuentra en la bibliografía utilizada en este proyecto.

Tabla 2.0, Comparación de plataformas de dispositivos móviles

Característica	PocketPC	Palm
Facilidad de uso	Por ser prácticamente una PC de mano, su funcionamiento puede dificultarse un poco si no se tiene conocimiento en el manejo de PC's.	Utiliza un sistema operativo desarrollado pensado en dispositivos móviles, con aplicaciones sencillas de manejar por lo que es más fácil de usar.
Aplicaciones	El software de Windows Mobile suele ser más veloz que los programas Palm OS, aunque existen menos para esta plataforma.	Muchas de las aplicaciones son gratuitas. Supera por mucho en cuanto al total de ellas disponibles, gracias a la abundancia de freeware para este dispositivo.
Procesador	400Mhz. (mod. Hp ipaq rx3715)	416 Mhz. (mod. Palmone lifedrive)
Memoria	Cuenta con una memoria suficiente para una "minicomputadora" con	Debido a que fueron construidos para ser agendas y sigue siendo

	capacidad de ejecutar diferentes aplicaciones.	esa su premisa, el espacio en memoria no es una prioridad.
Negocios	A nivel de negocios se presenta una plataforma de fácil expansión, basta con agregar un lector de barras, un puerto de salida externo en las ranuras de expansión, se configura el software y queda listo para trabajar, sin costosas adecuaciones.	Hay muchas aplicaciones disponibles, pero la gran mayoría de ellas están desarrolladas para darle un uso personal y no con propósitos específicos de los que se le pueda aprovechar en un negocio.
Soporte	Es una plataforma en crecimiento y cada vez más empresas dedicadas a la fabricación de dispositivos optan por él.	En enero de 2009 fue anunciada la desaparición de Palm OS y es relevado por WebOS, desarrollado desde cero. Por lo que el soporte para este tipo de PDA's tenderá a disminuir.
Acceso a Internet	Presenta buena recepción WiFi y no hay complicaciones con la conexión.	Presentan complicaciones cuando alguna aplicación utiliza acceso a Internet, es más lento.

Empresas como HP-Compaq, Dell y Toshiba utilizan el sistema operativo Windows CE para sus dispositivos móviles por lo que su crecimiento es mayor que el de Palm, aunque éste fue el primero en lanzarse.

2.8 Análisis de Herramientas para Desarrollo en PDA's

2.8.0 Microsoft Embedded Visual Tools

Es un paquete de herramientas de desarrollo ofrecido por Microsoft con el cual pueden ser creadas aplicaciones para la plataforma Pocket PC además de otras como Palm-size y Handheld PC. Cabe mencionar que esta herramienta no se ejecuta en el dispositivo móvil, sino en una PC en donde se realiza la edición, compilación y depuración de los proyectos para posteriormente transferirlos al dispositivo móvil e instalarlos.

En dicho paquete se incluyen distintos entornos de desarrollo con distintos lenguajes de programación, estos son: embedded visual basic, embedded visual C++, y embedded visual basic .NET.

2.8.1 Microsoft Embedded Visual Basic

Ofrece una forma sencilla de crear aplicaciones para Windows CE, el entorno de esta herramienta es básicamente el mismo que el de Visual Basic corriente solo que los componentes disponibles y tipos de proyectos es inferior, es decir, ésta es una versión recortada. Los tipos de datos están limitados pero es posible crear funciones y procedimientos aunque no existen otras posibilidades propias de la versión normal.

Microsoft Embedded Visual Basic no genera código directamente ejecutable, sino un código intermedio que, posteriormente, es necesario interpretar durante la ejecución. De esta forma se obtienen archivos de código realmente pequeños. Cada dispositivo donde deseemos usar las aplicaciones contará con ese intérprete o runtime que, dependiendo del tipo de procesador que tenga el dispositivo, se encargará de generar el código ejecutable correspondiente.

2.8.2 Microsoft Embedded Visual C++

Con esta versión del Visual C++ se pueden crear componentes, librerías de enlace dinámico y estático y aplicaciones con y sin ventanas.

Con Visual C++ embebido además de poder crear aplicaciones Windows CE o Pocket PC se tiene la posibilidad de crear componentes COM y controles ActiveX. Una ventaja de este entorno es que el compilador incluido es capaz de generar código altamente optimizado para varias familias de procesadores distintas. Esto significa que obtendremos aplicaciones pequeñas y, al tiempo, mucho más rápidas que las generadas por Microsoft eMbedded Visual Basic.

2.8.3 Microsoft Embedded Visual Basic .NET

Embedded Visual Basic no puede ser usado para desarrollar aplicaciones para Windows Mobile basados en celulares. Debido a esto y otros factores, la tecnología

de embedded visual Basic se ha transformado en un lenguaje más poderoso, el Visual Basic .NET.

Microsoft recomienda a todos los desarrolladores usar el Visual Basic .NET para crear nuevas aplicaciones sobre Pocket PC y celulares y usar el Embedded Visual Basic solo para mantener viejas aplicaciones.

2.8.4 PocketBuilder

De la empresa Sybase, se trata de una herramienta de desarrollo de para plataformas Windows CE, la cual permite crear aplicaciones para dispositivos móviles; además brinda altos niveles de facilidad de uso ya que el entorno de programación es muy parecido al de PowerBuilder, herramienta con la cual podemos crear aplicaciones de escritorio.

2.9 Análisis de Alternativas de Bases de Datos

2.9.0 SQL

Lenguaje de Consulta Estructurado (por sus siglas en inglés Structured Query Language) es un lenguaje para el acceso a a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. En la actualidad es el sistema de gestión de bases de datos más utilizado, ya que cuenta con una versión gratuita y con un amplio soporte en cuanto a información se refiere.

Entre las principales funciones que nos ofrece SQL están las siguientes: integridad, recuperación, manipulación y compartición de datos, así como control de acceso.

SQL Server tiene 6 tipos de licencias según el tipo de usuarios:

- SQL Server Enterprise Edition
- SQL Server Standard Edition
- SQL Server Workgroup Edition

- SQL Server Express Edition
- SQL Server Compact Edition
- SQL Server Developer Edition
- Editions on 64-bit Platform

Los costos varían desde Gratis (Express y Compact Edition), hasta $ 25 000 USD (Enterprise Edition). Estos costos dependen de los servidores a utilizar y el número de clientes (usuarios).

Ventajas

- Portabilidad a cualquier tipo de plataforma
- SQL está estandarizado
- Basado en el modelo relacional
- Barato
- Lenguaje de alto nivel
- Consultas interactivas ad-hoc
- Escalabilidad, estabilidad y seguridad
- Arquitectura cliente/servidor

Desventajas

- Costo de las licencias comparadas con otros competidores.
- La migración requiere un reinicio de la base de datos. El reinicio de todos los datos en una base de datos es un proceso en el que se implica la posibilidad de que haya pérdida de datos.

2.9.1 Oracle

Oracle es un sistema de gestión de bases de datos de tipo relacional desarrollado por Oracle Corporation, es considerado uno de los sistemas de bases de datos más completos por el soporte en transacciones que ofrece así como su estabilidad, escalabilidad y soporte multiplataforma.

Distintas fuentes de información afirman que aunque el dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server y de la oferta de otros sistemas con licencia libre como PostgreSQL, MySql o Firebird.

Oracle cuenta con 5 ediciones:

- Oracle Database Enterprise Edition(EE).
- Oracle Database Standard Edition (SE).
- Oracle Database Standard Edition One (SE1).
- Oracle Database Express Edition (XE).
- Oracle Database Personal Edition (PE).

La única edición gratuita es la Express Edition además de otras que pueden utilizarse con fines educativos.

Ventajas

- Puede ejecutarse en todas las plataformas, desde una PC hasta una supercomputadora.
- Permite el uso de particiones para la mejora de la eficiencia, de replicación y algunas versiones admiten la administración de bases de datos distribuidas.
- El software del servidor puede ejecutarse en diferentes de sistemas operativos.
- Objetos de Oracle (tipos de clases, referencias, tablas anidadas, matrices y otras estructuras de datos complejas).
- Oracle es la base de datos con más orientación hacia Internet.

Desventajas

- Desde el lanzamiento original de la versión 8.0 le sucedieron varias versiones con correcciones, hasta alcanzar la estabilidad en la 8.0.3. El motivo de tantos fallos fue, la remodelación del sistema de almacenamiento por causa de la introducción de extensiones orientadas a objetos.
- Su precio, incluso las licencias de Personal Oracle son excesivamente caras.

- Difícil configuración del sistema, un Oracle mal configurado puede ser muy lento.

2.9.2 SQL CE Server

SQL CE Server nos permite la creación y administración de bases datos en los dispositivos móviles, está apoyado en el lenguaje estructurado de consulta SQL; expone un sistema esencial de características de la base de datos relacionales. Incluye un procesador de consulta, además permite la entrada de datos remotos por lo que es ideal para los ambientes móviles y wireless.

Entre las principales características y funcionalidades de SQL CE se encuentran las siguientes:

- Capacidades robustas de la administración de datos en los dispositivos móviles.
- Un modelo operacional constantes con el resto de la familia del SQL Server, lo que significa que se pueden integrar fácilmente con los sistemas existentes y aprovecharse de capacidades existentes de desarrollo.
- Uso pequeño de la memoria, entregando toda su funcionalidad en aproximadamente 1 megabyte (MB). El funcionamiento se realza con un procesador QUERY óptimo.
- Cifrado 128-bit que proporciona en el dispositivo para la seguridad de archivo de base de datos.
- Acceso flexible de los datos.

2.10 Análisis de Alternativas de Conexión entre un Dispositivo Móvil y un Servidor

2.10.0 Sockets

Los sockets son una capa de abstracción con una arquitectura cliente-servidor, que permite que dos procesos (ubicados en diferentes máquinas o no) puedan intercambiar flujos de datos de manera transparente, sin conocer los detalles de

cómo se transmiten esos datos. Un socket queda definido por una dirección IP, un protocolo y un número de puerto.

Cómo funcionan

Para que dos programas puedan comunicarse entre sí, es necesario que se cumplan ciertos requisitos:

- Que un programa sea capaz de localizar al otro.
- Que ambos programas sean capaces de intercambiarse cualquier secuencia de bits.

Para ello son necesarios tres recursos:

- *Un protocolo de comunicaciones*, que permite el intercambio de octetos (conjunto de 8 bits).
- *Una dirección IP*, que identifica una computadora.
- *Un número de puerto*, que identifica a un programa dentro de una computadora[19].

El protocolo más utilizado para implementar sockets es el TCP[1], aunque también podemos utilizar UDP[2] o IPX[3]. Gracias al protocolo TCP, los sockets tienen las siguientes propiedades:

- Orientado a conexión.
- Se garantiza la transmisión de todos los octetos sin errores ni omisiones, además llegarán a su destino en el mismo orden en que se ha transmitido[19].

[1] Transmission Control Protocol (Protocolo de Control de Transmisión), permite crear conexiones entre programas dentro de una red de datos.
[2] User Datagram Protocol (Protocolo de Datagrama de Usuario) es un protocolo del nivel de transporte basado en el intercambio de datagramas. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión.

[3] Internetwork Packet Exchange (Intercambio de paquetes interred). Se utiliza para transferir datos entre el servidor y los programas de las estaciones de trabajo.

La primera especificación de sockets se dio a la necesidad de tener un medio sencillo y eficaz para escribir programas capaces de comunicarse entre sí. Hoy en día, los sockets están implementados como bibliotecas de programación para multitud de sistemas operativos, simplificando la tarea de los programadores.

Ventajas

- Los sockets son rápidos, al ser de bajo nivel introducen poca sobrecarga a las aplicaciones.
- Todos los lenguajes de programación y protocolos de transporte los soportan.

Desventajas

- No resultan muy cómodos para el programador. Al no permitir el paso directo de argumentos, el programador tiene que encargarse de abrir/cerrar los flujos de I/O, colocar en ellos los argumentos que quiere pasar, extraer los resultados, etcétera.
- Están muy ligados a la plataforma donde se ejecutan.
- El código es difícil de reutilizar.
- No proveen mucha funcionalidad.

2.10.1 Java RMI

Java RMI (Remote Method Invocation) es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y provee un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java[19].

Su principal característica es la facilidad de uso, ya que resulta natural para el programador de Java, pues no se tiene que aprender una nueva tecnología distinta

con la cual desarrollará; además con el fin de optimizar el uso de memoria, proporciona el paso de objetos por referencia y un Garbage Collector[4] distribuido.

Cómo funcionan

Por medio de RMI, un programa Java puede exportar un objeto, lo que significa que éste queda accesible a través de la red y el programa permanece a la espera de peticiones en un puerto TCP. A partir de este momento, un cliente puede conectarse e invocar los métodos proporcionados por el objeto.

La invocación se compone de los siguientes pasos:

- *Encapsulado de los parámetros,* mediante la serialización, es decir, generando una secuencia de bytes para su transmisión.
- *Invocación del método,* el cliente (invocador) se queda esperando una respuesta.
- *Servidor serializa el valor de retorno* (si lo hay) y lo envía al cliente.
- *Cliente recibe la respuesta* y continúa como si la invocación hubiera sido local.

En la siguiente figura se muestran los pasos de un llamado a un objeto remoto:

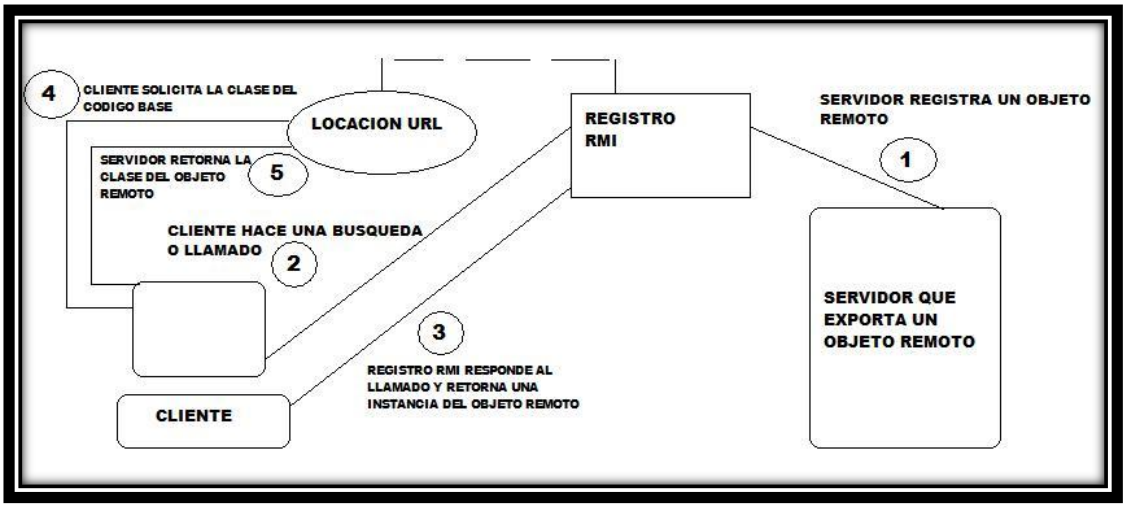


Fig. 2.18, Llamado a un objeto remoto Java RMI

[4] Rutina que busca en la memoria segmentos de programas o datos que ya no son activos, con el fin de recuperar ese espacio.

La arquitectura RMI se compone de cuatro capas

Capa 1. Aplicación

En esta capa tienen lugar las llamadas a alto nivel para acceder y exportar objetos remotos. Para que un objeto sea accesible remotamente se debe declarar una interfaz que extienda java.rmi.Remote.

Capa 2. Proxy

También conocida como “stub-skeleton”, esta capa es la que interactúa directamente con la capa de aplicación. Todas las llamadas a objetos remotos y acciones junto con sus parámetros y retorno de objetos tienen lugar en esta capa.

Capa 3. Referencia remota

Es responsable del manejo de la parte semántica de las invocaciones remotas. Así como de la gestión de la replicación de objetos y realización de tareas específicas de la implementación con los objetos remotos, como el establecimiento de las persistencias semánticas y estrategias adecuadas para la recuperación de conexiones perdidas.

Capa 4. Transporte

Aquí se realizan las conexiones necesarias y manejo del transporte de los datos de una máquina a otra. El protocolo de transporte subyacente para RMI es JRMP (Java Remote Method Protocol), que solamente es comprendido por programas Java.

Ventajas

- Aprovecha las ventajas del lenguaje java, sobre todo en el aspecto de seguridad.
- Los detalles de comunicación son transparentes para el programador.

Desventajas

- Debido a su estrecha integración con Java, esta tecnología no permite la interacción con aplicaciones escritas en otro lenguaje.
- Carece de rapidez en la transmisión de los datos.

2.10.2 CORBA

CORBA (Common Object Request Broker Architecture), es un estándar abierto para sistemas distribuidos definido por el *Object Management Group* (OMG), que es un consorcio dedicado al cuidado y establecimiento de distintos estándares de tecnologías orientadas a objetos[2].

Esta tecnología permite al cliente invocar métodos en objetos remotos en el servidor independientemente del lenguaje con el que los objetos hayan sido escritos y su locación.

Cómo funcionan

La interacción entre el cliente y el servidor es mediada por corredores de petición de objeto (ORB[5] por sus siglas en inglés) tanto del lado del cliente como del servidor, comunicándose vía IIOP (Internet Inter-ORB Protocol)[6]. CORBA envuelve el código escrito en otro lenguaje en un paquete que contiene información adicional sobre las capacidades del código que contiene, y sobre cómo llamar a sus métodos; de esta manera los objetos que resultan pueden ser invocados remotamente.

En la siguiente figura se ilustra el desarrollo de una aplicación CORBA:

[5] Object Request Broker (Corredor de Petición de Objeto) permite a los objetos realizar llamadas a métodos situados en máquinas remotas.
[6] IIOP (Internet Inter-Orb Protocol) es la implementación de GIOP ((Protocolo Entre ORBs General) para TCP/IP.

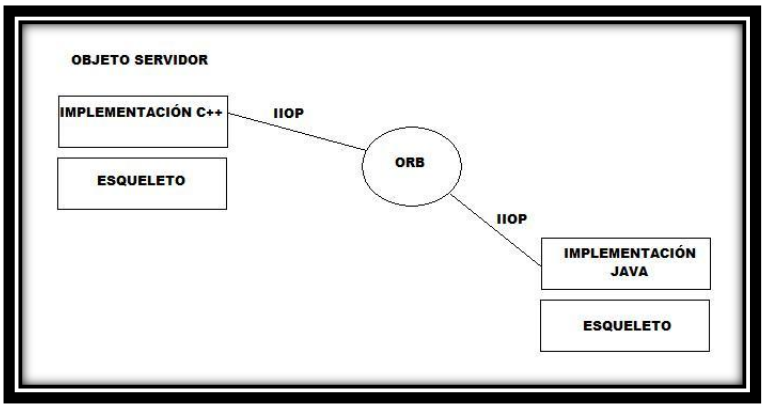


Fig. 2.19 Desarrollo de aplicación CORBA

El ciclo de vida típico de una aplicación CORBA es de la siguiente manera:

1.- Define el servicio como interfaz en IDL[7]

2.- Compila el IDL para generar matriz del cliente y esqueletos del servidor

3.- Ejecuta el servicio y asocia con los esqueletos a través del POA[8]

4.- Publica el servicio con un nombre para ser usado por los clientes

El proceso del cliente de CORBA consiste en lo siguiente:

1.- Contacta el nombre del servicio deseado y recupera la referencia del objeto apropiada.

2.- Se invocan operaciones en la referencia del objeto usando el compilador IDL. Alternativamente los clientes pueden deducir las operaciones soportadas por el servicio consultando un repositorio de interfaz (IR) y dinámicamente crear peticiones con los parámetros apropiados.

3.- Respuesta del proceso o excepciones.

Ventajas

- Las solicitudes no pasan directamente del cliente al objeto, sino que son administradas por un ORB lo que hace que los detalles de la distribución sean transparentes para los clientes y para los objetos.

[7] Interface Definition language (IDL) es un lenguaje de especificación de interfaces utilizado en software de computación distribuida.

[8] Portable Object Adapter (Adaptador de Objeto Portable), responsable de partir al encargado de la invocación remota del lado del servidor en el objeto remoto y su criado (blanco de invocación)

- Permite un acceso estándar a tipos de datos comunes y la funcionalidad necesaria para grupos de aplicaciones corporativos y específicos para las aplicaciones industriales.
- Diferentes implementaciones son interoperables, es decir, no importa el lenguaje del cliente.

Desventajas

- En un entorno móvil el cliente (y posiblemente el servidor) se mantiene moviendo lo cual requiere tratar con cambios de direcciones de red (y por lo tanto la necesidad de remitir servicios) y conexiones no confiables.

- El factor de movilidad crea algunas restricciones adicionales de las cuales se debe tener cuidado antes de tiempo debido a la estrecha asociación entre el cliente y el servidor.

2.10.3 Web Services

Los web services (servicios web) son aplicaciones que utilizan transporte estándar, códigos, y protocolos para el intercambio de información. Habilitan sistemas de computadoras en cualquier plataforma para comunicar sobre intranets corporativas, extranets y a través de la Internet con soporte para seguridad fin – a – fin, mensajería confiable, transacciones distribuidas, etcétera.

Los estándares en los que se basan los web service para describir la sintáxis y semántica de la comunicación de un software son los siguientes:

- *XML (Extensible Markup Language):* provee la sintaxis común para los datos representativos, un web service es una aplicación creada en XML.
- *SOAP (Simple Object Access Protocol):* provee la semántica para el intercambio de datos, permite que programas que corren en diferentes sistemas operativos se comuniquen.
- *WSDL (Web Services Description Language):* se encarga de proveer un mecanismo para describir las capacidades de un web service.

- *UDDI (Universal Description Discovery and Integration):* Es un directorio que permite la publicación y localización de los servicios. Los directorios UDDI actúan como una guía telefónica de web services[8].

Cómo funcionan

En un escenario típico de Web Services, una aplicación envía una petición vía HTTP a un servicio situado en una URL. El servicio recibe la petición, la procesa y devuelve una respuesta también sobre HTTP. Para realizarlo se requiere:

- Un protocolo de intercambio de mensajes petición/respuesta sobre HTTP.
- Una forma de que clientes y proveedores puedan interactuar a través de los mensajes, es decir, un *lenguaje de especificación de interfaces*.

Como se mencionó anteriormente, SOAP es utilizado para el intercambio de mensajes ya que se trata de un protocolo sencillo basado en XML. En lo que respecta al lenguaje de especificación de interfaces, se utiliza el WSDL que permite especificar en XML las operaciones y tipo de datos de un servicio web; de esta manera aunque el cliente y servidor estén escritos en lenguajes distintos pueden interactuar al utilizar un lenguaje neutral.

La petición de un web service consta de los siguientes pasos:

1. En el cliente se elabora una petición de una operación con unos parámetros.
2. La petición se transforma a formato XML utilizando WSDL.
3. La petición transformada se envía vía HTTP utilizando SOAP.
4. El servidor de servicios web recibe la petición.
5. El servidor determina que operación debe realizarse y transforma los parámetros de formato XML a su representación correspondiente en el lenguaje utilizado para implementar el servidor.
6. El servidor invoca la operación con los parámetros enviados, elabora una respuesta y se la envía al cliente de la misma forma en que llegó[4].

En la siguiente figura se representa de manera general la arquitectura de un web service.

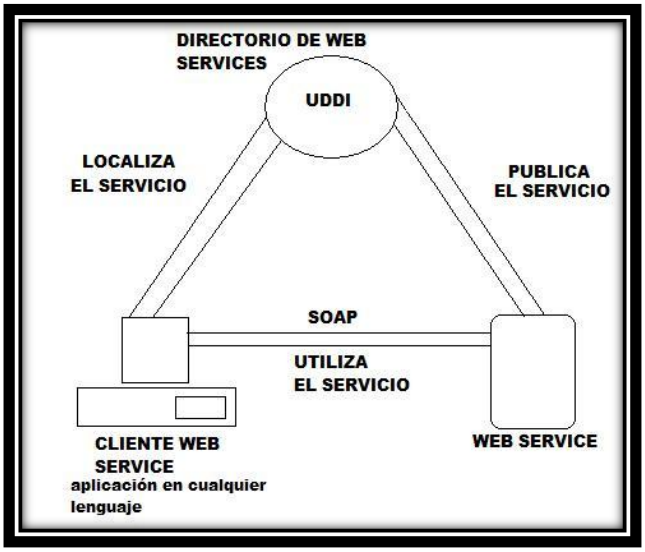


Fig. 2.20 Arquitectura de un web service

Ventajas

- Independencia del lenguaje de programación. El servidor y el cliente no necesitan estar escritos en el mismo lenguaje.
- Los Web services fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento[17].
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad *firewall* sin necesidad de cambiar las reglas de filtrado.
- Interoperabilidad e integración.
- Óptimos tiempos de respuesta.
- Sirve de base para la integración de múltiples aplicaciones.

Desventajas

- Para realizar transacciones no pueden compararse con los estándares abiertos de computación distribuida como CORBA[17].

- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, como RMI, CORBA, o DCOM (Distributed Component Object Model) [17].
- Los web services hacen uso de las mismas tecnologías que han sido atacadas en tantas ocasiones[10].

2.10.4 Comparación de Tecnologías

A continuación se presenta una tabla en la cual se realiza una comparación entre las alternativas de conexión entre un dispositivo móvil y un servidor para este proyecto.

Tabla 2.1, Comparación de tecnologías de sistemas distribuidos

	JAVA RMI	CORBA	WEB SERVICES
Mecanismo de invocación	Java RMI	CORBA RMI	.NET, JAX-RPC[9]…etc.
Formato de datos	Serialización	CDR[10]	XML
Formato de comunicación	Stream	GIOP[11]	SOAP
Protocolo de Transferencia	JRMP[12]	IIOP	HTTP
Lenguaje de especificación de interfaces	Interfaz Java	IDL	WSDL
Seguridad	Potente (seguridad java)	Potente (Servicio de seguridad CORBA)	Pobre (HTTP, XML)
Difusión	Registro Java	COS[13]	UDDI

[9] Permite que una aplicación Java invoque a un Web Service basado en Java, puede ser visto como un "Java RMI sobre Web Services".
[10] Common Data Representation. Se usa para representar tipos de datos estructurados o primitivos como argumentos o resultados durante invocacioes remotas.
[11] General Inter-ORB Protocol. Protocolo por el cual los ORB's se comunican.
[12] Java Remote Method Protocol. Especificación de Java para encontrar y referenciar objetos remotos.
[13] Common Object Services. Es un servidor para almacenar referencias de objetos CORBA.

Capítulo 3

Herramientas a Utilizar en este Proyecto

En este capítulo se dan a conocer las distintas herramientas que fueron utilizadas para el desarrollo del presente proyecto, desde el software utilizado para la programación de las aplicaciones hasta la marca del dispositivo PDA.

Capítulo 3 Herramientas a Utilizar en este Proyecto

3.0 Dispositivo Móvil

Después de analizar las alternativas de dispositivos móviles se optó por utilizar la ***PocketPC*** ya que al punto de vista del desarrollador de este proyecto, la interfaz es más amigable y es sencilla de utilizar, tomando en cuenta que los usuarios de este sistema son personas con conocimientos básicos en computación; además por la compatibilidad de su sistema operativo con el software de desarrollo utilizado.

Una razón más por la que se eligió esta plataforma es por su sistema operativo, Windows Mobile, una versión reducida del Windows para computadoras de escritorio y laptops, por lo que a los usuarios les resulta más fácil el uso del dispositivo por el previo conocimiento del sistema operativo Windows.

Funda de Protección

Debido a que el trabajo que realizan los técnicos (usuarios de la PDA) es considerado “trabajo rudo”, se debe tomar muy en cuenta la protección del dispositivo ya que si no se tiene el cuidado adecuado será muy fácil que se dañe el equipo.

Para este tipo de actividades existen fundas protectoras de PDA’s, las cuales cumplen con las características necesarias de protección al dispositivo. La funda a utilizar más apropiada para este proyecto es la siguiente:

Marca Armor, Modelo 1900, Costo aproximado: $1,300.00

Características:

- Membrana protectora
- Mica Protectora
- Acceso a todas las funciones gracias a partes removibles de goma y caucho.

Fig. 3.0, Funda Armor 1900

**Imágenes obtenidas de la página web www.otterbox.com*

3.1 Herramienta de Desarrollo

Después de analizar las alternativas de las herramientas de desarrollo para este proyecto, se decidió utilizar el entorno de desarrollo integrado ***Visual Studio .NET*** de Microsoft versión 2005. Con las características que nos ofrece esta herramienta fueron las idóneas para elegirla puesto que se aprovechó para desarrollar tanto la aplicación de escritorio como la del dispositivo móvil gracias a que Visual Studio soporta la plataforma Windows Mobile y Windows CE.

Para la creación del módulo de reportes se utilizó ***Power builder***, debido a las bastantes opciones que ofrece en cuanto a los formatos de los reportes, la sintaxis utilizada es fácil de entender además de que ya se tenían conocimientos previos sobre esta herramienta.

3.2 Conexión entre Servidor y Dispositivo Móvil

Para esta parte medular del proyecto se optó por desarrollar un ***Web Servic*e**, aplicación con la que obtenemos ciertas ventajas en el transporte de la información, una de ellas es la rapidez, gracias a que utiliza HTTP para la transferencia de datos,

el contenido que se envía es muy ligero obteniendo una rápida respuesta del lado de la aplicación cliente. En cuanto a la seguridad, se aprovecha el sistema firewall configurando las reglas de flitrado según las necesidades; y otra importante característica es que el lenguaje del servidor y el cliente no tiene que ser el mismo por lo que queda la posibilidad de que futuras aplicaciones puedan integrarse al sistema de datos de la empresa y hacer consumo del web service.

La entorno de desarrollo para la programación del web service fue Visual Studio .NET, mismo que se utilizó para la aplicación de escritorio y del dispositivo.

3.3 Base de Datos

Como gestor de la base de datos se utilizó ***SQL***, en la versión *SQL Server Express*, aunque es gratuita, está diseñada para construir aplicaciones robustas y fiables ofreciendo una sencilla pero potente base de datos.

Características:

Fácil de instalar y configurar
Fácil de utilizar y administrar
Seguridad robusta

- Valores predeterminados seguros
- Derechos de administración detallados
- Tres niveles de seguridad de acceso al código
  - Seguro
  - Acceso externo (verificable)
  - No seguro
- Aprovechamiento de los procedimientos almacenados como capa de abstracción
- Compatibilidad con Active Directory[14]
- Compatibilidad con la autenticación de Windows

Sencillez de precios y licencias

[14] Implementación de Microsoft de servicio de directorio en una red distribuida de computadoras.

- Siempre gratuito

  Amplia funcionalidad de bases de datos
- Procedimientos almacenados
- Vistas
- Triggers
- Cursores
- Índices ampliados
- Aislamiento a nivel de snapshot[15]
- Optimizador avanzado de consultas
- Compatibilidad con T-SQL[16]

  Compatibilidad con XML

  Profunda integración con Visual Studio 2005

SQL Server CE

Aplicación que se utilizó como sistema de gestión de la base de datos en los dispositivos PDA; ya que está diseñada para integrarse con el Visual Studio .NET de manera que ofrece bastantes herramientas para la creación y vinculación con aplicaciones de escritorio.

Query Analyzer 3.0

Parte del SQL Server CE 2.0, el Query Analyzer se utiliza como interfaz para la creación y administración de la base de datos del dispositivo móvil.

[15] Herramienta para instalar en sitios webs, para crear una visualización o pre visualización de cada vínculo de la página.
[16] Lenguaje de programación del SQL Server para simplificar código y ganar rapidez en las transacciones puesto que se ejecuta dentro del SQL Server y se compila la primera vez que se ejecuta un procedimiento almacenado.

Capítulo 4

Arquitectura del Sistema

La información contenida en este capítulo es parte esencial del presente proyecto, ya que se describe cómo está conformada la arquitectura del sistema desarrollado para poder hacer posible la conexión entre la PDA y el servidor; además se explica de manera detallada cada capa en la que se divide dicho sistema.

Capítulo 4

Arquitectura del Sistema

Este sistema se puede dividir en 4 capas:

Capa 1: Base de Datos Servidor
Capa 2 A: Aplicación de Escritorio
Capa 2 B: Web Service
Capa 3: Base de Datos Dispositivo Móvil / Dispositivo Móvil

La base principal del sistema se sirve de un servidor de base de datos (capa 1), en el cual como gestor se utiliza “SQL Server Express” para la administración de la base de datos.

De cara al usuario, tenemos la capa de Aplicación de escritorio (capa 2 A), el cual es el “SISTEMA DE CONTROL MATERIAL TTC” software desarrollado en Visual Basic .NET para su funcionamiento en una o más PC’s. Estas máquinas trabajan con el sistema operativo Windows que tengan instalado el .Net Framework, todos los movimientos realizados a través de la aplicación de escritorio quedan almacenados en la base de datos del servidor.

Para el intercambio de información entre la base de datos del servidor y el dispositivo móvil se utiliza la tecnología de los Web Services (capa 2 B), de acuerdo a las necesidades para este proyecto, se desarrolló el Web Service con Visual Basic .NET para que la comunicación con el dispositivo móvil se haga de forma local, es decir, utilizando una red WiFi, esta red se configura para que el acceso sea solamente dentro del área que cubra el ruteador inalámbrico; aunque las bases quedan para que en un futuro se pueda acceder a la base de datos desde cualquier punto en donde haya conexión a Internet.

En la capa 3 tenemos el dispositivo móvil, un Pocket PC mejor conocidos como PDA (Personal Digital Assistant), en este se instaló el “SISTEMA DE CONTROL

MATERIAL TTC / PDA" software también desarrollado con el Visual Basic .NET con las configuraciones necesarias para ser instalado y ejecutado en un dispositivo móvil y que cuenta con acceso al Web Service.

Para el almacenamiento de los datos en la PDA, siguiendo esta arquitectura, tenemos también en la capa 3 la base de datos del dispositivo móvil, el gestor que se utiliza es el SQL CE Server, el SISTEMA DE CONTROL MATERIAL TTC / PDA es la interfaz a la base de datos y cuenta con conexión al Web Service.

A continuación en la figura x.x tenemos el esquema de la arquitectura del sistema.

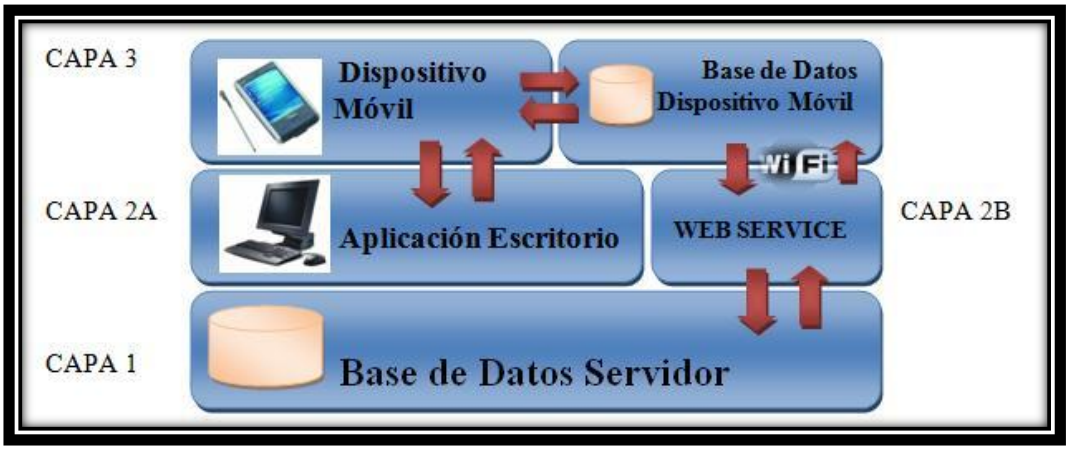


Fig. 4.0 Arquitectura del Sistema

En la siguiente figura se muestra un esquema con los dispositivos y tecnologías que se utilizan en este sistema.

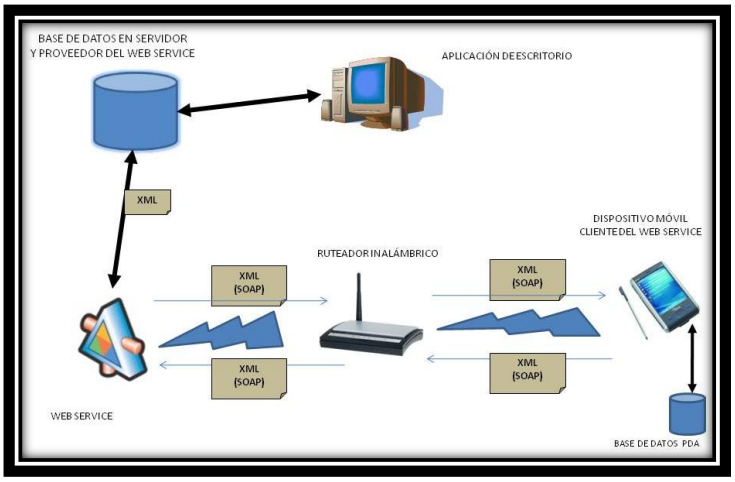


Fig. 4.1 Esquema sistema

Como se mencionó anteriormente, con esta arquitectura el sistema queda preparado para que en un futuro, realizando algunas modificaciones, se pueda tener acceso a la base de datos del servidor a través Internet, es decir, podríamos obtener los datos desde cualquier parte del mundo en donde haya conexión a esta.

4.0 Capa 1. Base de Datos Servidor

Es una base de datos operacional ya que para fines de este proyecto no sólo es necesario almacenar la información, sino que también debe de permitir funciones como actualización y adición de datos así como las operaciones fundamentales de consulta.

El modelo de base de datos que sigue es el de tipo relacional, es el más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente, además tiene la ventaja de que es más fácil de entender y de utilizar para un usuario casual de la base de datos.

En la siguiente figura se muestra el diagrama de relación de tablas de la base de datos.

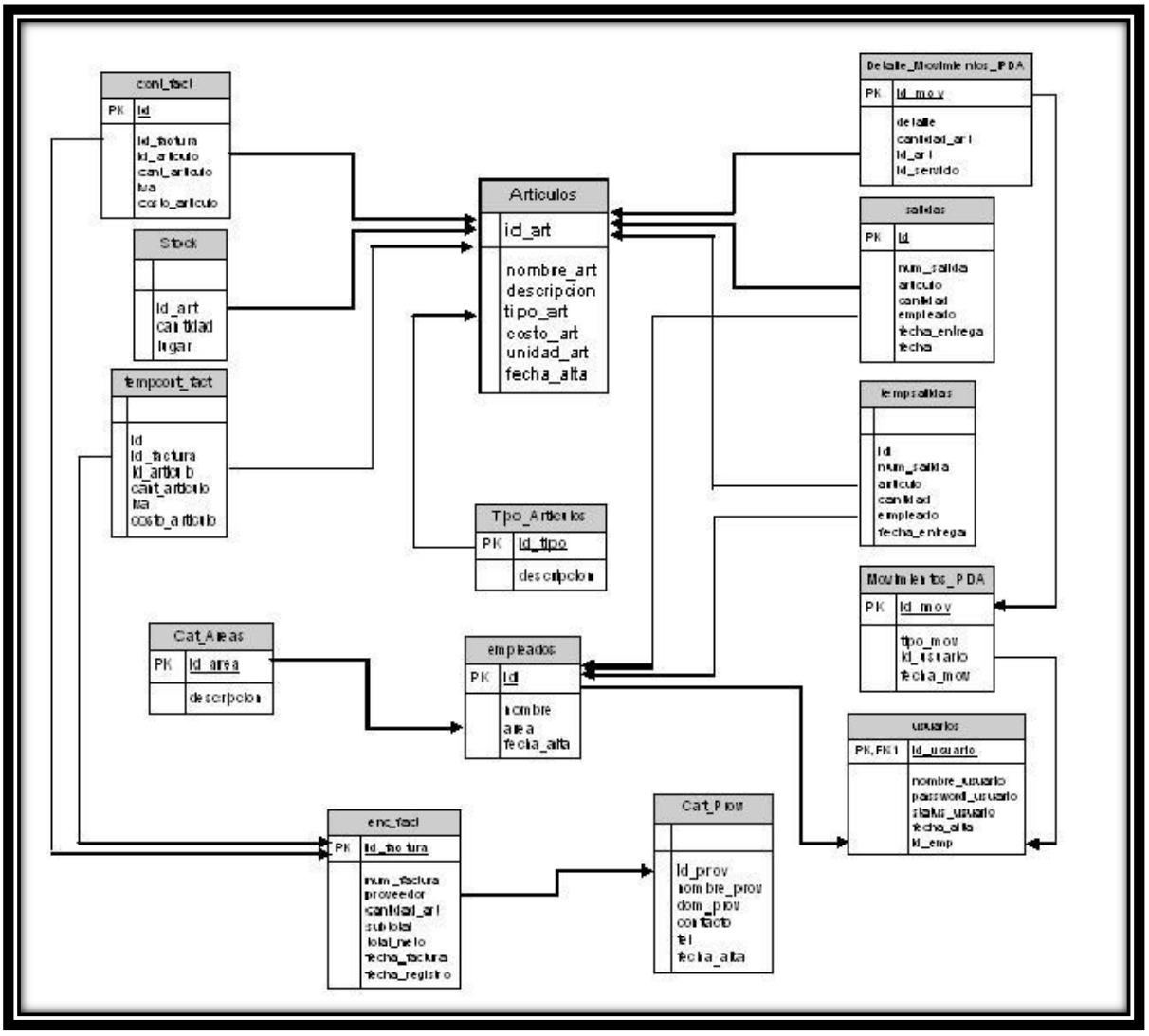


Fig. 4.2, Diagrama de relación de tablas de base de datos servidor

4.1 Capa 2 A. Aplicación de Escritorio

4.1.0 Sistema Control Material TTC

Interfaz gráfica desarrollada con la herramienta Visual Basic .NET 2005, parte del entorno de programación Microsoft Visual Studio .NET, ya que este permite crear una programación orientada a objetos y así obtener todos sus beneficios como reutilización y extensión de código, mejor interfaz gráfica, y facilitar el mantenimiento

del software. Otra característica importante por la que se eligió esta herramienta es la integración que existe con SQL Server Express.

Con este programa el usuario obtiene una mejor experiencia con el manejo del software ya que se programó para que tenga una interfaz más amigable y sea fácil de usar.

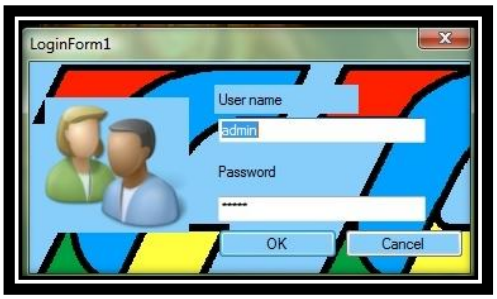


Fig. 4.3. Pantalla Login Sistema Control Material TTC

El sistema cuenta con autenticación de usuario para mantener la seguridad en el manejo de la información que respecta al material de la empresa.

El menú principal del programa contiene una interfaz amigable que permite identificar por medio de íconos o texto el módulo de nuestro interés.

Fig. 4.4. Menú principal Sistema de Control de Material TTC

Los módulos con los que cuenta el sistema son: Ingreso Material, Modificaciones, Catálogo Proveedores, Reportes, Alta Artículo, Alta Usuarios, Catálogo Artículos y Salidas.

4.2 Capa 2 B. Web Service

Es la aplicación que servirá como transporte para el intercambio de información entre el dispositivo móvil y la base de datos del servidor, es decir, la aplicación tendrá acceso a la base de datos “inventariottc” para consultar o actualizar datos según sea el caso. Un web service se puede dividir en dos partes: un servidor y un cliente o consumidor; a continuación se explica de manera detallada cómo fue desarrollada el servidor “web service“ de este proyecto.

4.2.0 ttcservice

Es el servidor del Web service, fue desarrollado según las necesidades para este proyecto, como ya se ha mencionado, el acceso a este web service es sólo de manera local. Está compuesto por 6 métodos expuestos para el acceso del cliente (dispositivo móvil); estos métodos son: *captsalida, descarga, inserta_detallemovimientosPDA, inserta_movimientosPDA, obtenerstock, ultsalida.*

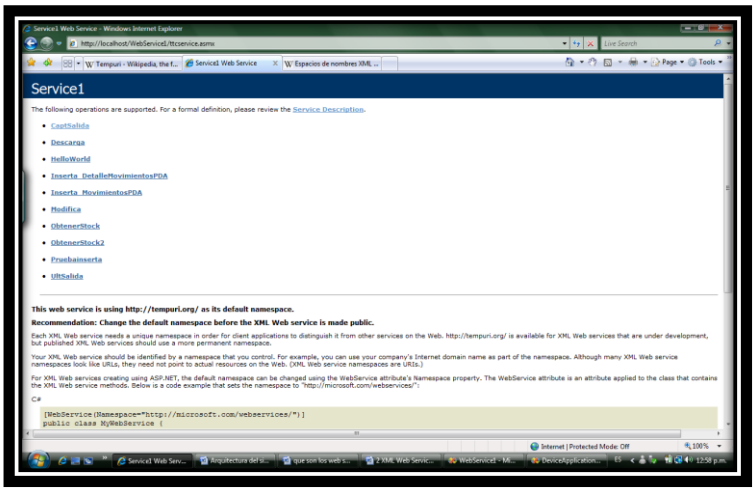


Fig. 4.5, Página de comprobación Web Service

4.2.1 Método CaptSalida

Este método es utilizado para actualizar la tabla Salidas de la base de datos “inventariottc”, recordemos que en esta tabla se va almacenando la información de las salidas de material que son registradas en la aplicación de escritorio y también en la aplicación del dispositivo móvil.

Script de programación:

```
<WebMethod()> _
Public Function CaptSalida(ByVal us As Decimal, ByVal art As Decimal, ByVal cant As Decimal, ByVal emp As
Decimal, ByVal fecha As String) As String

        Dim conexion As New SqlConnection()
        conexion.ConnectionString = "Data Source=JHUERTA-PC\SQLEXPRESS;Initial Catalog=inventariottc;Persist
Security Info=true;User ID=sa; Pwd=svsql2k3"
        If conexion.State <> Data.ConnectionState.Open Then
            conexion.Open()
        End If

        Dim sql As String
        sql = "INSERT INTO salidas(num_salida, articulo, cantidad, empleado, fecha_entrega) values('" & us &
"','" & art & "','" & cant & "','" & emp & "','" & fecha & "')"
        Try
            Dim transaction As SqlTransaction
            transaction = conexion.BeginTransaction("Transaccion SQL")

            Dim comando As New SqlCommand(sql, conexion)

            comando.Connection = conexion
            comando.Transaction = transaction

            Dim result As Integer
            result = comando.ExecuteNonQuery()
            transaction.Commit()
            transaction.Rollback()
            conexion.Close()
        Catch ex As Exception

        End Try

    End Function
```

Lo que logramos con este método es mantener un historial de salidas que se registran en el dispositivo móvil y mantener los datos de la base del servidor actualizados y así poder crear reportes de salida con datos correctos y al día.

4.2.2 Método Descarga

A través de este método se realiza la actualización a la tabla Stock de la base de datos del servidor, este método es invocado cuando desde el dispositivo móvil se realiza una descarga de artículos a la base de datos del servidor, es decir, se van agregar o sumar artículos a la base de datos del dispositivo móvil y se hace un "update" al stock del inventario del almacén.

Script de programación:

```
<WebMethod()> _
   Public Function Descarga(ByVal id As Decimal, ByVal n As Decimal) As String
        Dim conexion As New SqlConnection()
        conexion.ConnectionString = "Data Source=JHUERTA-PC\SQLEXPRESS;Initial Catalog=inventariottc;Persist
Security Info=true;User ID=sa; Pwd=svsql2k3"
        If conexion.State <> Data.ConnectionState.Open Then
            conexion.Open()
        End If

        Dim sql As String
        sql = "UPDATE Stock SET cantidad=cantidad -" & n & " WHERE id_art=" & id & ""
```

```
        Try

            Dim transaction As SqlTransaction
            transaction = conexion.BeginTransaction("Transaccion SQL")

            Dim comando As New SqlCommand(sql, conexion)
            comando.Connection = conexion
            comando.Transaction = transaction

            Dim result As Integer
            result = comando.ExecuteNonQuery()
            transaction.Commit()
            transaction.Rollback()
            conexion.Close()
        Catch ex As Exception

        End Try

    End Function
```

Con el uso de este método se mantiene actualizada la existencia de material en el almacén.

4.2.3 Método Inserta_DetalleMovimientosPDA

Con la invocación de este método se realiza el registro del detalle de movimientos registrados en el dispositivo móvil a la base de datos del servidor, es decir, una conciliación de datos entre la base de datos del dispositivo móvil y el servidor.

A continuación se presenta el script de programación de este método:

```
<WebMethod()> _
Public Function Inserta_DetalleMovimientosPDA(ByVal v As String, ByVal w As String, ByVal x As String, ByVal
y As String, ByVal z As String) As String

        Dim conexion As New SqlConnection()
        conexion.ConnectionString = "Data Source=JHUERTA-PC\SQLEXPRESS;Initial Catalog=inventariottc;Persist
Security Info=true;User ID=sa; Pwd=svsql2k3"
        If conexion.State <> Data.ConnectionState.Open Then
            conexion.Open()
        End If

        Dim id_mov, cantidad_art, id_art, id_servicio As Decimal
        Dim detalle As String
        id_mov = Convert.ToDecimal(v)
        detalle = Convert.ToString(w)
        cantidad_art = Convert.ToDecimal(x)
        id_art = Convert.ToDecimal(y)
        id_servicio = Convert.ToDecimal(z)

        Dim sql As String
        sql = "INSERT INTO Detalle_Movimientos_PDA(id_mov, detalle, cantidad_art, id_art, id_servicio)
values('" & id_mov & "','" & detalle & "','" & cantidad_art & "','" & id_art & "','" & id_servicio & "')"
        Try
            Dim transaction As SqlTransaction
            transaction = conexion.BeginTransaction("Transaccion SQL")

            Dim comando As New SqlCommand(sql, conexion)

            comando.Connection = conexion
            comando.Transaction = transaction

            Dim result As Integer
            result = comando.ExecuteNonQuery()
            transaction.Commit()
            transaction.Rollback()
            conexion.Close()
```

```
        Catch ex As Exception

        End Try
         End Function
```

El objetivo de este método es que se mantenga un historial de movimientos de material registrados en el dispositivo móvil en la base de datos del servidor para la creación de los reportes.

4.2.4 Método Inserta_MovimientosPDA

Con el uso de este método se insertan a la base de datos del servidor datos generales de los movimientos de material registrados en el dispositivo móvil.

Script de programación:

```
<WebMethod()> _
Public Function Inserta_MovimientosPDA(ByVal w As String, ByVal x As String, ByVal y As String, ByVal z As
String) As String

        Dim conexion As New SqlConnection()
        conexion.ConnectionString = "Data Source=JHUERTA-PC\SQLEXPRESS;Initial Catalog=inventariottc;Persist
Security Info=true;User ID=sa; Pwd=svsql2k3"
        If conexion.State <> Data.ConnectionState.Open Then
            conexion.Open()
        End If

        Dim id_mov, id_usuario As Decimal
        Dim tipo_mov, f2 As String
        Dim fecha_mov As Date
        id_mov = Convert.ToDecimal(w)
        tipo_mov = Convert.ToString(x)
        id_usuario = Convert.ToDecimal(y)
        fecha_mov = Convert.ToDateTime(z)
        f2 = Month(fecha_mov) & "-" & Microsoft.VisualBasic.DateAndTime.Day(fecha_mov) & "-" &
Year(fecha_mov)

        Dim sql As String
        sql = "INSERT INTO Movimientos_PDA(id_mov, tipo_mov, id_usuario, fecha_mov)  values('" & id_mov &
"','" & tipo_mov & "','" & id_usuario & "','" & f2 & "')"
        Try
            Dim transaction As SqlTransaction
            transaction = conexion.BeginTransaction("Transaccion SQL")

            Dim comando As New SqlCommand(sql, conexion)

            comando.Connection = conexion
            comando.Transaction = transaction

            Dim result As Integer
            result = comando.ExecuteNonQuery()
            transaction.Commit()
            transaction.Rollback()
            conexion.Close()
        Catch ex As Exception

        End Try
            End Function
```

Al igual que el método anterior, el objetivo de este es mantener el historial de movimientos de material que se registran en la PDA.

4.2.5 Método ObtenerStock

Como su nombre lo indica, con este método obtenemos la información del stock de artículos que se encuentra almacenada en la base de datos del servidor, esto, para verificar la existencia de cada artículo al momento de realizar una descarga de artículos a la base de datos del servidor.

Script de programación:

```
<WebMethod()> _
    Public Function ObtenerStock() As DataSet
        Dim conexion As New SqlConnection()
        Dim s2 As String

        conexion.ConnectionString = "Data Source=JHUERTA-PC\SQLEXPRESS;Initial Catalog=inventariottc;Persist
Security Info=true;User ID=sa; Pwd=svsql2k3"
        If conexion.State <> Data.ConnectionState.Open Then
            conexion.Open()
        End If

        Dim comando As New SqlCommand("SELECT a.id_art, a.nombre_art, s.cantidad ,a.unidad_art FROM
Articulos AS a INNER JOIN Stock AS s ON a.id_art = s.id_art", conexion)
        Dim adapt As New SqlDataAdapter
        adapt.SelectCommand = comando

        Dim dset As New DataSet
        adapt.Fill(dset, "Stock")
        Return dset
   End Function
```

Los datos obtenidos son cargados en un recuadro, dentro del SISTEMA DE CONTROL MATERIAL TTC / PDA.

4.2.6 Método Ultsalida

Obtiene el número de la última salida registrada en la tabla Salidas de la base de datos del servidor para que cuando se realice una descarga de artículos esta se registre como una salida más en el servidor.

Script de programación:

```
<WebMethod()> _
Public Function UltSalida() As String

        Dim conexion As New SqlConnection()
        conexion.ConnectionString = "Data Source=JHUERTA-PC\SQLEXPRESS;Initial Catalog=inventariottc;Persist
Security Info=true;User ID=sa; Pwd=svsql2k3"
        If conexion.State <> Data.ConnectionState.Open Then
            conexion.Open()
        End If

        Dim us As Decimal

        Try

            Dim comando As New SqlCommand("SELECT MAX(num_salida) as num_salida from salidas", conexion)

            Dim result As SqlDataReader
```

```
            result = comando.ExecuteReader()

            If result.Read() Then
                us = (result("num_salida"))
            End If


        Catch ex As Exception

        End Try
        Return us

    End Function
```

Cabe recalcar que en todos los métodos en donde se realizan operaciones con la base de datos del servidor se utilizan instrucciones de seguridad Transact-SQL, esto, para la confiabilidad en todas las transacciones.

4.3 Capa 3. Dispositivo Móvil

El dispositivo móvil que se utilizó en este proyecto cuenta con las características requeridas para este sistema; la marca de este es Hewlett Packard y el modelo es Pocket HP Ipaq RX 3715, se eligió este modelo debido a sus características de conectividad en lo que comunicaciones se refiere, y son las siguientes:

- Serie
- Infrarrojos
- USB
- Wi-Fi
- Bluetooth

Fig. 4.6, Pocket HP Ipaq RX 3715

A continuación se listan sus características principales.

- Software de sistema operativo: Microsoft Windows Mobile 2003 Second Edition Software
- Procesador: Procesador Samsung S3C 2440 de 400 Mhz
- Tipo de memoria: 152 MB totales, 64 MB de SDRAM / 128 MB de ROM (155 MB de memoria disponible para el usuario)
- Ranuras de memoria: Ranura para tarjetas SD: admite tarjetas SD, SDIO y MMC
- Dimensiones: 71,2 x 16,3 x 114,3 mm
- Pantalla: TFT transflectiva de color de 3,5 pulgadas (89 mm), modos vertical y horizontal, 65.000 colores
- Tecnologías inalámbricas: WLAN 802.11b, Bluetooth®, IrDA1 [17]

4.3.0 Sistema Control Material TTC/PDA

Aplicación desarrollada con Visual Basic .NET 2005, este software es la versión del sistema de control de material para su instalación en el dispositivo DPA. A través de este programa, el técnico captura los datos del material que utiliza en cada orden de trabajo para posteriormente ser transferidos al servidor a través del Web Service.

Fig. 4.7, Pantalla inicial Software

[17] Infrared Data Association (IrDA) define un estándar físico en la forma de transmisión y recepción de datos por rayos infrarrojo.

La interfaz gráfica es muy amigable y fácil de usar, pues como ya se mencionó anteriormente, los usuarios de estos dispositivos cuentan solo con conocimientos básicos en computación.

Para la integridad de los datos, el software cuenta con una ventana de autenticación de usuario, así como un menú exclusivo para el administrador y otro para usuario normal.

Fig. 4.8, Menú administrador

Los módulos del menú administrador son: consulta, movimientos, conexión ws y conciliar movimientos; en cuanto a las opciones del usuario, sólo se tienen habilitados dos módulos, consulta y movimientos.

Fig. 4.9, Menú usuario

4.3.1 Base de Datos en Dispositivo Móvil

Creada con la herramienta SQL Server CE, es la base de datos instalada en la PDA en la cual se almacenan los datos que se transfieran desde la base de datos del servidor a través del web service, además también en ella se registran, a través del SISTEMA DE CONTROL MATERIAL TTC / PDA, los movimientos con el material que realice el personal que utilice el dispositivo móvil (técnico).

Diagrama de relación de tablas:

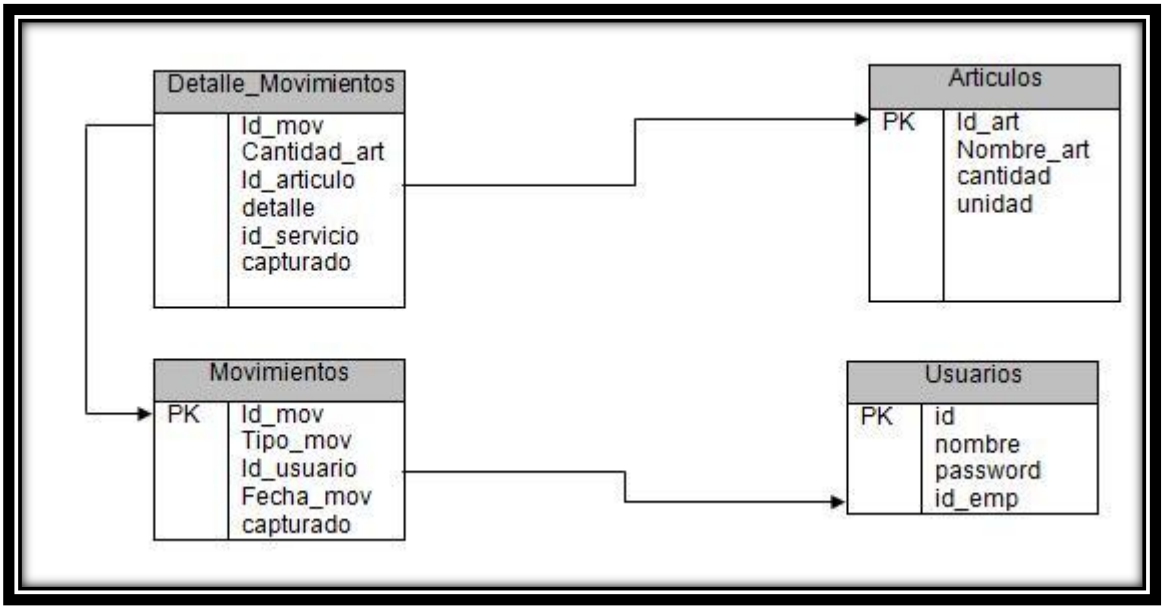


Fig. 4.10, Diagrama de relación de tablas base de datos dispositivo móvil

Capítulo 5

Resultados

Una vez desarrolladas las aplicaciones para PC y los PDA’s se realizaron una serie de pruebas para comprobar que cada uno de los módulos trabaje correctamente. Por lo que en este capítulo se describe en qué consistió cada prueba y los resultados obtenidos además de mencionar una serie de posibles fallas que pudiera tener el software.

Capítulo 5 Resultados

5.0 Pruebas al Software

En lo que respecta a la aplicación de escritorio se realizaron las siguientes pruebas:

- Captura de facturas (ingreso de material)

  Esta prueba consistió en capturar los datos de una factura por la compra de material, dichos datos que se almacenaron son: fecha y número de factura, proveedor, material con cantidades y costos.

Fig. 5.0, Datos ingresados en módulo Ingreso Material

- Configuración del catálogo de artículos y de proveedores

  Se realizó el alta, modificación y eliminación de datos de proveedores así como registrar los tipos de artículos.

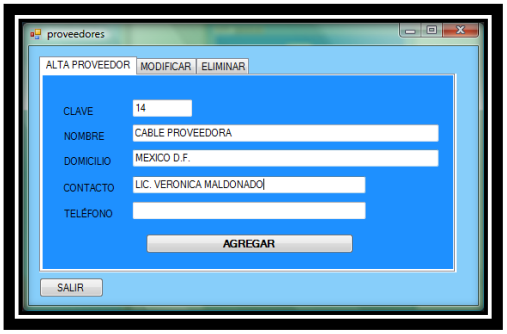


Fig. 5.1, Alta de datos de proveedor

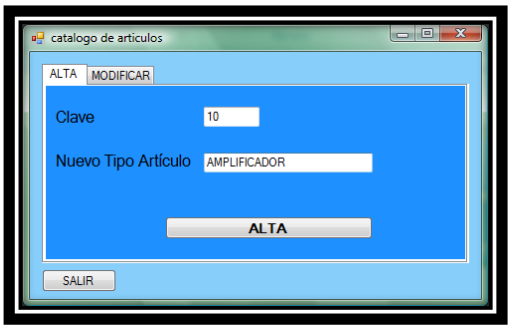


Fig. 5.2, Alta de tipo de artículos

- Alta de usuarios

  Se dieron de alta los datos de los usuarios y empleados a la base de datos para poder realizar captura de salidas

Fig. 5.3 Pantalla de módulo Alta Empleados

- Captura de salidas y modificaciones

  Se registraron los datos requeridos para realizar una salida de material, así como la modificación en los datos de los artículos registrados en la base de datos.

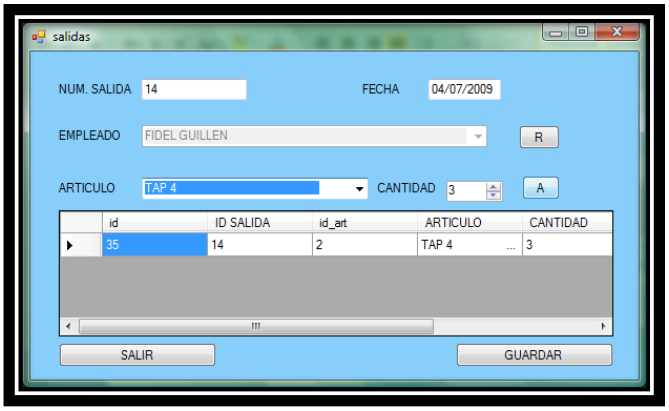


Fig. 5.4, Salida capturada en módulo Salidas

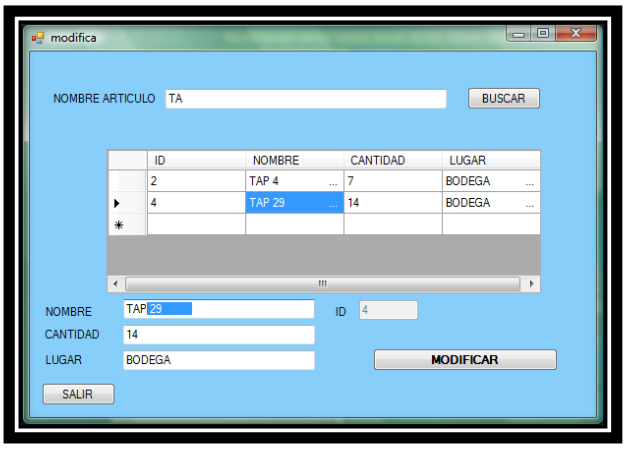


Fig. 5.5, Módulo de modificaciones

- Creación de reportes

  Se realizó la generación de cada uno de los diferentes tipos de reportes que ofrece el sistema.

Fig. 5.6, Módulo de Reportes (reporte generado de stock)

En cuanto a la aplicación desarrollada para la PDA las pruebas fueron:

- Conexión Web Service

  Se hizo la prueba de la conexión con el web service tanto para realizar la carga de artículos como la conciliación de movimientos, esta última consiste en transferir los datos de los movimientos capturados en la PDA a la base de datos del servidor.

Fig. 5.7, Módulo Conexión WS, lista de artículos mostrada de la base de datos del servidor

Fig. 5.8, Pantalla módulo Conciliar Movimientos

- Captura de movimientos

  Se realizaron varias capturas de movimientos de material para corroborar que no haya errores.

Fig. 5.9, Pantalla módulo Movimientos

- Consultas de material

  Una vez que se hizo la prueba de captura de movimientos se realizaron consultas al material cargado en la PDA para comprobar que no hubiera errores en la actualización del stock (de la base de datos de la PDA).

5.1 Posibles Fallas

- ***Errores de código***: si en algún campo de texto se introduce un carácter no válido, el sistema puede generar una excepción.
- ***Errores de funcionalidad***: cuando se tenga en uso esta aplicación es posible que el usuario desee agregar más funcionalidades al sistema que le permitan realizar otras tareas.
- ***Errores de infraestructura***: es posible que el equipo de cómputo utilizado no se encuentre en óptimas condiciones lo que imposibilitará el buen desempeño del sistema.
- ***Errores de conectividad***: para poder hacer consumo del web service de manera local se necesita de un router inalámbrico el cual está conectado al servidor y que la PDA se conecta vía wireless, se espera que el router siempre funcione correctamente para que se puedan realizar transferencias de datos entre la PDA y el servidor.

- ***Errores de usuario***: otra posible falla puede ser la falta de conocimiento del sistema y por ende el ingreso de datos erróneos, lo cual puede causar que el sistema despliegue datos incorrectos.

5.2 Posibles Riesgos

- El principal riesgo que se corre es que el sistema no sea utilizado, es decir, que la empresa una vez que realizó la inversión del software y equipo necesarios, no se ponga en marcha este nuevo sistema de control de inventario.
- El mal cuidado de los dispositivos PDA, por tratarse de un equipo electrónico siempre se debe tener las precauciones necesarias al momento de utilizarlos.

5.3 Conclusiones del desarrollo del Software

Una vez analizados los factores que intervienen en el traslado de información vía inalámbrica, desde el dispositivo por donde se envía hasta el servidor a donde llega, se definieron las herramientas y tecnologías necesarias para poder desarrollar el proyecto. Una vez desarrollado se comprobó la fiabilidad que existe hoy en día respecto a las comunicaciones inalámbricas, en particular, el intercambio de información entre una PDA y un servidor.

No cabe duda que este tipo de implementaciones se utiliza como un instrumento más para la recolección y análisis de datos, pero con la característica de la comodidad ya que los dispositivos PDA debido a su tamaño son fáciles de portar además de su eficiencia para el manejo de datos.

El desarrollo del proyecto será sumamente conveniente para la empresa TTC, pues le ayudará a mantener un mejor control de su información, en lo que respecta a una base de datos de inventarios, al poder actualizarla de una manera rápida y sencilla.

Capítulo 6

Conclusiones y Trabajo Futuro

En este capítulo se plasman las conclusiones a las que llegaron una vez terminada la investigación y desarrollo de este proyecto, además se mencionan algunas aplicaciones que se pueden desarrollar en un futuro sobre la base principal del proyecto realizado que es la conexión entre un servidor y un dispositivo móvil.

Capítulo 6 Conclusiones y Trabajo Futuro

6.0 Conclusiones

De manera personal, haber finalizado este proyecto es de gran importancia, ya que gracias a ello se enriquecen los conocimientos que se tenían previo al comienzo del trabajo de investigación, además se demuestra el esfuerzo realizado durante los cuatro años y medio de estudios de la carrera en la que se obtuvieron aprendizajes los cuales sirvieron como base para el desarrollo de este trabajo de tesis.

Durante el desarrollo de este proyecto se conocieron todos los factores que se deben tomar en cuenta para la creación de una aplicación de software, desde el estudio de la problemática y el objetivo a lograr, el análisis de las bastantes opciones de herramientas y tecnologías hoy en día disponibles, hasta la programación de dicha aplicación. El método a utilizar en un proyecto no se debe tomar a la ligera ya que si se eligen las herramientas inadecuadas las consecuencias pueden ser graves, se habrá hecho un desperdicio de tiempo y sobre todo de dinero. Por lo que todos estos aspectos fueron tomados en cuenta para el trabajo de este proyecto.

6.1 Trabajo Futuro

Con la creación del web service queda la base para desarrollar más servicios que ayuden a la mejora de la administración de la información de la empresa y por lo tanto ventajas competitivas. Algunas de las aplicaciones pueden ser:

- Actualización en tiempo real de la base de datos del inventario, es decir se evitaría que el personal (usuario del sistema) tenga que desplazarse hasta donde esté el servidor de la base de datos para poder descargar la información; pues esto no será necesario al tener la facilidad de actualizar en tiempo real a través de Internet.
- Creación de un módulo especial para la consulta de historial y cuentas de los clientes y así poder tener puntos de cobro remotos a la oficina central.

- Para cuando se llegan los cortes de servicio, en vez de tener que imprimir todas las órdenes de corte estas se pueden generar en la PDA si se desarrolla un módulo que tenga acceso a la base de datos de los clientes a través del web service.
- Unificar las bases de datos de los clientes de las distintas plazas de la empresa TTC, de esta manera se puede tener una base de datos central y se tendría acceso a ella a través del web service.

Estas mejoras y muchas más se pueden hacer al sistema actual, se irán desarrollando conforme el cliente lo requiera.

Apéndice 1

Diccionario de Datos de la Base de Datos en Servidor

► TABLA:	Articulos				
DESCRIPCIÓN:	**Relación de la información de los artículos que se manejan dentro de la empresa**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id_art	pk	Clave que se le asigna a cada artículo o material	int	4	Cont_fact.id_articulo, Detalle_Movimientos_PDA.id_art, salidas.articulo, Stock.id_art, Tempcont_fact.id_articulo, Tempsalidas.articulo
Nombre_art		Nombre del artículo	char	50	
Descripcion		Breve descripción general del artículo	char	100	
Tipo_art		Clave del tipo de artículo al que pertenece según en el catálogo de tipos de artículos	int	4	Tipo_articulos.id_tipo
Costo_art		Costo del artículo	money	8	
Unidad_art		Unidad del artículo, puede ser pieza, metro o kilómetros	char	10	
Fecha_alta		Fecha en que se registro el alta del artículo	smalldatetime	4	

► TABLA:	Cat_Areas				
DESCRIPCIÓN:	**Catálogo de las diferentes áreas de trabajo que existen en la empresa**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id_area	pk	Clave del área	int	4	Empleados.area
descripcion		Nombre del área	char	30	

► TABLA:	Cat_Prov				
DESCRIPCIÓN:	**Catálogo de las empresas proveedoras de material que se utiliza en TTC**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id_prov	pk	Clave única de identificación del proveedor	int	4	Enc_fact.proveedor
Nombre_prov		Nombre del proveedor, ya sea empresa o persona	char	70	
Dom_prov		Domicilio del proveedor	char	100	
tel		Teléfono de contacto al proveedor	char	45	
Fecha_alta		Fecha en que se registró como proveedor en el sistema	Smalldatetime	4	

► TABLA:	Cont_fact				
DESCRIPCIÓN:	**En esta tabla se almacena el contenido de las facturas que se registran a través del sistema, es decir, los ingresos de material a almacén**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id	pk	Consecutivo aplicado a esta tabla para evitar filas duplicadas	int	4	
Id_factura		Número consecutivo que se le asigna a cada factura	int	4	Enc_fact.id_factura
Id_articulo		Clave del artículo contenido en esa factura	int	4	Articulos.id_art
Cant_articulo		Cantidad del artículo cargado en la factura	int	4	
iva		Porcentaje del IVA de cada artículo	decimal	18,0	
Costo_articulo		Costo del artículo	money	8	

► TABLA:	Detalle_Movimientos_PDA				
DESCRIPCIÓN:	**Tabla para almacenar el detalle de los movimientos registrados en las PDA's, los datos se obtienen por medio del web service**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id_mov	pk	Consecutivo de número de cada	int	4	Movimientos_PDA.id_mov

		movimiento			
detalle		Descripción del movimiento	nchar	15	
Cantidad_art		Cantidad del artículo utilizado en movimiento	int	4	
Id_art		Id del artículo utilizado	int	4	Articulos.id_art
Id_servicio		Número de la orden de servicio en la que fue utilizado el artículo	int	4	

► TABLA:	**Empleados**				
DESCRIPCIÓN:	**Registro de los datos de los empleados de la empresa y área de trabajo**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id	pk	Id del usuario en el sistema	int	4	Usuarios.id_emp, Salidas.empleado
nombre		Nombre del empleado	char	50	
area		Clave del área de la empresa en la que labora	int	4	Cat_areas.id_area
Fecha_alta		Fecha del registro del empleado en la base de datos	smalldatetime	4	

► TABLA:	**Enc_fact**				
DESCRIPCIÓN:	**Se utiliza para almacenar los encabezados de cada factura, es decir, los datos generales de las mismas**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id_factura	pk	Consecutivo del Id de las facturas	int	4	Cont_fact.id_factura
Num_factura		Número de cada factura	nchar	10	
Proveedor		Clave del proveedor de acuerdo al catálogo de proveedores	int	4	Cat_prov.id_prov
Cantidad_art		Cantidad total de material adquirido en la factura	int	4	
subtotal		Subtotal de la factura	money	8	
Total_neto		Gran Total de la factura	money	8	
Fecha_factura		Fecha de la	datetime	8	

		factura o de la compra del material			
Fecha_registro		Fecha de registro de la factura en la base de datos	smalldatetime	4	

► TABLA:	Movimientos_PDA				
DESCRIPCIÓN:	**Complemento de la tabla Detalle_Movimientos_PDA, aquí se almacenan otros datos de los movimientos registrados en las PDA's y también se obtienen a través del web service**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id_mov	pk	Número de movimiento	int	4	Detalle_Movimientos_PDA.id_mov
Tipo_mov		Tipo de movimiento que se realiza al artículo	char	30	
Id_usuario		Id del usuario o empleado que realiza movimiento	int	4	Usuarios.id_emp
Fecha_mov		Fecha en la que se realiza movimiento	datetime	8	

► TABLA:	Salidas				
DESCRIPCIÓN:	**En esta tabla se almacenan los datos que corresponden a las salidas de material de almacén**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
id	pk	Id llave principal	int	4	
num_salida		Consecutivo correspondiente al número de salida	int	4	
articulo		Id del artículo que sale de almacén	int	4	Articulos.id_art
cantidad		Cantidad del artículo saliente	int	4	
empleado		Id del empleado que recibe el material	int	4	Empleados.id
fecha_entrega		Fecha en que se entrega el material al empleado	datetime	8	
fecha		Fecha de registro de salida en la base de datos	datetime	8	

► TABLA:	Stock				
DESCRIPCIÓN:	**Aquí se almacenará la cantidad en existencia de cada artículo y su ubicación**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id_art		Id del artículo	int	4	Articulos.id_art
cantidad		Existencia en almacén del artículo	int	4	
lugar		Ubicación del artículo	char	30	

► TABLA:	Tempcont_fact				
DESCRIPCIÓN:	**Se utiliza para almacenar temporalmente los contenidos de la factura, estos datos después son copiados a la tabla cont_fact**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
id		Consecutivo de las filas que se van agregando	int	4	
Id_factura		Número de la factura	int	4	Enc_fact.id_factura
Id_articulo		Id del artículo contenido en la factura	int	4	Articulos.id_art
iva		Porcentaje del IVA	decimal	18,0	
Costo_articulo		Costo del artículo adquirido	nchar	10	

► TABLA:	Tempsalidas				
DESCRIPCIÓN:	**Utilizada para almacenar temporalmente los datos correspondientes a las salidas de material de almacén, esto datos son copiados a la tabla salidas**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
id		Consecutivo de las filas de la tabla	int	4	
Num_salida		Número de salida de material de almacén	int	4	
articulo		Id del artículo que sale de almacén	int	4	Articulos.id_art
cantidad		Cantidad del artículo que sale	int	4	
empleado		Id del empleado al que se le entrega el artículo	int	4	Empleados.id
Fecha_entrega		Fecha en que es entregado el artículo a quien lo	datetime	8	

		solicita			

► TABLA:	Tipo_Articulos				
DESCRIPCIÓN:	**En esta tabla están contenidos los datos de los tipos de artículos que el usuario o administrador va dando de alta**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id_tipo	pk	Id correspondiente al tipo de artículo	int	4	Articulos.tipo_art
descripcion		Nombre que sirve para identificar a los artículos que sean del mismo tipo	char	40	

► TABLA:	Usuarios				
DESCRIPCIÓN:	**Aquí se almacenan los datos de los usuarios del sistema**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id_usuario	pk	Id de usuario del sistema	int	4	
Nombre_usuario		Nombre de usuario para acceso al sistema	char	10	
Password_usuario		Password del usuario para acceso al sistema	char	10	
Status_usuario		Estatus del usuario, si está en alta o ya ha sido dado de baja	char	10	
Fecha_alta		Fecha en que se da de alta al usuario en el sistema	smalldatetime	4	
Id_emp		Id del empleado que le corresponde	int	4	Empleados.id, Movimientos_PDA.id_usuario

Apéndice 2

Diccionario de Datos de la Base de Datos en Dispositivo Móvil

► TABLA:	Articulos				
DESCRIPCIÓN:	**En ella se almacena la información general referente a cada artículo que se transfiere desde la base de datos del servidor**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
id_art	Pk	Id del artículo, es el mismo que tiene en la base de datos del servidor	int	4	Detalle_Movimientos.id_articulo
nombre_art		Nombre del artículo	nchar	35	
cantidad		Cantidad del artículo que hay almacenado en esta base de datos	int	4	
unidad		Unidad de medida del artículo	nchar	25	

► TABLA:	Detalle_Movimientos				
DESCRIPCIÓN:	**Complemento de la tabla de Movimientos, aquí se almacenan datos relacionados con los movimientos de salida o traslado que se registran en el dispositivo; estos datos son transferidos a la base de datos del servidor para contar con un historial**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id_mov		Consecutivo del número de movimiento	int	4	Movimientos.id_mov
Cantidad_art		Cantidad del artículo que se utilizó en el movimiento	int	4	
id_articulo		Id del artículo utilizado	int	4	Articulos.id_art
detalle		Detalle del movimiento, puede ser acometida o reemplazo	nchar	25	
id_servicio		Número de la orden de servicio en el que es utilizado el artículo	int	4	

capturado		Bandera para diferenciar los movimientos que ya fueron transferidos al servidor de los que no, puede ser SI o NO	nchar	4	

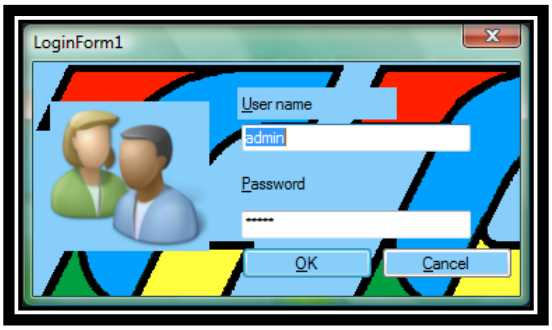
► TABLA:	**Movimientos**				
DESCRIPCIÓN:	**En esta tabla se almacenan los movimientos de salida o traslado de material que registra el usuario del dispositivo**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
Id_mov	Pk	Número id de movimiento	int	4	Detalle_Movimientos.id_mov
tipo_mov		Tipo de movimiento realizado con el artículo, puede ser Salida o Traslado	Nchar	25	
Id_usuario		Id de empleado del usuario que registra el movimiento	Int	4	Usuarios.id_emp
Fecha_mov		Fecha del movimiento	Datetime	8	
capturado		Bandera para diferenciar los movimientos que ya fueron transferidos al servidor de los que no, puede ser SI o NO	nchar	4	

► TABLA:	**Usuarios**				
DESCRIPCIÓN:	**Contendrá los datos de los usuarios que podrán accesar al sistema del dispositivo**				
CAMPOS:	**LLAVE**	**DESCRIPCIÓN**	**TIPO**	**LONGITUD**	**TABLA RELACIONADA**
id	Pk	Id de usuario de dispositivo	int	4	
nombre		Nombre de usuario	nchar	15	
password		Clave de acceso del usuario	nchar	10	
Id_emp		Id de empleado que le corresponde al del usuario del dispositivo	Int	4	Movimientos.id_usuario

Apéndice 3

Ventanas y scripts de programación aplicación de escritorio

Ventana login



Script de programación:

```
Imports System.Data
Imports System.Data.SqlClient

Public Class LoginForm1

    Dim conexion As New SqlConnection()
    Public str As String = "Data Source=JHUERTA-PC\SQLEXPRESS;Initial
Catalog=inventariottc;Persist Security Info=False;User ID=sa;Password=svsql2k3;"

        Private Sub OK_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles OK.Click
        Dim user, pass As String

        Try

            Dim comando As New SqlCommand("SELECT
id_usuario,nombre_usuario,password_usuario FROM usuarios where nombre_usuario='" +
usuario.Text + "' AND password_usuario='" + password.Text + "'", conexion)
            ' Crea el DataReader
            Dim lectorDatos As SqlDataReader

            lectorDatos = comando.ExecuteReader()

            If lectorDatos.Read() Then
                 menu1.Visible = True
            Else
                MsgBox("Nombre de usuario o contraseña incorrecto",
MsgBoxStyle.Exclamation)
                password.Focus()
                lectorDatos.Close()
                Return
            End If
            ' Cierra el objeto DataReader
            lectorDatos.Close()
            ' Desconecta de la Base de Datos
            conexion.Close()

        Catch ex As Exception
            MsgBox(ex.Message)
```

```
        End Try

        Me.Visible = False
    End Sub

    Private Sub Cancel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Cancel.Click
        Me.Close()
    End Sub

    Private Sub LoginForm1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        usuario.Text = "admin"
        password.Text = "admin"

        conexion.ConnectionString = str
        If conexion.State <> Data.ConnectionState.Open Then
            conexion.Open()
        End If

    End Sub

    Private Sub UsernameLabel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles UsernameLabel.Click

    End Sub
End Class
```

Menú principal.

Script de programación:

```
Imports System.Data
Imports System.Data.SqlClient

Public Class menu1

    Dim conexion As New SqlConnection()
```

```
    Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button5.Click
        LoginForm1.Dispose()

    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
        catalogo.Visible = True
    End Sub

    Private Sub menu1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        conexion.ConnectionString = LoginForm1.str
        If conexion.State <> Data.ConnectionState.Open Then
            conexion.Open()
        End If
    End Sub

    Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button6.Click
        proveedores.Visible = True
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        modifica.Visible = True
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        'alta.Visible = True
        facturas.Visible = True
    End Sub

    Private Sub Button7_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button7.Click
        salidas.Visible = True
    End Sub

    Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button8.Click

Dim p As New System.Diagnostics.Process
        p.StartInfo.FileName = "C:\Users\JHuerta\JAH\TESIS\PROG MODULO DE REPORTES
INVTTC\reportes.exe"
        p.Start()

    End Sub

    Private Sub Button9_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button9.Click
        alta.Visible = True
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        empleados.Visible = True
    End Sub
End Class
```

Módulo Alta Empleados.

Script de programación:

```
Imports System.Data
Imports System.Data.SqlClient
Public Class empleados

    Private Sub empleados_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        Me.Cat_AreasTableAdapter.Fill(Me.InventariottcDataSet.Cat_Areas)
        ComboBox1.Text = ""

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        Dim sql As String

        sql = "Insert into Empleados (nombre, area) values('" + nombre.Text + "','"
& ComboBox1.SelectedValue & "')"

        Try

            Dim conexion As New SqlConnection()
            conexion.ConnectionString = LoginForm1.str

            conexion.Open()

            Dim transaction As SqlTransaction
            transaction = conexion.BeginTransaction("Transaccion SQL")

            ' Crea el comando
            Dim comando As New SqlCommand(sql, conexion)


            comando.Connection = conexion
            comando.Transaction = transaction

            Dim result As Integer

            result = comando.ExecuteNonQuery()

            transaction.Commit()
            MsgBox("EMPLEADO DADO DE ALTA")
            transaction.Rollback()
            conexion.Close()
```

```
        Catch ex As Exception


        End Try

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Me.Close()
    End Sub
End Class
```

Módulo Catálogo Artículos

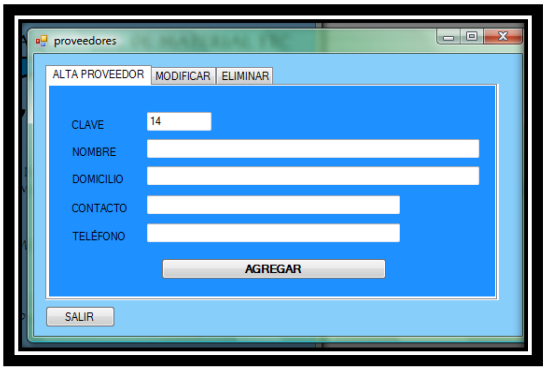
Script de programación:

```
Imports System.Data
Imports System.Data.SqlClient

Public Class catalogo

    Dim conexion As New SqlConnection()

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim sql As String

        Try

            sql = "Insert into Tipo_Articulos (descripcion) values('" +
TextBox1.Text + "')"

            Dim transaction As SqlTransaction
            transaction = conexion.BeginTransaction("Transaccion SQL")

            ' Crea el comando
            Dim comando As New SqlCommand(sql, conexion)

            comando.Connection = conexion
            comando.Transaction = transaction

            Dim result As Integer
            'conexion.Open()
            result = comando.ExecuteNonQuery()

            transaction.Commit()
```

```
            MsgBox("ALMACENADO")
            transaction.Rollback()

            Catch ex As Exception
        End Try
    End Sub

    Private Sub catalogo_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        conexion.ConnectionString = LoginForm1.str
        If conexion.State <> Data.ConnectionState.Open Then
            conexion.Open()
        End If

        Me.Tipo_ArticulosTableAdapter.Fill(Me.InventariottcDataSet.Tipo_Articulos)

        Dim id As Integer
        Try
            ' Crea el comando
            Dim comando As New SqlCommand("SELECT MAX(id_tipo) as idtipo FROM
Tipo_Articulos", conexion)
            ' Crea el DataReader
            Dim lectorDatos As SqlDataReader
            ' Conecta con la Base de Datos

            lectorDatos = comando.ExecuteReader()

            If lectorDatos.Read() Then
                id = (lectorDatos("idtipo"))
            End If
            ' Cierra el objeto DataReader
            lectorDatos.Close()

        Catch ex As Exception

        End Try
        id = id + 1
        TextBox2.Text = id
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Me.Close()
    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged

        Try
            ' Crea el comando
            Dim comando As New SqlCommand("SELECT * FROM Tipo_Articulos WHERE
id_tipo='" & ComboBox1.SelectedValue & "'", conexion)
            ' Crea el DataReader
            Dim lectorDatos As SqlDataReader

            lectorDatos = comando.ExecuteReader()

            If lectorDatos.Read() Then
                TextBox3.Text = (lectorDatos("descripcion"))
            End If
            ' Cierra el objeto DataReader
            lectorDatos.Close()
```

```
        Catch ex As Exception

        End Try


    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        Dim sql As String

        sql = "UPDATE Tipo_Articulos SET descripcion='" + TextBox3.Text + "' WHERE
id_tipo='" & ComboBox1.SelectedValue & "'"

        Try

            Dim conexion As New SqlConnection()
            conexion.ConnectionString = LoginForm1.str
            conexion.Open()

            Dim transaction As SqlTransaction
            transaction = conexion.BeginTransaction("Transaccion SQL")
            ' Crea el comando
            Dim comando As New SqlCommand(sql, conexion)

            comando.Connection = conexion
            comando.Transaction = transaction

            Dim result As Integer
            result = comando.ExecuteNonQuery()

            transaction.Commit()
            MsgBox("MODIFICACIÓN REALIZADA")
            transaction.Rollback()

            conexion.Close()

        Catch ex As Exception

        End Try
    End Sub
End Class
```

Módulo Catálogo de Proveedores.



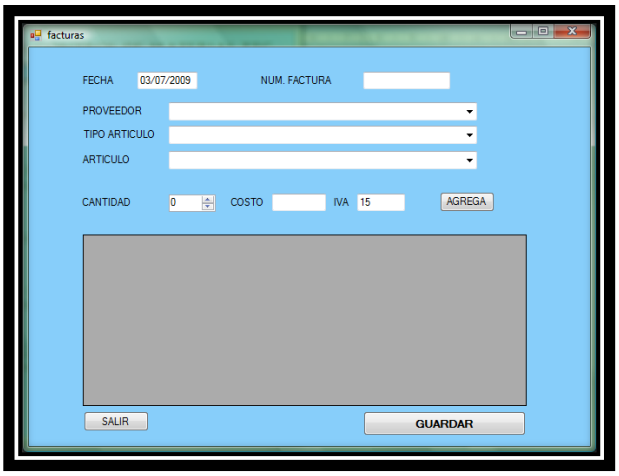
Script de programación:

```
Imports System.Data
Imports System.Data.SqlClient

Public Class proveedores

    Dim conexion As New SqlConnection()

    Private Sub proveedores_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        conexion.ConnectionString = LoginForm1.str
        If conexion.State <> Data.ConnectionState.Open Then
            conexion.Open()
        End If

            Me.Cat_ProvTableAdapter.Fill(Me.InventariottcDataSet.Cat_Prov)

            Dim id As Integer

            Try
                Dim comando As New SqlCommand("SELECT MAX(id_prov) AS idmax FROM
Cat_prov", conexion)
                ' Crea el DataReader
                Dim lectorDatos As SqlDataReader
                lectorDatos = comando.ExecuteReader()

                If lectorDatos.Read() Then
                    id = (lectorDatos("idmax"))
                End If
                lectorDatos.Close()
            Catch ex As Exception

            End Try
            id = id + 1
            TextBox1.Text = id
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim sql As String

        sql = "Insert into Cat_Prov (nombre_prov,dom_prov,contacto,tel) values('" +
TextBox2.Text + "','" + TextBox3.Text + "','" + TextBox4.Text + "','" +
TextBox5.Text + "')"
        Try

            Dim transaction As SqlTransaction
            transaction = conexion.BeginTransaction("Transaccion SQL")

            Dim comando As New SqlCommand(sql, conexion)

            comando.Connection = conexion
            comando.Transaction = transaction

            Dim result As Integer
            result = comando.ExecuteNonQuery()

            transaction.Commit()
            MsgBox("DATOS ALMACENADOS")

            transaction.Rollback()
```

```
        Catch ex As Exception
        End Try

    End Sub


    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Dim sql As String

        sql = "UPDATE Cat_Prov SET nombre_prov='" + TextBox7.Text + "',dom_prov='" +
TextBox8.Text + "',contacto='" + TextBox9.Text + "',tel='" + TextBox10.Text + "'
WHERE id_prov='" + TextBox6.Text + "'"

        Try

            Dim conexion As New SqlConnection()
            conexion.ConnectionString = LoginForm1.str
            conexion.Open()

            Dim transaction As SqlTransaction
            transaction = conexion.BeginTransaction("Transaccion SQL")
            ' Crea el comando
            Dim comando As New SqlCommand(sql, conexion)

            comando.Connection = conexion
            comando.Transaction = transaction

            Dim result As Integer
            result = comando.ExecuteNonQuery()

            transaction.Commit()
            MsgBox("MODIFICACIÓN REALIZADA")
            transaction.Rollback()

            conexion.Close()

        Catch ex As Exception

        End Try

    End Sub


    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        MsgBox("ESTA SEGURO QUE DESEA ELIMINAR AL PROVEEDOR SELECCIONADO?",
MsgBoxStyle.YesNo)
        If MsgBoxResult.Yes Then
            Dim sql As String
            sql = "DELETE FROM Cat_Prov WHERE id_prov='" & ComboBox2.SelectedValue &
"'"

            Try

                Dim conexion As New SqlConnection()
                conexion.ConnectionString = LoginForm1.str
                conexion.Open()

                Dim transaction As SqlTransaction
                transaction = conexion.BeginTransaction("Transaccion SQL")

                ' Crea el comando
```

```
                Dim comando As New SqlCommand(sql, conexion)

                comando.Connection = conexion
                comando.Transaction = transaction

                Dim result As Integer
                result = comando.ExecuteNonQuery()

                transaction.Commit()
                MsgBox("EL PROVEEDOR HA SIDO ELIMINADO DE LA BASE DE DATOS")
                transaction.Rollback()
                conexion.Close()

            Catch ex As Exception

            End Try
        End If

    End Sub


    Private Sub ComboBox1_SelectedIndexChanged_1(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged
        Try
            ' Crea el comando
            Dim comando As New SqlCommand("SELECT * FROM Cat_prov WHERE id_prov='" &
ComboBox1.SelectedValue & "'", conexion)
            ' Crea el DataReader
            Dim lectorDatos As SqlDataReader

            lectorDatos = comando.ExecuteReader()

            If lectorDatos.Read() Then
                TextBox6.Text = (lectorDatos("id_prov"))
                TextBox7.Text = (lectorDatos("nombre_prov"))
                TextBox8.Text = (lectorDatos("dom_prov"))
                TextBox9.Text = (lectorDatos("contacto"))
                TextBox10.Text = (lectorDatos("tel"))
            End If
            ' Cierra el objeto DataReader
            lectorDatos.Close()
        Catch ex As Exception

        End Try

    End Sub

    Private Sub TabPage2_GotFocus(ByVal sender As Object, ByVal e As
System.EventArgs) Handles TabPage2.GotFocus
        ComboBox1.Text = ""
    End Sub

    Private Sub TabPage1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles TabPage1.Click

    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
        Me.Close()
    End Sub
End Class
```

Módulo Ingreso Material.



Script de programación:

```
Imports System.Data
Imports System.Data.SqlClient

Public Class facturas
    Dim z As Integer
    Dim id_factura, id_art As Integer


    Private Sub facturas_Disposed(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Disposed

        'BORRAR LAS FILAS DE LAS ULTIMAS INSERCIONES A TABLA TEMPCONT_FACT EN
        'CASO DE SALIR DE LA VENTANA SIN ALMACENAR LA FACTURA
        Dim sql As String
        sql = "delete from tempcont_fact where id_factura=" & id_factura & ""

        Try

            Dim conexion As New SqlConnection()
            'conexion.ConnectionString = "Data Source=DESKTOP\SQLEXPRESS;Initial
Catalog=inventariottc;User ID=sa;Password=svsql2k3"
            conexion.ConnectionString = LoginForm1.str

            ' Crea el comando
            Dim comando As New SqlCommand(sql, conexion)

            Dim result3 As Integer

            conexion.Open()
            result3 = comando.ExecuteNonQuery()


            conexion.Close()

        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub
```

```
    Private Sub facturas_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Me.Tipo_ArticulosTableAdapter.Fill(Me.InventariottcDataSet1.Tipo_Articulos)
        Me.ArticulosTableAdapter.Fill(Me.InventariottcDataSet1.Articulos)
        Me.Cat_ProvTableAdapter.Fill(Me.InventariottcDataSet1.Cat_Prov)

        fecha.Text = Today()
        iva.Text = 15

        ComboBox1.Text = ""
        ComboBox2.Text = ""
        ComboBox3.Text = ""

        'PATA ASIGNAR EL ID DE FACTURA
        Dim sql As String

        sql = "SELECT MAX(id_factura) AS id_factura FROM enc_fact"

            Try

            Dim conexion As New SqlConnection()
            conexion.ConnectionString = LoginForm1.str
            ' Crea el comando
            Dim comando As New SqlCommand(sql, conexion)

            Dim result2 As SqlDataReader
            conexion.Open()

            result2 = comando.ExecuteReader()
            If result2.Read() Then
                id_factura = (result2("id_factura"))
            End If

            conexion.Close()

        Catch ex As Exception

        End Try
        id_factura = id_factura + 1

    End Sub

    Private Sub DomainUpDown1_SelectedItemChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs)

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click


        z = z + 1


        Dim cantidad, stock, iva3 As Integer
        Dim costo2, iva2, totalc, cantneto As Decimal


        cantidad = NumericUpDown1.Value
        costo2 = Convert.ToDecimal(costo.Text)
        iva2 = Convert.ToDecimal(iva.Text)
        iva3 = Convert.ToDecimal(iva.Text)
        iva2 = iva2 / 100
```

```
        totalc = Format((cantidad * costo2), "###0.#0")
        cantneto = totalc + (totalc * iva2)


        Dim sql, sql1, sql2, sql3, sql4 As String

              ' If z <= 1 Then
        sql2 = "insert into tempcont_fact (id_factura, id_articulo, cant_articulo,
iva, costo_articulo) values (" & id_factura & ",'" & id_art & "','" & cantidad &
"','" & iva3 & "'," & costo2 & ")"

        Try

            Dim conexion As New SqlConnection()
            conexion.ConnectionString = LoginForm1.str

            conexion.Open()

            Dim transaction As SqlTransaction
            transaction = conexion.BeginTransaction("Transaccion SQL")

            ' Crea el comando
            Dim comando As New SqlCommand(sql2, conexion)

            comando.Connection = conexion
            comando.Transaction = transaction

            Dim result3 As Integer


            result3 = comando.ExecuteNonQuery()

            transaction.Commit()
            transaction.Rollback()

            conexion.Close()

        Catch ex As Exception
        End Try
        ' End If

        Dim sr As New dataservice
        DataGridView1.DataSource = sr.GetDataSet("Exec cont_fact1 " & id_factura &
"").Tables(0)
    End Sub

    Private Sub ComboBox2_SelectedIndexChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ComboBox2.SelectedIndexChanged

        Dim sql As String

        sql = ("Select id_art, costo_art From Articulos where nombre_art LIKE'" +
ComboBox2.Text + "'")

        Try

            Dim conexion As New SqlConnection()
            conexion.ConnectionString = LoginForm1.str
            ' Crea el comando
            Dim comando As New SqlCommand(sql, conexion)
            Dim result As SqlDataReader
            conexion.Open()
            result = comando.ExecuteReader()
```

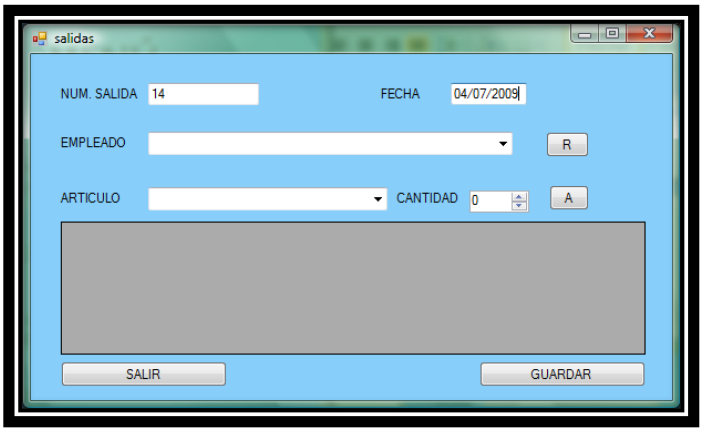
```
            While result.Read()
                id_art = (result("id_art"))
                costo.Text = Format((result("costo_art")), "###0.#0")
            End While

        Catch ex As Exception

        End Try
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click


        Dim ban As Integer = 0
        Dim r As Integer
        Dim s As String
        Dim iv, cant, cost, tot, total, subt As Decimal
        Dim suma As Decimal = 0
        r = DataGridView1.RowCount
        s = DataGridView1.CurrentRow.Cells(0).Value.ToString()
        Try


            For i As Integer = 0 To DataGridView1.RowCount
                cant = DataGridView1.Rows.Item(i).Cells(3).Value
                iv = DataGridView1.Rows.Item(i).Cells(4).Value
                cost = DataGridView1.Rows.Item(i).Cells(5).Value
                tot = cant * cost
                total = Format((total + tot), "######0.#0")
            Next

        Catch ex As Exception

        End Try
        subt = Format((total - (total * 0.15)), "######0.#0")
        MsgBox("suma total")
        MsgBox(total)
        MsgBox(subt)

        'ACTUALIZA ENCABEZADO FACTURA
        Dim sql As String

        sql = "UPDATE enc_fact SET subtotal=" & subt & ", total_neto=" & total & "
WHERE id_factura='" & id_factura & "'"

        Try

            Dim conexion As New SqlConnection()
            conexion.ConnectionString = LoginForm1.str
            ' Crea el comando
            Dim comando As New SqlCommand(sql, conexion)


            Dim result As Integer
            conexion.Open()
            result = comando.ExecuteNonQuery
            MsgBox("actualizado")
            conexion.Close()


        Catch ex As Exception
```

```
        End Try
    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged
        ComboBox1.Enabled = False
    End Sub


    Private Sub DataGridView1_DoubleClick(ByVal sender As Object, ByVal e As
System.EventArgs) Handles DataGridView1.DoubleClick

        Dim sr As New dataservice
        DataGridView1.DataSource = sr.GetDataSet("Exec elim_cont_fact " &
DataGridView1.CurrentRow.Cells(0).Value & "")
        DataGridView1.DataSource = sr.GetDataSet("Exec cont_fact1 " & id_factura &
"").Tables(0)
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click

        Dim fecha2 As DateTime
        Dim f3 As String
        fecha2 = Convert.ToDateTime(fecha.Text)

        f3 = Month(fecha2) & "-" & Microsoft.VisualBasic.DateAndTime.Day(fecha2) &
"-" & Year(fecha2)
        Dim iv, cant, cost, tot, total, subt As Decimal
        Dim id, id_articulo As Integer
        Dim nombre_art, sql, sql2, sql_1 As String
        Dim conexion As New SqlConnection()
        Dim result, result2, result_1 As Integer

        'PARA GUARDAR DE TABLA TEMPORAL CONT_FACT A CONTENIDO DE FACTURA
        Try

            For i As Integer = 0 To DataGridView1.RowCount
                'id = DataGridView1.Rows.Item(i).Cells(0).Value
                id_articulo = DataGridView1.Rows.Item(i).Cells(1).Value
                nombre_art = DataGridView1.Rows.Item(i).Cells(2).Value.ToString
                cant = DataGridView1.Rows.Item(i).Cells(3).Value
                iv = DataGridView1.Rows.Item(i).Cells(4).Value
                cost = DataGridView1.Rows.Item(i).Cells(5).Value
                tot = cant * cost
                total = Format((total + tot), "######0.#0")

                Try
                    sql = "insert into cont_fact (id_factura, id_articulo,
cant_articulo, iva, costo_articulo) values (" & id_factura & "," & id_articulo &
",'" & cant & "','" & iv & "'," & cost & ")"
                    sql_1 = "UPDATE Stock SET cantidad = cantidad +" & cant & "
WHERE id_art= " & id_articulo & ""

                    conexion.ConnectionString = LoginForm1.str

                    conexion.Open()
                    ' Crea el comando
                    Dim transaction As SqlTransaction
                    transaction = conexion.BeginTransaction("Transaccion SQL")

                    Dim comando As New SqlCommand(sql, conexion)
```

```
                    Dim comando_1 As New SqlCommand(sql_1, conexion)

                    comando.Connection = conexion
                    comando.Transaction = transaction
                    comando_1.Connection = conexion
                    comando_1.Transaction = transaction


                    result = comando.ExecuteNonQuery()
                    result_1 = comando_1.ExecuteNonQuery()

                    transaction.Commit()
                    transaction.Rollback()

                    conexion.Close()
                    comando.Cancel()
                    comando_1.Cancel()
                Catch ex As Exception
                    'MsgBox(ex.Message)
                End Try
            Next

        Catch ex As Exception
            'MsgBox(ex.Message)
        End Try

        subt = Format((total - (total * 0.15)), "######0.#0")

        'PARA ACTUALIZAR SUBTOTAL Y TOTAL DE ENCABEZADO FACTURA
        'PARA ALMACENAR EL ENCABEZADO FACTURA
        Try
            sql2 = "Insert into enc_fact(id_factura,num_factura,
proveedor,subtotal,total_neto, fecha_factura) values(" & id_factura & ",'" +
num_factura.Text + "','" & ComboBox1.SelectedValue & "','" & subt & "','" & total &
"','" & f3 & "')"
            'sql2 = "UPDATE enc_fact SET subtotal=" & subt & ", total_neto=" & total
& " WHERE id_factura='" & id_factura & "'"
            Dim transaction As SqlTransaction
            transaction = conexion.BeginTransaction("Transaccion SQL")
            Dim comando2 As New SqlCommand(sql2, conexion)

            ' conexion.Open()

            comando2.Connection = conexion
            comando2.Transaction = transaction


            result2 = comando2.ExecuteNonQuery()

            transaction.Commit()
            MsgBox("ALMACENADA")
            transaction.Rollback()

            conexion.Close()
        Catch ex As Exception

        End Try

    End Sub

    Private Sub NumericUpDown1_ValueChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles NumericUpDown1.ValueChanged
```

```
    End Sub

    Private Sub ComboBox3_SelectedIndexChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ComboBox3.SelectedIndexChanged
        ComboBox2.Items.Clear()
        Dim sql As String

        sql = ("Select nombre_art From Articulos where tipo_art=" &
ComboBox3.SelectedValue & "")

        Try

            Dim conexion As New SqlConnection()
            conexion.ConnectionString = LoginForm1.str
            ' Crea el comando
            Dim comando As New SqlCommand(sql, conexion)
            Dim result As SqlDataReader
            conexion.Open()
            result = comando.ExecuteReader()
            While result.Read()
                Me.ComboBox2.Items.Add(result("nombre_art"))
            End While
        Catch ex As Exception

        End Try
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
        Me.Close()
    End Sub
End Class
```

Módulo Salidas



Script de programación:

```
Imports System.Data
Imports System.Data.SqlClient

Public Class salidas

    Dim fechas As DateTime
```

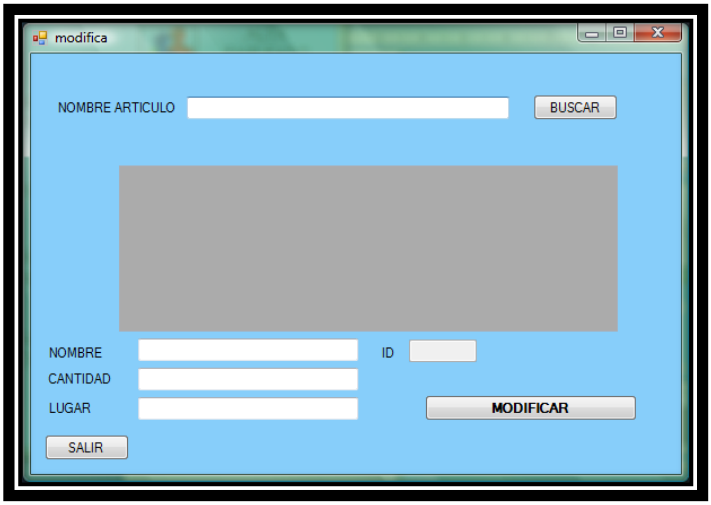
```
    Dim nsalida As Decimal
    Dim stock, cantidad As Integer
    Dim f2 As String

    Private Sub salidas_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Me.EmpleadosTableAdapter.Fill(Me.InventariottcDataSet1.empleados)
        Me.ArticulosTableAdapter.Fill(Me.InventariottcDataSet1.Articulos)
        Me.UsuariosTableAdapter.Fill(Me.InventariottcDataSet1.usuarios)

        ComboBox1.Text = ""
        ComboBox2.Text = ""
        fecha.Text = Today()

        Dim sql1 As String
        Dim numsal As Integer
        sql1 = "SELECT MAX(num_salida) as num_salida from salidas"
        Try

            Dim conexion As New SqlConnection()
            conexion.ConnectionString = LoginForm1.str
            Dim comando As New SqlCommand(sql1, conexion)
            Dim result As SqlDataReader
            conexion.Open()
            result = comando.ExecuteReader()
            If result.Read() Then
                numsal = (result("num_salida"))
            End If
            conexion.Close()
        Catch ex As Exception
        End Try
        salida.Text = numsal + 1

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
       Try

            nsalida = Convert.ToDecimal(salida.Text)
            fechas = Convert.ToDateTime(fecha.Text)
            f2 = Month(fechas) & "-" & Microsoft.VisualBasic.DateAndTime.Day(fechas)
& "-" & Year(fechas)
        Catch ex As Exception
        End Try

        cantidad = NumericUpDown1.Value

        'CHECA SI HAY SUFICIENTE CANTIDAD DE ARTICULO
        Dim sql, sql1 As String
        sql = "SELECT cantidad from Stock where id_art='" & ComboBox2.SelectedValue
& "'"
        Try

            Dim conexion As New SqlConnection()
            conexion.ConnectionString = LoginForm1.str
            Dim comando As New SqlCommand(sql, conexion)
            Dim result As SqlDataReader
            conexion.Open()
            result = comando.ExecuteReader()
            If result.Read() Then
                stock = (result("cantidad"))
            End If
```

```
            conexion.Close()
        Catch ex As Exception
        End Try
        If cantidad > stock Then
            MsgBox("NO HAY SUFICIENTE CANTIDAD DEL ARTÍCULO SELECCIONADO")
            Return
        End If

        sql1 = "Insert into tempsalidas(num_salida, articulo, cantidad, empleado,
fecha_entrega) values('" & nsalida & "','" & ComboBox2.SelectedValue & "','" &
cantidad & "','" & ComboBox1.SelectedValue & "','" & f2 & "')"
        Try

            Dim conexion As New SqlConnection()
            conexion.ConnectionString = LoginForm1.str

            conexion.Open()

            Dim transaction As SqlTransaction
            transaction = conexion.BeginTransaction("Transaccion SQL")

            Dim comando As New SqlCommand(sql1, conexion)

            comando.Connection = conexion
            comando.Transaction = transaction

            Dim result1 As Integer
            result1 = comando.ExecuteNonQuery()

            transaction.Commit()
            transaction.Rollback()

            conexion.Close()
        Catch ex As Exception
        End Try

        Dim sr As New dataservice
        DataGridView1.DataSource = sr.GetDataSet("Exec salida " & nsalida &
"").Tables(0)

        ComboBox2.Focus()

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

        Dim id_art, id_emp As Integer
        Dim sql, sql2 As String
        Dim conexion As New SqlConnection()
        Dim result, result2 As Integer

        'PARA GUARDAR DE TABLA TEMPORAL SALIDAS A SALIDAS Y ACTUALIZA STOCK
        Try

            For i As Integer = 0 To DataGridView1.RowCount
                nsalida = DataGridView1.Rows.Item(i).Cells(1).Value
                id_art = DataGridView1.Rows.Item(i).Cells(2).Value
                cantidad = DataGridView1.Rows.Item(i).Cells(4).Value
                id_emp = DataGridView1.Rows.Item(i).Cells(6).Value
                Try
```

```
                    sql = "Insert into salidas(num_salida, articulo, cantidad,
empleado, fecha_entrega) values('" & nsalida & "','" & id_art & "','" & cantidad &
"','" & id_emp & "','" & f2 & "')"
                    sql2 = "UPDATE Stock SET cantidad = cantidad -" & cantidad & "
WHERE id_art= " & id_art & ""
                    conexion.ConnectionString = LoginForm1.str
                    conexion.Open()

                    Dim transaction As SqlTransaction
                    transaction = conexion.BeginTransaction("Transaccion SQL")

                    ' Crea el comando
                    Dim comando As New SqlCommand(sql, conexion)
                    Dim comando2 As New SqlCommand(sql2, conexion)

                    comando.Connection = conexion
                    comando.Transaction = transaction
                    comando2.Connection = conexion
                    comando2.Transaction = transaction

                    result = comando.ExecuteNonQuery()
                    result2 = comando2.ExecuteNonQuery()

                    transaction.Commit()
                    MsgBox("SALIDA ALMACENADA")
                    transaction.Rollback()

                    conexion.Close()
                    comando.Cancel()
                    comando2.Cancel()
                Catch ex As Exception

                End Try
            Next
        Catch ex As Exception

        End Try


    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged
        ComboBox1.Enabled = False
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        ComboBox1.Enabled = True
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
        Try
            Dim conexion As New SqlConnection()
            Dim result As Integer
            Dim sql As String
            sql = "DELETE FROM tempsalidas where num_salida='" & nsalida & "'"
            conexion.ConnectionString = LoginForm1.str
            Dim comando As New SqlCommand(sql, conexion)
            conexion.Open()
            result = comando.ExecuteNonQuery()
            conexion.Close()
```

```
        Catch ex As Exception
        End Try

        Me.Close()
    End Sub

    Private Sub DataGridView1_CellContentDoubleClick(ByVal sender As Object, ByVal e
As System.Windows.Forms.DataGridViewCellEventArgs) Handles
DataGridView1.CellContentDoubleClick
        Dim sr As New dataservice
        DataGridView1.DataSource = sr.GetDataSet("Exec elim_salida " &
DataGridView1.CurrentRow.Cells(1).Value & "")
    End Sub

End Class
```

Módulo Modificaciones.

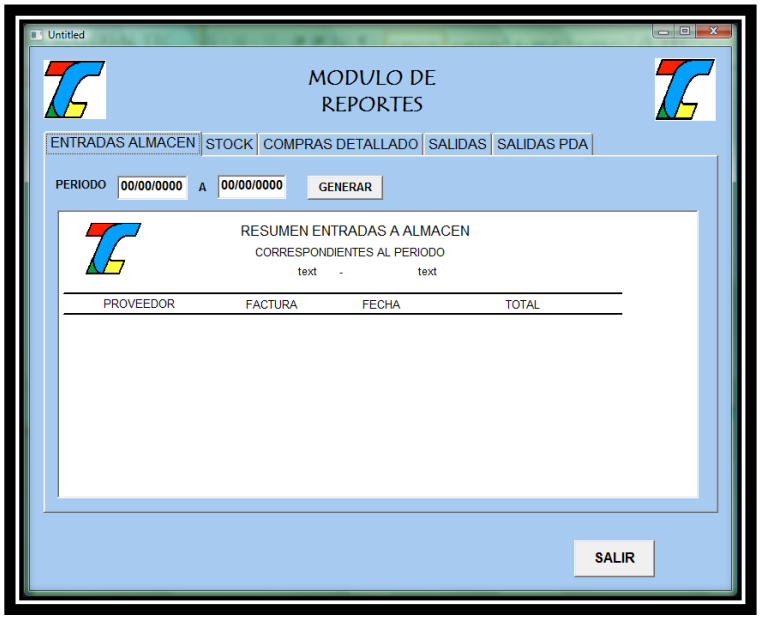


Script de programación:

```
Public Class modifica

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        Dim sr As New dataservice
        Dim ds As DataSet
        Dim ban As Integer = 0

        If TextBox1.Text = "" Then
            DataGridView1.DataSource = sr.GetDataSet("Exec
lista_articulos").Tables(0)
            If (DataGridView1.Rows.Count < 2) Then
                MsgBox("NO HAY DATOS")

            End If
        ElseIf ban = 0 Then
            DataGridView1.DataSource = sr.GetDataSet("Exec lista_articulos2 '%" &
TextBox1.Text & "%'").Tables(0)
            If (DataGridView1.Rows.Count < 2) Then
```

```
                MsgBox("NO HAY DATOS")

            End If
        End If

    End Sub

    Private Sub DataGridView1_CellContentClick(ByVal sender As System.Object, ByVal
e As System.Windows.Forms.DataGridViewCellEventArgs) Handles
DataGridView1.CellContentClick
        id.Text = DataGridView1.CurrentRow.Cells(0).Value.ToString
        nombre.Text = DataGridView1.CurrentRow.Cells(1).Value.ToString
        cantidad.Text = DataGridView1.CurrentRow.Cells(2).Value.ToString
        lugar.Text = DataGridView1.CurrentRow.Cells(3).Value.ToString
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Dim sr As New dataservice
        DataGridView1.DataSource = sr.GetDataSet("Exec modifica_articulos '" +
id.Text + "','" + nombre.Text + "','" + cantidad.Text + "','" + lugar.Text + "'")
        DataGridView1.Refresh()
        DataGridView1.DataSource = sr.GetDataSet("Exec lista_articulos").Tables(0)
    End Sub

    Private Sub modifica_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        Me.Close()
    End Sub

End Class
```

Módulo Reportes.



Script de programación:

```
reportes.conexion()
long colnum
string a
dw_3.settransobject(SQLCA);
dw_3.Retrieve(SQLCA);
dw_6.settransobject(SQLCA);
dw_6.Retrieve(SQLCA);

//PARA DESPLEGAR TIPOS DE ARTICULOS
FOR colNum = 1 TO dw_3.RowCount()
tab_1.tabpage_2.ddlb_1.additem(String(dw_3.object.descripcion[colnum]))
NEXT

//PARA DESPLEGAR LAS AREAS
FOR colNum = 1 TO dw_6.rowcount()
tab_1.tabpage_4.ddlb_2.additem(string(dw_6.object.descripcion2[colnum]))
NEXT
disconnect using sqlca;
dw_1.setFilter("enc_fact_fecha_factura >= date('"+em_1.text+"') and enc_fact_fecha_factura <=
                                        date('"+em_2.text+"')" );
dw_1.Filter();
dw_1.settransobject(SQLCA);
dw_1.Retrieve(SQLCA);

disconnect using SQLCA;
if dw_1.rowcount()=0 then
messagebox("AVISO","NO HAY DATOS A MOSTRAR CON EL PERIODO ESTABLECIDO")
RETURN 0
end if
dw_1.object.t_3.text=em_1.text;
dw_1.object.t_4.text=em_2.text;
dw_2.setFilter("stock_cantidad < 15");
dw_2.Filter();
dw_2.settransobject(SQLCA);
dw_2.Retrieve(SQLCA);

disconnect using SQLCA;
if dw_2.rowcount()=0 then
        messagebox("AVISO","NO HAY DATOS A MOSTRAR")
RETURN 0
```

```
end if
dw_2.setFilter("stock_cantidad = 0");
dw_2.Filter();
dw_2.settransobject(SQLCA);
dw_2.Retrieve(SQLCA);
disconnect using SQLCA;
if dw_2.rowcount()=0 then
messagebox("AVISO","NO HAY DATOS A MOSTRAR")
RETURN 0
end if
reportes.conexion()
dw_2.setFilter("");
dw_2.Filter();
dw_2.settransobject(SQLCA);
dw_2.Retrieve(SQLCA);
disconnect using SQLCA;
reportes.conexion()

dw_2.setFilter("tipo_articulos_descripcion = string('"+ddlb_1.text+"')");
dw_2.Filter();
dw_2.settransobject(SQLCA);
dw_2.Retrieve(SQLCA);
disconnect using SQLCA;
if dw_2.rowcount()=0 then
messagebox("AVISO","NO HAY DATOS A MOSTRAR")
RETURN 0
end if
reportes.conexion()
dw_4.setFilter("enc_fact_fecha_factura >= date('"+em_3.text+"') and enc_fact_fecha_factura <=
                                        date('"+em_4.text+"')" );
dw_4.Filter();
dw_4.settransobject(SQLCA);
dw_4.Retrieve(SQLCA);
disconnect using SQLCA;
if dw_4.rowcount()=0 then
messagebox("AVISO","NO HAY DATOS A MOSTRAR CON EL PERIODO ESTABLECIDO")
RETURN 0
end if
dw_4.object.t_12.text=em_3.text;
dw_4.object.t_13.text=em_4.text;
reportes.conexion()
```

```
if (rb_1.checked=true) then
dw_5.setFilter("salidas_fecha_entrega >= date("'+em_5.text+'") and salidas_fecha_entrega <=
                                        date("'+em_6.text+'") and cat_areas_descripcion =
                                        string("'+ddlb_2.text+'")");
dw_5.Filter();
dw_5.settransobject(SQLCA);
dw_5.Retrieve(SQLCA);
if dw_5.rowcount()=0 then
messagebox("AVISO","NO HAY DATOS A MOSTRAR CON LOS DATOS PROPORCIONADOS")
RETURN 0
  end if
else
dw_5.setFilter("salidas_fecha_entrega >= date("'+em_5.text+'") and salidas_fecha_entrega <=
                                        date("'+em_6.text+'")" );
dw_5.Filter();
dw_5.settransobject(SQLCA);
dw_5.Retrieve(SQLCA);
if dw_5.rowcount()=0 then
messagebox("AVISO","NO HAY DATOS A MOSTRAR CON LOS DATOS PROPORCIONADOS")
RETURN 0
end if
end if
disconnect using SQLCA;
dw_5.object.t_9.text=em_5.text;
dw_5.object.t_10.text=em_6.text;
reportes.conexion()
dw_7.setFilter("movimientos_pda_fecha_mov >= date("'+em_7.text+'") and movimientos_pda_fecha_mov <=
                                        date("'+em_8.text+'")");
dw_7.Filter();
dw_7.settransobject(SQLCA);
dw_7.Retrieve(SQLCA);
if dw_7.rowcount()=0 then
messagebox("AVISO","NO HAY DATOS A MOSTRAR CON LOS DATOS PROPORCIONADOS")
RETURN 0
  end if
disconnect using SQLCA;
dw_7.object.t_7.text=em_7.text;
dw_7.object.t_8.text=em_8.text;
```

Apéndice 4

Ventanas y scripts de programación aplicación de dispositivo móvil

Ventana de login.

Script de programación:

```
Imports System.Data.SqlServerCe
Imports System.Data.SqlClient

Public Class Form1

    Dim cn As SqlCeConnection
    Dim usuario As String
    Public id_usuario As Integer
    Public nom3 As String

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        Me.Close()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

        Dim pass, pass2, nom, nom2 As String
        Dim cmd As SqlCeCommand = cn.CreateCommand
        Dim consulta As String = "SELECT nombre,password, id_emp FROM usuarios
where nombre='" & ComboBox1.SelectedItem & "'"

        cmd.CommandText = consulta
        Dim lectorDatos As SqlCeDataReader
        lectorDatos = cmd.ExecuteReader()

        Dim rst As SqlCeResultSet
        rst = cmd.ExecuteResultSet(Data.SqlServerCe.ResultSetOptions.None)
        While rst.Read
            nom = (rst.GetString(0))
```

```
            pass = (rst.GetString(1))
            id_usuario = (rst.GetValue(2))
        End While
        lectorDatos.Close()
        nom = nom.Trim()
        pass = pass.Trim()
        nom2 = ComboBox1.SelectedItem
        pass2 = password.Text
        nom2 = nom2.Trim()
        pass2 = pass2.Trim()
        'MsgBox(nom2 + pass2)
        nom3 = nom2

        If (nom2 = nom) And (pass2 = pass) Then
            menu1.Visible = True
            password.Text = ""
        Else
            MsgBox("Nombre de usuario y/o contraseña incorrectos",
MsgBoxStyle.Information)
            Return
        End If
    End Sub

    Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles password.TextChanged
        password.PasswordChar = "*"
        password.MaxLength = 14
    End Sub

    Private Sub Form1_Closed(ByVal sender As Object, ByVal e As System.EventArgs)
Handles MyBase.Closed
        Me.Close()
    End Sub

    Private Sub Form1_Disposed(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Disposed

        If cn.State = Data.ConnectionState.Open Then
            cn.Close()
        End If
        Me.Close()
    End Sub

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        cn = New System.Data.SqlServerCe.SqlCeConnection("Data Source=\Archivos
de programa\DeviceApplication1\inventariop.sdf; Password ='")
        If cn.State <> Data.ConnectionState.Open Then
            cn.Open()
        End If

        Dim cmd As SqlCeCommand = cn.CreateCommand
        Dim consulta As String = "SELECT * FROM usuarios"

        cmd.CommandText = consulta

        Dim rst As SqlCeResultSet
        rst = cmd.ExecuteResultSet(Data.SqlServerCe.ResultSetOptions.None)
        While rst.Read
            ComboBox1.Items.Add(rst.GetString(1))
        End While
```

```
        password.Text = "admin"
    End Sub
    Private Sub InitializeMyControl()
        password.Text = ""
        password.PasswordChar = "*"
        password.MaxLength = 14
    End Sub

    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
        altausuarios.Visible = True
    End Sub
End Class
```

Menú Principal (Administrador y Usuario)

Script de programación:

```
Public Class menu1

    Private Sub Label1_ParentChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs)
    End Sub
    Private Sub Label2_ParentChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Label2.ParentChanged
    End Sub
    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs)
    End Sub
    Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Me.Close()
    End Sub
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        altausuarios.Visible = True
    End Sub
    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click
        consulta.Visible = True
    End Sub
```

```
    Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button4.Click
        movimientos.Visible = True
    End Sub
    Private Sub menu1_Disposed(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Disposed
        Me.Close()
    End Sub
    Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button5.Click
        conexion.Visible = True
    End Sub
    Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button6.Click
        concilia.Visible = True
    End Sub
    Private Sub menu1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        If (Form1.nom3 = "admin") Then
            Button5.Enabled = True
            Button6.Enabled = True
            Button2.Enabled = True
            Label3.Visible = True
        ElseIf (Form1.nom3 <> "admin") Then
            Button5.Enabled = False
            Button6.Enabled = False
           Button2.Enabled = False
            Label3.Visible = False
        End If
    End Sub
End Class
```

Módulo Consulta.

Script de programación:

```
Imports System.Data.SqlServerCe
Imports System.Data.SqlClient
Imports System.xml
Imports System.Web.Services
Imports System.Xml.Schema
```

```
Public Class consulta

    Dim cn As SqlCeConnection

    Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        '        Me.Close()
        Me.Visible = False
    End Sub

    Private Sub consulta_Closed(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Closed
        If cn.State = Data.ConnectionState.Open Then
            cn.Close()
        End If
        Me.Close()
    End Sub


    Private Sub consulta_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        cn = New System.Data.SqlServerCe.SqlCeConnection("Data Source=\Archivos
de programa\DeviceApplication1\inventariop.sdf; Password ='")
        If cn.State <> Data.ConnectionState.Open Then
            cn.Open()
        End If

        'CONSULTA PARA LLENAR EL COMBOBOX
        Dim cmd As SqlCeCommand = cn.CreateCommand
        Dim consulta As String = "SELECT * FROM Articulos"
        cmd.CommandText = consulta
        Dim lectorDatos As SqlCeDataReader
        lectorDatos = cmd.ExecuteReader()
        While lectorDatos.Read()
            Me.ComboBox1.Items.Add(lectorDatos("nombre_art"))
            ComboBox1.ValueMember = (lectorDatos("id_art"))
        End While
        lectorDatos.Close()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Dim nomart As String
        nomart = ComboBox1.SelectedItem
        nomart = nomart.Trim()

        Dim cmd As SqlCeCommand = cn.CreateCommand
        Dim consulta As String = "SELECT * FROM Articulos WHERE nombre_art='" +
nomart + "'"
        cmd.CommandText = consulta
        Dim lectorDatos As SqlCeDataReader
        lectorDatos = cmd.ExecuteReader()
        If lectorDatos.Read() Then
            stock.Text = (lectorDatos("cantidad"))
            uni.Text = (lectorDatos("unidad"))
        End If
        lectorDatos.Close()
    End Sub

    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button3.Click

End Class
```

Módulo Movimientos.

Script de programación:

```
Public Class movimientos

    Dim cn As SqlCeConnection
    Dim id_art As Integer

    Private Sub movimientos_Disposed(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Disposed
        If cn.State = Data.ConnectionState.Open Then
            cn.Close()
        End If
    End Sub
    Private Sub movimientos_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        cn = New System.Data.SqlServerCe.SqlCeConnection("Data Source=\Archivos
de programa\DeviceApplication1\inventariop.sdf; Password ='")
        If cn.State <> Data.ConnectionState.Open Then
            cn.Open()
        End If

        'CONSULTA PARA LLENAR EL COMBOBOX
        Dim cmd As SqlCeCommand = cn.CreateCommand
        Dim consulta As String = "SELECT * FROM Articulos"
        cmd.CommandText = consulta
        Dim lectorDatos As SqlCeDataReader
        lectorDatos = cmd.ExecuteReader()
        While lectorDatos.Read()
            Me.ComboBox1.Items.Add(lectorDatos("nombre_art"))
        End While
    End Sub

    Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles ComboBox1.SelectedIndexChanged

        Dim nomart As String
```

```
        nomart = ComboBox1.SelectedItem
        nomart = nomart.Trim()

        Dim cmd As SqlCeCommand = cn.CreateCommand
        Dim consulta As String = "SELECT id_art FROM Articulos WHERE
nombre_art='" + nomart + "'"
        cmd.CommandText = consulta
        Dim lectorDatos As SqlCeDataReader
        lectorDatos = cmd.ExecuteReader()
        If lectorDatos.Read() Then
            id_art = (lectorDatos("id_art"))
        End If

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Try
            Dim id_servicio, cantidad, id_mov As Integer
            id_servicio = Convert.ToDecimal(TextBox1.Text)
            cantidad = Convert.ToDecimal(NumericUpDown1.Value)
            Dim cmd As SqlCeCommand = cn.CreateCommand
            Dim consulta As String = "INSERT Movimientos(tipo_mov,id_usuario,
capturado) values('" + ComboBox2.Text + "','" & Form1.id_usuario & "', 'NO')"

            cmd.CommandText = consulta
            cmd.ExecuteNonQuery()

            Dim cmd2 As SqlCeCommand = cn.CreateCommand
            Dim consulta2 As String = "SELECT MAX(id_mov) as id_mov FROM
Movimientos"
            cmd2.CommandText = consulta2
            Dim lectorDatos As SqlCeDataReader
            lectorDatos = cmd2.ExecuteReader()
            If lectorDatos.Read() Then
                id_mov = (lectorDatos("id_mov"))
            End If

            Dim cmd3 As SqlCeCommand = cn.CreateCommand
            Dim consulta3 As String = "INSERT
Detalle_Movimientos(id_mov,cantidad_art,id_articulo,detalle,id_servicio,
capturado) values('" & id_mov & "','" & cantidad & "','" & id_art & "','" +
ComboBox3.Text + "','" & id_servicio & "', 'NO')"

            cmd3.CommandText = consulta3
            cmd3.ExecuteNonQuery()

            Dim cmd4 As SqlCeCommand = cn.CreateCommand
            Dim consulta4 As String = "UPDATE Articulos SET cantidad=cantidad -"
& cantidad & " WHERE id_art= " & id_art & ""

            cmd4.CommandText = consulta4
            cmd4.ExecuteNonQuery()
        Catch ex As Exception

        End Try
        MessageBox.Show("Movimiento realizado")

    End Sub

End Class
```

Módulo Conexión WS.

Script de programación:

```
Imports System.Data.SqlServerCe
Imports System.Data.SqlClient
Imports System.xml
Imports System.Web.Services
Imports System.Xml.Schema

Public Class conexion

    Dim cn As SqlCeConnection
    Dim id_art, cantidad As Integer
    Dim desc, unidad As String

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        Dim f As Integer
        Dim Midataset As New Data.DataSet()
        'PRUEBA PARA COMUNICACION WEB SERVICE
        Dim JWS1 As New ttcservice.Service1
        Dim OBJXML As New XmlDocument

        Try
               Midataset = JWS1.ObtenerStock
            Dim src As New BindingSource(Midataset, "stock")
            DataGrid1.DataSource = src
            f = Midataset.Tables("stock").Rows.Count
        Catch XmlExp As XmlException
            MsgBox(XmlExp.Message)
        Catch XmlSchExp As XmlSchemaException
            MsgBox(XmlSchExp.Message)
        Catch GenExp As Exception
            MsgBox(GenExp.Message)
        End Try
        MsgBox(f)
    End Sub

    Private Sub DataGrid1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles DataGrid1.Click
        Dim currentcell As DataGridCell
```

```
        Dim currentcelldata As String
        Dim currentcelldata1 As String
        Dim currentcelldata2 As String

        'obtiente celda actual
        currentcell = DataGrid1.CurrentCell

        'obtiene el dato de la celda actual
        currentcelldata = CStr(DataGrid1(currentcell.RowNumber, 0))
        currentcelldata1 = CStr(DataGrid1(currentcell.RowNumber, 1))
        currentcelldata2 = CStr(DataGrid1(currentcell.RowNumber, 3))

        'al textbox se le inserta el dato
        TextBox1.Text = currentcelldata
        TextBox2.Text = currentcelldata1
        TextBox3.Text = currentcelldata2
    End Sub

    Private Sub conexion_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        cn = New System.Data.SqlServerCe.SqlCeConnection("Data Source=\Archivos
de programa\DeviceApplication1\inventariop.sdf; Password ='")
        If cn.State <> Data.ConnectionState.Open Then
            cn.Open()
        End If

        'CONSULTA PARA LLENAR EL COMBOBOX
        Dim cmd As SqlCeCommand = cn.CreateCommand
        Dim consulta As String = "SELECT * FROM Usuarios WHERE nombre!='juan'"
        cmd.CommandText = consulta
        Dim lectorDatos As SqlCeDataReader
        lectorDatos = cmd.ExecuteReader()
        While lectorDatos.Read()
            Me.ComboBox1.Items.Add(lectorDatos("nombre"))
            ComboBox1.ValueMember = (lectorDatos("id_emp"))
        End While
        lectorDatos.Close()

        TextBox4.Text = Today()
    End Sub
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

        'CONSULTA DEL ID DEL EMPLEADO
        Dim id_emp As Decimal
        Dim cmdz As SqlCeCommand = cn.CreateCommand
        Dim consult As String = "SELECT id_emp FROM Usuarios WHERE nombre='" +
ComboBox1.Text + "'"
        cmdz.CommandText = consult
        Dim lectorDato As SqlCeDataReader
        lectorDato = cmdz.ExecuteReader()
        If lectorDato.Read() Then
            id_emp = (lectorDato("id_emp"))
        End If
        lectorDato.Close()

        Try
            id_art = Convert.ToDecimal(TextBox1.Text)
            cantidad = Convert.ToDecimal(NumericUpDown1.Value)
            desc = TextBox2.Text
            unidad = TextBox3.Text
```

```
'CONSULTA PARA VERIFICAR SI EL ARTICULO YA EXISTE EN LA BD DE LA PDA
            Dim cmd As SqlCeCommand = cn.CreateCommand
            Dim consulta As String = "SELECT * FROM Articulos WHERE id_art='" &
id_art & "'"
            cmd.CommandText = consulta
            Dim lectorDatos As SqlCeDataReader
            lectorDatos = cmd.ExecuteReader()

            'si el articulo ya existe entonces modificar
            'la cantidad
            If lectorDatos.Read() Then

                Dim cmd2 As SqlCeCommand = cn.CreateCommand
                Dim consulta2 As String = "UPDATE Articulos SET cantidad=cantidad
+" & cantidad & " WHERE id_art= " & id_art & ""
                cmd2.CommandText = consulta2
                cmd2.ExecuteNonQuery()
                MessageBox.Show("Movimiento almacenado")

                'si el articulo no existe entonces dar de alta con todos sus
datos
            ElseIf lectorDatos.Read() < 1 Then

                Dim cmd3 As SqlCeCommand = cn.CreateCommand
                Dim consulta3 As String = "INSERT INTO
Articulos(id_art,nombre_art,cantidad,unidad) values('" & id_art & "','" + desc +
"','" & cantidad & "','" + unidad + "');"

                cmd3.CommandText = consulta3
                cmd3.ExecuteNonQuery()
                MessageBox.Show("Movimiento almacenado, articulo nuevo")

            End If

        Catch ex As Exception
            MsgBox(ex.Message)

        End Try

        'HACER EL LLAMADO AL METODO DESCARGAR DE WEB SERVICE PARA
        'ACTUALIZAR LA DB DEL SERVIDOR
        Dim JWS1 As New ttcservice.Service1
        Dim OBJXML As New XmlDocument
        Dim ws As String
        Try
            ws = (JWS1.Descarga(id_art, cantidad))
        Catch XmlExp As XmlException
            'Console.WriteLine(XmlExp.Message)
            MsgBox(XmlExp.Message)
        Catch XmlSchExp As XmlSchemaException
            'Console.WriteLine(XmlSchExp.Message)
            MsgBox(XmlSchExp.Message)
        Catch GenExp As Exception
            ' Console.WriteLine(GenExp.Message)
            MsgBox(GenExp.Message)
        End Try


        'HACER EL LLAMADO AL METODO ULTSAL DE WEB SERVICE PARA
        'OBTENER EL NUMERO DE LA ULTIMA SALIDA
        Dim JWS2 As New ttcservice.Service1
        Dim OBJXML2 As New XmlDocument
        Dim us As Decimal
```

```
        Try
            us = (JWS2.UltSalida())
            us = us + 1
        Catch XmlExp As XmlException
            'Console.WriteLine(XmlExp.Message)
            MsgBox(XmlExp.Message)
        Catch XmlSchExp As XmlSchemaException
            'Console.WriteLine(XmlSchExp.Message)
            MsgBox(XmlSchExp.Message)
        Catch GenExp As Exception
            ' Console.WriteLine(GenExp.Message)
            MsgBox(GenExp.Message)
        End Try

        'HACER EL LLAMADO AL METODO CAPTSAL DE WEB SERVICE PARA
        'ACTUALIZAR LA TABLA SALIDAS
        Dim JWS3 As New ttcservice.Service1
        Dim OBJXML3 As New XmlDocument
        Dim ws1 As String
        Try
            ws1 = (JWS3.CaptSalida(us, id_art, cantidad, id_emp, TextBox4.Text))
        Catch XmlExp As XmlException
                MsgBox(XmlExp.Message)
        Catch XmlSchExp As XmlSchemaException
               MsgBox(XmlSchExp.Message)
        Catch GenExp As Exception
            MsgBox(GenExp.Message)
        End Try
        MsgBox("La db en servidor ha sido actualizada")
    End Sub

End Class
```

Módulo Conciliar Movimientos:

Script de programación:

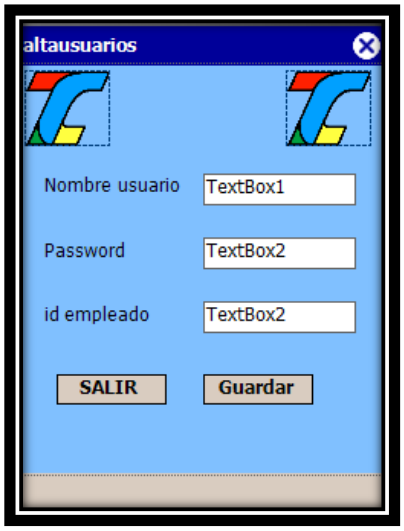
```
Imports System.Data.SqlServerCe
Imports System.Data.SqlClient
Imports System.xml
Imports System.Web.Services
```

```
Imports System.Xml.Schema
Imports System.Data

Public Class concilia

    Dim cn As SqlCeConnection
    Dim JWS1 As New ttcservice.Service1
    Dim OBJXML As New XmlDocument

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

        'CODIGO PARA SABER CUANTAS FILAS TENGO EN TABLAS DE MOVIMIENTOS
        Dim f As Integer
        Try
            MsgBox("consulta filas")
            Dim cmd As SqlCeCommand = cn.CreateCommand
            Dim consulta As String = "SELECT count (id_mov) as mov FROM
Movimientos WHERE Capturado='NO'"
            cmd.CommandText = consulta
            Dim lectorDatos As SqlCeDataReader
            lectorDatos = cmd.ExecuteReader()
            If lectorDatos.Read() Then
                f = (lectorDatos("mov"))
            End If
            lectorDatos.Close()
        Catch ex As Exception

        End Try
        MsgBox(f)
        If f > 0 Then
            f = f - 1
            Dim Result As MsgBoxResult

            Result = MsgBox("A continuación se trasladarán los datos de los
movimientos realizados a la base de datos del servidor," & _
            "para proseguir dé clic en Si", MsgBoxStyle.OkCancel)
            If Result = MsgBoxResult.Cancel Then
                Return
            ElseIf Result = MsgBoxResult.Ok Then

                'CREA MATRIZ PARA GUARDAR DATOS
                Dim movs(f, 3) As String
                Dim det_movs(f, 4) As String

                Dim i, z As Integer
                i = 0

                'CODIGO PARA LLENAR MATRIZ
                Try
                    Dim cmd As SqlCeCommand = cn.CreateCommand
                    Dim consulta As String = "SELECT id_mov, tipo_mov,
id_usuario,fecha_mov FROM Movimientos WHERE Capturado='NO'"
                    cmd.CommandText = consulta
                    Dim lectorDatos As SqlCeDataReader
                    lectorDatos = cmd.ExecuteReader()
                    While lectorDatos.Read()
                        For z = 0 To 3
                            movs(i, z) =
Convert.ToString(lectorDatos.GetValue(z))
                            movs(i, z) = movs(i, z).Trim
                            'MsgBox(movs(i, z))
                        Next
```

```
                        i = i + 1
                    End While
                    lectorDatos.Close()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try

                Try
                    i = 0
                    Dim cmd As SqlCeCommand = cn.CreateCommand
                    Dim consulta As String = "SELECT id_mov, detalle,
cantidad_art, id_articulo, id_servicio FROM Detalle_Movimientos WHERE
Capturado='NO'"
                    cmd.CommandText = consulta
                    Dim lectorDatos As SqlCeDataReader
                    lectorDatos = cmd.ExecuteReader()
                    While lectorDatos.Read()
                        For z = 0 To 4
                            det_movs(i, z) =
Convert.ToString((lectorDatos.GetValue(z)))
                            det_movs(i, z) = det_movs(i, z).Trim
                            'MsgBox(det_movs)
                        Next
                        i = i + 1
                    End While
                    lectorDatos.Close()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try

                'CODIGO PARA LLAMAR METODO WEB SERVICE
                MsgBox("llama web service")
                Dim sql, sql1 As String
                Try

                    For i = 0 To f

                        sql = (JWS1.Inserta_MovimientosPDA(movs(i, 0), movs(i,
1), movs(i, 2), movs(i, 3)))
                        sql1 = (JWS1.Inserta_DetalleMovimientosPDA(det_movs(i,
0), det_movs(i, 1), det_movs(i, 2), det_movs(i, 3), det_movs(i, 4)))
                        'Next
                    Next
                Catch XmlExp As XmlException
                    MsgBox(XmlExp.Message)
                Catch XmlSchExp As XmlSchemaException
                    MsgBox(XmlSchExp.Message)
                Catch GenExp As Exception
                    MsgBox(GenExp.Message)
                End Try

                'CODIGO PARA ACTUALIZAR CAMPO CAPTURADO A SI
                Try

                    Dim cmd As SqlCeCommand = cn.CreateCommand
                    Dim cmd2 As SqlCeCommand = cn.CreateCommand
                    Dim consulta As String = "UPDATE Movimientos set
Capturado='SI' where Capturado = 'NO'"
                    Dim consulta2 As String = "UPDATE Detalle_Movimientos set
Capturado='SI' where Capturado = 'NO'"
                    cmd.CommandText = consulta
                    cmd2.CommandText = consulta2
                    cmd.ExecuteNonQuery()
```

```
                    cmd2.ExecuteNonQuery()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try
                MsgBox("final")
            End If 'final if msgbox
        ElseIf f <= 0 Then
            MsgBox("NO HAY MOVIMIENTOS PARA CONCILIAR")
            Return
        End If  ' FINAL IF DEL F
    End Sub

    Private Sub Button1_Disposed(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Disposed
        If cn.State = Data.ConnectionState.Open Then
            cn.Close()
        End If
    End Sub

    Private Sub concilia_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        cn = New System.Data.SqlServerCe.SqlCeConnection("Data Source=\Archivos
de programa\DeviceApplication1\inventariop.sdf; Password ='")
        If cn.State <> Data.ConnectionState.Open Then
            cn.Open()
        End If
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Me.Visible = False
    End Sub
End Class
```

Módulo Alta Usuarios:



Script de programación:

```
Imports System.Data.SqlServerCe
Imports System.Data.SqlClient

Public Class altausuarios
```

```
    Dim cn As SqlCeConnection

    Private Sub Label1_ParentChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Label1.ParentChanged
    End Sub
    Private Sub Label2_ParentChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Label2.ParentChanged
    End Sub
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Try

            Dim cmd As SqlCeCommand = cn.CreateCommand
            Dim user, pass As String
            Dim id_emp2 As Integer
            id_emp2 = Convert.ToDecimal(id_emp.Text)
              user = Me.user.Text
            pass = Me.pass.Text

            ' Agregamos registro
            cmd.CommandText = "INSERT INTO usuarios" & _
            "(nombre, password, id_emp)" & _
            " VALUES (?, ?, ?)"

            cmd.Parameters.Add(New SqlCeParameter("@nombre_usuario", user))
            cmd.Parameters.Add(New SqlCeParameter("@password_usuario", pass))
            cmd.Parameters.Add(New SqlCeParameter("@id_emp", id_emp2))

            cmd.Prepare()
            cmd.ExecuteNonQuery()

            MessageBox.Show("Registro agregado")

            'limpiamos los textbox
            LimpiaTextbox()

        Catch sqlex As SqlCeException
            Dim sqlError As SqlCeError

            For Each sqlError In sqlex.Errors
                MessageBox.Show(sqlError.Message)
            Next

        Catch ex As Exception
            MessageBox.Show(ex.Message)

        Finally
            If cn.State <> System.Data.ConnectionState.Closed Then
                cn.Close()
            End If
        End Try
    End Sub
    'limpia textbox
    Private Sub LimpiaTextbox()
        Me.user.Text = ""
        Me.pass.Text = ""
    End Sub

    Private Sub altausuarios_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        cn = New System.Data.SqlServerCe.SqlCeConnection("Data Source=\Archivos
de programa\DeviceApplication1\inventariop.sdf; Password ='")
        If cn.State <> Data.ConnectionState.Open Then
```

```
            cn.Open()
        End If
        Me.LimpiaTextbox()
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Me.Visible = False
    End Sub

    Private Sub pass_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles pass.TextChanged
        pass.PasswordChar = "*"
        pass.MaxLength = 14
    End Sub
End Class
```

BIBLIOGRAFÍA

[1] Sascha Segan. “¿Cuál me llevo?”. PC Magazine en Español. Vol. XVII. Fascículo 03. Pp 85-90. Agosto 2006.

[2] Aniruddha Gokhale, Bharat Kumar, Arnaud Sahuguet. “Reinventing the Wheel? CORBA vs. Web Services”. Disponible en http://www2002.org/CDROM/alternate/395. Fecha de última visita: 6 de diciembre de 2008.

[3] Payam Shodjai. “Web Services and the Microsoft Platform”. Disponible en http://msdn.microsoft.com/en-us/library/aa480728.aspx#wsmsplat_topic2. Publicado en junio 2006. Fecha de última visita: 18 de enero 2009.

[4] Noelia Barreira Rodríguez. “Introducción a servicios web”. Disponible en http://www.elrincondelprogramador.com/default.asp?pag=articulos/leer.asp&id=32. Publicado en septiembre 2002. Fecha de última visita: 22 de agosto de 2008.

[5] Joaquín Salvachúa. “Comunicación mediante sockets”. Disponible en http://www.lab.dit.upm.es/~cdatlab/cursos/cdatlab/sockets/sld001.htm. Fecha de última visita: 3 de junio de 2008.

[6] Francisco Charte. “De Palms, Pockets y Otros Informática móvil I”. Disponible en http://www.idg.es/pcworld/De-Palms_-Pockets-y-otros_Informatica-movil-_I_/art113968.htm. Fecha de última visita: 23 de junio de 2009.

[7] Gabrierl Agustín Praino. “Introducción a la Programación TCP/IP sobre Windows”. Disponible en http://www.7542.fi.uba.ar/herramientas.php?i=socketst. Publicado en noviembre 2001. Fecha de última visita: 13 de julio de 2008.

[8] “Principios de Web Services”. Disponible en http://www.cepeu.edu.py/LIBROS_ELECTRONICOS_3/lpcu104%20-%2001.pdf. Fecha de última visita: 5 de junio de 2009.

[9] Oscar González Moreno. “Introducción a Web Services con Herramientas de Desarrollo Microsoft”. Disponible en www.danysoft.com. Publicado en abril 2002. Fecha de última visita: 5 de junio de 2009.

[10] “Web Services”. Disponible en http://www.desarrolloweb.com/articulos/1537.php. Fecha de última visita: 15 de noviembre de 2008.

[11] http://msdn.microsoft.com/vbasic/learning/mobile/. Fecha de última visita: 27 de octubre de 2008.

[12] http://www.microsoft.com. Fecha de última visita: 11 de junio de 2009.

[13] http://www.sybase.com/products/ developmentintegration/pocketbuilder. Fecha de última visita: 14 de mayo de 2008.

[14] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html. Fecha de última visita: 25 de mayo de 2008.

[15] http://www.otterbox.com. Fecha de última visita: 29 de junio de 2009.

[16] http://www.maximotec.com/showthread.php?t=22803. Fecha de última visita: 18 de enero de 2009.

[17] http://sistemas3.wordpress.com/2007/06/14/web-services/. Fecha de última visita: 27 de agosto de 2008.

[18] http://pegaso.ls.fi.upm.es/BD/Documentacion/06-SQL.pdf. Fecha de última visita: 24 de junio de 2008.

[19] http://es.wikipedia.org. Fecha de última visita: 3 de agosto de 2009.

[20] David Livingstone Vaught. “Pocket PC vs PALM” Disponible en http://www.pc-doctor.com.mx/TIPS/PDA%20VS%20%20PALM.htm. Fecha última visita: 2 de noviembre de 2009.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO DE TÉRMINOS

Acceso Remoto

Significa que se puede acceder desde una computadora a un recurso ubicado físicamente en otra computadora, a través de una red local o externa (Internet).

Computación Distribuida / Sistema Distribuido

Colección de computadoras separadas físicamente y conectados entre sí por una red de comunicaciones distribuida. El usuario accede a los recursos remotos de la misma manera en que accede a recursos locales, o un grupo de computadoras que usan un software para conseguir un objetivo en común.

Embedded

Se refiere a un equipo con un propósito determinado, integrado en el sistema que controla. Necesita especificaciones particulares y realiza tareas predefinidas. Embedded significa encajado, embutido, algo metido dentro de una cosa.

Extranet

Red que utiliza la tecnología de Internet para conectar la red local (LAN) de una organización con otras redes externas.

Firebird

Sistema de administración de bases de datos relacional de código abierto, basado en la versión 6 de Interbase, cuyo código fue liberado por Borland en el 2000.

Información

Es un conjunto de datos que al relacionarse adquieren sentido o un valor de contexto o de cambio.

Interoperabilidad

Capacidad de un programa para acceder a múltiples sistemas diferentes.

Intranet

Es una red de computadoras dentro de una red de área local (LAN) privada empresarial o educativa que proporciona herramientas de Internet. Tiene como función principal proveer lógica de negocios para aplicaciones de captura, reportes y consultas con el fin de facilitar la producción de dichos grupos de trabajo; es también un importante medio de difusión de información interna a nivel de grupo de trabajo.

MODEM

Modulator/Demodulator. Modulador/Demodulador. Dispositivo que adapta las señales digitales para su transmisión a través de una línea analógica. Normalmente telefónica.

MySQL

Sistema de gestión de bases de datos multiusuario, multiplataforma y de código abierto. Pertenece a la compañía sueca MySQL AB.

.NET Framework

Componente integral de Windows que admite la creación y la ejecución de la siguiente generación de aplicaciones y servicios Web XML. Proporciona un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota; un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones. Ofrece al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.

PostgreSQL

Motor de base de datos de código abierto.

Switch Inalámbrico

Dispositivo que interconecta por si solo varias PC’s inalámbricamente.

Tecnología de la Información

Son aquellas herramientas y métodos empleados para recabar, retener, manipular o distribuir información.

Tecnología Inalámbrica

Se refiere al uso de la tecnología sin cables la cual permite la conexión de varias computadoras entre sí.

Transacción distribuida

Es una transacción que afecta a varios recursos. Para que una transacción distribuida se confirme, todos los participantes deben garantizar que los cambios en los datos serán permanentes. Los cambios deben mantenerse a pesar de bloqueos del sistema u otros acontecimientos imprevistos. Si alguno de los participantes no cumple esta garantía, toda la transacción da error y se desharán los cambios en los datos en el ámbito de la transacción.

WiFi

Significa Wireless Fidelity (Fidelidad Inalámbrica), es un sistema de envío de datos sobre redes de computadoras que utiliza ondas de radio en lugar de cables.

Wireless

Es un término que significa “SIN CABLES”, y que designa a todos aquellos aparatos que, en su funcionamiento no requieren la conexión física entre él y otro.