Universidad del Azuay

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas

Interfaz en Visual Basic .Net para el mantenimiento de Inventarios utilizando equipos móviles Pocket PC

Trabajo de graduación previo a la obtención del título de

Ingeniero en Sistemas

Autores:

Juan Carlos Torres Piedra

Juan Fernando Villavicencio Zambrano

Director:

Ing. Lenin Erazo Garzón

Cuenca-Ecuador

2009

DEDICATORIA

La dedico a Dios por darme todo lo necesario para lograr esta meta de mi vida.

A mis padres Giordano y Janeth, quienes me dieron la vida, amor, educación y la formación de valores, ética y de principios que me han servido durante toda mi vida.

Juan Carlos.

Dedicado a las personas que han hecho posible que cada despertar sea motivo de alegría y el motor que me anima a seguir siempre adelante, por ellos y para ellos: Alfonso, Esthela, Valeria, JuanFer, Paola, Mateo.

Juan Fernando.

AGRADECIMIENTOS

Agradezco a Dios por brindarme la oportunidad y hacer posible todo esto.

Agradezco a mis padres Giordano y Janeth quienes me han apoyado y siempre han estado pendientes, alentándome y preocupados por mis estudios y mi vida, al igual que mi abuela Ma que de una u otra forma se ha preocupado por mis estudios.

Juan Carlos.

Un agradecimiento muy especial a mi padre que está en el cielo y que siempre se preocupó por darme la mejor educación. A mi madre que siempre ha estado a mi lado siempre y me ha apoyado en todo.

Juan Fernando.

Agradecemos a la Universidad del Azuay, por darnos la oportunidad de educarnos en este prestigioso establecimiento, a todos los profesores que nos transmitieron su conocimiento y experiencia y de manera especial al Ing. Lenin Erazo por haber dirigido el desarrollo de esta monografía.

Índice de contenidos

Índice de ilustraciones

Resumen

Se implementará una solución informática que permita al personal de auditoría, registrar la constatación del saldo físico en un dispositivo móvil para la actualización y conciliación de saldos en el sistema general de la empresa. Se utilizarán dispositivos móviles Pocket PC. La programación se realiza en Visual Basic .Net, utilizando los conceptos de programación para dispositivos móviles, archivos XML, servicios Web y redes inalámbricas Wi-Fi.

Abstract

An informatics solution will be implemented that will allow auditory personnel to register the presence of physical stock in a mobile unit for the updating and conciliation of stock in the company's general system. Mobile Pocket PCs will be used. The programming will be done in Visual Basic .Net, using the programming concepts for mobile units, XML files, Web services and Wi-Fi wireless connections.

Introducción

El Centro Comercial "Center Plaza San Blas" ubicado en la ciudad de Cuenca, cuenta con varias plantas de exhibición y ventas de varios tipos de productos, entre los principales: artículos de comisariato, artículos de bazar, accesorios para bebe, juguetes, artículos navideños, ferretería y electrodomésticos.

La empresa cuenta con un sistema de gestión desarrollado por el personal propio, así como con una infraestructura tecnológica actualizada y en función de los requerimientos que un negocio de este tipo genera.

Uno de los procesos que requiere mayor atención es el de revisar y consolidar los saldos de los artículos, este proceso es continuo y permanente.

Es necesario contar con un inventario actualizado en todo momento, pero es común que los saldos en el sistema no sean los reales por varias razones, entre las principales están:

- Pérdida de productos, al ser artículos pequeños es muy común que las personas tomen estos productos del lugar en que están exhibidos.
- Por "cruces" entre productos, esto sucede con frecuencia por errores del personal que registra los productos al recibirlos del proveedor, en especial con los productos que tienen descripciones, características y presentaciones muy parecidas, registran un producto con el código de otro con características similares.
- Una razón también es por la mala ubicación de los productos, es decir un producto que tiene un lugar asignado dentro de todo el edificio de exhibición y ventas se encuentra en otra planta o en otro piso de exhibición. Un producto es colocado en una planta sin que haya la transferencia respectiva en el sistema.

Debido al alto número de artículos el proceso de revisión de inventarios por parte del personal de auditoría es muy arduo.

El proceso actual comprende imprimir un reporte de saldos en el sistema, luego se procede con la constatación física de los saldos de cada producto del listado. Una vez que se ha terminado de anotar el saldo actual de los productos, se verifican cada uno de los productos con el saldo del sistema y se registran en el sistema las regulaciones necesarias para igualar los saldos.

Tanto para el personal de auditoría como para la dirección de la empresa es importante que este proceso de constatación y conciliación de saldos sea cada vez más eficiente, optimizando el recurso humano y tiempo necesarios.

Tomando en cuenta todo lo anterior se decidió implementar una solución informática para la constatación física de los saldos de los productos, con el fin de que el trabajo del personal sea más rápido y eficiente.

La solución informática utilizará dispositivos móviles Pocket PC con dispositivo de lector de código de barras que permitirán registrar la constatación de los saldos físicos y su actualización de la manera más rápida y fácil.

Se implementarán una solución que permite al usuario actualizar los saldos de 2 maneras, la primera ***online*** que consiste en actualizar directamente los saldos en el sistema central a través de una conexión inalámbrica y la segunda ***offline***, la cual permite llevar los datos en los dispositivos móviles y luego de registrar en estos los saldos se actualiza en el sistema central mediante una conexión del dispositivo al sistema.

CAPITULO 1

VISUAL BASIC.NET

1.1. Introducción

Visual Basic.NET (VB.NET) es una mejora del lenguaje de programación Visual Basic (VB), el que ha tenido una gran evolución convirtiéndose en un lenguaje de programación orientado a objetos y eventos, implementado y basado en Framework .NET requiriendo de esté para su ejecución. Las diferencias entre VB y VB.NET son profundas, sobre todo en cuanto a metodología de programación y librerías, pero ambos lenguajes siguen manteniendo un gran parecido.

VB.NET se encuentra dentro una plataforma llamada Visual Studio.NET (VS.NET), la que cuenta con varios lenguajes de programación como: VB.NET, C#, C++ y J#. VS.NET contiene un amplio conjunto de bibliotecas de desarrollo, permitiéndonos el desarrollo de todo tipo de funcionalidades: desde programas de consola o servicios Windows hasta aplicaciones para dispositivos móviles (Pocket PC), pasando por desarrollos de escritorio o para Internet.

Cada lenguaje de programación dentro de VS.NET tendrá sus propias sentencias y métodos de programación, para esto VS.NET contiene un entorno de ejecución conocido como CLR (Lenguaje Común de Ejecución), él cual es una implementación de Microsoft de un estándar llamado CLI (Infraestructura de Lenguaje Común), lo que permite que trabaje con cualquier Framework .NET.

Cualquier aplicación creada en .NET puede ser utilizado de forma transparente desde cualquier otro lenguaje .NET, para esto cuenta con el CLS (Lenguaje Común de Especificación), esté define un conjunto de tipos de datos comunes CTS (Sistema de Tipos Comunes), lo que nos permite identificar los diferentes tipos de datos que se pueden manejar, cómo se declaran y se utilizan éstos y de qué manera se deben gestionar durante la ejecución.

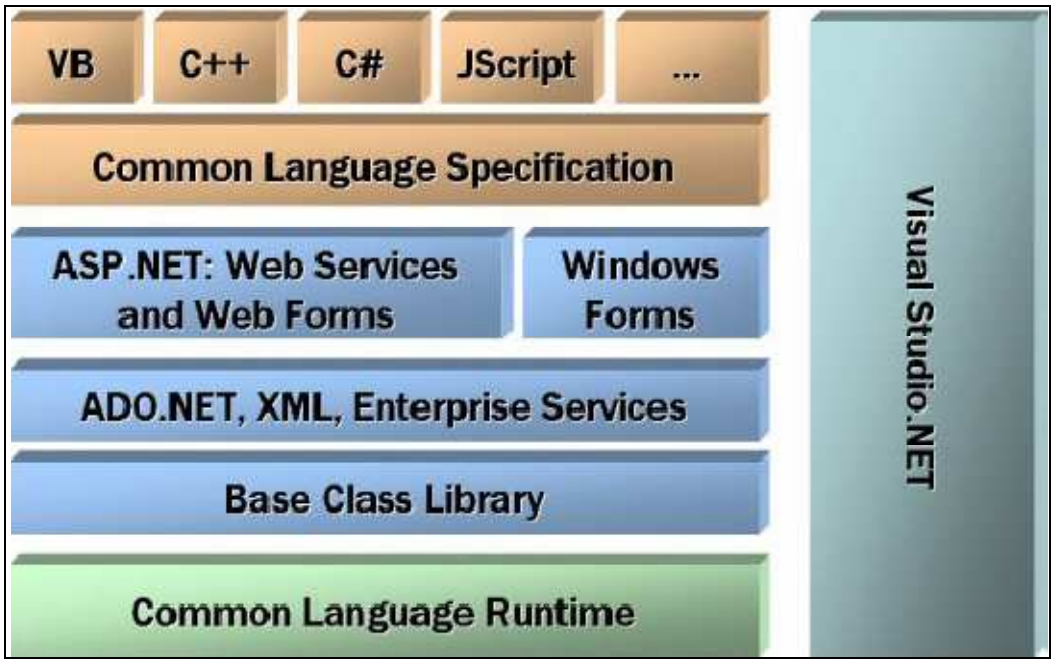


Ilustración 1. Elementos de la Plataforma Visual Studio.NET

1.2. Windows Forms

Windows Forms es la denominación que reciben los Formulario Windows, que son la representación de las muy conocidas ventanas utilizadas en el Sistema Operativo Windows, un cuadro de diálogo que constituye la interfaz de usuario de una aplicación.

Los formularios Windows se encuentran dentro de un espacio de nombres el espacio System.Windows.Forms, el cual contiene un conjunto de clases, estructuras, enumeraciones, etc., que permiten la creación de aplicaciones del tipo Windows.

La clase necesaria para la creación de los formularios es la Forms, esta contiene todos los elementos necesarios para esta función. Dentro de una ventana podremos colocar diferentes controles como: cajas de texto, botones, ListBox, etc., lo que permitirá la interacción con el usuario.

Cabe recalcar que los Windows Forms sólo se podrán ejecutar en un Sistema Operativo Windows, por lo que no ejecutaran en el internet o en algún otro sistema.

1.3. Web Forms

Web Forms (Formularios Web), son la base para la creación de una página Web en .NET, y son los que se ejecutan en las paginas ASP.NET (Active Server Pages .NET) que son el medio con el que podemos programar aplicaciones para Internet.

La principal aportación de ASP.NET al mundo de la programación es que ha llevado a la Web el paradigma de la programación orientada a eventos propia de aplicaciones de escritorio, ofreciendo:

Separación entre diseño y lógica.

Componentes de interfaz de usuario, tanto estándar como de terceras empresas o propios.

Diseñadores gráficos.

Eventos.

Estado.

Enlazado a datos desde la interfaz.

1.4. ADO .NET

El acceso a fuentes de datos es algo indispensable en cualquier lenguaje o plataforma de desarrollo. La biblioteca de clases base que se especializa en el acceso a datos se denomina ADO.NET.

ADO.NET se puede definir como:

- Un conjunto de interfaces, clases, estructuras y enumeraciones que permiten el acceso a datos desde la plataforma .NET de Microsoft.
- Permite un modo de acceso desconectado a los datos, los cuales pueden provenir de múltiples fuentes de datos y de diferentes arquitecturas de almacenamiento.
- Soporta un completo modelo de programación y adaptación, basado en el estándar XML.

1.4.1. Beneficios de ADO.NET

- **Interoperabilidad**

Las aplicaciones basadas en ADO.NET obtienen ventaja de la flexibilidad y la masiva aceptación del estándar XML para el intercambio de datos. Puesto que XML es el estándar de envío de información entre capas, cualquier componente capaz de interpretar los datos XML puede acceder a la información de ADO .NET, se encuentre donde se encuentre, y procesarla. Además, puesto que la información se envía en flujos de XML, no importa la implementación empleada para enviar o recoger la información así como la plataforma empleada. Simplemente se exige a los componentes que reconozcan el formato XML empleado para el proceso, envío y recepción de un DataSet.

- **Mantenimiento**

En el ciclo de vida de una aplicación los cambios poco sustanciales y modestos son permisibles. Pero cuando es necesario abordar un cambio estructural o arquitectónico del sistema, la tarea se vuelve demasiado compleja y a veces inviable. Es por ello que se separa la estructura de un programa en varias capas. Una de esas capas es la de datos, que es fundamental desarrollar correctamente. Gracias a los DataSet, la tarea de portar y aumentar los procesos de datos y de negocio será más sencillo: el intercambio de información a través de XML, hace que sea más sencilla la tarea de estructurar en más capas la aplicación, convirtiéndola en más modular y fácil de mantener.

- **Rendimiento**

Puesto que trabajamos con objetos de datos desconectados, todo el proceso se acelera, ya que no tenemos que estar comunicándonos con el servidor. Además, gracias al modelo de XML la conversión de tipos no es necesaria. Se reduce pues el ancho de banda disponible, se independiza más el cliente del servidor, y se descarga más a éste, que puede estar dedicado a otras tareas en lo que el cliente analiza sus datos.

- **Escalabilidad**

Las aplicaciones Web tienen un número ilimitado de conexiones potenciales debido a la naturaleza de Internet. Los servidores son capaces de atender muy bien decenas y decenas de conexiones. Pero cuando hablamos de miles y millones, los servidores ya

no son capaces de realizar correctamente su trabajo. Esto es debido a que por cada usuario se mantiene una memoria de proceso y conexión, un conjunto de bloqueos de recursos como puedan ser tablas, índices, etc., y una comprobación de sus permisos; todo ello consume tiempo y recursos.

ADO .NET favorece la escalabilidad, puesto que su modelo de conexión Off-Line evita que se mantengan los recursos reservados más tiempo del considerado necesario. Esto permite que más usuarios por unidad de tiempo puedan acceder a la aplicación sin problemas de tiempos.

1.4.2. Arquitectura de ADO .NET

El objeto más importante a la hora de trabajar con el nuevo modelo de acceso a datos es el ***DataSet***. Se podría calificarlo casi como un motor de datos relacionales en memoria.

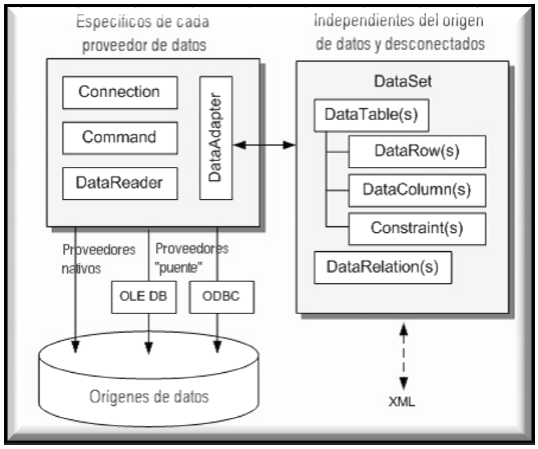


Ilustración 2. Arquitectura de ADO.NET

Existen dos capas fundamentales dentro de su arquitectura: la ***capa conectada*** y la ***desconectada***.

1.4.2.1. Capa conectada

La primera de ellas contiene objetos especializados en la conexión con los orígenes de datos. Así, la clase genérica ***Connection*** se utiliza para establecer conexiones a los orígenes de datos. La clase ***Command*** se encarga de enviar comandos de toda índole al origen de datos. Por fin la clase ***Data-Reader*** está especializada en leer los resultados de los comandos mientras se permanece conectado al origen de datos.

La clase ***DataAdapter*** hace uso de las tres anteriores para actuar de puente entre la capa conectada y la desconectada.

Estas clases son abstractas, es decir, no tienen una implementación real de la que se pueda hacer uso directamente. Es en este punto en donde entran en juego los ***proveedores de datos***. Cada origen de datos tiene un modo especial de comunicarse con los programas que los utilizan, además de otras particularidades que se deben contemplar. Un proveedor de datos de ADO.NET es una implementación concreta de

las clases conectadas abstractas que hemos visto, que hereda de éstas y que tiene en cuenta ya todas las particularidades del origen de datos en cuestión.

Así, por ejemplo, las clases específicas para acceder a SQL Server se llaman ***SqlConnection***, ***SqlCommand***, ***SqlDataReader*** y ***SqlDataAdapter*** y se encuentran bajo el espacio de nombres ***System.Data.SqlClient***.

Para acceder a Oracle versión 10 se utiliza el espacio de nombres Oracle.DataAcces que contiene sus propias clases para interactuar con los datos, entre las principales están ***OracleCommand, OracleCommandBuilder, OracleConnection, OracleDataAdapter, OracleDataReader, OracleParameter***.

Existen ***proveedores nativos***, que son los que se comunican directamente con el origen de datos (por ejemplo el de SQL Server o el de Oracle), y proveedores "puente", que se utilizan para acceder a través de ODBC u OLEDB cuando no existe un proveedor nativo para un determinado origen de datos. Estos proveedores puente, si bien muy útiles en determinadas circunstancias, ofrecen un rendimiento menor debido a la capa intermedia que están utilizando (ODBC u OLEDB).

1.4.2.2. Capa desconectada

ADO .NET está basado en una arquitectura desconectada de los datos. En una aplicación de datos se ha comprobado que mantener los recursos reservados mucho tiempo, implica reducir el número de usuarios conectados y aumenta el proceso del sistema al mantener una política de bloqueos y transacciones. Al mismo tiempo, si la aplicación mantiene más de un objeto simultáneamente, se encuentra con el problema de tener que estar continuamente conectando con el servidor para alimentar las relaciones existentes entre ambas, subiendo y bajando información.

Con ADO .NET se consigue estar conectado al servidor sólo lo estrictamente necesario para realizar la operación de carga de los datos en el DataSet. De esta manera se reducen los bloqueos y las conexiones a la mínima expresión. Se pueden soportar muchos más usuarios por unidad de tiempo y disminuyen los tiempos de respuesta, a la par que se aceleran las ejecuciones de los programas.

Tradicionalmente, el recoger información de una base de datos ha ido destinado a realizar un proceso con dicha información: mostrarla por pantalla, procesarla o enviarla a algún componente.

Frecuentemente, la aplicación no necesita una única fila, sino un buen conjunto de ellas. Además, también frecuentemente, ese conjunto de filas procede no de una tabla sino de una unión de múltiples tablas (join de tablas). Una vez que estos datos son cargados, la aplicación los trata como un bloque compacto. En un modelo desconectado, es inviable el tener que conectar con la base de datos cada vez que avanzamos un registro para recoger la información asociada a ese registro (condiciones del join).

Para solucionarlo, lo que se realiza es almacenar temporalmente toda la información necesaria donde sea necesario y trabajar con ella. Esto es lo que representa un DataSet en el modelo ADO .NET.

Un DataSet es una caché de registros recuperados de una base de datos que actúa como un sistema de almacenamiento virtual, y que contiene una o más tablas basadas en las tablas reales de la base de datos. Adicionalmente, almacena las relaciones y reglas de integridad existentes entre ellas para garantizar la estabilidad e integridad de la información de la base de datos. Muy importante es recalcar, que los DataSet son almacenes pasivos de datos, esto es, no se ven alterados ante cambios subyacentes de la base de datos. Es necesario recargarlos siempre que queramos estar *al día* en cuanto a datos se refiere.

Una de las mayores ventajas de esta implementación, es que una vez obtenido el DataSet, éste puede ser enviado (en forma de flujo XML) entre distintos componentes de la capa de negocio, como si de una variable más se tratase, ahorrando así comunicaciones a través de la base de datos. Una consecuencia lógica de este tipo de arquitecturas, es la de conseguir que los DataSet sean independientes de los orígenes de datos. Los drivers OLE-DB transformarán la consulta SQL en un cursor representado con una estructura XML, que es independiente del motor de la base de datos.

Esto nos permitirá trabajar con múltiples orígenes de datos, de distintos fabricante e incluso en formatos que no pertenezcan a bases de datos, por ejemplo, ficheros

planos u hojas de cálculo, lo que representa un importante punto de compatibilidad y flexibilidad.

La persistencia es un concepto muy interesante en el mundo del desarrollo. Es un mecanismo por el cual un componente puede almacenar su estado (valores de variables, propiedades, datos...en un momento concreto del tiempo) en un soporte de almacenamiento fijo. De manera, que cuando es necesario, se puede recargar el componente tal y como quedó en una operación anterior.

En un sistema de trabajo Off-Line como el que plantea ADO .NET, la persistencia es un mecanismo fundamental. Podemos cerrar la aplicación y mantener persistentes todos los DataSet necesarios, de manera que al reiniciarla, nos encontramos los DataSet tal y como los dejamos. Ahorrando el tiempo que hubiera sido necesario para recuperar de nuevo toda esa información del servidor. Optimizando todavía más el rendimiento del sistema distribuido.

El formato que emplea ADO .NET para almacenar su estado es XML. Puesto que ya es un estándar de la industria, esta persistencia nos ofrece las siguientes cualidades:

- La información puede estar accesible para cualquier componente del sistema que entienda XML.
- Es un formato de texto plano, no binario, que lo hace compatible con cualquier componente de cualquier plataforma, y recuperable en cualquier circunstancia.

1.4.3. DataSet

El API de ADO .NET proporciona una superclase, DataSet, que encapsula lo que sería la base de datos a un nivel lógico: tablas, vistas, relaciones, integridad entre todos ellos, etc., pero siempre con independencia del tipo de fabricante que la diseñó. Aquí se tiene el mejor concepto de datos desconectados: una copia en el cliente de

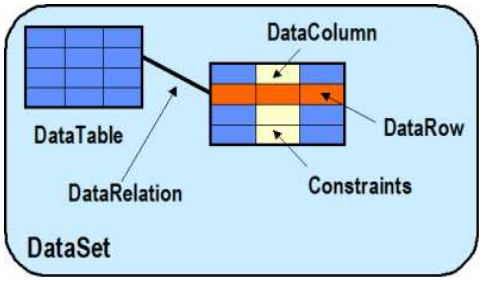


Ilustración 3: Esquema de un DataSet

la arquitectura de la base de datos, basada en un esquema XML que la independiza del fabricante, proporcionando al desarrollador la libertad de trabajo independiente de la plataforma.

Esta clase se compone a su vez, de clases de soporte, que representan cada una, los elementos arquitecturales de la base de datos: tablas, columnas, filas, sus reglas de chequeo, sus relaciones, las vistas asociadas a la tabla, etc.

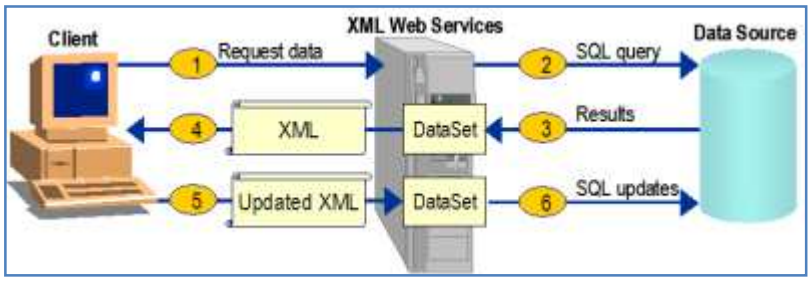


Ilustración 4. Esquema de la relación entre ADO.NET y XML

1.5. XML

XML, sigla en inglés de *Extensible Markup Language* ("lenguaje de marcas ampliable"), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C)[1]. XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo, entre otros.

XML, con todas las tecnologías relacionadas, representa una manera distinta de hacer las cosas, más avanzada, cuya principal novedad consiste en permitir compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. Así pues, el XML juega un papel importantísimo en este mundo actual, que tiende a la globalización y la compatibilidad entre los sistemas, ya que es la

[1] El Consorcio World Wide Web (W3C) es un internacional donde las organizaciones miembro, trabajan conjuntamente para desarrollar estándares Web. La misión del consorcio W3C es: **Guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web.**

tecnología que permitirá compartir la información de una manera segura, fiable y fácil. Además, XML permite al programador y al personal de soporte y mantenimiento de la aplicación dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos, ya que algunas tareas tediosas como la validación de estos o el recorrido de las estructuras corre a cargo del lenguaje y está especificado por el estándar, de modo que el programador no utilizará recursos en controlar dichas tareas.

El XML se creó para que cumpliera varios objetivos.

- Que fuera idéntico a la hora de servir, recibir y procesar la información.
- Que fuera formal y conciso desde el punto de vista de los datos y la manera de guardarlos.
- Que fuera extensible, para que lo puedan utilizar en todos los campos del conocimiento.
- Que fuese fácil de leer y editar.
- Que fuese fácil de implantar, programar y aplicar a los distintos sistemas.

El XML se puede usar para infinidad de trabajos y aporta muchas ventajas en amplios escenarios. Veamos algunas ventajas del XML en algunos campos prácticos.

- Comunicación de datos. Si la información se transfiere en XML, cualquier aplicación podría escribir un documento de texto plano con los datos que estaba manejando en formato XML y otra aplicación recibir esta información y trabajar con ella.
- Migración de datos. Si tenemos que mover los datos de una base de datos a otra sería muy sencillo si las dos trabajasen en formato XML.
- Aplicaciones web. Hasta ahora cada navegador interpreta la información a su manera y los programadores del web tienen que hacer unas cosas u otras en

función del navegador del usuario. Con XML tenemos una sola aplicación que maneja los datos y para cada navegador o soporte podremos tener una hoja de estilo o similar para aplicarle el estilo adecuado.

CAPITULO 2

PROGRAMACION PARA EQUIPOS MÓVILES CON VISUAL BASIC .NET

2.1 Introducción

Hace ya varios años que Microsoft disponía de entornos específicos para la programación de dispositivos móviles (PDAs, smartphones, entre otros.). Se trataba de entornos particulares, exclusivamente para este tipo de programación y que requerían de perfiles de desarrolladores muy concretos y especializados.

Esto no solo ocurría con los dispositivos móviles, teníamos entornos especializados para la programación de aplicaciones Web, integración para office, etc. Obviamente esto representaba un problema, y era que cada programador o diseñador de software debía tener un perfil específico por tipos de aplicaciones, Web, móviles, office, etc. No solo era un problema para los propios programadores que debían conocer múltiples SDKs y sobre todo entornos diferentes, sino sobre todo para las empresas, que no podían reutilizar a sus desarrolladores en diferentes tipos de proyectos.

Otro problema que suponía este paradigma de programación es que a menudo aplicaciones muy similares para diferentes entornos (Web, móviles) o ya incluso para diferentes versiones de los dispositivos móviles (Windows CE) requerían ser codificadas por separado, pudiéndose reutilizar poco código entre todas ellas.

.NET tenía justamente esta visión, un entorno de desarrollo único, independientemente del tipo de aplicación (Web, móvil, etc.) y que permitiese poder ejecutar el mismo código en diferentes dispositivos.

Para ello se diseñó el .NET Framework.

2.2 .NET Framework.

.NET Framework, una capa de abstracción entre el hardware del dispositivo y el código fuente, que permitía aislar al programador del tipo de hardware y sistema para el que se codificaba en la mayor medida de lo posible. Toda aplicación .NET requiere por lo tanto este framework instalado en la máquina. Un framework que en

muchos casos ya viene con el sistema operativo o, que en caso contrario, puede descargarse de forma rápida y sencilla.

Independientemente del lenguaje que usemos en Visual Studio (Visual Basic, C#, J# o cualquier otro manejado de otro fabricante), todos generan lo que se conoce como Código Intermedio (IL Code). Este es el código que se compila en tiempo real por el .NET Framework y luego es ejecutado en última instancia directamente sobre el procesador.

2.3 Requisitos para la programación de dispositivos móviles en Visual Basic .Net

Lo primero que se necesita es Visual Studio 2005 o Visual Studio 2008, cualquiera de estas dos versiones ofrece un entorno de desarrollo para aplicaciones sobre Pocket PC 2003, Smartphone 2003 y Windows CE 5.0. Dependiendo de los nuevos dispositivos que van apareciendo en el mercado se pueden descargar el SDK de actualización para Visual Studio.

Otro requisito es ActiveSync, el módulo que permite la conexión desde un PC con un dispositivo móvil y está disponible para descargarse de forma gratuita desde la Web.

Se necesita obviamente un dispositivo sobre el que se pueda probar la aplicación. Se puede usar un dispositivo físico conectado al PC a través de ActiveSync o uno de los emuladores que vienen por defecto con Visual Studio. Estos emuladores utilizan tecnología de virtualización y contienen una imagen binaria del Sistema Operativo original, con lo cual la compatibilidad con el dispositivo físico real es del 100%.

Independientemente del tipo de dispositivo que escojamos veremos como en tiempo real podremos hacer el despliegue de la aplicación al mismo tiempo y depurar la ejecución en el dispositivo desde el Visual Studio en el PC.

Una vez que se tiene el Interfaz de Usuario creado en Visual Studio se podrá:

- Utilizar componentes de negocio que ya tenemos escritos en .NET simplemente agregando la referencia en el proyecto. Si tenemos ya una aplicación 3 capas con un buen diseño donde la lógica de negocio son

librerías de componentes con código de negocio exclusivamente (cálculos, acceso a BDs, etc.) podremos utilizar dicho assembly en el dispositivo.

- Utilizar una Base de Datos. Podremos agregar una referencia a un servidor de Base de Datos remoto como lo haríamos desde una aplicación Windows o Web (al que conectaríamos de forma transparente vía Wi-Fi/GPRS/3G, etc.) o por ejemplo a una versión de SQL CE instalada en el dispositivo móvil. Esta versión de SQL soporta hasta 2Gb de BDs en el dispositivo y la funcionalidad necesaria para programar aplicaciones móviles. La sincronización con una BD central de Oracle podremos hacerla manualmente o dejar al motor que la realice vía HTTP (de nuevo de forma transparente sobre Wi-Fi/GPRS/3G u otras conexiones que tengamos en el dispositivo).

- Llamar a un Web Service. Podremos agregar una referencia Web para hacer una llamada a un Web Service vía HTTP de forma transparente. De esta forma tendríamos un cliente ligero en el dispositivo con la Interfaz de Usuario y nos conectaríamos a nuestro sistema vía HTTP por Servicios Web.

Eso son solo algunas de las opciones de lo que se puede realizar, pero obviamente las posibilidades son ilimitadas con todo el .NET Compact Framework. La cantidad de dispositivos son enormes, desde Windows CE hasta SmartPhones pasando por PDAs, así que las posibilidades son infinitas.

CAPITULO 3

REDES INALAMBRICAS (Wi-Fi)

3.1. Introducción

Wi-Fi es un sistema de envío de datos sobre redes computacionales que utiliza ondas de radio en lugar de cables, además es una marca de la Wi-Fi Alliance, anteriormente conocida como WECA: Wireless Ethernet Compatibility Alliance, Alianza de Compatibilidad Ethernet Inalámbrica, creada en 1999 por Nokia y Symbol Technologies, que en el 2003 pasaría a denominarse Wi-Fi Alliance la organización comercial que adopta, prueba y certifica que los equipos cumplen los estándares 802.11., fomenta más fácilmente la tecnología inalámbrica y asegura la compatibilidad de equipos.

De esta forma en abril de 2000 WECA certifica la interoperabilidad de equipos según la norma IEEE 802.11b. Esto quiere decir que el usuario tiene la garantía de que todos los equipos que tengan el sello Wi-Fi pueden trabajar juntos sin problemas, independientemente del fabricante de cada uno de ellos.

3.2 Estándares Existentes.

La norma IEEE 802.11 fue diseñada para sustituir el equivalente a las capas físicas y MAC de la norma 802.3 (Ethernet). Esto quiere decir que en lo único que se diferencia una red Wi-Fi de una red Ethernet es en cómo se transmiten las tramas o paquetes de datos; el resto es idéntico. Por tanto, una red local inalámbrica 802.11 es completamente compatible con todos los servicios de las redes locales (LAN) de cable 802.3 (Ethernet).

Los estándares IEEE 802.11b e IEEE 802.11g disfrutan de una aceptación internacional debido a que la banda de 2.4 GHz está disponible casi universalmente, con una velocidad de hasta 11 Mbps y 54 Mbps, respectivamente.

En la actualidad ya se maneja también el estándar IEEE 802.11a, conocido como WI-FI 5, que opera en la banda de 5 GHz y que disfruta de una operatividad con canales relativamente limpios. La banda de 5 GHz ha sido recientemente habilitada y,

además no existen otras tecnologías (Bluetooth, microondas, ZigBee, WUSB) que la estén utilizando, por lo tanto existen muy pocas interferencias. Su alcance es algo menor que el de los estándares que trabajan a 2.4 GHz (aproximadamente un 10%), debido a que la frecuencia es mayor (a mayor frecuencia, menor alcance).

El estándar 802.11g es capaz de alcanzar transferencias a 108 Mbps.

3.3 Seguridad y fiabilidad

Uno de los problemas más graves a los cuales se enfrenta actualmente la tecnología Wi-Fi es la progresiva saturación del espectro radioeléctrico, debida a la masificación de usuarios, esto afecta especialmente en las conexiones de larga distancia (mayor de 100 metros). En realidad Wi-Fi está diseñado para conectar ordenadores a la red a distancias reducidas, cualquier uso de mayor alcance está expuesto a un excesivo riesgo de interferencias.

Un muy elevado porcentaje de redes son instaladas sin tener en consideración la seguridad convirtiendo así sus redes en redes abiertas (o muy vulnerables a los crackers), sin proteger la información que por ellas circulan.

3.4 Ventajas y desventajas

Las redes Wi-Fi poseen una serie de ventajas, entre las cuales podemos destacar:

- Al ser redes inalámbricas, la comodidad que ofrecen es muy superior a las redes cableadas porque cualquiera que tenga acceso a la red puede conectarse desde distintos puntos dentro de un rango suficientemente amplio de espacio.

- Una vez configuradas, las redes Wi-Fi permiten el acceso de múltiples ordenadores sin ningún problema ni gasto en infraestructura, no así en la tecnología por cable.

- La Wi-Fi Alliance asegura que la compatibilidad entre dispositivos con la marca Wi-Fi es total, con lo que en cualquier parte del mundo podremos utilizar la tecnología Wi-Fi con una compatibilidad total. Esto no ocurre, por ejemplo, en móviles.

Pero como red inalámbrica, la tecnología Wi-Fi presenta los problemas intrínsecos de cualquier tecnología inalámbrica. Algunos de ellos son:

- Una de las desventajas que tiene el sistema Wi-Fi es la pérdida de velocidad en comparación a una conexión con cables, debido a las interferencias y pérdidas de señal que el ambiente puede acarrear.

- La desventaja fundamental de estas redes existe en el campo de la seguridad. Existen algunos programas capaces de capturar paquetes, trabajando con su tarjeta Wi-Fi, de forma que puedan calcular la contraseña de la red y de esta forma acceder a ella. Las claves de tipo WEP son relativamente fáciles de conseguir con este sistema. La alianza Wi-Fi arregló estos problemas sacando el estándar WPA y posteriormente WPA2, basados en el grupo de trabajo 802.11i. Las redes protegidas con WPA2 se consideran robustas dado que proporcionan muy buena seguridad. De todos modos muchas compañías no permiten a sus empleados tener una red inalámbrica ya que sigue siendo difícil para lo que representa la seguridad de una empresa estar "seguro". Uno de los puntos débiles (sino el gran punto débil) es el hecho de no poder controlar el área que la señal de la red cubre, por esto es posible que la señal exceda el perímetro del edificio y alguien desde afuera pueda visualizar la red y esto es sin lugar a dudas una mano para el posible atacante.

- Hay que señalar que esta tecnología no es compatible con otros tipos de conexiones sin cables como Bluetooth, GPRS, UMTS, etc.

CAPITULO 4

ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE (ERS)

4.1 Introducción

Este documento es una Especificación de Requisitos de Software del Interfaz en Visual Basic .Net para el mantenimiento de inventarios utilizando equipos móviles PocketPC. Todo su contenido será elaborado con la colaboración de los usuarios, directivos y desarrolladores de la empresa.

4.2 Propósito

El objetivo de la especificación es definir de manera clara y precisa las funcionalidades y condicionantes técnicas del sistema computacional que se desea desarrollar.

4.3 Ámbito del sistema

El sistema contempla lo siguiente:

- Registro de la constatación física del inventario.
- Movimientos de ingreso o egreso de mercadería en el sistema general.

El objetivo del sistema es contar con una solución que permita realizar la tarea de control de saldos de inventario, registrar el inventario físico y actualizar el inventario en el sistema general.

4.4 Definiciones

Ubicación	Lugar de exhibición o almacenamiento de mercadería dentro del local comercial (bodega)
Kárdex	Registro de movimientos del inventario
Movimiento	Se refiere a los diferentes tipos de ingresos, compromisos de egreso y despachos que se puede realizar con los ítems.
Característica	Se refiere a las cualidades físicas o lógicas de un ítem

4.5 Requisitos funcionales

4.5.1 Sistema fuera de línea

4.5.1.1 Casos de uso

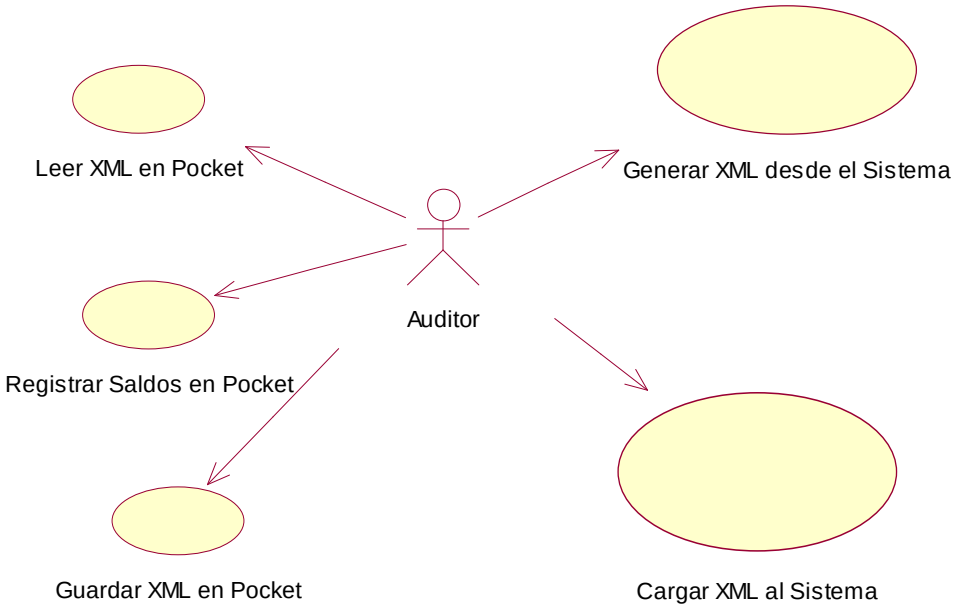


Ilustración 5. Casos de Uso, Sistema fuera de línea

4.5.1.2 Descripción de actores

Auditor: Encargado de realizar todos los procesos del sistema, tanto en el interfaz del sistema como en la interfaz del Pocket PC.

4.5.1.3 Descripción de casos de uso

Caso de uso 10	Generar XML desde Sistema
Actor:	Auditor
Descripción:	Módulo donde se generarán los archivos XML que serán utilizados en la aplicación de los Pocket PC.
Prioridad:	Obligatorio
REQUISITOS ASOCIADOS	

R.10.1 El sistema permitirá consultar los productos existentes de la base de datos. R.10.2 El sistema permitirá seleccionar los productos que se generarán en los archivos XML de acuerdo al criterio seleccionado por el usuario. R.10.3 El sistema generará los archivos XML en el disco local. R.10.4 El sistema actualizará los archivos XML en los Pocket PC utilizando el software ActiveSync.

Caso de uso 20	Leer XML en Pocket
Actor:	Auditor
Descripción:	La interfaz en el Pocket PC permite con esta opción leer y cargar los datos contenidos en los archivos XML a las tablas de datos.
Prioridad:	Obligatorio
REQUISITOS ASOCIADOS	
R.20.1 El sistema leerá los archivos XML. R.20.2 El sistema llenará las tablas de datos con los contenidos de los archivos XML. R.20.3 El sistema notificará cuando el proceso se haya completado.	

Caso de uso 30	Registrar saldos en Pocket
Actor:	Auditor
Descripción:	La interfaz en el Pocket PC permite con esta opción registrar el saldo físico de los productos.
Prioridad:	Obligatorio
REQUISITOS ASOCIADOS	

R.30.1 El sistema permitirá ingresar un código de producto en la interfaz. R.30.2 El sistema validará que el código del producto este registrado en las tablas. R.30.3 El sistema presentará las siguientes características del producto seleccionado: descripción, referencia, unidad de medida. R.30.4 El sistema presentará los saldos en cada una de las ubicaciones del producto seleccionado, los datos presentados son: Nombre de la ubicación, Saldo del sistema, Saldo Físico. R.30.5 El sistema permitirá seleccionar la ubicación en la que se va a registrar el saldo R.30.6 El sistema permitirá ingresar la información del saldo físico en la ubicación seleccionada. R.30.7 El sistema permitirá seleccionar entre grabar los cambios o salir sin grabar. R.30.8 El sistema actualizará las tablas de datos con los saldos ingresados.

Caso de uso 40	Grabar XML en Pocket
Actor:	Auditor
Descripción:	La interfaz en el Pocket PC permite con esta opción generar los archivos XML con la nueva información de saldos.
Prioridad:	Obligatorio
REQUISITOS ASOCIADOS	
R.40.1 El sistema generará los archivos XML con la información de las tablas de datos del Pocket PC.	

Caso de uso 50	Cargar XML al Sistema
Actor:	Auditor
Descripción:	Se leen los archivos XML y actualiza los inventarios en la base de datos
Prioridad:	Obligatorio
REQUISITOS ASOCIADOS	

R.50.1 El sistema actualizará los archivos XML en la PC utilizando el software ActiveSync.

R.50.2 El sistema llenará las tablas de datos con la información contenida en los XML.

R.50.3 El sistema permitirá seleccionar la opción de actualizar información en la Base de Datos.

R.50.4 El sistema generará los registros necesarios en la Base de Datos para actualizar los saldos de los productos con la información del nuevo saldo físico.

4.5.1.4 Diagrama de secuencias

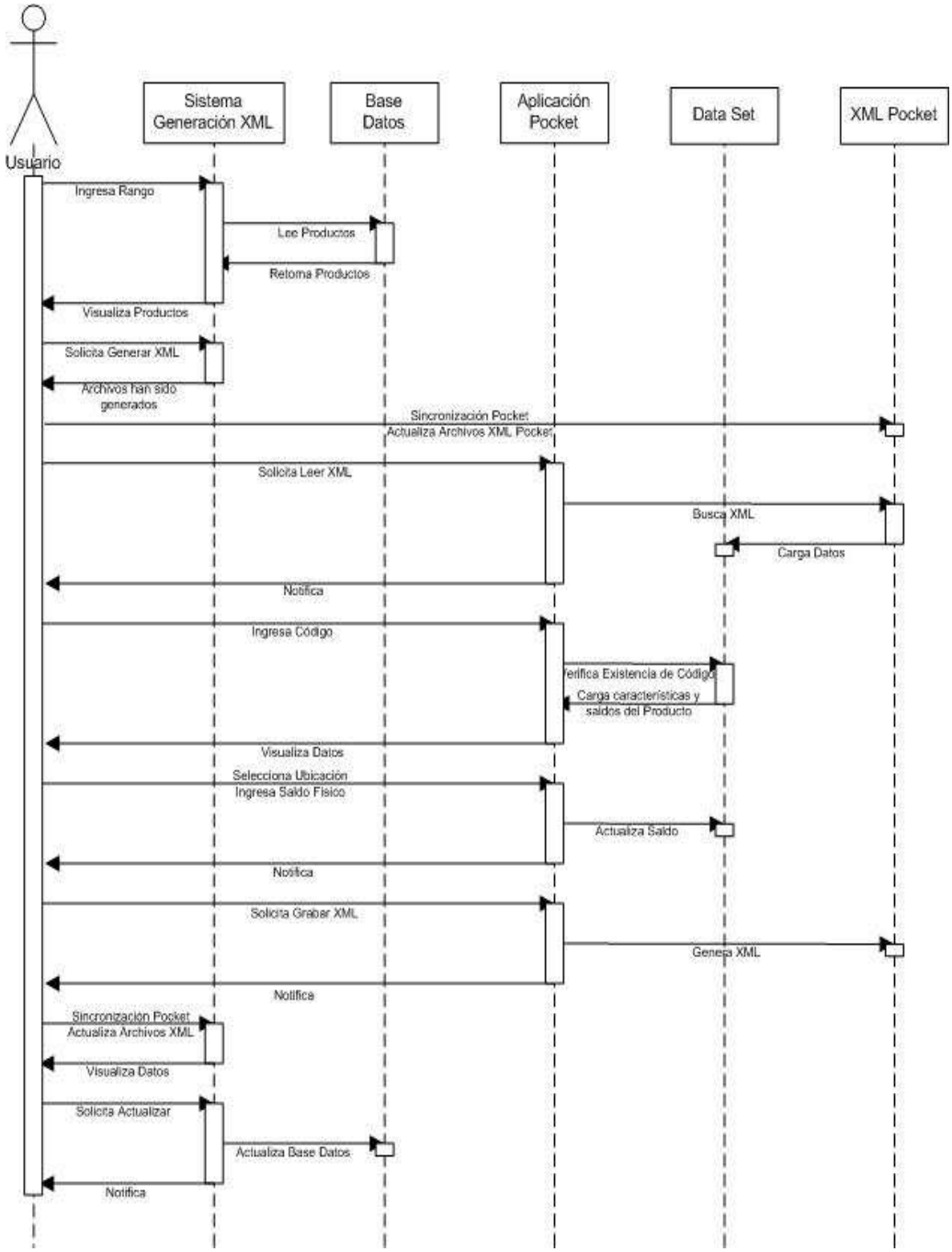


Ilustración 6. Diagrama de Secuencias Sistema fuera de línea

4.5.2 Sistema en línea

4.5.2.1 Caso de Uso

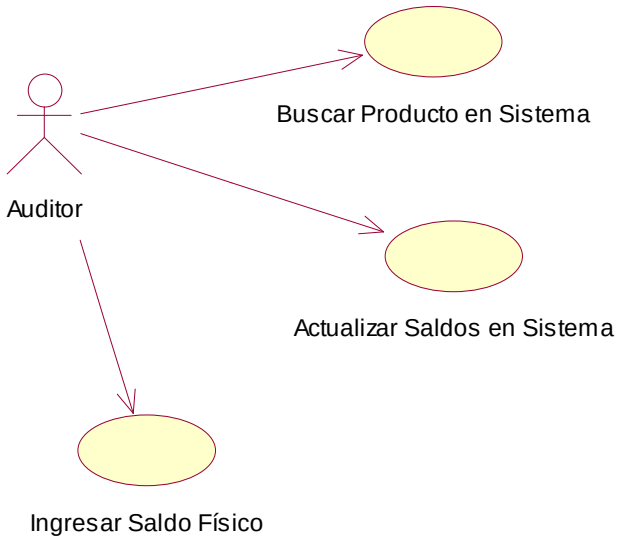


Ilustración 7. Casos de Uso. Sistema Conectado

4.5.2.2 Descripción de actores

Usuario: Encargado de realizar todos los procesos del sistema, tanto en el interfaz del sistema como en la interfaz del Pocket PC.

4.5.2.3 Descripción de casos de uso

Caso de uso 10	Buscar Producto
Actor:	Auditor
Descripción:	El usuario ingresa a la página alojada en el servidor de aplicaciones web y busca el producto para registrar el saldo
Prioridad:	Obligatorio
REQUISITOS ASOCIADOS	

R.10.1 El sistema permitirá ingresar un código de producto en la interfaz. R.10.2 El sistema validará que el código del producto este registrado en las tablas de la base de datos. R.10.3 El sistema presentará las siguientes características del producto seleccionado: descripción, referencia, unidad de medida. R.10.4 El sistema presentará los saldos en cada una de las ubicaciones del producto seleccionado, los datos presentados son: Nombre de la ubicación, Saldo del sistema, Saldo Físico.

Caso de uso 20	Ingresar saldo físico
Actor:	Auditor
Descripción:	Se ingresa el saldo físico del productos seleccionado en la interfaz
Prioridad:	Obligatorio
REQUISITOS ASOCIADOS	
R.20.1 El sistema permitirá seleccionar la ubicación en la que se va a registrar el saldo R.20.2 El sistema permitirá ingresar la información del saldo físico en la ubicación seleccionada. R.20.3 El sistema permitirá seleccionar entre grabar los cambios o salir sin grabar.	

Caso de uso 30	Actualizar Saldos
Actor:	Auditor
Descripción:	En la base de datos se generan los registros de movimiento necesarios para actualizar el saldo del producto
Prioridad:	Obligatorio
REQUISITOS ASOCIADOS	
R.30.1 El sistema verificará el saldo actual del producto R.30.2 El sistema comparará el saldo actual con el físico ingresado por el usuario R.30.3 El sistema generará los registros necesarios en la Base de Datos para actualizar los saldos de los productos con la información del nuevo saldo físico.	

4.5.2.4 Diagrama de secuencias

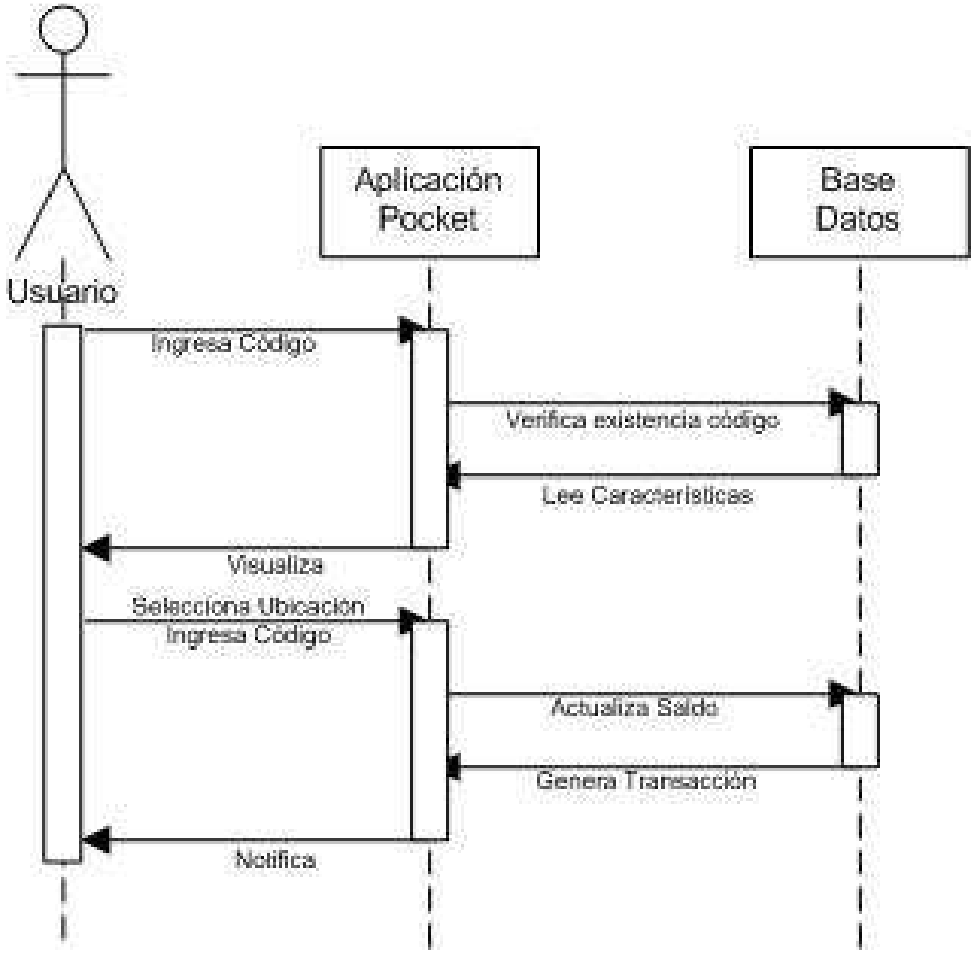


Ilustración 8. Diagrama de Secuencias. Sistema Conectado

4.6 Interfaces de software

- El sistema debe comunicarse con el sistema de inventarios.
- En el primer caso (no conectado) se utilizará el software ActiveSync para sincronizar y actualizar los archivos con los Pocket PC.
- Para la conexión en el segundo caso (conectado) se utilizarán los servicios Web generados en Visual Basic .Net y se conectarán directamente con la base de datos.

4.7 Interfaces de usuario

El sistema de información deberá ofrecer una interfaz de usuario intuitivo, fácil de aprender y sencillo de manejar. El sistema deberá presentar un alto grado de

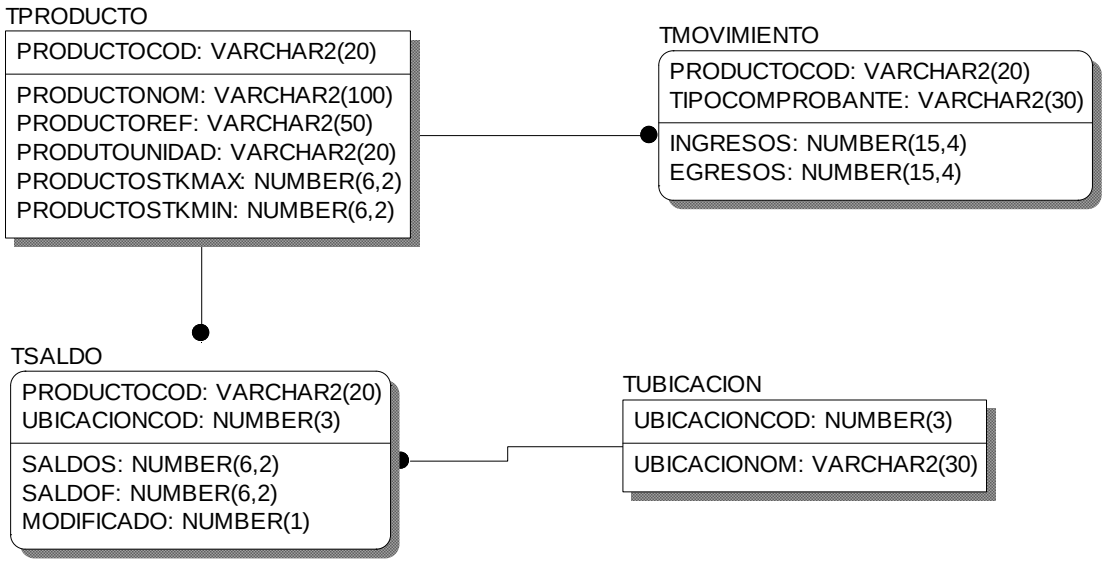
usabilidad. Lo deseable sería que el usuario se familiarice con el sistema en cuestión de pocas horas.

Las interfaces de usuario deben estar diseñadas para ambientes Web y para dispositivos móviles Pocket PC.

CAPITULO 5

DISEÑO E IMPLEMENTACIÓN DE LA APLICACIÓN

5.1 Modelo Entidad Relación



5.2 Definición de tablas y atributos

TABLA	**TPRODUCTO**

CAMPOS			
Name	Type	Null	Comment
PRODUCTOCOD	VARCHAR2(20)	N	Código de barras del producto
PRODUCTONOM	VARCHAR2(100)	N	Nombre del producto
PRODUCTOREF	VARCHAR2(50)	N	Referencia del producto
PRODUCTOUNIDAD	VARCHAR2(20)	N	Nombre de la unidad de medida del producto
PRODUCTOSTKMIN	NUMBER(6,2)	N	Stock mínimo
PRODUCTOSTKMAX	NUMBER(6,2)	N	Stock máximo

La tabla TPRODUCTO contiene los productos seleccionados por el usuario para generar el archivo PRODUCTO.XML.

TABLA	**TSALDO**

CAMPOS			
Name	Type	Null	Comment
PRODUCTOCOD	VARCHAR2(20)	N	Código de barras del producto
UBICACIONCOD	NUMBER(3)	N	Código de la Ubicación (bodega)
SALDOS	NUMBER(15,4)	N	Saldo del sistema
SALDOF	NUMBER(15,4)	N	Saldo Físico
MODIFICADO	NUMBER(1)	N	Si el saldo fue modificado o no desde la aplicación

La tabla TPRODUCTO contiene los saldos de los productos seleccionados en cada una de las ubicaciones o bodegas, de esta tabla se genera el archivo SALDO.XML.

TABLA	**TUBICACION**

CAMPOS			
Name	Type	Null	Comment
UBICACIONCOD	NUMBER(3)	N	Código de la Ubicación o bodega
UBICACIONNOM	VARCHAR2(30)	N	Nombre de la Ubicación o bodega

La tabla TUBICACION contiene las ubicaciones o bodegas que identifican el lugar o el piso de exhibición de la mercadería en el sistema general.

TABLA	**TMOVIMIENTO**

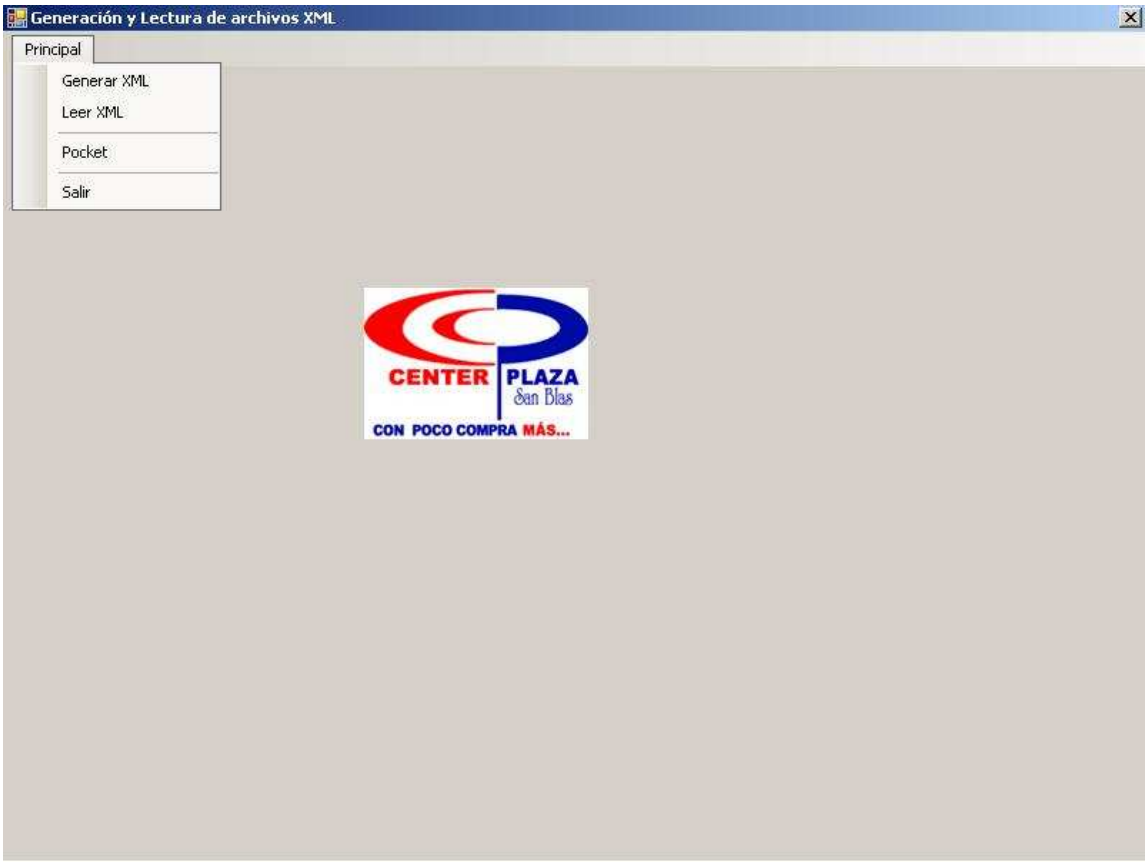
CAMPOS			
Name	Type	Null	Comment
PRODUCTOCOD	VARCHAR2(20)	N	Código de barras del producto
TIPOMOVIMIENTO	VARCHAR2(30)	N	Nombre del tipo de movimiento
INGRESOS	NUMBER(15,4)	N	Total de movimiento de ingresos
EGRESOS	NUMBER(15,4)	N	Total de movimiento de egreso

La tabla TMOVIMIENTO contiene un **resumen** de los movimientos del año en curso del producto, agrupados por tipo de comprobante. Se totalizan las cantidades registradas en los movimientos de un producto, agrupadas por tipo de comprobante, cada tipo de comprobante contiene la suma de ingresos y egresos del producto.

5.3 Diseño de interfaces de la aplicación

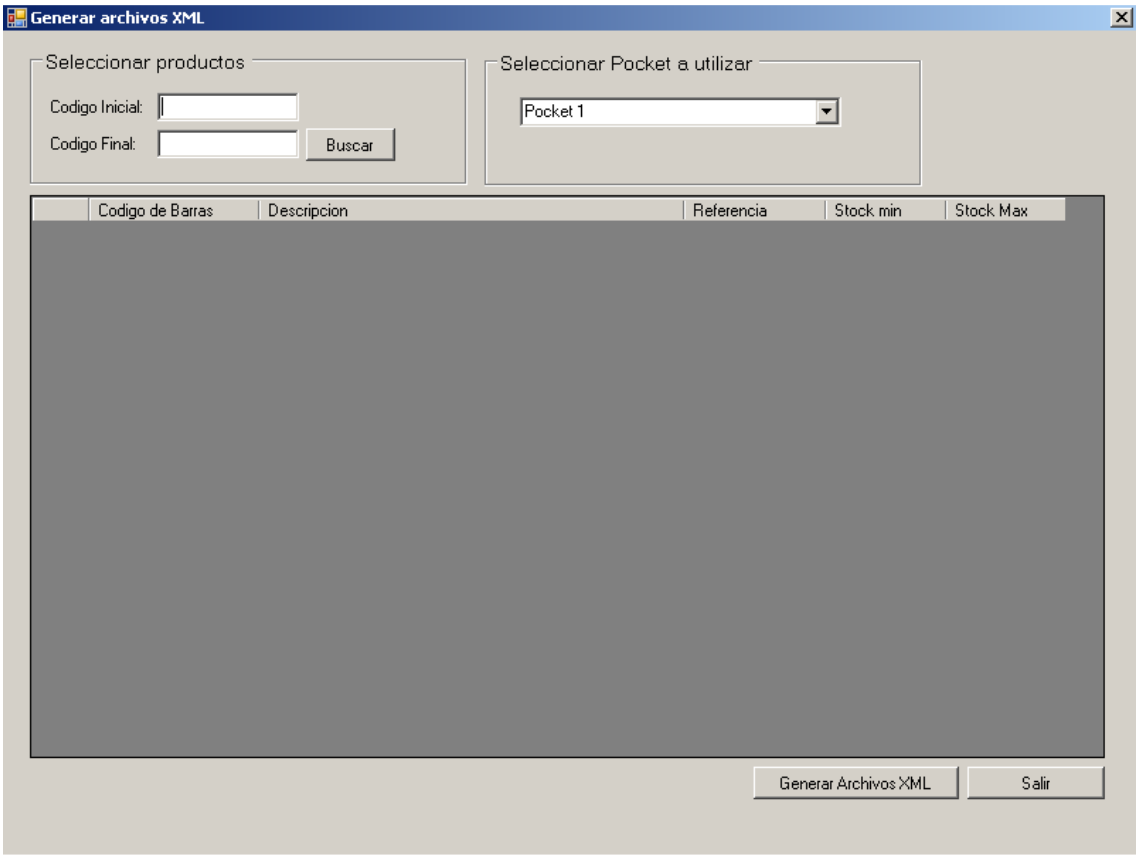
5.3.1 Interfaz de generación y lectura de archivos XML

5.3.1.1 Pantalla principal



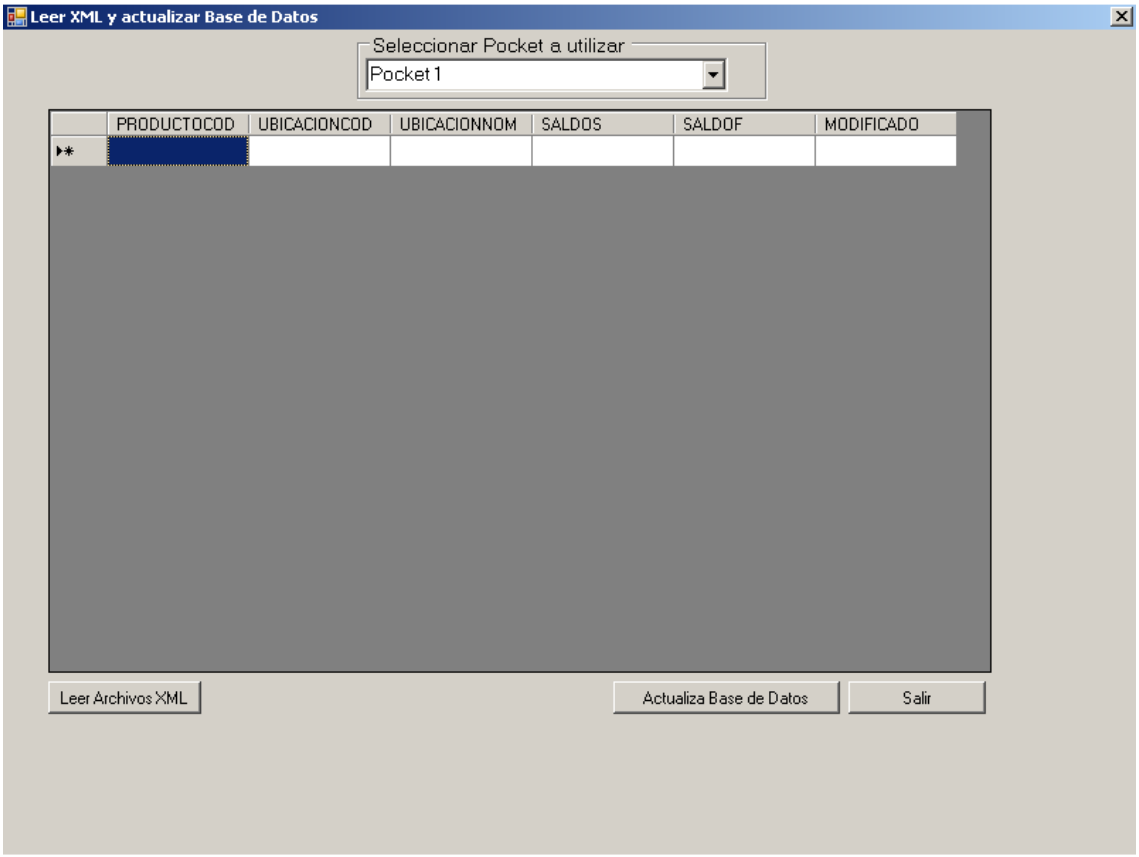
Esta pantalla contiene el menú principal de la aplicación: Generar XML, Leer XML, Pocket y Salir.

5.3.1.2 Pantalla de generación de archivos XML



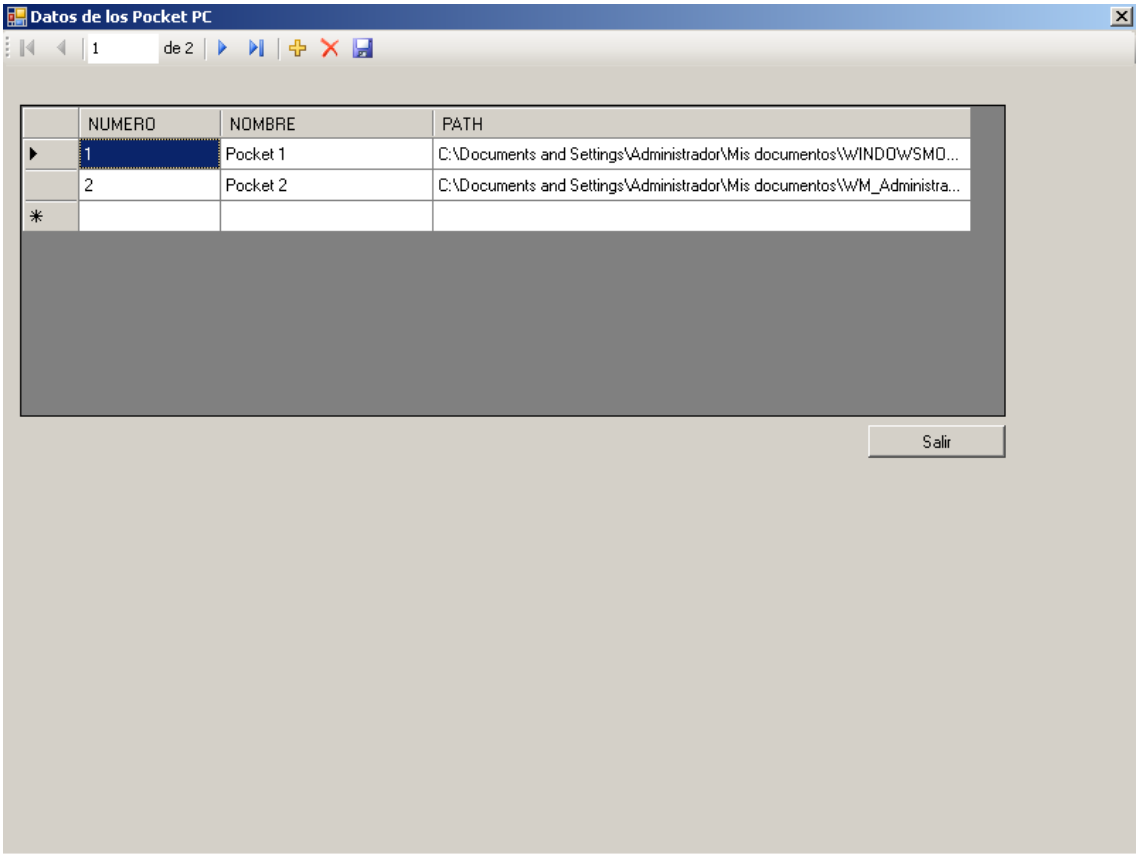
En esta pantalla el usuario selecciona el rango de códigos a consultar ej.: código inicial 7LNA y código final 7LNAZ y luego genera los archivos XML que serán actualizados en el Pocket PC.

5.3.1.3 Pantalla de Lectura de XML y actualización de Base de Datos



En esta pantalla el usuario lee los archivos XML y luego de visualizar, actualiza la base de datos.

5.3.1.4 Pantalla de administración de Pocket PC



En esta pantalla se ingresan los registros que contienen la ubicación del directorio específico para cada Pocket PC.

5.3.2 Interfaz de la aplicación, fuera de línea

5.3.2.1 Pantalla inicial

Es la pantalla de inicio de la aplicación en el Pocket PC, aquí tenemos el menú principal con las diferentes opciones: Leer XML, Productos, Grabar XML, Salir.

5.3.2.2 Pantalla de lectura de archivos XML

Esta pantalla nos muestra una notificación luego de cargar los archivos XML a las tablas de la aplicación, nos muestra un listado de cada una de las tablas: PRODUCTOS, SALDOS, MOVIMIENTOS.

5.3.2.3 Pantalla de consulta de producto

En esta pantalla el usuario ingresa el código del producto, ya sea digitando o mediante el lector de código de barras, luego la aplicación nos visualiza en diferentes pestañas las características del producto, el resumen de movimientos y los saldos del producto en las diferentes ubicaciones.

5.3.2.4 Pantalla de Resumen de Movimientos

En esta pantalla tenemos un resumen de movimientos del producto en el presente año, nos indica el saldo inicial, y la suma de ingresos y egresos realizados en el sistema, es una pantalla de información y no permite modificar ni ingresar datos.

5.3.2.5 Pantalla de Registro de Saldo Físico

En la pantalla de registro de saldos, el usuario selecciona la ubicación en la cual va a registrar el saldo y luego ingresa el saldo físico, con el botón Grabar se registran los cambios en las tablas de la aplicación.

5.3.3 Interfaz de la aplicación, en línea

5.3.3.1 Pantalla Inicial

Esta pantalla es la inicial de la aplicación, nos visualiza el logo de la empresa y el título de la aplicación, con el botón continuar ingresamos al sistema.

5.3.3.2 Pantalla de Datos

Se ingresa el código del producto ya sea digitando o con el lector del código de barras del Pocket PC y el sistema nos visualiza las características del producto.

5.3.3.3 Pantalla de Movimientos

En esta pantalla tenemos el resumen de movimientos del producto, el saldo inicial, un resumen de ingresos y egresos de acuerdo al tipo de movimiento y el saldo final a la fecha actual.

5.3.3.4 Pantalla de Registro de Saldo Físico

En esta pantalla visualizamos los saldos actuales del producto en cada una de las ubicaciones, el usuario selecciona de la lista la ubicación en la cual se va a registrar el saldo, ingresa el saldo físico y procede a grabar.

5.4 Esquemas de conexión

5.4.1 Modelo en línea

La aplicación en modo en línea es una aplicación a 3 capas.

Todos los componentes generados son componentes .Net (Assemblies), que siguen el estándar de "Factory Design Pattern".

La forma de comunicación entre los componentes es usando .Net remoting, esto implica que para el transporte de la información se crean mensajes que viajan bajo HTTP o TCP (.Net Channel Services).

En el servidor de aplicaciones (Internet Information Server) se publica un único objeto, que es compartido por todos los clientes. Este permite al cliente, acceder al objeto remoto y activarlo. El servidor de aplicaciones se encarga de mantenerlos vivos mientras exista interacción frecuente entre el objeto y el cliente.

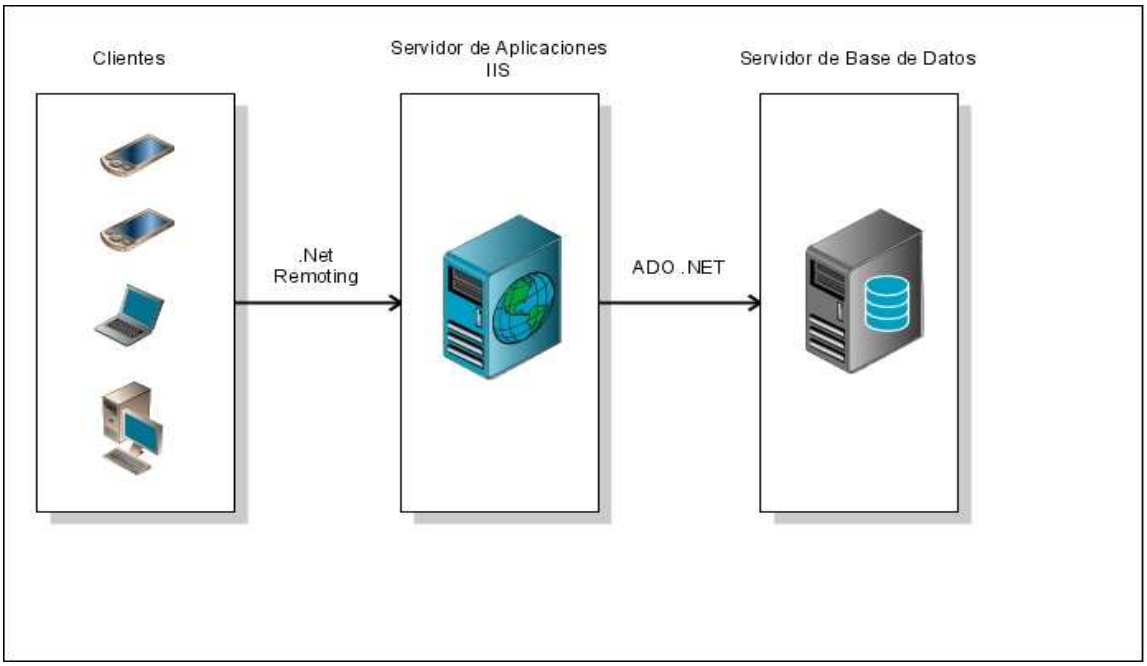


Ilustración 9. Esquema de conexión, Modelo en línea

5.4.2 Modelo fuera de línea

La aplicación en el dispositivo móvil es una aplicación de 2 capas, contiene un DataSet que se carga con los datos de los archivos XML y la aplicación que manipula esos datos.

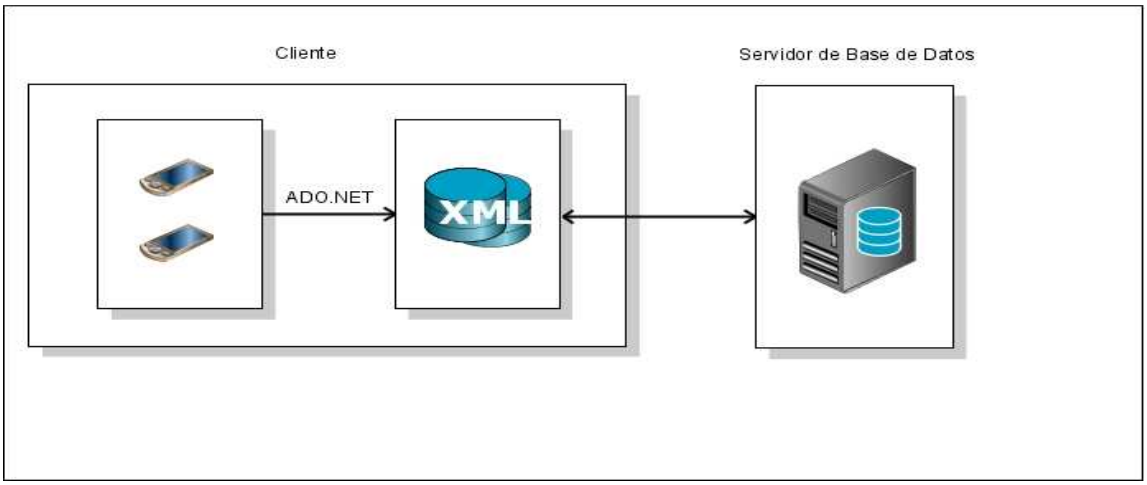


Ilustración 10. Esquema de conexión. Modelo fuera de línea

5.5 Esquema general de la conexión con Wi-Fi

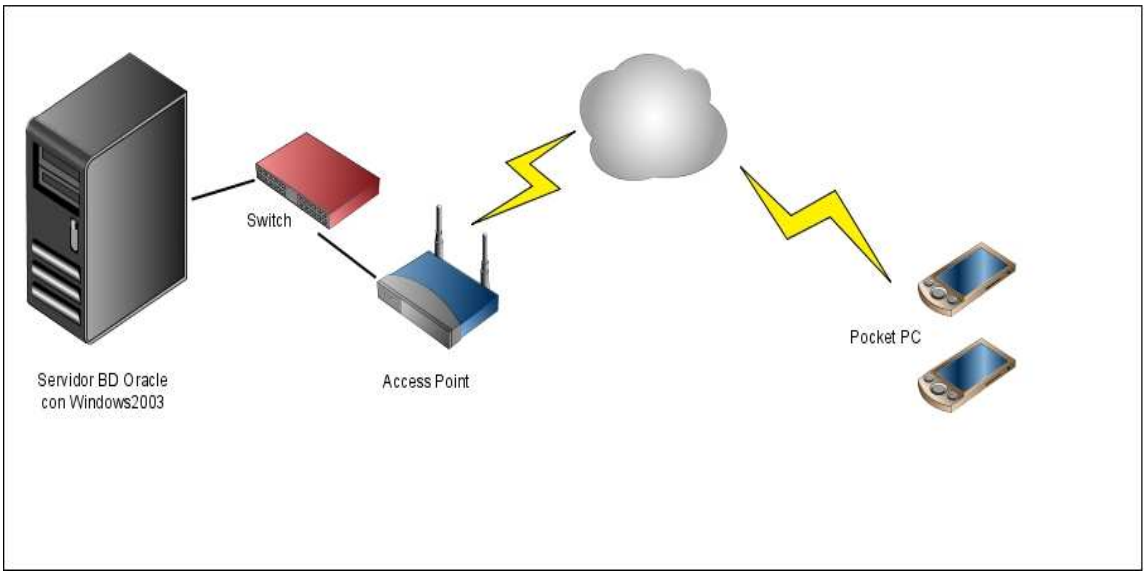


Ilustración 11. Esquema general de la conexión con Wi-Fi

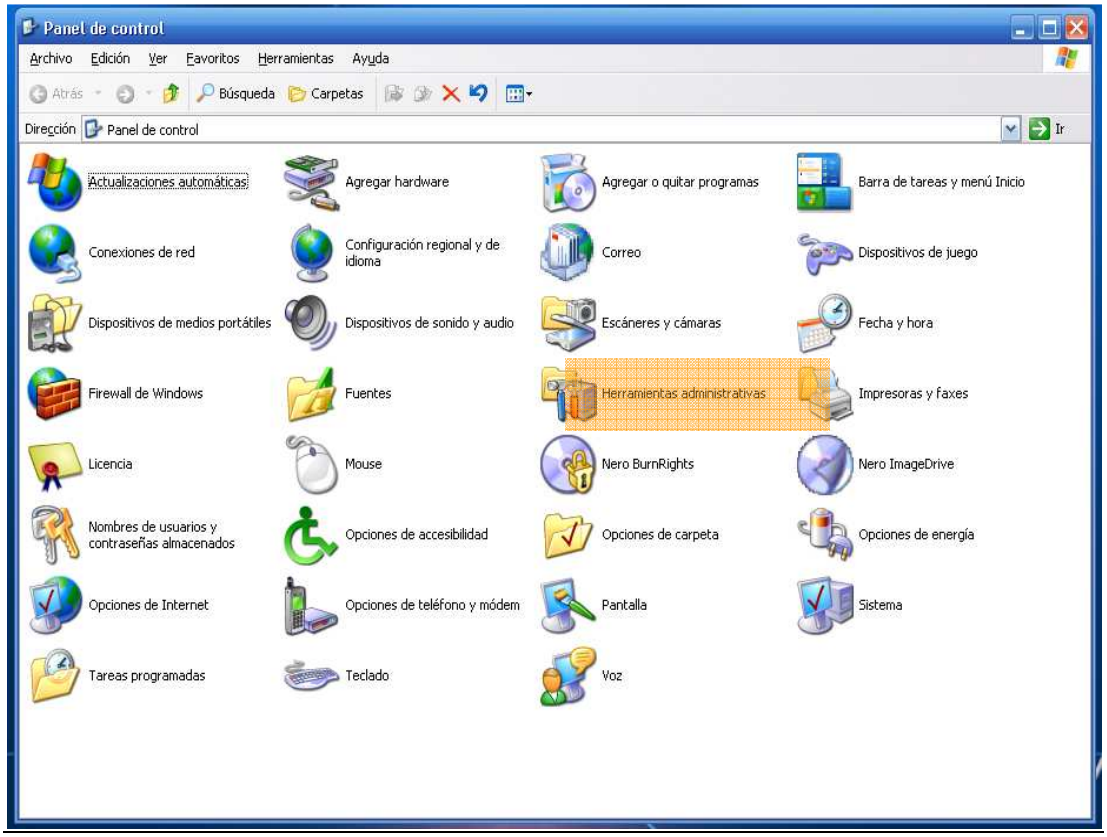
5.6 Configuración del Internet Information Server (IIS)

Internet Information Server , IIS, es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS.

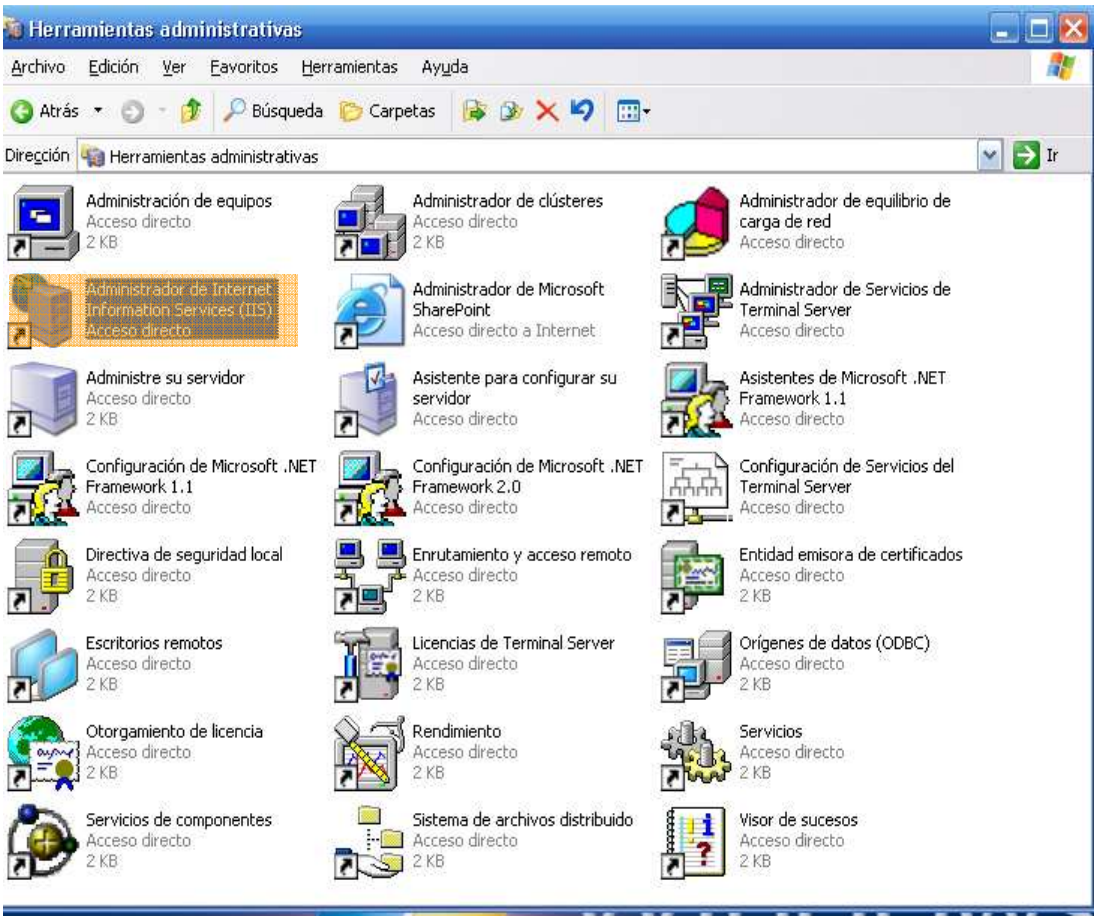
Este servicio convierte a un ordenador en un servidor de Internet o Intranet es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente (servidor web).

El servidor web se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas, por ejemplo Microsoft incluye los de Active Server Pages (ASP) y ASP.NET. También pueden ser incluidos los de otros fabricantes, como PHP o Perl.

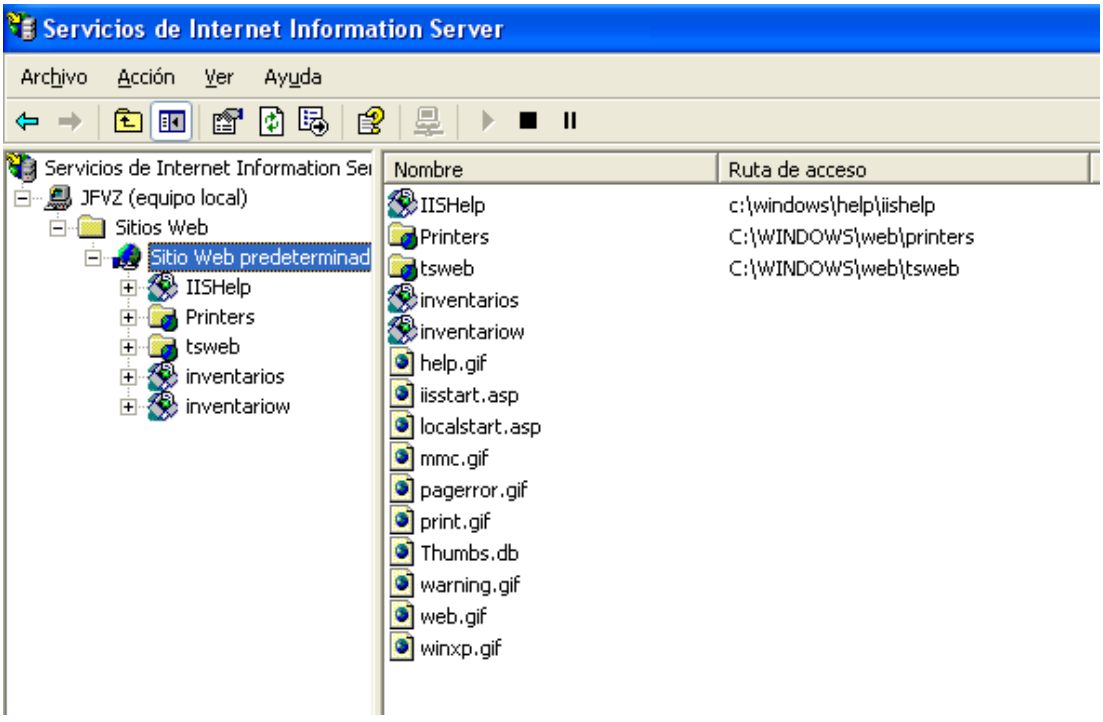
Como primer paso tenemos que entrar al panel de control y damos doble Click en Herramientas Administrativas para abrir la siguiente ventana.



Ya una vez que estamos en la ventana de Herramientas Administrativas seleccionamos y damos doble click en el Icono de Administrador de Internet Information Server (IIS)



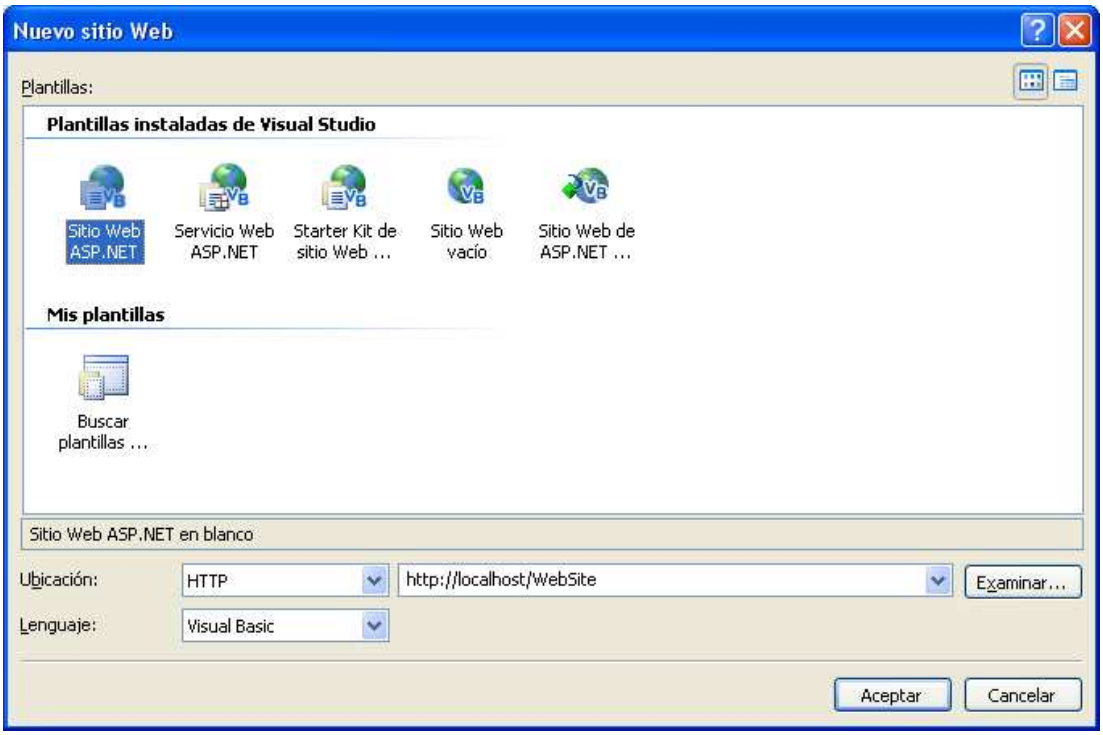
En este punto tenemos la estructura del IIS, el Sitio Web predeterminado, dentro del cual se irán generando nuestras páginas web que desarrollamos en Visual Studio.



Luego de generar los sitios web en Visual Studio como veremos a continuación, vamos a configurar en el IIS una serie de opciones necesarias para poder ejecutar nuestras páginas desde el explorador de internet.

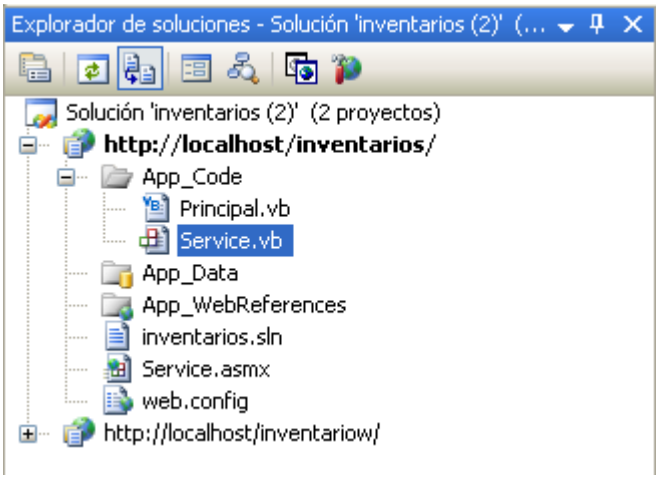
5.7 Creación de los servicios Web en Visual Basic .Net

En el entorno de Visual Studio ingresamos a la opción Archivo -> Nuevo Sitio Web y seleccionamos Sitio Web ASP .Net, en ubicación seleccionamos HTTP y como dirección **http://localhost/inventarios** y en lenguaje seleccionamos Visual Basic .



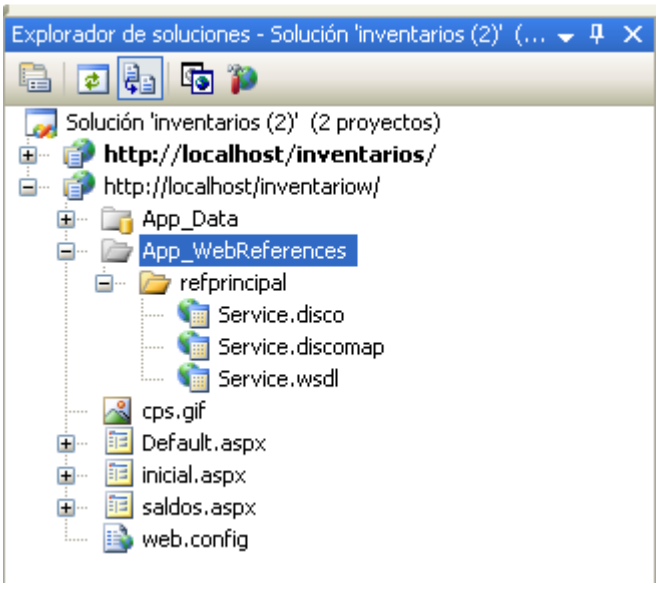
El sitio que generamos va a ser el que contiene los Servicios Web, ahora generamos un nuevo sitio web en la ubicación **http://localhost/inventariow** el cual va a contener la aplicación que va a consumir los servicios web creados anteriormente.

Los servicios web generado es Service.vb, este contiene los métodos web necesarios para la aplicación, y tenemos la clase Principal.vb en donde están los procedimientos y funciones que se utilizan en Service.vb. La estructura del servicio web es el siguiente:



En el segundo sitio web desarrollamos las interfaces de la aplicación, en este sitio web agregamos una referencia web, la referencia web es la que creamos en el primer sitio web, es decir nuestro servicio web. El nombre de nuestra referencia web es refprincipal.

La estructura del segundo sitio web es la siguiente:

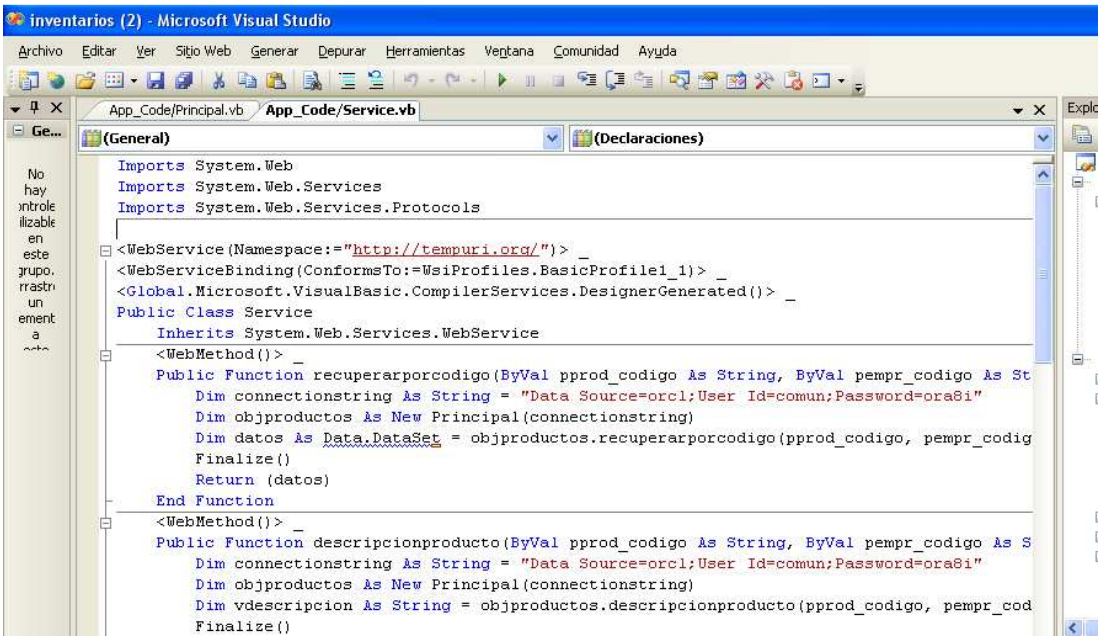


Una vez generados los sitios web, regresamos a la pantalla de configuración del IIS para configurar la página por defecto y además la configuración de acceso a las páginas por direcciones IP, se selecciono esta forma de restringir el acceso para que puedan ingresar a nuestras páginas únicamente los equipos autorizados, cada Pocket PC tiene su dirección IP, entonces podrán ingresar únicamente dichos equipos.

Cuando tenemos configurados los Servicios Web en Visual Studio y el IIS podemos ingresar desde el explorador de internet con la siguiente dirección: **http://localhost/inventariow**.

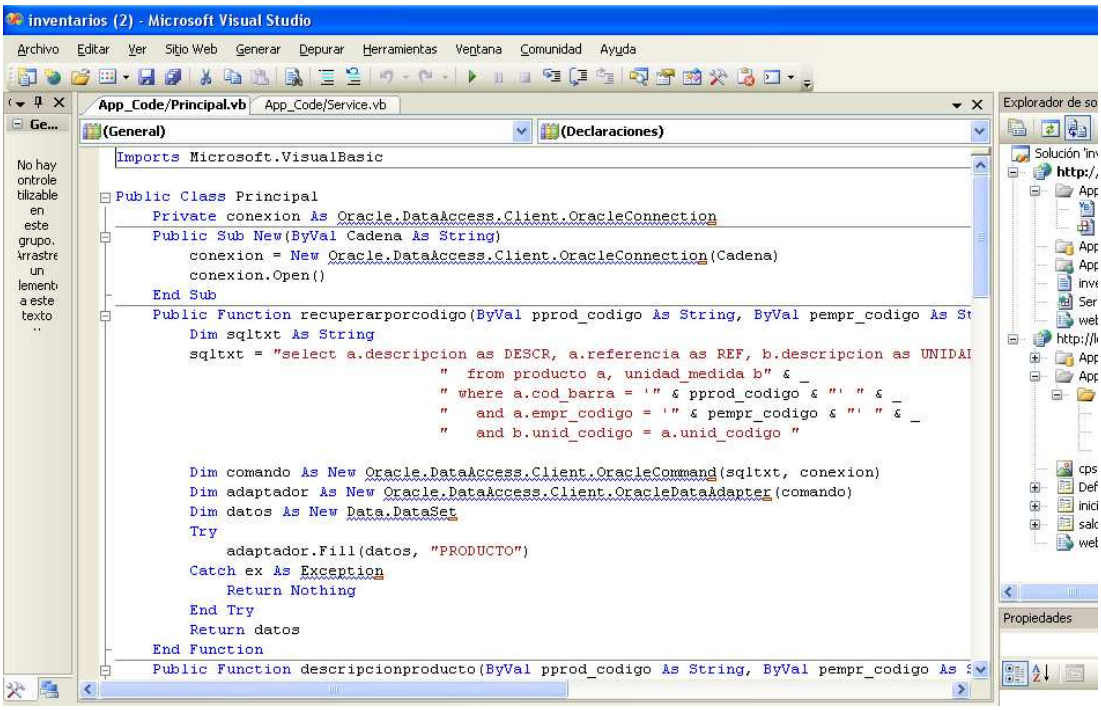
5.8 El Servicio Web: Service.vb

Este es el servicio web que utilizaremos en nuestras páginas asp, contiene los métodos necesarios para acceder a la base de datos (ver código en Anexo 1).



5.9 La Clase Principal.vb

Esta clase contiene los métodos, procedimientos y funciones que se utilizan para interactuar con la base de datos y son utilizados por Service.vb (ver código en Anexo 1).



En esta clase en los diferentes procesos y funciones se utilizaron varias de las funcionalidades que presenta Visual Basic .Net para interactuar con la base de datos Oracle, por ejemplo se utilizó el método de escribir la sentencia SQL que se va a utilizar para recuperar los datos, otro método utilizado fue el de ejecutar procedimientos almacenados de la base de datos, declarando los parámetros y el nombre del procedimiento.

5.10 Actualización en la base de datos.

Los datos ya sea en el modo en línea como en el fuera de línea utilizan un procedimiento almacenado de la base de datos, en el primer caso desde la aplicación de actualización de archivos XML, en el segundo es llamado directamente desde el servicio web.

Este procedimiento verifica la diferencia entre los saldos del sistema y el físico y registra los movimientos de ajuste (Regulaciones) ya sea de ingreso o egreso para consolidar los saldos (ver código en Anexo 1).

Para el modo fuera de línea la lectura de los datos de la Base de Datos para generar los archivos XML se utilizan procedimientos almacenados de la base de datos. Esos

llenan las tablas definidas en el modelo entidad relación y estos a su vez son los orígenes de datos para los archivos XML (ver código en Anexo 2).

CAPITULO 6

CONCLUSIONES Y RECOMENDACIONES

Luego del desarrollo de la aplicación la empresa cuenta con una herramienta útil para sus procesos de auditoría de inventarios, ya que se los puede realizar de forma más rápida y eficiente.

Para desarrollar la aplicación se aprovecharon las facilidades que ofrece el Visual Studio .Net en lo que se refiere a la programación con archivos XML y acceso a base de datos. Estas facilidades permiten interactuar con diferentes tipos de orígenes de datos, dando así una flexibilidad en el desarrollo y evitando una dependencia a un solo proveedor de datos. En el presente trabajo se desarrollo sobre una base de datos Oracle y no hubo ningún inconveniente en trabajar con Visual Basic .Net.

Observando los resultados obtenidos con esta aplicación, se recomienda poner en marcha proyectos de desarrollo de nuevas aplicaciones para los dispositivos móviles que dispone la empresa para automatizar algunos procesos que actualmente son realizados manualmente.

Uno de los procesos que se puede automatizar es el de registros de cobros a clientes, actualmente el personal lleva un registro manual de los cobros realizados en las visitas a los clientes, entonces sería de enorme ayuda una aplicación para registrar los cobros y su posterior conciliación con el sistema general de la empresa.

Otra aplicación sería el registro de pedidos en las ferias, la empresa participa en ferias principalmente con sus productos de bazar y juguetería, en dichas ferias se registran los pedidos manualmente, entonces se desarrollaría una aplicación para registrar los pedidos en los dispositivos móviles y luego registrar dichos pedidos en el sistema general de la empresa.

Bibliografía

Referencias Bibliográficas

- Visual Basic .Net: Windows Forms, FACI SANTIAGO, http://santi.rastafurbi.org/dotnet/vbnet/VBNET_forms.pdf
- Curso de Introducción a .Net con Visual 2005, GUILLERMO SOM, SERRANO JORGE
- Programación en Visual .Net, BLANCO LUIS MIGUEL, grupo EIDOS.

Referencias electrónicas

http://www.programatium.com/vbnet.htm#vbNet

http://www.programatium.com/vbnet.htm#exe

http://es.wikipedia.org/wiki/VB.NET

http://www.elguruprogramador.com.ar/articulos/que-es-un-webform-net.htm

http://www.desarrolloweb.com/articulos/477.php

http://es.wikipedia.org/wiki/XML

http://www.w3c.es/Consorcio/

ANEXOS

Anexo 1. Código fuente de servicios web y clase del sistema en línea

Servicio Web: Service.vb

```
Imports System.Web
Imports System.Web.Services
Imports System.Web.Services.Protocols

<WebService(Namespace:="http://tempuri.org/")> _
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Public Class Service
    Inherits System.Web.Services.WebService
    <WebMethod()> _
    Public Function recuperarporcodigo(ByVal pprod_codigo As String, ByVal
pempr_codigo As String) As Data.DataSet
        Dim connectionstring As String = "Data Source=orcl;User
Id=comun;Password=ora8i"
        Dim objproductos As New Principal(connectionstring)
        Dim datos As Data.DataSet = objproductos.recuperarporcodigo(pprod_codigo,
pempr_codigo)
        Finalize()
        Return (datos)
    End Function
    <WebMethod()> _
    Public Function descripcionproducto(ByVal pprod_codigo As String, ByVal
pempr_codigo As String) As String
        Dim connectionstring As String = "Data Source=orcl;User
Id=comun;Password=ora8i"
        Dim objproductos As New Principal(connectionstring)
        Dim vdescripcion As String = objproductos.descripcionproducto(pprod_codigo,
pempr_codigo)
        Finalize()
        Return (vdescripcion)
    End Function

    <WebMethod()> _
    Public Function QuerySaldos(ByVal pprod_codigo As String, ByVal pempr_codigo As
String) As Data.DataSet
        Dim connectionstring As String = "Data Source=orcl;User
Id=comun;Password=ora8i"
        Dim objproductos As New Principal(connectionstring)
        Dim datos As Data.DataSet = objproductos.QuerySaldos(pprod_codigo,
pempr_codigo)
        Finalize()
        Return (datos)
    End Function
    <WebMethod()> _
    Public Sub registramovimiento(ByVal pprod_codigo As String, ByVal pbod_codigo As
String, ByVal pempr_codigo As String, ByVal psaldo As String)
        Dim connectionstring As String = "Data Source=orcl;User
Id=comun;Password=ora8i"
        Dim objproductos As New Principal(connectionstring)
        objproductos.registramovimiento(pprod_codigo, pbod_codigo, pempr_codigo,
psaldo)
        Finalize()
    End Sub
    <WebMethod()> _
    Protected Overrides Sub Finalize()
        MyBase.Finalize()
    End Sub

End Class
```

Clase Principal.vb

```
Imports Microsoft.VisualBasic

Public Class Principal
    Private conexion As Oracle.DataAccess.Client.OracleConnection
    Public Sub New(ByVal Cadena As String)
        conexion = New Oracle.DataAccess.Client.OracleConnection(Cadena)
        conexion.Open()
    End Sub
    Public Function recuperarporcodigo(ByVal pprod_codigo As String, ByVal
pempr_codigo As String) As Data.DataSet
        Dim sqltxt As String
        sqltxt = "select a.descripcion as DESCR, a.referencia as REF, b.descripcion
as UNIDAD, a.stock_minimo as STOCKmin, a.stock_maximo as STOCKmax " & _
                                    "  from producto a, unidad_medida b" & _
                                    " where a.cod_barra = '" & pprod_codigo & "' " & _
                                    "   and a.empr_codigo = '" & pempr_codigo & "' " &
_
                                    "   and b.unid_codigo = a.unid_codigo "

        Dim comando As New Oracle.DataAccess.Client.OracleCommand(sqltxt, conexion)
        Dim adaptador As New Oracle.DataAccess.Client.OracleDataAdapter(comando)
        Dim datos As New Data.DataSet
        Try
            adaptador.Fill(datos, "PRODUCTO")
        Catch ex As Exception
            Return Nothing
        End Try
        Return datos
    End Function
    Public Function descripcionproducto(ByVal pprod_codigo As String, ByVal
pempr_codigo As String) As String
        Dim sqltxt As String
        sqltxt = "select a.descripcion as DESCR " & _
                 "  from producto a, unidad_medida b" & _
                 " where a.cod_barra = '" & pprod_codigo & "' " & _
                 "   and a.empr_codigo = '" & pempr_codigo & "' " & _
                 "   and b.unid_codigo = a.unid_codigo "
        Dim comando As New Oracle.DataAccess.Client.OracleCommand(sqltxt, conexion)
        Dim adaptador As New Oracle.DataAccess.Client.OracleDataAdapter(comando)
        Dim datos As New Data.DataSet
        Try
            adaptador.Fill(datos, "PRODUCTO")
        Catch ex As Exception
            Return Nothing
        End Try
        Return datos.Tables("PRODUCTO").Rows(0).Item("DESCR")
    End Function

    Public Function QuerySaldos(ByVal pprod_codigo As String, ByVal pempr_codigo As
String) As Data.DataSet
        Dim comando As New Oracle.DataAccess.Client.OracleCommand()
        comando.Parameters.Add("pprod_codigo", pprod_codigo)
        comando.Parameters.Add("pempr_codigo", pempr_codigo)
        comando.Parameters.Add("resulset",
Oracle.DataAccess.Client.OracleDbType.RefCursor, Data.ParameterDirection.InputOutput)
        comando.CommandText = "pkg_producto.queryxsaldo"
        comando.CommandType = Data.CommandType.StoredProcedure
        comando.Connection = conexion
        Dim da As New Oracle.DataAccess.Client.OracleDataAdapter(comando)
        Dim datos As New Data.DataSet
        da.Fill(datos, "SALDOS")
        Return datos
    End Function
    Public Sub registramovimiento(ByVal pprod_codigo As String, ByVal pbod_codigo As
String, ByVal pempr_codigo As String, ByVal psaldo As String)
        Dim comando As New Oracle.DataAccess.Client.OracleCommand()
        comando.Parameters.Add("pprod_codigo", pprod_codigo)
        comando.Parameters.Add("pbod_codigo", CInt(pbod_codigo))
        comando.Parameters.Add("pempr_codigo", pempr_codigo)
        comando.Parameters.Add("psaldo", CInt(psaldo))
        comando.CommandText = "pkg_producto.registramovimiento"
        comando.CommandType = Data.CommandType.StoredProcedure
        comando.Connection = conexion
```

```
        Dim da As New Oracle.DataAccess.Client.OracleDataAdapter(comando)
        Dim ds As New Data.DataSet()
        da.Fill(ds)
        ds.AcceptChanges()
    End Sub
    Protected Overrides Sub Finalize()
        MyBase.Finalize()
        Try
            conexion.Close()
        Catch ex As Exception
        End Try
    End Sub
End Class
```

Procedimiento que actualiza la base de datos **pkg_producto.registramovimiento**

```
procedure registramovimiento(pprod_codigo varchar2,
                             pbod_codigo  number,
                             pempr_codigo varchar2,
                             psaldo       number) is
    vsaldo_actual number(15,4);
    vage_codigo   number := 0;
    vproducto_rec producto%rowtype;
    vcantidad     number(15,4) := 0;
    vdebito_credito number(1);
    vtipoc_codigo   tipo_comprobante.tipoc_codigo%type;
    vdiferencia     number(15,4);
    vcomp_numero    comprobante.comp_numero%type;
    vprod_codigo    producto.prod_codigo%type;
begin
    select *
      into vproducto_rec
      from producto
     where cod_barra = pprod_codigo
       and empr_codigo = pempr_codigo;
    vsaldo_actual :=
pkg_producto.obtener_saldo_producto(vproducto_rec.prod_codigo,1,pbod_codigo,
vage_codigo,vproducto_rec.unid_codigo,pempr_codigo,to_char(sysdate,'yyyy'),t
o_char(sysdate,'yyyy-mm-dd'),'N');
    vdiferencia := vsaldo_actual - psaldo;
    vcantidad := abs(vdiferencia);
    if vdiferencia <> 0 then
      if vdiferencia > 0 then
        vtipoc_codigo := 'RE';
        vdebito_credito := 2;
      else
        vtipoc_codigo := 'RI';
        vdebito_credito := 1;
      end if;
      vcomp_numero :=
secuencia_comprobantes(pempr_codigo,vtipoc_codigo,vage_codigo,pbod_codigo);
      insert into comprobante
       (comp_numero, tipoc_codigo, empr_codigo,
        subbod_codigo, bod_codigo, age_codigo,
        cod_ente, cod_tipo_ente, fecha,
        liq_numero, fpago_codigo, numero_cuotas,
        estado, comprobante_manual, usuario,comp_numero_ref,
        tipoc_codigo_ref,empr_codigo_ref,cod_ente_ven,
        cod_tipo_ente_ven, cod_ente_tran,cod_tipo_ente_tran,
        dui_numero, subtotal_sin_iva, subtotal_con_iva,
        descuento1_porce, descuento1_valor, descuento2_porce,
        descuento2_valor, iva_porce, iva_valor, flete, gastos,
        seguro, otros_impuestos_porce, otros_impuestos_valor,
        otros_impuestos_razon, financiamiento, contado,
        credito, otros1_valor, otros1_porce, otros1_razon,
        otros2_valor, otros2_porce, otros2_razon, saldo,
        total_factura, fecha_generacion, valor_retencion,
```

```
     agente_nro, precios_con_iva, comprobante_con_iva, tipoc_codigo_ven,
     ruta_numero, age_serie, caja_serie,
      numero_comprobante, caja_numero)
   values
     (vcomp_numero, vtipoc_codigo, pempr_codigo,
      1, pbod_codigo, vage_codigo,
      'AJUSTE','PRO',sysdate,
      null,'EFE',0,
      2,'POCKET',USER,
      NULL, NULL, NULL, NULL, NULL, NULL,
      NULL, NULL, 0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
      0,0,0,0,0,0,0,0,0, sysdate, null,null,
      'S', null, null, null, null,null,
      null,null);
    insert into movimiento
      (comp_numero, tipoc_codigo, empr_codigo,
       secuencia, prod_codigo, debito_credito,
       cantidad, costo, precio,
       iva_valor, iva_porce, total_descuento,
       total_movimiento, age_codigo, bod_codigo,
       subbod_codigo, unid_codigo, descuento_comprobante,
       otros_comprobante, precio_ref, usua_codigo,
       fecha, cantidad_devuelta, cantidad_pedida,
       porce_arancel, estado_codigo)
    values(vcomp_numero, vtipoc_codigo, pempr_codigo,
        1, vproducto_rec.prod_codigo, vdebito_credito,
        vcantidad, 0,0,
        0,0,0,
        0, vage_codigo, pbod_codigo,
        1, vproducto_rec.unid_codigo, 0,
        0, 0, lower(user),
        sysdate, 0, 0,
        0, 2);
   commit;
  end if;
end;
```

Anexo 2. Procedimientos para generar y leer los archivos XML

En la pantalla de generación de archivos XML cuando el usuario ingresa el rango de códigos, inicial y final, se llama al procedimiento almacenado pkg_producto.generatablas, este procedimiento llena las tablas TPRODUCTO, TSALDO, TMOVIMIENTO y TUBICACION.

```
procedure generatablas(pprod_codigo_inicial in varchar2, pprod_codigo_final
in varchar2) is
    vsaldo_actual NUMBER(15,4);
    vproducto_rec producto%rowtype;

begin
    delete from tmovimiento;
    delete from tsaldo;
    delete from tproducto;
    delete from tubicacion;
    COMMIT;
    insert into tubicacion
    select bod_codigo,  descripcion
      from bodega
     where empr_codigo = vempr_codigo
       and age_codigo = 0
       and orden < 10;
    insert into tproducto
    select a.cod_barra, a.descripcion, a.referencia, b.descripcion,
a.stock_minimo, a.stock_maximo
      from producto a, unidad_medida b
     where a.empr_codigo = vvempr_codigo
       and length(a.prod_codigo) = 11
       and b.unid_codigo = a.unid_codigo
       and a.prod_codigo >= pprod_codigo_inicial
       and a.prod_codigo <= pprod_codigo_final;
    commit;
    insert into tsaldo
    select productocod, bod_codigo, descripcion, saldos, 0 saldof, 0
modificado
      from (select c.productocod,
                   b.bod_codigo,
                   d.descripcion,
                   b.cant_inicial +
                   (ing_cantidad_01 + ing_cantidad_02 + ing_cantidad_03 +
                   ing_cantidad_04 + ing_cantidad_05 + ing_cantidad_06 +
                   ing_cantidad_07 + ing_cantidad_08 + ing_cantidad_09 +
                   ing_cantidad_10 + ing_cantidad_11 + ing_cantidad_12 -
                   egr_cantidad_01 - egr_cantidad_02 - egr_cantidad_03 -
                   egr_cantidad_04 - egr_cantidad_05 - egr_cantidad_06 -
                   egr_cantidad_07 - egr_cantidad_08 - egr_cantidad_09 -
                   egr_cantidad_10 - egr_cantidad_11 - egr_cantidad_12)
saldos
              from tproducto c, producto a, saldo_producto b, bodega d
             where a.empr_codigo = vvempr_codigo
               and c.productocod = a.cod_barra
               and a.prod_codigo = b.prod_codigo
               and a.empr_codigo = b.empr_codigo
               and b.unid_codigo = a.unid_codigo
               and b.prod_bod_aaaa = to_char(sysdate, 'yyyy')
               and b.age_codigo = 0
               and d.bod_codigo = b.bod_codigo
               and d.age_codigo = b.age_codigo
               and d.orden < 10);
    commit;
    for c1_rec in (select * from tproducto) loop
```

```
      select *
        into vproducto_rec
        from producto
       where cod_barra = c1_rec.productocod
         and empr_codigo = vempr_codigo;
      vsaldo_actual :=
pkg_producto.obtener_saldo_producto(vproducto_rec.prod_codigo,1,null,0,vprod
ucto_rec.unid_codigo,vempr_codigo,to_char(sysdate,'yyyy'),to_char(trunc(sysd
ate,'yyyy')-1),'N');
      insert into tmovimiento values (c1_rec.productocod, '
S.Inicial',vsaldo_actual,null);
      insert into tmovimiento
      select productocod, nombre_corto , sum(ingresos) , sum(egresos)
      from (
      select e.productocod, b.nombre_corto,
decode(a.debito_credito,1,a.cantidad,0) ingresos,
      decode(a.debito_credito,2,a.cantidad,0) egresos
        from movimiento a, tipo_comprobante b, producto d, tproducto e
       where e.productocod = c1_rec.productocod
         and e.productocod = d.cod_barra
         and a.empr_codigo = vempr_codigo
         and a.fecha >= trunc(sysdate,'yyyy')
         and d.prod_codigo = a.prod_codigo
         and d.empr_codigo = a.empr_codigo
         and b.tipoc_codigo = a.tipoc_codigo
         and b.empr_codigo = a.empr_codigo
         and b.afecta_inventario = 'S'
         and not exists (Select 'x' from comprobante c
                where c.comp_numero = a.comp_numero
         and c.tipoc_codigo = a.tipoc_codigo
         and c.empr_codigo = a.empr_codigo
         and c.estado = 22))
      group by productocod, nombre_corto;
    end loop;
    commit;
  end;
```

Con estas tablas se llenan los DataGrid desde los cuales se generan los archivos XML.

En la pantalla de lectura de archivos XML y actualización de base de datos, se llenan las tablas con el DataGrid que lee los archivos XML y ejecuta el procedimiento **pkg_producto.registramovimiento** en la base de datos.