UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE SEGURIDAD REMOTO PARA EL LABORATORIO MICROSOFT RESEARCH.

TESIS

QUE PARA OBTENER EL TÍTULO DE:

ÍNGENIERO EN COMPUTACIÓN

PRESENTA:

FRANK JONATHAN TEHUACANERO CASTRO

DIRECTOR DE TESIS:

M. EN I. JORGE VALERIANO ASSEM

MÉXICO, D.F. **2006**

ÍNDICE

INTRODUCCIÓN

En la actualidad desafortunadamente la necesidad de contar con sistemas de seguridad es cada día mayor. Debido al valor monetario del equipo que se encuentra dentro del Laboratorio Microsoft Research, es de suma importancia contar con un sistema de seguridad que ayude a evitar el robo de dichos bienes. Al momento de iniciar esta tesis, no existe ningún sistema de seguridad instalado en el Laboratorio Microsoft Research. La única forma de protección con la cual se dispone para evitar el robo del equipo es mediante cerraduras mecánicas en las puertas, las cuales pueden ser fácilmente violadas. Además, no se puede saber quién está dentro del laboratorio, para que en caso de detectar algún intruso, alertar a las autoridades pertinentes.

Debido a la inseguridad que se vive en el país, es necesario desarrollar cada vez más y mejores sistemas de seguridad. Se desea que una vez instalado el Sistema de Seguridad Remoto para el Laboratorio Microsoft Research, permita ser modificado en el futuro, permitiendo así agregarle mayor funcionalidad y poder personalizarlo según sea necesario, con lo cual podría ser instalado también en otras instalaciones si así se deseara.

Al término de esta tesis, se dispondrá de un sistema de seguridad para el Laboratorio Microsoft Research de la Facultad de Ingeniería, el cual mediante una página ASP.NET permitirá controlar de manera remota, a través de Internet, el acceso a este laboratorio. Se pretende que mediante dicha página ASP.NET alojada en una computadora del Laboratorio Microsoft Research, se pueda tener una vista en tiempo real del laboratorio a través de una webcam, así como un sistema de detección de intrusos en los periodos de inactividad del laboratorio. También, dentro de esta página ASP.NET se tendrá la opción de activar mediante el puerto paralelo de la computadora que alojará dicha página, dos solenoides, uno instalado en la puerta de entrada al laboratorio, y otro en la puerta del cubículo donde se encuentra el servidor del laboratorio, con el fin de impedir que dichas puertas sean abiertas. Sólo podrán acceder a esta página los administradores del laboratorio, mediante un Login y Password. Estas credenciales utilizarán un algoritmo de encriptación para evitar que sean robadas al ser transmitidas a través de Internet, y se creará un registro de quién y a qué hora ingresó a la página.

CAPITULO I. MARCO TEÓRICO

I.1.- Plataforma .NET

.NET es la estrategia de Microsoft para servicios Web, con el propósito de conectar información, gente, sistemas y dispositivos a través de software. Integrada dentro de la plataforma Microsoft, la tecnología .NET provee la habilidad de construir, entregar, administrar y utilizar soluciones de seguridad mejorada, con servicios Web.

Las soluciones .NET permiten integrar sistemas más rápidamente y de una manera más ágil, y permite el acceso a la información en cualquier momento, en cualquier lugar, y a través de cualquier dispositivo.

Fig. 1.1 Logo de la tecnología Microsoft .NET. [1]

¿Qué son los servicios Web?

Si se le pregunta a un desarrollador que son los servicios Web, responderá: “módulos de software auto descriptivos, encapsulando semánticamente funcionalidad discreta, envueltos y accesibles vía protocolos estándar de comunicación por Internet, como XML o SOAP”. Pero si se le pregunta a un empresario que ha implementado soluciones basadas en servicios Web, se obtiene una respuesta diferente. Dirá que los servicios Web son un enfoque que permite a negocios conectarse con clientes, compañeros y empleados. Estos servicios permiten ampliar los servicios existentes a nuevos clientes o usuarios. Los servicios Web permiten trabajar más eficientemente con compañeros y proveedores. Permiten abrir el acceso a la información de manera que pueda fluir hacia cualquier usuario que la necesite. Reducen el tiempo de desarrollo y costos de nuevos proyectos.

Beneficios de los Servicios Web.

Al permitir a las aplicaciones compartir información a través de diferentes plataformas de hardware y sistemas operativos, los servicios Web proveen diferentes beneficios, como:

- Abrir la puerta a nuevas oportunidades de negocio al volver más fácil el conectarse con socios.
- Entregar experiencias integradas, dramáticamente más personales a los usuarios a través de una nueva variedad de dispositivos inteligentes, incluyendo PC’s.

- Ahorro de tiempo y dinero al cortar tiempo de desarrollo.
- Incrementar los flujos de ingresos al permitir crear servicios Web, que estén disponibles a las demás personas.

Conectando aplicaciones a través de servicios Web:

Los servicios Web están revolucionando el cómo las aplicaciones se comunican con otras aplicaciones, o poniéndolo de otra manera, como las computadoras se comunican con otras computadoras, al proveer un formato universal de datos que permite que la información pueda ser fácilmente adaptada o transformada. Basados en XML, el lenguaje universal de intercambio de información a través de Internet, los servicios Web pueden comunicarse a través de diferentes plataformas y sistemas operativos, sin importar el lenguaje de programación en el cual las aplicaciones fueron desarrolladas.

Cada servicio Web es una unidad discreta de código que maneja un set limitado de tareas. De cualquier manera, a pesar de que los servicios Web son independientes unos de otros, pueden ligarse dentro de un grupo de colaboración que ejecute una tarea particular.

Los servicios Web utilizan protocolos estándares en la industria de IT.

Los servicios Web también permiten a los programadores escoger entre construir todas las piezas en sus aplicaciones, o utilizar servicios Web desarrollados por otros programadores. Esto significa que una compañía individual no necesita suministrar cada una de las piezas de una solución. La habilidad de exponer (anunciar y ofrecer) los servicios Web propios a otros puede crear nuevos ingresos personales o para la empresa. Los servicios Web son invocados por Internet a través de protocolos estándares en la industria de IT, como SOAP, XML, y Universal Description, Discovery and Integration (UDDI). Ellos son definidos por organizaciones estándares publicas como el World Wide Web Consortium (W3C).

SOAP es una tecnología de mensajería basada en XML, estandarizada por el W3C, el cual especifica todas las reglas necesarias para localizar servicios Web, integrándolos en aplicaciones, y permitiendo la comunicaciones entre ellos. UDDI es un registro público, sin ningún costo, donde alguien puede publicar e inquirir acerca de servicios Web. [1]

I.2. .NET Framework

Microsoft .NET Framework, o más comúnmente conocido simplemente como .NET Framework, es una plataforma de desarrollo de software creado por Microsoft. .NET Framework se encuentra actualmente disponible en la versión 2.0, la cual fue liberada en Noviembre de 2005 y es la sucesora de dos versiones previas: 1.0 y 1.1.

.NET es la tecnología de Microsoft que permite el desarrollo multilenguaje, y provee una amplia librería estándar. Otros enfoques competentes son el de lenguajes multiplataforma, como por ejemplo Perl, utilizando una multiplataforma en tiempo de ejecución como Java Virtual Machina (JVM), o compilar ANSI estándar en cada plataforma.

Descripción:

.NET Framework fue diseñado con varias intenciones:

- Interoperabilidad: Debido a que muchas librerías COM han sido previamente creadas, .NET Framework provee métodos que permiten la interoperabilidad entre nuevo código y librerías existentes.
- Common Language Runtime: Lenguajes de programación en .NET Framework son compilados en un lenguaje intermedio conocido como Common Intermediate Language (CLI); la implementación de Microsoft del CIL se conoce como Microsoft Intermediate Language, o MSIL. A diferencia de la plataforma Java, este lenguaje no es interpretado, sino que es compilado de una manera conocida como just-in-time-compilation (JIT), a código nativo. La combinación de estos conceptos es llamada Common Language Infrastructure (CLI); la implementación de Microsoft del CLI es conocida como Common Language Runtime (CLR).
- Independencia del hardware: .NET Framework introduce al Common Type System, o CTS. La especificación CTS define todos los posibles tipos de datos y constructores de programación soportados por el CLR y cómo deberán y no deberán interactuar entre ellos. Debido a esta característica, .NET Framework soporta el desarrollo en múltiples lenguajes de programación.
- Base Class Library: Base Class Library (BCL), a veces referido como Framework Class Library (FCL), es una librería de tipos disponibles a todos los lenguajes que utilizan .NET Framework. BCL provee clases que encapsulan un número de funciones comunes como lectura y escritura de archivos, procesamiento de gráficos, interacción con bases de datos, manipulación de documentos XML, etc.
- Entrega simplificada: La configuración del registro, distribución de archivos, y problemas con DLL's han sido casi completamente eliminados a través de nuevos mecanismos de instalación en .NET Framework.
- Seguridad: .NET permite que el código se ejecute con diferentes niveles de seguridad, sin necesidad de utilizar un "*sandbox*" separado.

Debido a la inherente naturaleza de .NET Framework, a través del uso de un lenguaje intermedio, es independiente de la plataforma. Mientras que la creencia es que .NET Framework está únicamente disponible para el sistema operativo Windows, esto es incorrecto. Microsoft provee Shared Source Common Language Infrastructure, una versión del CLI para Windows, FreeBSD y Mac OS X. Adicionalmente, debido a que el CLI es ahora un estándar ECMA internacional, varios proyectos de desarrollo *Open Source* han surgido para proveer soporte a plataformas adicionales, los más notables de estos proyectos son Mono y DotGNU. Microsoft también provee una versión reducida de .NET Framework para ser utilizada en dispositivos inteligentes, como Pocket PC y Smartphones, llamada .NET Compact Framework.

A pesar de que .NET Framework está disponible y provee compiladores sin costo alguno, Microsoft ofrece un número adicional de herramientas para facilitar el desarrollo de procesos. La más prominente herramienta es Visual Studio .NET Integrated Development

Environment. La IDE de código abierto SharpDevelop y Eclipse también soportan el desarrollo .NET.

Historia:

A pesar de que algunos creen que las tecnologías utilizadas en .NET fueron originalmente desarrolladas por Microsoft como su versión propia de la plataforma Java, la verdad es que muchos de los equipos trabajando en .NET inicialmente comenzaron creando COM+ 2.5. Otros departamentos estaban también mejorando otras tecnologías Microsoft; los departamentos de servidores Web estaban creando ASP 4.0 y el departamento de cómputo distribuido de Microsoft estaba creando lo que se llamaba "Servicios Windows de Siguiente Generación". El trabajo de los varios departamentos fueron fusionados en lo que ahora se llama .NET.

Cuando Microsoft decidió finalizar el uso de las tecnologías Java de Sun Microsystems en 1998 debido a una demanda legal, relativa a una extensión Java, los productos existentes Microsoft J++ (Java) se convirtieron en el inicio del proyecto .NET. Se dijo que el código de .NET Common Language Runtime (CLR) venía de OmniVM de Colusa Software, la cual Microsoft adquirió el 12 de Marzo de 1996.

El CLR es la implementación de Microsoft del estándar ECMA CLI para la plataforma Windows. El Framework Mono provee una implementación parcial del CLI y algunas partes de .NET Framework para Linux y Solaris.

Mientras que el modelo original de .NET era el de un fundamento general (.NET Framework) con tres pilares primarios (ASP .NET, Windows y servicios Web), el modelo para .NET 2.0 es el de ser un fundamento para la plataforma de siguiente generación de Microsoft conocida como WinFX, la cual es la unificación de las tecnologías de desarrollo de Microsoft, dentro de un modelo de programación. WinFX es también el reemplazo de la longeva API Win32 introducida a principios de 1990. FX en WinFX es la abreviación para el .NET Framework, asegurando así que la mayoría de los proyectos basados en Microsoft tendrán un fundamento .NET.

Un cuarto componente del .NET Framework es el .NET Compact Framework, el cual fue diseñado para ejecutarse en dispositivos Windows CE como Pocket PC, Smartphone y otros paquetes generalizados de Windows CE (o WinCE). El Compact Framework se ejecuta como un componente del sistema operativo Windows CE, el cual es un sistema operativo *Open Source*, y el cual puede ser modificado fácilmente. El Compact Framework es más pequeño que .NET Framework y se ajusta fácilmente dentro de WinCE. Compact Framework permite a mucho del código diseñado para versiones más grandes de Windows, trabajar en dispositivos compactos.

Versiones:

- .NET Framework 1.0.3705: La versión inicial de .NET Framework, liberada en enero de 2002. Está disponible por sí sola como un paquete distribuido o dentro de

un kit de desarrollo de software. Es también parte de la primera versión de Microsoft Visual Studio .NET (también conocida como Visual Studio .NET 2002).

- .NET Framework 1.1.4322: Primera actualización importante de .NET Framework, liberada en Abril de 2003. Está disponible como un paquete distribuido o en un kit de desarrollo de software. Es también parte de la segunda versión de Microsoft Visual Studio .NET (liberada como Visual Studio .NET 2003). Ésta es la primera versión de .NET Framework que se incluye como parte del sistema operativo Windows, y de Windows Server 2003.

- .NET Framework 2.0.50727.42: Liberado el 7 de Noviembre de 2005, junto con Visual Studio 2005, SQL 2005 y BizTalk 2006. Puede obtenerse gratuitamente del sitio Web de Microsoft; también se puede obtener gratuitamente el Software Development Kit. Incluye una nueva API para aplicaciones nativas que deseen albergar una nueva instancia del .NET Runtime. La nueva API da un control de grano fino en el comportamiento del runtime con respecto al multithreading, alocación de memoria, carga de ensamblajes, etc. Tiene soporte de 64 bits para hardware de plataformas x64 y IA64; soporte para Generics construidos directamente dentro de .NET CLR.

Arquitectura .NET Framework: Common Language Infrastructure (CLI).

El componente más importante de .NET Framework reside en el Common Language Infrastructure, o CLI (Fig. 1.2). El propósito de CLI es el de proveer una plataforma de lenguaje agnóstico para el desarrollo de aplicaciones, incluyendo componentes para: manejo de excepciones, recolector de basura, seguridad e interoperabilidad. La implementación de Microsoft del CLI es llamada Common Language Runtime, o CLR. CLI se compone de varias partes primarias:

- Common Type System (CTS).
- Common Language Specification (CLS).
- Common Intermediate Language (CIL).
- Just-in-Time Compiler (JIT).
- Virtual Extension System (VES).

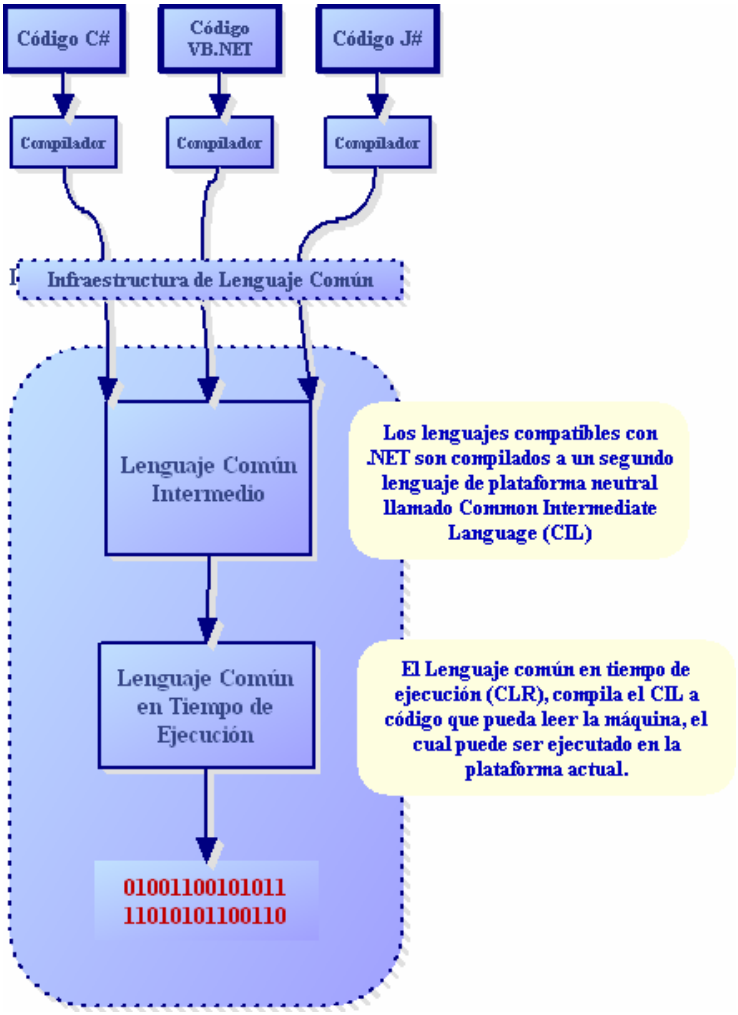

Fig. 1.2 Perspectiva general del Common Language Infrastructure (CLI). [1]

Ensamblajes:

Para la implementación de Windows, un ensamblaje (assembly) es el equivalente a un Portable Executable (PE) o DLL. Los ensamblajes son la unidad .NET de entrega, versión y seguridad. Un ensamblaje puede construirse de uno o varios archivos, pero alguno de éstos debe contener un manifiesto, el cual posee metadatos acerca del ensamblaje. El nombre completo del ensamblaje contiene: nombre de archivo PE, número de versión, cultura y token de llave pública; sólo el PE es necesario, los demás son opcionales. El token de llave pública es un hash de 64 bits de la llave pública de una llave par pública-privada, y es utilizado para asegurar la singularidad del nombre del ensamblaje. Se dice que un ensamblaje que tiene un token de llave pública, tiene un nombre fuerte. Este esquema de nombres significa que puede ser identificado de manera única por el CLR, y cuando se combina con el Global Assembly Cache, permite que existan múltiples versiones de una misma librería, en la misma máquina sin el peligro de cargar una versión errónea y sin forzar a los desarrolladores a enumerar correctamente o colocar inteligentemente ejecutables.

Metadatos:

Todo el CIL está descrito a través de metadatos .NET. El CLR checa los metadatos para asegurarse que el método correcto ha sido invocado. Los metadatos son generalmente utilizados por compiladores de lenguaje, pero los desarrolladores pueden crear sus propios metadatos a través de atributos personalizados.

Base Class Library (BCL):

Base Class Library, algunas veces referido como Framework Class Library (FCL), es una librería de tipos disponibles para todos los lenguajes que utilizan .NET Framework. BCL provee clases que encapsulan un número de funciones comunes como lectura y escritura de archivos, manipulación de gráficos, interacción con bases de datos, manipulación de documentos XML, etc. BCL es más grande que otras librerías, pero tiene mucho más funcionalidad en un solo paquete.

Seguridad:

.NET posee sus propios mecanismos de seguridad, con dos características generales: seguridad de acceso de código, validación y verificación. La seguridad de acceso de código está basada en evidencia que está asociada con un ensamblaje específico. Típicamente la evidencia es la fuente del ensamblaje (si está instalado en una máquina local, o ha sido descargado desde una Intranet o Internet). Utiliza evidencia para determinar cuales permisos son otorgados al código. Esta demanda ocasiona que el CLR realice un recorrido de la pila de memoria: cada ensamblaje de cada método en la llamada a la pila es revisada para comprobar los permisos requeridos y si a algún ensamblaje no se le da el permiso, entonces una excepción de seguridad es lanzada. Cuando un ensamblaje es cargado, el CLR realiza varias pruebas. Dos de esas pruebas son validación y verificación. Durante la validación el CLR revisa que el ensamblaje contenga metadatos y CIL, y verifica que las tablas internas sean correctas. La verificación no es exacta. El mecanismo de verificación ve si el código contiene código que es inseguro, o *"unsafe"*. El algoritmo utilizado es algo conservador y por lo tanto, algunas veces el código seguro no es verificado. El código inseguro sólo será ejecutado si el ensamblaje tiene un permiso de "ignorar verificación", lo cual generalmente significa código que está instalado en la máquina local.

Windows Forms:

Windows Forms, o WinForms, es la porción de .NET Framework que provee envolturas administradas, o "wrappers", para los widgets contenidos en la API Win32. Una interfaz de usuario usualmente consta de una ventana principal o Windows Form, y varios controles dentro de ella, como botones, campos de texto, etc.

ASP.NET:

ASP.NET, el reemplazo de Microsoft de la tecnología de programación Web clásica Active Server Pages (ASP). Implementa .NET y tiene una librería de clases .NET. ASP.NET 2.0 es una actualización monumental para ASP.NET incluyendo muchos controles que reducen dramáticamente la cantidad de código que los desarrolladores escriben. El diseño de ASP.NET 2.0 fue posterior al rediseño completo de .NET, que promete a los programadores escribir 70% menos código, comparado con lo que se necesitaría utilizando .NET 1.x.

ADO.NET:

ADO.NET es un conjunto de clases que exponen servicios de acceso a datos para los programadores .NET. ADO.NET provee un conjunto de componentes para crear aplicaciones distribuidas y con intercambio de datos. XML juega un papel importante, ya que procedimientos RPC se proveen para serializar objetos de datos como datos XML.

.NET Remoting:

La infraestructura remota .NET es un acercamiento abstracto para interprocesar comunicaciones. Microsoft .NET Remoting provee un rico y extensible Framework para objetos viviendo en diferentes AppDomains, en diferentes procesos, y en diferentes máquinas para comunicarse unas a otras sin problemas. .NET Remoting ofrece un poderoso e incluso simple modelo de programación y soporte en tiempo de ejecución para volver estas interacciones transparentes.

Web Services:

Los servicios Web extienden la infraestructura World Wide Web para proveer medios para que las aplicaciones puedan conectarse con otras aplicaciones vía protocolos Web y formatos de datos como HTTP, XML y SOAP, sin necesidad de preocuparse de cómo cada servicio Web es implementado.

Lenguajes .NET:

El núcleo de .NET Framework reside en la implementación de Microsoft del lenguaje agnóstico CLI, el CLR, el cual ejecuta el lenguaje intermedio MSIL. Debido a que el motor de tiempo de ejecución está basado en un lenguaje intermedio, cualquier compilador de código fuente que emite MSIL puede ser utilizado para crear ensamblajes que pueden ser ejecutados por el CLR. El CLR entonces, soporta lenguajes orientados a objetos y lenguajes de procedimiento.

Mientras que actualmente existen más de 40 lenguajes con compiladores para .NET Framework, solo un pequeño número de éstos son ampliamente utilizados y promovidos por Microsoft. El resto está compuesto de lenguajes desarrollados por terceros. Debido a

que los requerimientos específicos de la compatibilidad del CLR, muchos de los lenguajes alternativos sufrieron vastos cambios para no sólo cumplir con estos requerimientos, sino que también para mejorar muchos otros aspectos de la implementación del lenguaje. Muchos de estos lenguajes alternativos proveen compiladores gratuitos y algunos proveedores venden IDE's.

Lenguajes soportados por Microsoft:

- C#: El lenguaje insignia de .NET Framework que posee similitudes con los lenguajes Java y C++.
- Visual Basic .NET: Una nueva versión completamente rediseñada del lenguaje Visual Basic para .NET Framework.
- C++/CLI y Manager C++: Una versión administrada del lenguaje C++.
- J#: Un lenguaje transicional .NET de Java y J++.
- JScript .NET: Una versión compilada del lenguaje JScript.
- Eiffel: El lenguaje de método Eiffel.

Estandarización y licencia:

En agosto de 2001, Microsoft, HP e Intel trabajaron para estandartizar el CLI y el lenguaje de programación C#. Para diciembre de 2001, ambos fueron ratificados como estándares ECMA (ECMA 335 y ECMA 334). ISO siguió en abril de 2003 (ISO/IEC 23271 e ISO/IEC23270).

Éste es un riesgo calculado, pero animará a implementaciones que cumplan con los estándares, para crear un puente para que software no basado en Windows pueda ser convertido a Microsoft .NET. Un grupo promoviendo esto es el International .NET Association (INETA).

.NET vs. Java EE:

CLI, CIL, y C# (versión 1.x) poseen similitudes con JVM y Java de Sun. Ambos utilizan su propio lenguaje de código bit intermedio, Microsoft llamándolo Microsoft Intermediate Language (MSIL). MSIL fue diseñado para compilación just–in-time (JITing), mientras que el bytecode de Java originalmente fue diseñado para interpretar. .NET actualmente está disponible para plataformas Windows, mientras que Java está disponible para varias plataformas. Sin embargo, Microsoft, liberó el código fuente completo de su implementación inicial para el .NET Framework completo, incluyendo el código fuente de C++ para el CLR. Actualmente existen algunas otras implementaciones en curso, como Portable .NET, y Mono que puede ser utilizado para ejecutar aplicaciones .NET en sistemas operativos parecidos a Unix, como por ejemplo Linux, FreeBSD y Mac OS X.

.NET vs. COM:

La tecnología previa de componente de software aprobada por Microsoft para sistemas de software de gran escala, fue COM, utilizando mejoras de COM+ o MTS para componentes

transicionales distribuidos. Mientras que .NET puede envolver objetos COM y viceversa, a lo cual Microsoft llamó Runtime Callable Wrapper (RCW) y COM Callable Wrapper (CCW) respectivamente, ha sido claramente estipulado por Microsoft que .NET eventualmente remplazará a COM como una arquitectura de componente de software. Microsoft espera que desarrolladores escribiendo nuevas aplicaciones para la plataforma Win32 empiecen a utilizar .NET en vez de COM, con el uso de servicios existentes vía interfaces abstractas. [1]

I.3. Visual Studio .NET

Microsoft Visual Studio es un ambiente integrado de programación avanzada. Permite a los programadores crear programas que se ejecuten en Microsoft Windows y el World Wide Web.

Fig1.3 Logo oficial de Microsoft Visual Studio .NET. [6]

En Visual Studio se incluyen:

- Visual Basic .NET
- Visual C++
- Visual C#
- Visual J#
- SQL Server Express

En versiones anteriores, se incluían:

- Visual InterDev: Una aplicación de desarrollo de páginas Web utilizado para modificar Active Server Pages (ASP), así como HTML y otros archivos de script Web.
- Visual J++: Una herramienta de desarrollo Java.
- Visual Fox Pro: Un lenguaje de programación xBase.

Historia:

Visual Studio 97:

Microsoft liberó Visual Studio en 1997, que incluía muchas de sus herramientas de programación. Visual Studio 97 fue liberado en dos ediciones, Professional y Enterprise. Incluía Visual Basic 5.0 y Visual C++ 5.0, esencialmente para programación Windows; Visual J++ 1.1 para programación Windows y Java, y Visual FoxPro 5.0 para programación xBase. También se introducía a Visual InterDev para crear websites

dinámicamente generados utilizando Active Server Pages (ASP), además se incluía una versión de la librería Microsoft Developer Network. Visual Studio 97 fue el primer intento de Microsoft de utilizar un mismo ambiente de desarrollo para múltiples lenguajes. Visual C++, J++, InterDev, y la librería MSDN utilizaban este ambiente, llamado Developer Studio. Visual Basic utilizaba un ambiente separado, al igual que Visual FoxPro.

Visual Studio 6.0:

La siguiente versión, 6.0, fue liberada en 1998. El número de versión de todas sus partes constituyentes también se renombraron como 6.0, incluyendo Visual J++, el cual pasó de la versión 1.1 a 6.0, y Visual InterDev 6.0, que era la versión 1.0. Esta versión fue la base de los sistemas de desarrollo Microsoft de los siguientes 4 años, mientras que Microsoft pasaba al desarrollo .NET.

Visual Studio 6.0 fue la última versión en incluir Visual Basic como la mayoría de los programadores VB lo conocían; las versiones subsecuentes incluirían una versión algo diferente de VB basada en .NET. También fue la última versión que incluiría Visual J++, la cual tenía lazos más estrechos del lenguaje Java con Windows, pero que era incompatible con la versión de Sun. Esto ocasionó que Sun Microsystems demandara a Microsoft. Como parte de un acuerdo, Microsoft no volvería a vender herramientas de programación que utilizaran Java Virtual Machina (JVM).

A pesar de que la meta a largo plazo de Microsoft era el unificar sus herramientas de programación dentro de un solo ambiente, esta versión utilizaba más de un ambiente. Visual J++ y Visual InterDev se apartaron del ambiente Visual C++, mientras que Visual Basic y Visual FoxPro mantenían herramientas separadas.

Visual Studio .NET (2002):

Microsoft liberó Visual Studio .NET (mas específicamente Visual Studio 7.0) en el 2002. El mayor cambio fue la introducción de .NET Framework. Los programas desarrollados utilizando el .NET Framework no son compilados a lenguaje máquina (como C++, por ejemplo) sino que son compilados a un formato llamado MIL o CIL. Cuando una aplicación MIL es ejecutada, es compilada al lenguaje máquina apropiado de la plataforma en que se ejecuta, mientras esta aplicación MIL es ejecutada. A través de este método, Microsoft espera ser capaz de soportar varias implementaciones de sus sistemas operativos Windows, como por ejemplo Windows CE.

Los programas compilados a MIL pueden ser ejecutados sólo en plataformas que tengan una implementación de .NET Framework. Es posible correr programas en Linux o Mac OS X utilizando implementaciones no basadas en .NET Framework, como Mono o DotGNU.
También Microsoft introdujo C#, un lenguaje similar a Java, específicamente diseñado para .NET. También introdujo el sucesor de Visual J++ llamado Visual J#. Los programas Visual J# utilizan una sintaxis de Java. Sin embargo, a diferencia de Visual J++, los programas Visual J# solo pueden utilizar .NET Framework, no la máquina virtual de Java JVM.

Visual Basic fue drásticamente rediseñado para cumplir con los requerimientos del nuevo framework, y la nueva versión fue llamada Visual Basic .NET. Microsoft también añadió extensiones para C++, llamadas Managed C++, para que así los programadores de C++ pudieran crear programas .NET. Visual Studio .NET puede ser utilizado para crear aplicaciones diseñadas para Windows (utilizando Windows Forms, parte de .NET Framework), Web (utilizando ASP.NET y Web Services), y dispositivos portátiles (utilizando .NET Compact Framework).

El ambiente Visual Studio .NET fue reescrito para utilizar parcialmente .NET. Todos los lenguajes fueron finalmente unificados bajo un solo ambiente, con excepción de Visual FoxPro. Comparado con versiones anteriores de Visual Studio, tiene una interfaz más limpia y mayor cohesividad. Es también más fácil de personalizar, con ventanas de herramientas que automáticamente se ocultan cuando no son utilizadas.

Visual Studio .NET 2003:

Microsoft introdujo una actualización menor de Visual Studio .NET en el 2003 llamada Visual Studio .NET 2003. Incluía una mejora del .NET Framework, versión 1.1, también contenía un soporte integrado para dispositivos móviles, utilizando ya sea ASP.NET, o .NET Compact Framework. Además, el compilador Visual C++ fue mejorado para cumplir con estándares, específicamente en el área de especialización parcial de plantillas, y una versión del compilador C++ incluida en Visual Studio .NET 2003, llamada Visual C++ Toolkit 2003, fue liberada al público de manera gratuita, aunque no incluía ninguna IDE.

Visual Studio 2005:

Visual Studio 2005, nombre clave Whidbey (una referencia a la isla NAS Whidbey en el Pacífico), fue liberada en línea en octubre de 2005 y disponible en tiendas unas semanas después. Aunque Microsoft removió la frase .NET de Visual Studio y de cualquier otro producto que utilizara .NET dentro de su nombre, sigue utilizando primariamente a .NET Framework, el cual fue actualizado a la versión 2.0. En la Fig. 1.4 se puede apreciar la IDE de Visual Studio 2005.

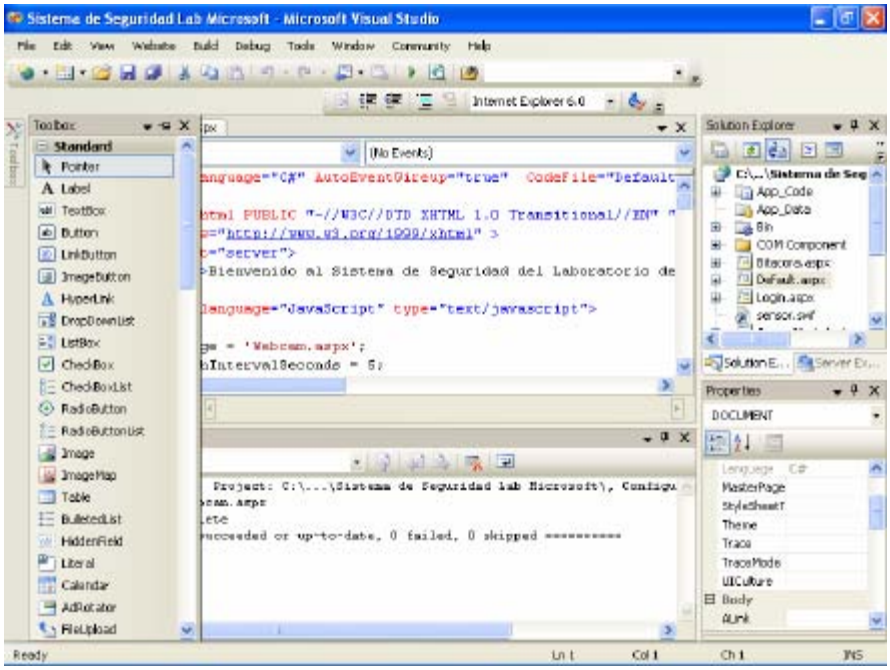

Fig. 1.4 GUI o Interfaz Gráfica de Microsoft Visual Studio 2005

La característica más importante del lenguaje C# añadida a esta versión fue la introducción de Generics, que es similar a las plantillas en C++. Esto incrementa potencialmente el número de *bugs* atrapados en tiempo de compilación, en lugar de ser atrapados en tiempo de ejecución. C++ también tuvo una mejora similar con la adición de C++/CLI, la cual se prevé remplazará al Managed C++. En esta versión no se incluye Visual FoxPro, el cual es vendido de forma separada. Otras características de Visual Studio 2005 son Deployment Designer, el cual permite que el diseño de la aplicación sea validado antes de ser entregada, y un ambiente mejorado para la publicación Web combinada con ASP.NET 2.0.

Fig. 1.5 Logo oficial de Microsoft Visual Studio 2005 [1]

Visual Studio 2005 también agrega soporte extensivo de 64 bits. Visual C++ 2005 soporta compilación para x64 (AMD64 y EM64T), así como IA-64(Itanium). Versiones previas de Visual Studio no soportaban compilación de 64 bits. Platform SDK solo incluía los compiladores de 64 bits y librerías de versiones 64 bits de Visual C++ 6.0.

Además de versiones Standard y Professional, Visual Studio 2005 introdujo varias nuevas versiones. Team System incluye soporte para organizaciones de grandes desarrollos, y viene en ediciones separadas para arquitectos de software, desarrolladores, y testers. También se introdujeron las versiones Express para amateurs y pequeños negocios. Herramientas para Microsoft Office System permite a los desarrolladores crear extensiones para Microsoft Office.

Existen versiones Express separadas para cada lenguaje (Visual Basic, Visual C++, Visual C#, Visual J#), cada una utilizando .NET Framework en Windows, así como Visual Web Developer para crear sitios Web ASP.NET.

Versiones Futuras:

El sucesor de Visual Studio 2005 está actualmente bajo desarrollo bajo el nombre clave Orcas (haciendo referencia a la isla Orcas en el Pacífico). Se prevé que Orcas será liberado al mismo tiempo que Windows Vista. Al sucesor de Orcas se le conoce con el nombre clave Hawai. [11]

I.4. Visual C#

C# es un lenguaje de programación derivado del C/C++, con la finalidad de proporcionar un método sencillo de creación de aplicaciones de propósito general y con programación orientada a objetos (OOP). Inicialmente fue desarrollado por Microsoft, como parte de su plataforma .NET Framework (bajo el entorno Visual Studio .NET). Se trata de un lenguaje de alto nivel, compilado y que proporciona un método más simple de control de la memoria que el disponible en el lenguaje C/C++. Fue ideado para la creación de aplicaciones simples, incluso en dispositivos de mano y multiplataforma, como una alternativa a Java y C/C++, aunque no está diseñado específicamente para competir con este último en grandes aplicaciones.

Historia:

El principal diseñador de C# y arquitecto principal en Microsoft fue Anders Heijlsberg. Su experiencia previa en lenguajes de programación y diseño de frameworks (Visual J++, Borland Delphi, Turbo Pascal) puede ser observada en la sintaxis del lenguaje C#, así como en todo el Common Language Runtime.

Características del lenguaje:

C# es en muchos sentidos, el lenguaje de programación que más directamente refleja el Common Language Runtime (CLR) en el cual todos los programas .NET se ejecutan, y depende fuertemente de este framework debido a que fue diseñado específicamente para sacar provecho de las características que el CLR provee. La mayoría de los tipos intrínsecos de C# corresponden a valores-tipo implementados por .NET Framework. Una creencia errónea común es que estos tipos son recolectados por el *garbage collector (GC)*, a pesar de que no lo son. Son verdaderos valores-tipos y son alocados en el stack (con excepción de System.Object y System.String). Aplicaciones escritas en C# requieren una implementación del CLR para ejecutarse.

A diferencia de las clases de Java, los programas .NET son compilados en dos pasadas, almacenados como código binario en la primera pasada y compilado en la estación de

trabajo cliente en la segunda pasada. Los programas de Java requieren una máquina virtual JVM. Los programas VB6 requieren una librería de soporte. De la misma manera, los programas .NET requieren un conjunto de librerías de soporte y un ambiente de ejecución central, el cual maneja la inicialización y proceso inicial JIT. Comparado con C y C++, el lenguaje está restringido de varias maneras:

- Los apuntadores sólo pueden ser utilizados dentro de un alcance inseguro o "unsafe", y sólo los programas con permisos apropiados pueden ejecutar código marcado como inseguro. La mayoría del acceso a objetos se realiza a través de referencias seguras, las cuales no pueden volverse inválidas, y la mayoría de la aritmética se verifica por sobreflujo. Los apuntadores pueden sólo ser construidos para tipos de valor; los objetos administrados por el *garbage collector* sólo pueden ser referidos. Un apuntador inseguro puede ser construido no sólo para valores-tipo, sino que también para subclases de System.Object. También puede ser escrito código seguro que utilice apuntadores (System.IntPtr), sin embargo, no se puede realizar manipulación de memoria en la dirección de almacenamiento desde C# sin las palabras clave *unsafe* y *fixed*, al menos que se llamen a las funciones de la API Win32 como rtlMoveMemory.
- La memoria administrada no puede ser explícitamente liberada, sino que es recolectada por el recolector de basura (GC) cuando no existen más referencias a la memoria. Los objetos que hacen referencia a recursos no administrados, como HBRUSH, pueden ser instruidos para liberar esos recursos a través de la interfaz estándar IDisposable, la cual provee un patrón para la distribución determinística de recursos.
- Solo la herencia singular está disponible, aunque una clase puede implementar cualquier número de interfaces. Esta fue una decisión de diseño del arquitecto principal del lenguaje Anders Heijlsberg para evitar complicaciones, evitar el "infierno de la dependencia" y simplificar los requerimientos arquitectónicos a través del CLR.
- En C#, la única conversión implícita por default son conversiones seguras, como la ampliación de enteros y conversión de un tipo derivado a un tipo base. No existen las conversiones implícitas entre boolean y enteros, entre miembros de enumeraciones y enteros, no hay apuntadores vacíos (aunque las referencias a Object son similares), y cualquier conversión implícita definida por el usuario debe ser explícitamente marcada como tal, a diferencia del constructor copy de C++.
- La sintaxis para la declaración de arreglos es diferente ("int[] a=new int[5]"; en lugar de int a[5]).
- Los miembros de enumeraciones son colocados en su propio namespace.
- C# 1.0 carece de plantillas, sin embargo, C# 2.0 provee Generics.
- Las propiedades están disponibles, lo cual resulta en una sintaxis que se parece al acceso de miembros de campos de C++, similar a VB.
- Reflexión total de tipos y descubrimiento están disponibles.
- A diferencia de C++ en .NET, el cual tiene la opción de ser compilado como código administrado o como código no administrado (código máquina nativa), C# puede ser sólo compilado como código administrado, el cual es más propenso a ser evitado por aplicaciones de recursos intensivos. [8]

Librerías de código:

La especificación ECMA para C# detalla un conjunto mínimo de librerías de clases que el compilador espera tener disponible y definen las bases requeridas. .NET Framework es una librería de clases que puede ser utilizada desde el lenguaje .NET para realizar tareas desde una representación de datos simples y manipulación de cadenas para generar páginas Web dinámicas (ASP.NET), análisis y reflexión XML. El código es organizado dentro de conjuntos de Namespaces los cuales agrupan clases con funciones similares, por ejemplo System.Drawing para gráficos, System.Collections para estructuras de datos y System.Windows.Forms para Windows Forms.

Un mayor nivel de organización se provee con el concepto de ensamblaje. Un ensamblaje puede ser un solo archivo o archivos múltiples ligados los cuales pueden contener namespaces y objetos. Programas que necesitan clases para realizar una función particular pueden referenciar ensamblajes, como por ejemplo System.Drawing.dll y System.Windows.Forms.dll, así como también la librería central conocida como mscorlib.dll.

Ejemplo "Hola Mundo":

El siguiente es un programa simple en C#:

```
public class HolaMundo
{
        public static void Main()
        {
                System.Console.WriteLine("¡Hola Mundo!");
        }
}
```

Este código imprime en pantalla: ¡Hola Mundo!.

Las partes que constituyen al código anterior son:

- public class HolaMundo: Define una clase. La palabra clave public permite que objetos en otros proyectos pueden libremente utilizar esta clase.
- public static void Main(): Este es el punto de entrada donde el programa empieza su ejecución. Puede ser invocada desde otro código utilizando la sintaxis HolaMundo.Main().
- System.Console.WriteLine("¡Hola Mundo!"); : Esta línea realiza la tarea de escribir un mensaje al dispositivo de salida predeterminado. Console es un objeto del sistema, representando una consola de línea de comandos donde el programa puede ingresar e imprimir texto. El programa invoca el método de Console, WriteLine, el cual causa que la cadena que se le pasa se muestre en la consola.

Nombre del lenguaje:

El nombre C? (Fig. 1.6) pudo haber sido escogido por Microsoft para suponer una progresión del lenguaje C++, con el símbolo ? asemejándose a dos símbolos ++, o cuatro símbolos ++ colocados en un cuadrado.

Fig. 1.6 Logo del lenguaje de programación C?. [9]

Debido a las limitaciones técnicas para desplegarlo (fuentes, exploradores, etc.), y al hecho de que el símbolo sharp ? no está presente en un teclado estándar, el signo de número # fue escogido para representar el símbolo sharp en el nombre escrito del lenguaje. Microsoft clarifica el nombre del lenguaje como:

"*El nombre oral del lenguaje es "C Sharp" haciendo referencia al signo musical "sharp", el cual incrementa un tono denotado por una letra (entre A y G) en medio tono. Sin embargo, debido a la dificultad para teclearlo se decidió representar el signo sharp con un símbolo de número #, el cual está disponible en cualquier teclado, en lugar que el signo sharp Unicode ? correcto* ".

El sufijo "sharp" ha sido emulado por otros lenguajes .NET que son variantes de lenguajes existentes, incluidos J# (la implementación Microsoft de Java), A# (de Ada), F# (se presume proviene de System F, el sistema de tipo utilizado por la familia ML) y Gtk# (un wrapper .NET para GTK+). [9]

I.5 ASP.NET

ASP.NET es un conjunto de técnicas de desarrollo de red comercializadas por Microsoft. Los programadores las utilizan para construir sitios Web dinámicos, aplicaciones Web y servicios XML. Es parte de la plataforma .NET de Microsoft y es el sucesor de la tecnología Active Server Pages (ASP) de Microsoft.

Principios de ASP.NET:

A pesar de que ASP.NET ha tomado su nombre de la antigua tecnología de desarrollo Web de Microsoft, las dos difieren de manera significativa. Microsoft ha completamente reconstruido ASP.NET, basados en el Common Language Runtime (CLR) compartido por todas las aplicaciones Microsoft .NET.

Los programadores pueden escribir código ASP.NET utilizando cualquiera de los diferentes lenguajes soportados por .NET Framework, usualmente Visual Basic .NET,

JScript.NET o C#, pero también se incluyen lenguajes Open Source como Perl y Python. ASP.NET posee beneficios de desempeño sobre otras tecnologías basadas en scripts debido a que el código *Server-Side* es compilado en uno o unos cuantos archivos DLL en un servidor Web.

ASP.NET intenta simplificar la transición de los desarrolladores del desarrollo de aplicaciones Windows hacia el desarrollo Web al ofrecer la habilidad de construir páginas compuestas de controles similares a la interfaz de usuario de Windows. Un control Web, tal como un button o label, funciona casi del mismo modo que su contraparte en Windows, se puede asignar sus propiedades mediante código y responder a eventos. Los controles saben cómo construirse a ellos mismos, mientras que los controles Windows se dibujan a ellos mismo en la pantalla, los controles Web producen segmentos de HTML y JavaScript los cuales forman parte del envío de la página al explorador del usuario final.

ASP.NET obliga al programador a desarrollar aplicaciones utilizando una GUI controlada por eventos, o event-driven, en lugar de un ambiente de scripts Web convencional como ASP y PHP. El framework intenta combinar las tecnologías existentes como JavaScript con componentes internos como “Viewstate” para generar estados persistentes (petición interna) al ambiente Web.

ASP.NET utiliza .NET Framework como su infraestructura. .NET Framework ofrece un ambiente de tiempo de ejecución administrado (como Java), que proveen una máquina virtual con JIT y una librería de clases. Los numerosos controles .NET, clases y herramientas pueden disminuir el tiempo de desarrollo al proveer un conjunto rico de características para tareas comunes de programación. El acceso a datos provee un buen ejemplo, y viene de la mano con ASP.NET. Un desarrollador puede construir una página que muestre una lista de registros en una base de datos, de manera significativamente más fácil utilizando ASP.NET que con ASP.

Formato de archivos ASPX:

ASPX es un formato de archivo de texto utilizado como páginas Web Form en el ambiente .NET. El archivo ASPX típicamente contiene solamente código HTML estático en donde el desarrollador coloca todos los campos de forma y contenido de texto requerido por la página Web. El código dinámico que implica peticiones y respuestas del servidor es colocado en la página HTML con la etiqueta <% -- *código dinámico* -- >, el cual es similar a cualquier otra tecnología de desarrollo Web como ASP y JSP. ASP.NET soporta bloques de código dentro de un archivo ASPX, pero esta práctica no es recomendada.

Cuando los proyectos son desarrollados con la tecnología .NET, los archivos de formas o páginas Web con código HTML son renombrados a formato ASPX con código dinámico insertado dentro de una etiqueta. Cuando un cliente pide información al servidor, por ejemplo, para buscar el precio de un boleto de una agencia de viajes, la página ASPX con contenido de texto y campos de formas obtiene la información del cliente y se la envía al servidor. Con la ayuda del código dinámico dentro de una etiqueta, el cliente obtiene la respuesta o la información peticionada por el servidor.

Los archivos ASPX y otros archivos de recursos son colocados en un host virtual dentro de Internet Information Services (IIS), u otro servidor compatible con ASP.NET. Cuando un cliente pide información, .NET Framework analiza y compila el archivo en una clase .NET y envía la respuesta. A diferencia de otras tecnologías de desarrollo Web, las cuales compilan sus archivos cada vez que peticiona el cliente, los archivos ASPX son compilados sólo la primera vez que son accedidos y entonces son reutilizados para reducir el tiempo de respuesta. Los desarrolladores pueden también optar por precompilar su código antes de ser enviado, eliminando la necesidad de la compilación just-in-time (JIT).

Ventajas de ASP.NET sobre ASP:

- Código compilado significa que las aplicaciones se ejecutan más rápido, con menos errores de tiempo de diseño atrapados en la fase de desarrollo.
- Mejora significativa en el manejo de errores de tiempo de ejecución, haciendo uso de excepciones y bloques try-catch.
- Los controles definidos por el usuario permiten comúnmente utilizar plantillas, tales como menús.
- Metáforas similares a las aplicaciones Windows como controles y eventos, los cuales vuelven posible el desarrollo de interfaces de usuario más ricas.
- Un conjunto escalable de controles y librerías de clases que permiten la construcción rápida de aplicaciones.
- ASP.NET aumenta las capacidades multilenguaje del .NET CLR, permitiendo a las páginas Web ser codificadas en VB.NET, C#, J#, etc.
- Capacidad de guardar en caché páginas Web completas o sólo partes de ellas para mejorar el rendimiento.
- Habilidad de utilizar el modelo de desarrollo “detrás del código” para separar la lógica de negocios de la presentación.
- Si una aplicación ASP.NET malgasta la memoria, el ASP.NET Runtime descarga el AppDomain que hospeda el error y recarga la aplicación con un nuevo AppDomain.
- Los estados de sesión de ASP.NET pueden ser guardados en una base de datos SQL Server o en un proceso separado ejecutándose en la misma máquina, como el servidor Web o en una máquina diferente. De esta manera los valores de sesión no se pierden cuando el servidor Web es reiniciado o el proceso ASP.NET es reciclado.

Desventajas con otras plataformas:

El framework del servidor ejecuta Microsoft IIS 5.0 nativo o posterior, y Cassini. Sin embargo, puede correr en Linux o en cualquier de los frameworks alternativos basados en el estándar ECMA. El más conocido es el proyecto Mono, un framework free/opensource.
Versiones previas de ASP.NET (1.0 y 1.1) fueron criticadas por su falta de cumplimiento con los estándares. El HTML y JavaScript generado enviado al explorador cliente no siempre se validaba con los estándares W3C/ECMA. Además, la característica del framework de detección de navegador algunas veces identificaban incorrectamente exploradores Web que no fueran el Internet Explorer como de nivel bajo, y regresaban

HTML o JavaScript roto o incompleto a los clientes. De cualquier forma, en la versión 2.0 todos los controles generan HTML 4.0, XHTML 1.0 o XHTML 1.1 válido, dependiendo de la configuración del sitio, la detección de exploradores Web que cumplan con estándares ha sido mejorada, y el soporte de Cascading Style Sheets (CCS) es más amplio. [7]

I.6. Dispositivos Móviles

Dispositivos móviles es un término genérico que se refiere a la habilidad de utilizar la tecnología para conectar y utilizar información centralizada o aplicaciones de software a través de la utilización de dispositivos de cómputo y comunicaciones que sean pequeños, portátiles e inalámbricos. Algunos de estos dispositivos son laptops con tecnología Wi-Fi, teléfonos móviles, memorias flash USB y Asistentes Personales Digitales (PDA) con interfaces Bluetooth o IRDA.

I.6.1 Historia de los dispositivos móviles

Originalmente, los dispositivos electrónicos como transmisores de radio, sistemas de comunicación inalámbrica, etc., eran estaciones base, o sea, dentro de un edificio u otra arquitectura, típicamente con grandes torres de antenas. El uso masivo de los automóviles provocó el surgimiento de pequeños dispositivos que operaban a 6 volts. En 1950, la transición del automóvil al sistema eléctrico de 12 volts, provocaron el surgimiento de dispositivos que operan a 12 volts, como los radios bidireccionales, conocidos como *mobile rigs* o plataformas móviles. Entonces surgieron industrias como Motorola que empezaron a crear dispositivos móviles, como radios para cabinas de taxi, radios para la policía, y otros dispositivos de 12 volts que se podían colocar debajo del tablero del carro, así como también ser montados en la cajuela. Actualmente existen una amplia variedad de plataformas de cómputo móviles, como displays VGA montados en el tablero de los autos, y computadoras que proveen funciones de navegación GPS para los automovilistas.

A principios de 1980 la frontera entre lo móvil (montados en un vehículo) y portátil (dispositivos de mano) empezó a desvanecerse. Esto comenzó con los “teléfonos de bolsillo”, los cuales tenían un conector de alimentación a través del encendedor de cigarrillos del automóvil, y los cuales eran lo suficientemente pequeños para ser transportados por las personas en sus bolsillos. Los teléfonos móviles de 12 volts empezaron a ser utilizados como teléfonos móviles, y al mismo tiempo, computadoras portátiles como la Osborne comenzaron a surgir. Estos dispositivos podían ser utilizados dentro de un vehículo, conectados a una fuente de poder dentro del vehículo, o también podían ser transportados por las personas, y alimentados con baterías.

Actualmente, ya no existe la barrera entre lo móvil y lo portátil, ya que muchos teléfonos celulares y computadoras pueden operar con 12 volts provenientes del socket del encendedor de cigarrillos del automóvil, así como por baterías integradas dentro de los dispositivos.

Actualmente, casi todos los vehículos son construidos con computadoras a bordo. Incluso existe la propuesta de intercomunicar a estos vehículos mediante una red inalámbrica *ad-hoc*. Debido al uso tan amplio del automóvil, dicha red podría ser utilizada para unir computadoras portátiles y dispositivos móviles, y así tener casi cobertura completa dentro de la mayoría de las áreas urbanas y suburbanas. [5]

I.6.2 PocketPC

Nombre por el que se conoce a cierto tipo de dispositivos de mano, similares a computadoras de dimensiones reducidas (computadora personal de bolsillo, Pocket Personal Computer) que ejecutan el sistema operativo Windows CE, lo que les permite trabajar con aplicaciones similares a las computadoras tipo PC Windows, como Excel o Word, en versiones reducidas. [5]

I.6.3 Infrared Data Association (IrDA)

Asociación creada en 1993 para desarrollar estándares de comunicación inalámbrica entre computadoras por medio de radiaciones infrarrojas, similares a las que existen entre el televisor y el control remoto. Es necesario que emisor y receptor dispongan de puertos de infrarrojo (puertos IrDA), que estén situados en línea, sin ningún obstáculo entre ellos y a pequeña distancia (entre unos centímetros y unos pocos metros, dependiendo del puerto); en esto se diferencia de la tecnología Bluetooth que es omnidireccional y puede traspasar paredes.

Los primeros productos con puertos IrDA aparecieron en 1995 y su uso se fue extendiendo; hoy los encontramos, por ejemplo, en impresoras, ratones, teclados, PDA, computadoras portátiles y cámaras digitales.

Para este fin, existen dos tipos de aplicaciones, según la función que vayan a realizar: IrDA-Data e IrDA-Control. La primera permite la comunicación bidireccional (síncrona o asíncrona) con velocidades que oscilan entre los 9.600 bps y los 4 Mbps; se utiliza para intercambiar información entre PDA o computadoras portátiles y su alcance varía entre unos pocos centímetros y un metro. La segunda se utiliza para conectar periféricos de control como ratones, teclados o joysticks; su velocidad de transmisión llega a los 75 Kbps y permite alcances de hasta cinco metros. [5]

I.6.4 Bluetooth

Es una norma que define un estándar global de comunicación inalámbrica de corto alcance para transmitir voz y datos entre dispositivos móviles (como teléfonos y computadoras portátiles) y dispositivos de escritorio (como las computadoras fijas), mediante un enlace de radiofrecuencias. Su capacidad de transferencia llega a los 720 Kbps y tiene un alcance entre 10 y 100 metros; a diferencia de la tecnología IrDA, es omnidireccional y puede atravesar paredes y otras barreras no metálicas. Esta tecnología de comunicación comprende hardware, software y requerimientos de interoperatibilidad. Para el

establecimiento de la norma se creó en 1998 un grupo de interés especial (Special Interest Group) formado por las empresas Ericsson, IBM, Intel, Nokia y Toshiba.

El nombre procede del rey vikingo Harald Blatan, conocido por los ingleses como Bluetooth, que reinó en Dinamarca y Noruega en la segunda mitad del siglo X y fue uno de los hombres más poderosos de Europa. [5]

I.6.5 Personal Digital Assistant (PDA)

Son dispositivos de mano que fueron diseñados originalmente para ser organizadores personales, pero se volvieron mucho más versátiles con el paso del tiempo. Entre las muchas tareas realizadas por un PDA básico se encuentran: cálculos, reloj y calendario, juegos de computadora, acceso Internet, envío y recepción de e-mail, almacenamiento de notas, libro de contactos, y hojas de cálculo. Equipos PDA más recientes incluyen también pantallas a color y capacidad de reproducir audio, ser utilizados como teléfonos móviles (PDA Phones), exploradores Web, y reproductores de música y video. Muchos PDA pueden acceder redes Intranet, Extranet e Internet vía Wi-Fi, o a través de Wireless Wide-Area Networks (WWAN).

El principal propósito de los PDA es el de actuar como un organizador electrónico o planificador diario que sea portátil, fácil de utilizar y capaz de compartir información con una PC. Se creó para ser una extensión de la PC, no un suplente.

Los PDA, también llamados *handhelds* o *palmtops*, se clasifican en varios tipos:

- PDA Tradicional: Los PDA actuales son descendientes del Palm Pilot original y de los dispositivos Microsoft Handheld PC. Los dispositivos Palm utilizan el sistema operativo Palm OS, y Microsoft Pocket PC utiliza Windows Mobile. Con el tiempo, la diferencia entre ambos sistemas operativos se ha vuelto mínima. Los PDA Palm OS ofrecen:

    - Una vasta librería de aplicaciones de terceros (más de 20000), las cuales pueden ser añadidas al sistema.
    - Una aplicación actualizada del software de reconocimiento de escritura a mano conocido como Graffiti.
    - Sincronización con computadoras Windows y Macintosh utilizando Palm Desktop.
    - Pantallas más pequeñas para dedicar un área exclusiva para Graffiti en el dispositivo, aunque también existen dispositivos Palm OS que ofrecen un área graffiti virtual en la pantalla, con lo cual la pantalla se vuelve más grande.

  La mayoría de los dispositivos son fabricados por palmOne, quién ofrece las líneas Zire y Tungsten. La compañía se formó en 2003 cuando Palm Computing adquirió Handspring Inc. Sony.

Fig. 1.7 palmOne Tungsten T5 Handheld [17]

Fig. 1.8 palmOne Treo 650 [17]

- Pocket PC: Es el nombre genérico de los PDA Windows Mobile. Sus características estándar son:

    - Versiones de bolsillo de aplicaciones Microsoft como Microsoft Word, Excel y Outlook.
    - Sincronización con Microsoft Outlook en una PC Windows.
    - Tres aplicaciones de reconocimiento de escritura a mano: Transcriber, Letter Recognizer (similar a la nueva versión de Graffiti) y Block Recognizer (similar al Graffiti original).
    - Un área de escritura virtual, la cual maximiza el tamaño de la pantalla.

- Windows Media Player para reproducir contenido multimedia.

- ***Smart Phones:*** Un smart phone es un teléfono celular con capacidades PDA, o viceversa, o sea, un PDA con capacidades de teléfono celular. Algunas características de estos dispositivos son:

  - Se necesita un proveedor de servicios celular para proporcionar el servicio de telefonía.
  - Acceso a Internet a través de redes de datos celulares.
  - Variedad de sistemas operativos, incluyendo Windows Mobile Pocket PC Phone Edition, PalmOS, Blackberry OS para teléfonos Blackberry, y Symbian OS para smart phones Panasonic, Nokia, Samsung, y otros. [16]

I.6.5.1 Historia de los PDA

Los predecesores del PDA actual son Psion Organizar y Sharp Wizard. Estos dispositivos, los cuales se diseñaron para ser computadoras portátiles, surgieron a mediados de 1980. Incluían pequeños teclados, una pequeña pantalla, y funciones básicas como un reloj alarma, calendario, calculadora, etc. También soportaban software especializado como hojas de cálculo y juegos.

En 1993, Apple introdujo el Newton MessagePad que costaba $700 USD. Tenía un bloc de notas electrónico, lista de pendientes, calendario, registro de direcciones y teléfonos, etc. Algunas innovaciones del Newton se han convertido en funciones estándar de los PDA actuales, incluyendo el touch screen sensible a la presión con stylus, reconocimiento de escritura a mano, puerto infrarrojo y ranuras de expansión. Sin embargo, el Newton MessagePad era grande, complicado de usar y caro, por lo cual fue descontinuado en 1998.

El Palm Pilot original fue introducido en marzo de 1996 por Palm Computing. Costaba menos de $300 USD, utilizaba su propio sistema operativo Palm OS, cabía en un bolsillo, y era capaz de sincronizarse con una PC. Las baterías AAA duraban semanas, era fácil de utilizar, y podía guardar miles de contactos, notas y citas. En parte su pequeño tamaño se debía a la falta de un teclado. Los usuarios utilizaban un stylus y Graffiti para capturar caracteres.

En Noviembre de 1996, Microsoft introdujo Windows CE, su primer sistema operativo para dispositivos móviles. Fabricantes como HP, Compaq y Casio lo adoptaron para sus dispositivos apodados *Handheld.* [16]

I.6.5.2 Partes de un PDA

Microprocesador y memoria:

Al igual que una computadora desktop o laptop, un PDA es controlado por un microprocesador. El microprocesador es el cerebro del PDA, y coordina las funciones del PDA de acuerdo con instrucciones programadas. A diferencia de una laptop o desktop, un PDA utiliza microprocesadores más pequeños y baratos. A pesar de que estos procesadores tienden a ser más lentos, son adecuados para realizar las funciones del PDA. En la figura 1.9 se muestra la interrelación entre el microprocesador y las demás partes del PDA.

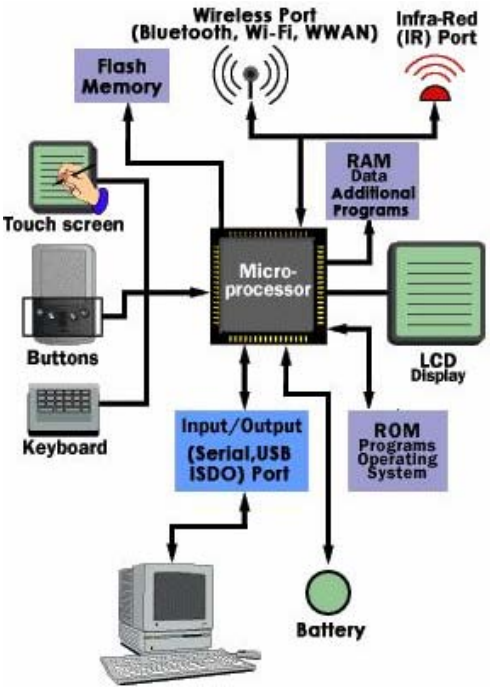

Fig. 1.9 Partes que constituyen un PDA. [17]

Los PDA no poseen disco duro. Almacena sus programas básicos, como libro de contactos, calendario, memos y sistema operativo, en un chip read-only memory (ROM). La información y programas que se añadan posteriormente se almacenan en random-access memory (RAM) del dispositivo. A pesar de que esta memoria es volátil, la información en RAM no se pierde al apagarse el PDA, debido a que la RAM se alimenta de pequeñas cargas provenientes de las baterías del PDA, incluso cuando el dispositivo está apagado.

Algunos PDA, como el Palm Tungsten E2, utilizan una memoria flash en lugar de memoria RAM. Debido a que la memoria flash no es volátil, no hay peligro de que se pierda la información almacenada en el PDA cuando las baterías se agoten. En la figura 1.10 se muestra la estructura interna de un PDA. A la izquierda se muestran los chips de memoria y el microprocesador, y a la derecha se muestra el touch screen.

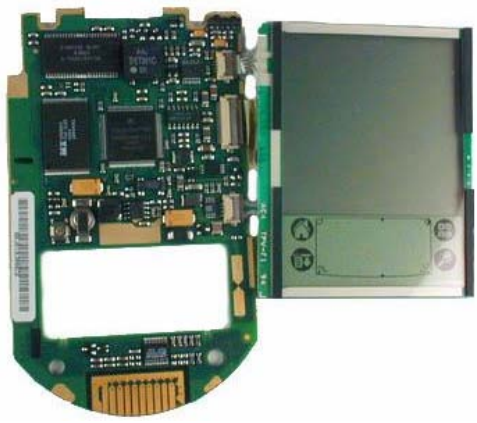

Fig. 1.10 Vista interna de un PDA. [17]

Sistema Operativo:

El sistema operativo contiene las instrucciones preprogramadas que le ordenan al microprocesador que tiene que hacer. Los sistemas operativos utilizados por los PDA no son tan complejos como los de una PC. Contienen menos instrucciones, lo cual requiere menos memoria. [17]

Touch Screen

Muchos de los PDA originales, como la Palm Pilot, utilizan touch screen para interaccionar con el usuario, y sólo tienen pocos botones reservados usualmente como atajos para acceder programas comunes. Estos PDA, incluyendo los dispositivos Windows Pocket PC, usualmente tienen un s*tylus* removible que es utilizado en el touch screen. Los botones y menús se activan al tocar la pantalla touch screen con el *stylus*, e incluso se puede arrastrar texto con el *stylus* para seleccionarlo y moverlo, etc.

Algunos PDA utilizan un *stylus* y *touch screen* en combinación con un programa de reconocimiento de escritura a mano. Utilizando un stylus de plástico, se dibujan caracteres sobre el touch screen del dispositivo, o en el área dedicada para escritura. El software dentro del PDA convierte los caracteres en letras y números. En los dispositivos Palm, el software de reconocimiento se llama Graffiti. Graffiti requiere que cada letra sea escrita de cierta forma, por lo cual se requiere utilizar un alfabeto especial. Por ejemplo, para capturar la letra “A”, se dibuja ”V” en el touch screen. Para que Graffiti sea más preciso, se dibujan las letras en una parte de la pantalla, y los números en otra parte de la pantalla.

Los PDA Pocket PC ofrecen tres programas de reconocimiento: Transcriber, Letter Recognizer y Block Recognizer. Letter Recognizer y Block Recognizer son similares a Graffiti ya que requieren alfabetos especiales. Transcriber reconoce la escritura a mano “regular”, o sea, escritura legible, similar al reconocimiento de escritura de una Tablet PC.

Algunos PDA, como el Blackberry y Treo, poseen un teclado completo y un *scroll wheel* o *thumb wheel* para facilitar el ingreso de datos y la navegación. Incluso existen teclados plegables de gran tamaño que pueden ser conectados directamente al PDA y que permiten escribir como si fuera un teclado de una PC Desktop.

I.6.5.3 Sincronización

Un aspecto importante de los PDA es la posibilidad de sincronizar automáticamente datos dentro de una base de datos de contactos, como Microsoft Outlook o ACT!, instalada en una computadora personal o en un servidor. Los datos sincronizados aseguran que el PDA posee una lista actualizada de contactos, citas e e-mail, permitiendo a los usuarios acceder la información dentro del servidor, a través del PDA. Así se evita la pérdida de datos almacenados en el dispositivo en caso de que éstos sean borrados, robados, o destruidos.

Otra ventaja de la sincronización es que, debido a que el ingreso de datos a través del touch screen es demasiado lento, mediante la sincronización este ingreso de datos se vuelve sumamente rápido. La mayoría de los PDA tienen ésta habilidad de sincronizarse con una PC. Esto se realiza a través de algún programa de sincronización instalado en el PDA, como por ejemplo, HotSynch Manager, el cual viene integrado en dispositivos Palm OS, o Microsoft ActiveSynch, el cual viene integrado dentro de dispositivos Windows Mobile. Estos programas le permiten al PDA sincronizarse con un administrador de información personal. Este administrador de información personal puede ser un programa exterior o un programa patentado. Por ejemplo, el PDA Blackberry incluye el programa Desktop Manager el cual se sincroniza con Microsoft Outlook y también con ACT!. Otros PDA sólo incluyen su propio software patentado. También existe software de sincronización desarrollado por compañías como Intellisync y CompanionLink. Este software sincroniza a los PDA con administradores de información personal no soportados por los fabricantes de PDA, como Goldmine o Lotus Notes.

I.6.5.4 Personalización

Como en un PC, es posible instalar software adicional en la mayoría de los PDA. Este software puede ser comprado o descargado a través de Internet, permitiendo a los usuarios personalizar sus PDA a su propio gusto. Algunos PDA también permiten agregar hardware. El hardware más común es una tarjeta de memoria, las cuales permiten a los usuarios tener espacio de almacenamiento intercambiable adicional para sus PDA. Los teclados plegables es otro hardware que puede ser añadido al PDA. Dispositivos PDA con Bluetooth pueden utilizar audífonos, ratones y teclados que funcionen a través de Bluetooth. [16]

I.7 Wi-Fi

Wi-Fi es una marca originalmente autorizada por Wi-Fi Alliance, para describir la tecnología de redes de área local inalámbricas (WLAN), basada en la especificación IEEE 802.11.

Fig. 1.11 Logo oficial de la tecnología inalámbrica Wi-Fi. [15]

Wi-Fi (Fig. 1.11) fue creada para ser utilizada en dispositivos de cómputo portátiles, como laptops, pero ahora es utilizada en más aplicaciones, como acceso a Internet, juegos, y conectividad básica para componentes electrónicos domésticos como TV's y reproductores de DVD.

Una persona con un dispositivo Wi-Fi, como por ejemplo una computadora, teléfono o PDA, puede conectarse a Internet cuando se encuentra en proximidad con un *access point*. La región cubierta por uno o varios access points se denomina *hotspot.* Los hotspots pueden abarcar desde una habitación hasta kilómetros cuadrados. Wi-Fi permite conectividad de forma peer-to-peer, la cual permite que varios dispositivos se conecten directamente entre sí. Wi-Fi utiliza tecnología de radio *Single Carrier Direct-Sequence Spread Spectrum*, y *Multi Carrier OFDM (Orthogonal Frecuency Division Multiplexing).*

Historia de Wi-Fi

Wi-Fi fue inventada en 1991 por NCR Corporation/AT&T en Nieuwegein, Holanda. Fue originalmente creada para cajeros automáticos; los primeros productos inalámbricos fueron introducidos con el nombre WaveLAN con velocidades de 1Mbit/s a 2Mbit/s. Vic Hayes, quién fue el inventor primario de Wi-Fi, estuvo involucrado en el diseño de los estándares IEEE 802.11a, 802.11b y 802.11g.

La palabra Wi-Fi fue inventada por Interbrand, empresa contratada por The Wireless Ethernet Compatibility Alliance (luego Wi-Fi Alliance) para idear un nombre atractivo para sus productos, y así no tener que utilizar el nombre *IEEE 802.11b Direct Sequence.* Cabe mencionar que aunque se suele describir a Wi-Fi como *Wireless Fidelity* debido a su similitud con Hi-Fi (High Fidelity), este término es erróneo. Wi-Fi es simplemente una marca para describir productos WLAN basados en IEEE 802.11b.

Funcionamiento de Wi-Fi

El típico sistema Wi-Fi contiene uno o más access points y uno o más clientes. Un access point (AP) transmite su SSID (Service Set Identifier) a través de packets que son llamados *beacons*, los cuales son transmitidos cada 100 ms. Los *beacons* son transmitidos a 1Mbit/s, con lo cual se asegura que el cliente que recibe el beacon puede comunicarse al menos a 1Mbit/s. Basados en esta configuración, el cliente decidirá si se conecta al AP. Además, el *firmware* corriendo en la tarjeta Wi-FI del cliente también influye. Por ejemplo, si dos AP con el mismo SSID se encuentran dentro del rango del cliente, el *firmware* decidirá a cual de los dos AP conectarse basado en la potencia de la señal. Esto le da al cliente un criterio de conexión y *roaming* totalmente abierto. Esto tiene la desventaja de que un adaptador inalámbrico a veces se desempeñe mejor que otro. Debido a que Wi-Fi se transmite a través del aire, posee las mismas propiedades de una red Ethernet sin switch. Por lo tanto, coaliciones similares a las de una red LAN sin switch pueden aparecer en una red Wi-Fi.

Fig. 1.12 Un Access Point (en el centro), permite a varios dispositivos conectarse a Internet al mismo tiempo. [14]

Canales

Excepto por 802.11a, la cual opera a 5 GHz, Wi-Fi utiliza un espectro cercano a los 2.4 GHz, el cual está estandarizado y sin licencia por acuerdo internacional, aunque la alocación exacta del espectro varía ligeramente en el mundo, al igual que la potencia máxima permitida. Sin embargo, el número de canales están estandarizados por frecuencias en el mundo, así las frecuencias autorizadas pueden ser identificadas por números de canales. La siguiente tabla muestra los diferentes canales con sus respectivas frecuencias, así como su disponibilidad en diferentes partes del mundo.

Canales Wi-Fi

Canal	Frecuencia Central (GHz)	USA y Canadá	Europa (ETSI)	Japón
1	2.412	Sí	Sí	Sí
2	2.417	Sí	Sí	Sí
3	2.422	Sí	Sí	Sí
4	2.427	Sí	Sí	Sí
5	2.432	Sí	Sí	Sí
6	2.437	Sí	Sí	Sí
7	2.442	Sí	Sí	Sí
8	2.447	Sí	Sí	Sí
9	2.452	Sí	Sí	Sí
10	2.457	Sí	Sí	Sí
11	2.462	Sí	Sí	Sí
12	2.467		Sí	Sí
13	2.472		Sí	Sí
14	2.484			Sí

Características:

- La potencia máxima permitida varía con la región.
- El distanciamiento entre canales es de 0.005 GHz, excepto por el canal 14.
- Cada canal se traslapa con su vecino. La cantidad de interferencia disminuye entre más separados estén entre ellos.
- La mayoría de las interferencias es con los dos canales adyacentes en cada dirección (arriba y abajo). Por ejemplo, el canal 6 interfiere en su mayoría con los canales 4 y 5, 7 y 8.
- Existe interferencia significativa con dos canales más en cada dirección. Por ejemplo, el canal 6 interfiere significativamente con los canales 2, 3,9 y 10.
- Puede existir algo de interferencia más allá de cuatro canales en cada dirección, particularmente con transmisores poderosos. Por ejemplo, el canal 6 puede interferir con los canales 1 y 11.
- Como regla, debe existir mínima interferencia entre canales con más de 5 números de distancia, excepto el canal 14, el cual tiene interferencia mínima con los demás canales.
- En los Estados Unidos:

    - Existen tres canales de interferencia mínima: 1, 6,11.
    - Para cuatro canales con algo de interferencia que sigan siendo utilizables, los canales pueden estar tres o cuatro números separados:

Opción #1	Opción #2	Opción #3	
1, 4, 7, 11	1, 4, 8, 11	1, 5, 8, 11	[15]

1.8 Puerto Paralelo

El puerto paralelo es una interfaz de sistema de cómputo donde los datos son transferidos hacia adentro y afuera de éste de forma paralela, es decir, a través de más de un alambre. Un puerto paralelo transporta un bit a través de cada alambre multiplicando así la tasa de transferencia disponible a través de un solo cable. Hay también muchos otros alambres extra en el puerto que son utilizados para señales de estado y de control que indican cuando los datos han sido recibidos o enviados, inicializar un reset, indicar una condición de error (como falta de papel), etc. Actualmente, la interfaz USB ha reemplazado al puerto paralelo, ya que la mayoría de las impresoras modernas son conectadas a través de un puerto USB. En muchas de las computadoras modernas, el puerto paralelo no se incluye con el propósito de ahorrar costos de producción, y se le considera como un puerto en parte obsoleto.

Usos:

Los puertos paralelos son frecuentemente utilizados por los microprocesadores para comunicarse con los periféricos. El más común de los puertos paralelos es el puerto de la impresora, como por ejemplo el puerto Centronics que transfiere 8 bits simultáneamente. Los discos son también conectados vía puertos paralelos especiales, como por ejemplo SCSI o ATA.

Antes de que las conexiones USB estuvieran disponibles de manera masiva, muchos dispositivos externos, como unidades de discos portátiles para Windows y MS-DOS, utilizaba un conector de paso que les permitía compartir el puerto paralelo con la impresora. Esto se debía a que los sistemas Windows de esa época no poseían conexiones equivalentes a SCSI, la única conexión conveniente era usualmente el puerto de la impresora. El puerto paralelo IBM-PC compatible, es por mucho, el puerto de computadora estándar más común. Utiliza voltajes lógicos estándar, directamente de la computadora hacia un conjunto de pines. El voltaje estándar lógico de 5 volts DC, es virtualmente inocuo.

Conectores:

Comúnmente, los conectores del puerto paralelo tienen al menos 25 pines, como por ejemplo el DB25 de 25 pines (Fig. 1.13). La mayoría de ellos son utilizados, lo cual genera que los cables sean anchos. Estos cables además están limitados en longitud a un máximo de 3 a 8 metros, dependiendo del puerto específico y características del cable. A pesar de que varios estándares de puerto paralelo existen, no son siempre utilizados, especialmente en dispositivos viejos, por lo cual a veces es algo complicado encontrar el cable apropiado o el driver apropiado.

Fig. 1.13 Conector DB25 en la parte posterior de la PC. [12]

El puerto paralelo tiene cuatro tipos de pines:

- Pines de datos: Usualmente 8, a veces 16, y algunas veces con un pin extra para bit de paridad. Pueden ser unidireccionales o bidireccionales.
- Pines de control: Utilizados para envío de señales de control, como por ejemplo STROBE, que indica que el pin de datos está listo, o R/W que especifica si los puertos bidireccionales leen o escriben datos.
- Pines de estados: Utilizados para enviar señales de estado, como BUSY para indicar que el dispositivo no se encuentra listo para recibir datos, o ACK para notificar que se recibió satisfactoriamente un símbolo.
- Pines de tierra (GND): Son utilizar para cerrar el circuito con otros pines.

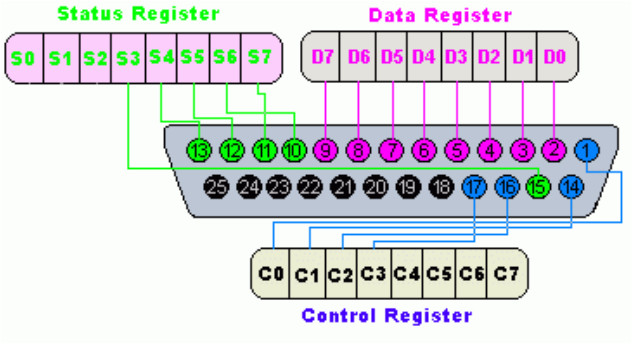

Fig. 1.14 Localización de los pines de datos, de estado y de control en el puerto paralelo. [4]

Pin No	Señal	Dirección	Register - bit	Invertido
1	nStrobe	Out	Control-0	Sí
2	Data0	In/Out	Data-0	No
3	Data1	In/Out	Data-1	No
4	Data2	In/Out	Data-2	No
5	Data3	In/Out	Data-3	No
6	Data4	In/Out	Data-4	No
7	Data5	In/Out	Data-5	No
8	Data6	In/Out	Data-6	No
9	Data7	In/Out	Data-7	No
10	nAck	In	Status-6	No
11	Busy	In	Status-7	Sí
12	Paper Out	In	Status-5	No
13	Select	In	Status-4	No
14	Line Feed	Out	Control-1	Sí

15	nError	In	Status-3	No
16	nInitialize	Out	Control-2	No
17	nSelect-Printer	Out	Control-3	Sí
18-25	Ground	-	-	-

Como sus nombres lo especifican, los registros de datos están conectados a las líneas de datos, los registros de control a las líneas de control y los registros de estados a las líneas de estados. Cuando se escribe en alguno de estos registros, cambios de voltajes aparecen en las líneas correspondientes. Por ejemplo, si se escribe un “1” en el registro de datos, en la línea de datos Data0 se mandará un voltaje de +5V.

En una PC IBM, estos registros son mapeados como I/O y tienen una única dirección. En una PC típica, la dirección base de LPT1 es 0x378 y para LPT2 es 0x278. Los registros de datos residen en su dirección base, los registros de estados en la dirección base + 1, y los registros de control en la dirección base + 2. [12]

Registro	**LPT1**	**LPT2**
Registro de datos (dirección base + 0)	0x378	0x278
Registro de estado (dirección base + 1)	0x379	0x279
Registro de control (dirección base + 2)	0x37a	0x27a

Historia:

Los puertos paralelos fueron originalmente desarrollados por IBM como una manera de conectar una impresora a una PC. Cuando IBM se encontraba en el proceso de diseñar su PC, la compañía quería que la computadora trabajara con impresoras de Centronics, uno de los más importantes fabricantes de impresoras en ese tiempo. Sin embargo, IBM después decidió no utilizar la misma interfaz de puerto que Centronics utilizaba en sus impresoras.

DB 25		Centronics 36	
Pin	Signal	Pin	Signal
1	Strobe	1	Strobe
2	data0	2	data0
3	data1	3	data1
4	data2	4	data2
5	data3	5	data3
6	data4	6	data4
7	data5	7	data5
8	data6	8	data6
9	data7	9	data7
10	Acknowledge	10	Acknowledge
11	Busy	11	Busy
12	Paper End	12	Paper End
13	Select	13	Select
14	Auto Feed	14	Auto Feed
15	Error	15	Error
16	Init	16	Init
17	Select In	17	Select In
18	GND	18	GND
19	GND	19	GND
20	GND	20	GND
21	GND	21	GND
22	GND	22	GND
23	GND	23	GND
24	GND	24	GND
25	GND	25	GND
		26	GND
		27	GND
		28	GND
		29	GND
		30	GND
		31	Init
		32	Error
		33	Ground
		34	NC
		35	NC
		36	Select In

Fig. 1.15 Diferencia de pines entre conectores DB25 y Centronics. [12]

En vez de esto, los ingenieros en IBM emparejaron un conector DB-25 (Fig. 1.16) con 25 pines, con un conector Centronics de 36 pines para crear un cable especial que conectara la impresora con la computadora. Otros fabricantes de impresoras terminaron adoptando la interfaz Centronics, volviendo a este extraño cable un improbable estándar *de facto*.

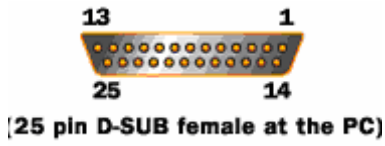

Fig. 1.16 Conector DB25 hembra [12]

Cuando una PC envía datos a la impresora u otro dispositivo utilizando el puerto paralelo, envía 8 bits de datos (1 Byte) al mismo tiempo. Estos 8 bits son transmitidos de manera paralela, opuesto a los 8 bits trasmitidos de manera serial (todos en una sola fila) a través del puerto serie. El puerto paralelo estándar es capaz de enviar de 50 a 100 kilobytes de datos por segundo.

A continuación se describe lo que cada pin hace:

- El pin 1 transmite la señal STROBE. Ésta mantiene un nivel de entre 2.8 y 5 volts, pero este voltaje cae por debajo de los 0.5 volts cuando la computadora envía un byte de información. Esta caída de voltaje le indica a la impresora que los datos están siendo enviados.
- Los pines 2 al 9 son utilizados para transmitir datos. Para indicar que un bit tiene un valor de 1, una carga de 5 volts es enviada a través del pin correcto. Ninguna carga en el pin indica un valor de 0. Esto es una simple pero sumamente efectiva forma de transmitir información digital a través de un cable analógico en tiempo real.
- El pin 10 envía la señal acknowledge de la impresora a la computadora. Como en el pin 1, mantiene una carga de entre 2.8 y 5 volts, y baja el voltaje por debajo de 0.5 volts para indicarle a la computadora cómo se recibieron los datos.
- Si la impresora está ocupada, enviará una carga al pin 11. Posteriormente, bajará el voltaje por debajo de 0.5 volts para indicarle a la computadora que está lista para recibir más datos.
- La impresora le indica a la computadora si no tiene papel, al enviar una carga por el pin 12.
- Mientras que la computadora reciba una carga a través del pin 13, sabe que el dispositivo está en línea.
- La computadora envía una señal *autofeed* a la impresora a través del pin 14 utilizando una carga de 5 volts.
- Si la impresora tiene algún problema, baja el voltaje a menos de 0.5 volts en el pin 15 para indicarle a la computadora que hay un error.
- Siempre que un nuevo trabajo de impresión esté listo, la computadora baja el voltaje en el pin 16 para inicializar la impresora.
- El pin 17 es utilizado por la computadora para apagar remotamente la impresora. Esto se logra al enviar una carga a la impresora y mantenerla todo el tiempo que se deseé que la impresora esté apagada.
- Los pines 18 a 25 son tierras y son utilizadas como una señal de referencia para la carga baja (debajo de los 0.5 volts).

Fig. 1.17 Conectores DB25 tipo hembra y tipo macho. [12]

SPP/EPP/ECP:

La especificación original para puertos paralelos era unidireccional, esto significa que los datos sólo viajan en una sola dirección en cada pin. Con la introducción de PS/2 en 1997, IBM ofreció un nuevo diseño de puerto paralelo bidireccional. Este modo es comúnmente

conocido como Standard Parallel Port (SSP) y ha remplazado completamente el diseño original. Las comunicaciones bidireccionales permiten a cada dispositivo recibir datos además de enviarlos. Muchos dispositivos utilizan los pines 2 al 9, originalmente designados para datos. Utilizar los 8 pines limita las comunicaciones a half duplex, lo que significa que la información sólo puede viajar en una sola dirección al mismo tiempo. Pero los pines 18 al 25, originalmente sólo utilizados como tierra, pueden ser también utilizados como pines de datos. Esto permite una comunicación full duplex (ambas direcciones al mismo tiempo).

Enhanced Parallel Port (EPP) fue creado por Intel, Xircom y Zenith en 1991. EPP permite que más datos sean transmitidos, de 500 kilobytes a 2 megabytes cada segundo. Fue diseñado específicamente para dispositivos que no fueran impresoras y que se conectaran al puerto paralelo, particularmente dispositivos de almacenamiento que necesitaban la máxima velocidad de transferencia posible.

EPP

Pin	EPP Signal	Pin	EPP Signal	Pin	EPP Signal
1	Write	10	Interrupt	19	Ground
2	Data 0	11	Wait	20	Ground
3	Data 1	12	Spare	21	Ground
4	Data 2	13	Spare	22	Ground
5	Data 3	14	Data Strobe	23	Ground
6	Data 4	15	Spare	24	Ground
7	Data 5	16	Reset	25	Ground
8	Data 6	17	Address Strobe		
9	Data 7	18	Ground		

Fig. 1.18 Configuración de pines de EPP. [12]

Antes de la introducción de EPP; Microsoft y Hewlett Packard conjuntamente anunciaron una especificación llamada Extended Capabilities Port (ECP) en 1992. Mientras que EPP fue diseñado para dispositivos que no fueran impresoras, ECP fue diseñado para mejorar la velocidad y funcionalidad de impresoras.

ECP

Pin	ECP Signal	Pin	ECP Signal	Pin	ECP Signal
1	HostCLK	10	PeriphCLK	19	Ground
2	Data 0	11	PeriphAck	20	Ground
3	Data 1	12	nAckReverse	21	Ground
4	Data 2	13	X-Flag	22	Ground
5	Data 3	14	Host Ack	23	Ground
6	Data 4	15	PeriphRequest	24	Ground
7	Data 5	16	nReverseRequest	25	Ground
8	Data 6	17	1284 Active		
9	Data 7	18	Ground		

Fig. 1.19 Configuración de pines de ECP [12]

En 1994, el estándar IEEE 1284 fue liberado. Incluía dos especificaciones para dispositivos de puerto paralelo, EPP y ECP. Para que pudieran operar correctamente, el sistema

operativo y el dispositivo debían soportar la especificación requerida. En la actualidad, esto es a veces un problema ya que la mayoría de las computadoras soportan SPP, ECP y EPP y detectan automáticamente cuál modo se necesita utilizar, dependiendo del dispositivo colocado. En la mayoría de las computadoras, para seleccionar manualmente alguno de estos modos, se necesita hacerlo vía el BIOS. [12]

I.9 Laboratorio Microsoft Research

El Laboratorio Microsoft de la Facultad de Ingeniería fue creado en el año 2002 con motivo de la donación de un servidor, ocho PC's, Windows 2000 Server y Visual Studio .NET por el corporativo Microsoft.

OBJETIVOS

El Laboratorio tiene por objetivos:

1.- Exponer a los estudiantes y académicos de la Facultad de Ingeniería las últimas tecnologías Microsoft disponibles, incorporando su aplicación en asignaturas relacionadas con las carreras de la DIE.

2.- Desarrollar investigación.

3.- Desarrollar aplicaciones.

4-. Formar recursos humanos, Tesis y Servicio Social.

5.- Organizar cursos intersemestrales sobre tecnologías Microsoft.

Responsable: M. en I. Jorge Valeriano Assem.

RED WIRELESS LABORATORIO MICROSOFT

Dispositivos de Interconexión de la red inalámbrica

- 1 Wireless Access Point marca 3COM, soporta los estándares 802.11a, b y g.
- 1 Switch 3COM OfficeConnect de 16 puertos
- IP Real: 132.248.59.84
- 2 Servidores: HP ProLiant ML350 y Compaq ProLiant ML530
- 10 HP Workstation xw4200 con monitor plano de 17", 1 GB RAM, tarjeta de video ATI FireGL 3100 128MB
- 6 computadoras de escritorio Compaq, 512 MB RAM
- 5 computadoras de escritorio HP modelo d330, 512 MB RAM
- 6 Tablet PC HP/Compaq, 256MB y 512MB RAM

Diseño

Descripción: La red WLAN instalada en el laboratorio de Microsoft (Fig. 1.20) da servicio a 21 nodos de red asignándoles direcciones dinámicamente (soporta hasta 38 nodos). La máquina que proporciona el servicio de salida a Internet es un servidor HP ProLiant ML350 con procesador Intel Xeon a 3 GHz, 1.5 GB de RAM, trabaja bajo la plataforma Windows 2003 Standard Edition, además para el control de la red interna, se tiene un segundo servidor Compaq ProLiant ML530 con procesador Intel Pentium III Xeon a 1 GHz, 1GB de RAM, trabaja bajo la plataforma Windows 2003 Standard Edition.

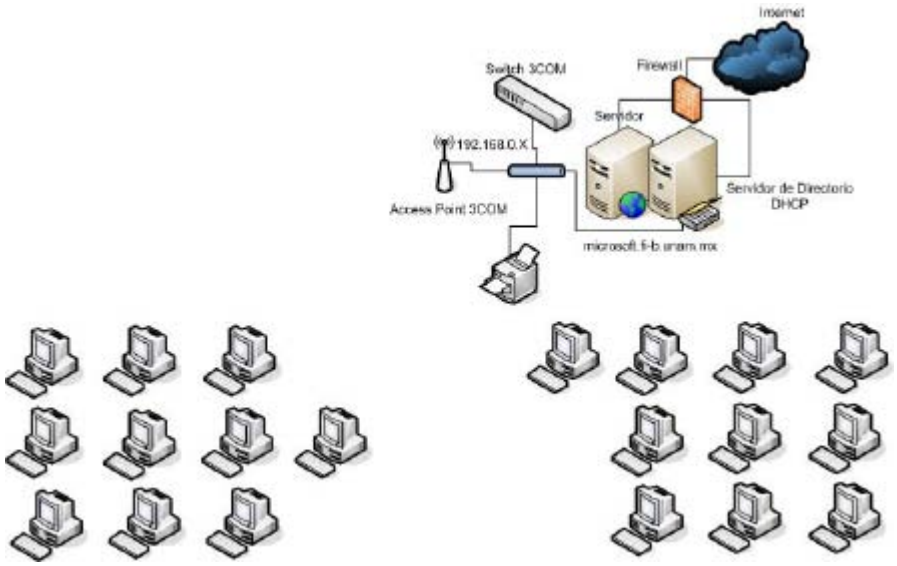

Fig. 1.20 Estructura de la red WLAN del laboratorio Microsoft Research [18]

La red WLAN soporta los protocolos 802.11a, 802.11b y 802.11g con velocidad máxima de 54Mbps, tiene un alcance de 60 m. sin obstáculos. El área real de cobertura es el segundo piso del edificio Valdés Vallejo (Área de Laboratorios)

Como medidas de seguridad en cuanto a la conexión a Internet se tiene instalado el Servidor de Seguridad ISA Server 2004, en cuanto a la red interna se tiene filtrado de MAC, y como segunda medida las transferencias son encriptadas mediante el protocolo WEP. [18]

CAPITULO II. PROPUESTA DE SISTEMA DE SEGURIDAD

II.1. Sistema de monitoreo con webcam

II.1.1 Página principal.

Este sistema consiste de una página ASP.NET en la cual se mostrará cada determinado tiempo una imagen en tiempo real del Laboratorio Microsoft Research. Esta imagen proviene de una webcam instalada en el servidor que hospeda la página ASP.NET.

Al inicio de la aplicación Web, se muestra una página llamada Login.aspx (Fig.2.1), la cual sirve para autenticar a los administradores que quieran acceder al Sistema de Seguridad, mediante un login y password, los cuales son enviados a través Internet de manera segura utilizando un algoritmo de encriptación Hash SHA1. Estos login y passwords son encriptados y almacenados dentro de una base de datos llamada ASPNETDB.MDF, dentro de la carpeta C:\Inetpub\wwwroot\SegLab\App_Data.

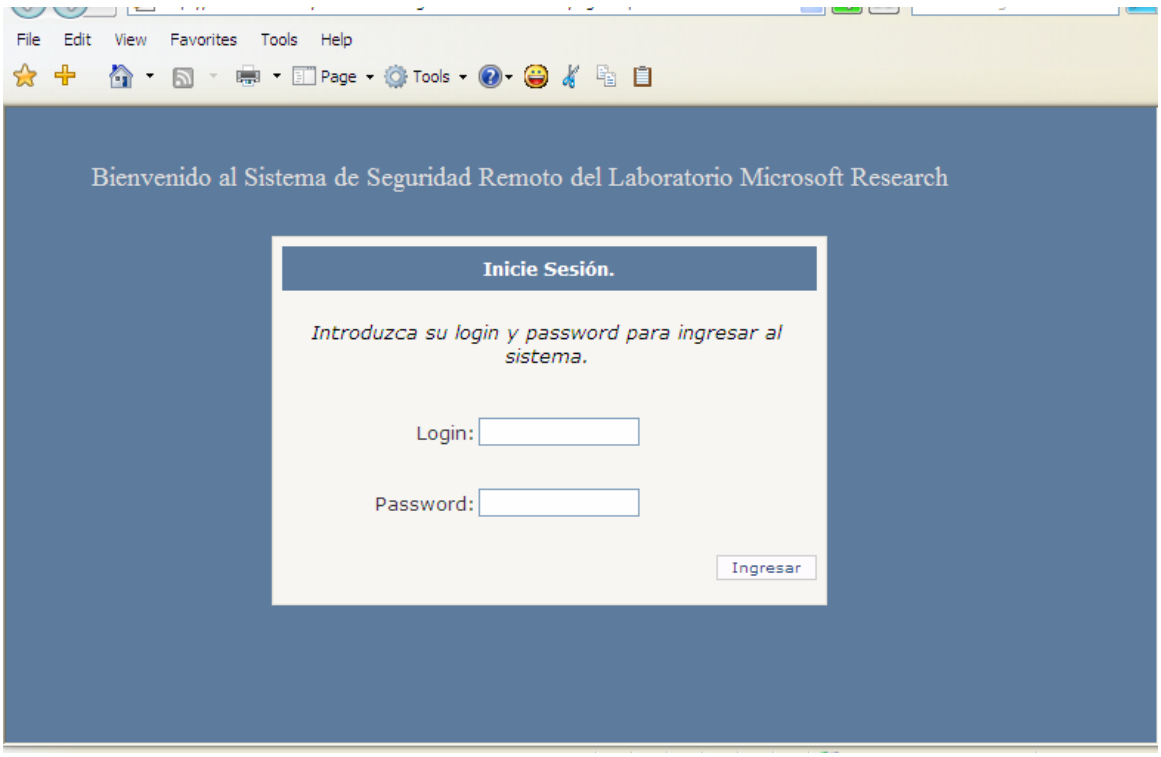

Fig. 2.1 Página Login.aspx, la cual sirve para autenticar a los administradores que quieran entrar al Sistema de Seguridad.

Para poder registrar a los usuarios autorizados, Visual Studio 2005 posee una herramienta llamada ASP.NET Configuration Tool (Fig. 2.3), la cual nos permite de manera fácil, crear usuarios y roles, como administrador o usuario básico, crear sus logins y passwords, etc.

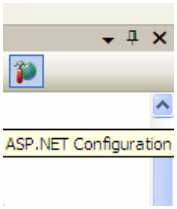

Fig. 2.2 Botón de ASP.NET Configuration Tool,
dentro de la ventana Solution Explorer

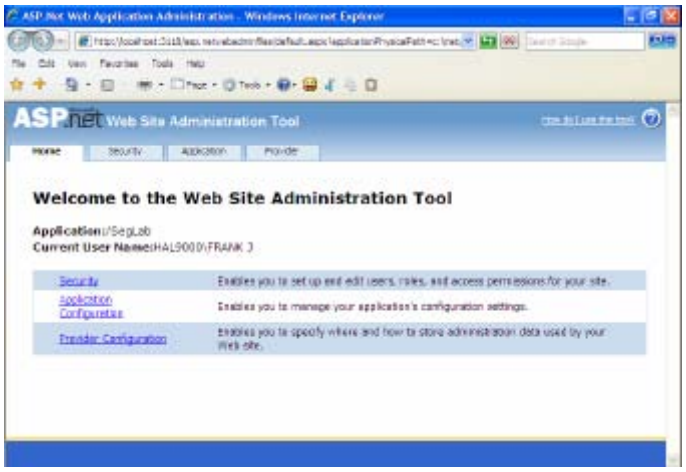

Fig. 2.3 Página principal de ASP.NET Configuration Tool. En ella se
puede configurar la aplicación Web, crear usuarios y administrar
la base de datos de manera fácil.

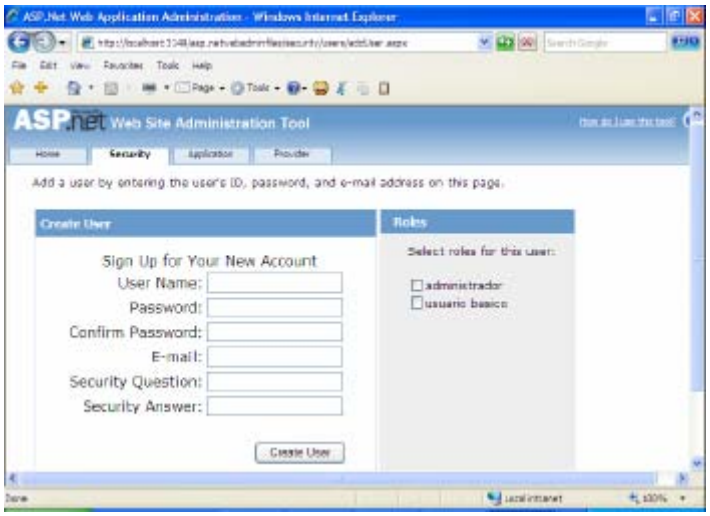

Fig. 2.4 Página de ASP.NET Configuration Tool donde se pueden crear
cuentas de usuario. Los datos son automáticamente encriptados.

Regresando a la página Login.aspx (Fig. 2.1), en caso de que no se ingrese el login o password correcto, se vuelve a mostrar esta misma página hasta que se introduzcan los datos correctos. Si el login y password son correctos, se muestra la página Default.aspx (Fig. 2.5), en la cual se encuentran los botones que controlan las cerraduras de las puertas del laboratorio, y donde también se muestra la imagen de la webcam que nos sirve para observar lo que sucede dentro del laboratorio.

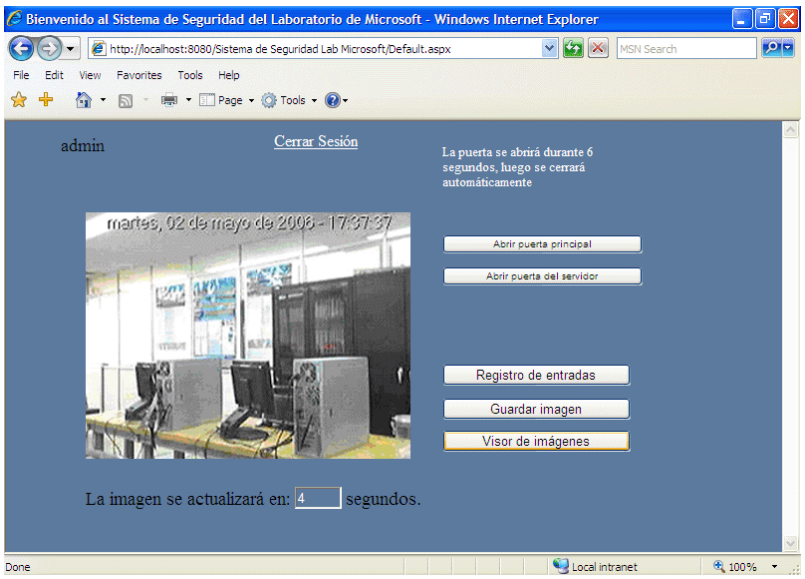

Fig. 2.5 Default.aspx, página donde se muestra la imagen de la webcam y botones para controlar cerraduras.

Para obtener la imagen de la webcam y mostrarla en la página, se hace uso de CamServer.dll, un archivo obtenido de manera gratuita en Internet [13] el cual permite interactuar con el driver de la webcam para que esta tome una imagen. Se registró CamServer.dll en el sistema operativo mediante la línea de comandos del sistema, escribiendo regsvr32 Camserver.dll, como se muestra en la figura 2.6.

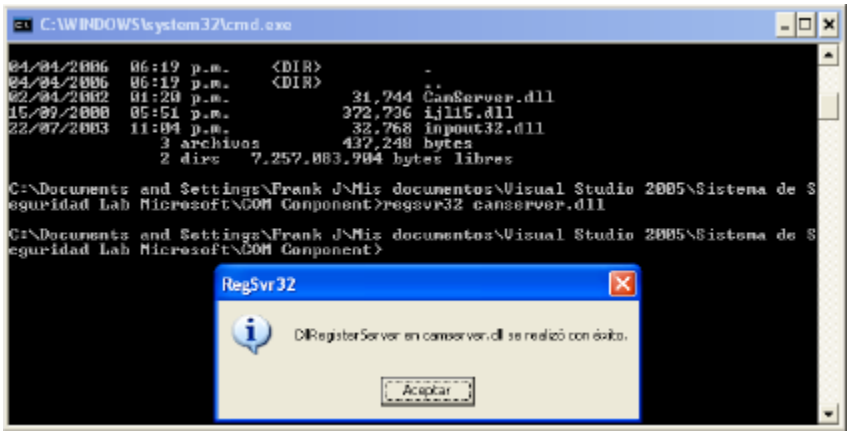

Fig. 2.6 Mediante RegSvr 32 se registran archivos .dll en el sistema

Posteriormente, dentro de Visual Studio 2005 se importó este archivo .dll al proyecto mediante *Website-> Add Reference ->Com -> CamServer 1.0 Type Library*

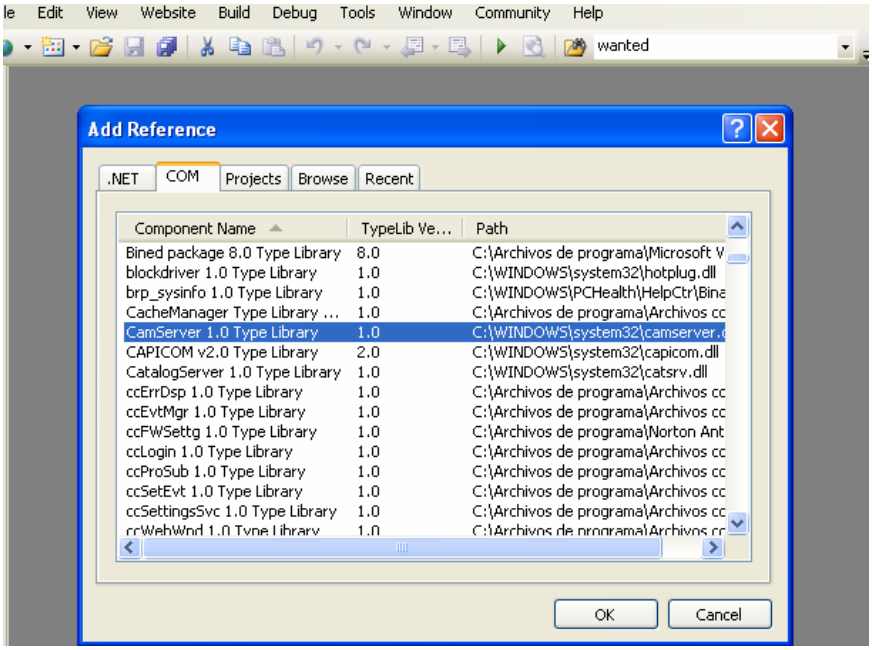

Fig. 2.7 Al haberse registrado CamServer.dll en el sistema, dentro de Visual Studio 2005, CamServer aparece como componente COM

Luego se agregó una página Web llamada Webcam.aspx al proyecto. Básicamente lo que esta página hará es mandar la imagen capturada por la webcam a un control HTML Image, cuyo img src será la página .aspx creada anteriormente, algo parecido a lo siguiente: .

Mediante `CAMSERVERLib.Camera cam = new CAMSERVERLib.CameraClass();` se crea un objeto de tipo `CAMSERVERLib.CameraClass();`, el cual le pide al driver de la webcam que tome una imagen. Con `MemoryStream ms = new MemoryStream( picture );` y `Bitmap bmp = new Bitmap( ms );`se asigna al objeto `ms` lo que se encuentra en memoria (en este caso la imagen tomada por la webcam) y después se crea el objeto bmp, el cual contendrá a la imagen tomada por la webcam. Con `g.DrawString` se le agrega a la parte superior de la imagen la fecha y hora en que la imagen fue tomada. Por último `bmp.Save( Response.OutputStream, icf[1], encps );` guarda en memoria la imagen, con la fecha y hora incluida, para después ser referenciada por

La página donde se muestra la imagen de la webcam (Fig. 2.5), utiliza un control HTML Image, el cual el URL de la imagen que mostrará será el de Webcam.aspx.

```
    <img src="Webcam.aspx" name="webCamImage" alt="web cam" title="web
cam" width="320" height="240" border="0" style="z-index: 110; left: 91px;
position: absolute; top: 59px">
```

Se incluye el siguiente JavaScript para permitir que esta imagen se recargue cada determinado tiempo, en este caso 5 segundos:

```
  <script language="JavaScript" type="text/javascript">
<!--
var camImage = 'Webcam.aspx';
var refreshIntervalSeconds = 5;

var secondsLeft = refreshIntervalSeconds;

function startClock() {
  if (secondsLeft > 0) {
    if (secondsLeft < 5) {
      document.webCam.timer.value = secondsLeft;
    } else {
      document.webCam.timer.value = secondsLeft;
    }
    secondsLeft = secondsLeft - 1
    timerID = setTimeout('startClock()', 1000);
  } else {
  date = new Date();
  imageNumber = date.getTime();
  document.webCamImage.src = camImage + '?' + imageNumber;
  secondsLeft = refreshIntervalSeconds;
  startClock();
  }
}
// -->
</script>
```

También se incluye un cuadro de texto en la página que indicará el tiempo que falta para que la imagen se recargue, de la siguiente forma:

```
    <form name="webCam" action="#">
      <span style="font-size: 14pt"><br />
         La imagen se actualizará en: </span>
        <input type="text" name="timer" size="2" style="background-
color: #5d7b9d; color: white;">
        <span style="font-size: 14pt">segundos. 
        </span>
</form>
```

II.1.2 Registro de entradas

Dentro de la página Default.aspx, existe un botón llamado Registro de entradas .Tal botón al ser presionado abre una ventana emergente llamada Bitacora.aspx (Fig. 2.8) en donde se muestran dentro de una tabla diversos datos que les permitirán a los administradores del

laboratorio tener un registro de todos los usuarios que han accedido al Sistema de Seguridad.

En dicha tabla, se muestran los siguientes datos:

- Login del usuario que accedió la página.
- Hora de inicio de sesión.
- IP de la máquina del usuario.
- Nombre de la máquina del usuario.
- Hora de salida del usuario.

Estos datos permitirán a los administradores del laboratorio, detectar comportamientos inusuales en el sistema, que un usuario entre en un horario fuera de lo normal, como por ejemplo, en la madrugada, lo cual probablemente significaría que alguna persona malintencionada trata de ingresar a la página de manera ilegal.

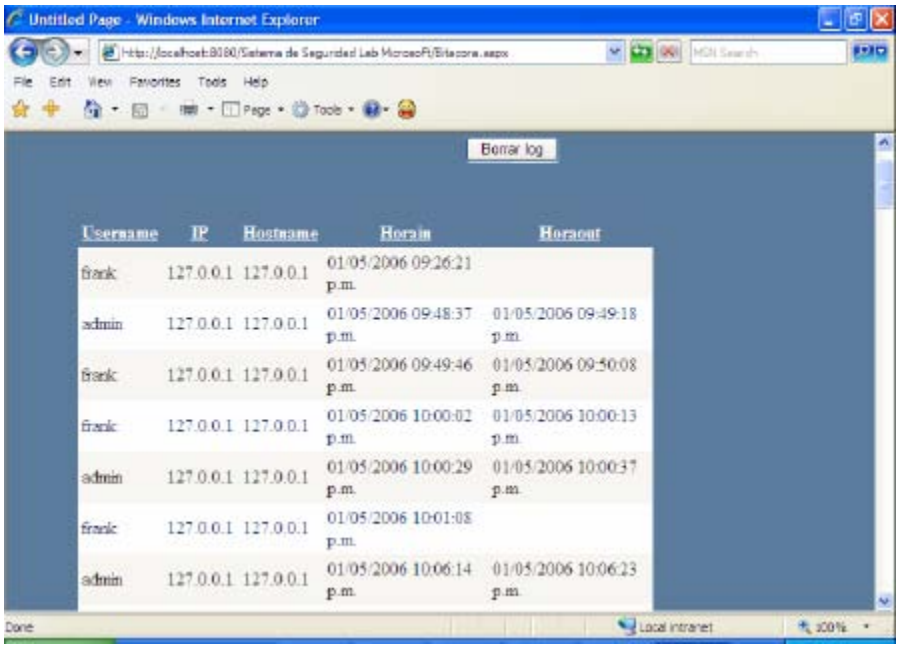

Fig. 2.8 Página Bitacora.aspx, donde se muestra el registro de visitas.

Para obtener estos datos, se crearon variables de sesión, las cuales mantienen su valor a lo largo de la sesión del usuario. En el momento en que el usuario cierra sesión, estas variables son borradas. A estas variables de sesión se les asignaron las variables `User.Identity.Name` y `Server.HtmlEncode` dentro del namespace System.Diagnostics de .NET Framework. Estas variables almacenan la IP, hora de inicio de sesión, hora de cierre de sesión, etc.

```
Session["usuario"] = User.Identity.Name.ToString();

Session["horain"]=Server.HtmlEncode(DateTime.Now.ToString());
Session["ip"] = Server.HtmlEncode(Request.UserHostAddress);
```

```
Session["host"] = Server.HtmlEncode(Request.UserHostName);
Session["horaout"] = "";
```

Estos datos se guardan en una base de datos SQL Server llamada Logs. El ingreso y recuperación de estos datos desde y hacia la base de datos se realiza fácilmente mediante un control ASP.NET 2.0 llamado SqlDataSource. Insertando este control en la página Default.aspx (Fig. 2.5), lo único que hay que hacer es, por ejemplo, para insertar en la base de datos, escribir `SqlDataSource1.Insert();` ASP.NET se encarga de realizar la conexión con SQL Server.

Al control `SqlDataSource1` sólo hay que modificarle los queries de Insert, Update y Delete mediante las propiedades de dicho control. En el query Update se ingresó:

```
UPDATE Logs SET Horaout = @horaout WHERE (Username = @user) AND (Horain = @horain)
```

En el query Insert:

```
INSERT INTO Logs(Username, IP, Hostname, Horain, Horaout) VALUES ( @user , @ip, @host, @horain, @horaout)
```

Y en Delete:

```
DELETE FROM Logs
```

Por ultimo, se cuenta con un botón (borrar log) para borrar los registros en la base de datos, con el fin de evitar que la base de datos se vuelva muy grande al tener demasiados registros.

II.1.3 Guardar Imágenes.

Dentro de la página principal (Fig.2.5), se encuentra un botón llamado *Guardar imagen,* el cual sirve para que el administrador guarde la imagen más reciente mostrada por la webcam en el disco duro de la máquina servidor, en caso de que detecte algo extraño. Por ejemplo, si el administrador ingresa a la página principal del Sistema de Seguridad Remoto (Fig.2.5), dentro de un horario en el que sabe que no hay actividades, como por ejemplo domingo, podrá ver dentro de la página las imágenes del laboratorio en tiempo real a través de la webcam. En caso de que el administrador observe imágenes sospechosas, como por ejemplo, una persona moviéndose dentro del laboratorio, el administrador tendrá la oportunidad de grabar en disco duro las imágenes de esta persona, como evidencia para ser mostrada cuando se reporte este suceso a las autoridades de la facultad.

Las imágenes serán guardadas en la carpeta `C:\ImagenesCapturadas` de la máquina servidor, mediante el siguiente código:

```
bmp.Save(@"C:\ImagenesCapturadas\imagen" + hora, icf[1], encps);
```

El nombre que se le asigna a la imagen proviene de tomar la hora del sistema. Dentro de la imagen, se agrega la fecha y hora en que ésta fue tomada. En la figura 2.9 se muestra un ejemplo de imagen guardada en disco duro.

Fig. 2.9 Imagen guardada en disco duro mediante el botón Guardar imagen.

II.1.4 Visor de Imágenes

El visor de imágenes es una página llamada Imagenes.aspx (Fig. 2.10), en la cual se muestran todas las imágenes guardadas dentro de la carpeta `C:\ImagenesCapturadas`. El propósito del visor de imágenes es el de permitirle al administrador ver las imágenes que se han capturado, tanto por el botón Guardar imagen de la página principal, como las capturadas por el Sistema de Detección de Movimiento, el cual se describe posteriormente.

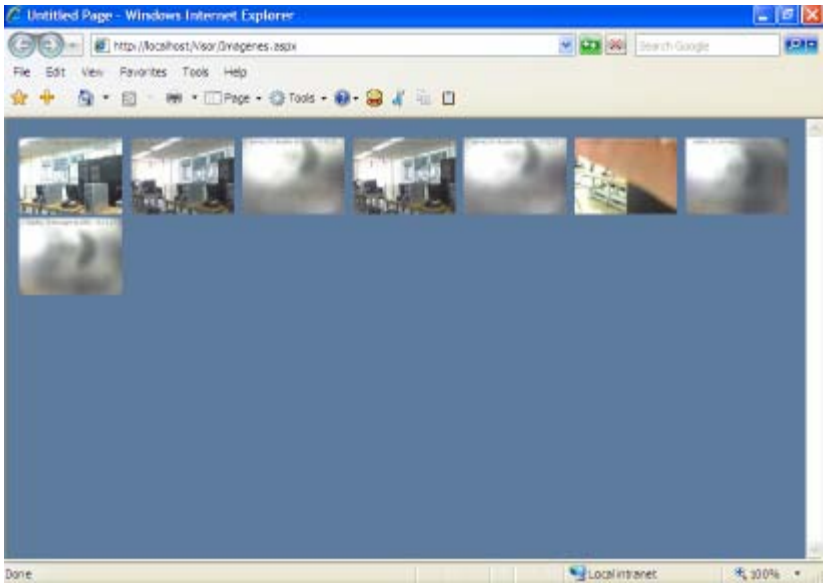

Fig. 2.10 Imagenes.aspx, página donde se muestran todas las imágenes gurdadas en c:\ImagenesCapturadas.

Debido a que estas imágenes se generan de manera dinámica, mostrar estas imágenes no es fácil, ya que no se le puede asignar el url de la imagen a un control HTML:IMAGE o ASP: IMAGE, ya que sólo se cuenta con el nombre de la imagen hasta que es asignado dinámicamente en el momento en que ésta es guardada en disco duro.

Por lo tanto, se tuvo que encontrar una manera de obtener todos los archivos con términación .jpeg dentro de un directorio. Esto se logró mediante el siguiente código

```
public static SubDirectoryWrapper GetSubDirectoryWrapper(string dir)
            {
                  return new SubDirectoryWrapper(dir,
HttpContext.Current.Request.ApplicationPath + "/folder.jpeg");
            }
```

Para permitirle al administrador ver todas las imágenes dentro del fólder, se crearon imágenes *thumbnail*, para que así pudieran ser vistas todas juntas en la misma página. Estas imágenes *thumbnail* son copias de las imágenes dentro del fólder `C:\ImagenesCapturadas`. Éstas imágenes son de dimensiones 100 ancho x 75 largo, menor dimension que las originales (320 x 240). Estas imágenes *thumbnail* se crean dentro de un subfolder especial llamado *thumbs*, dentro del folder ImagenesCapturadas:

`C:\ImagenesCapturadas\thumbs`

Dentro de la página Imagenes.aspx (Fig 2.10), cuando se da click en alguna de las imágenes *thumbnail*, se muestra la imagen original con su tamaño original (Fig. 2.11).

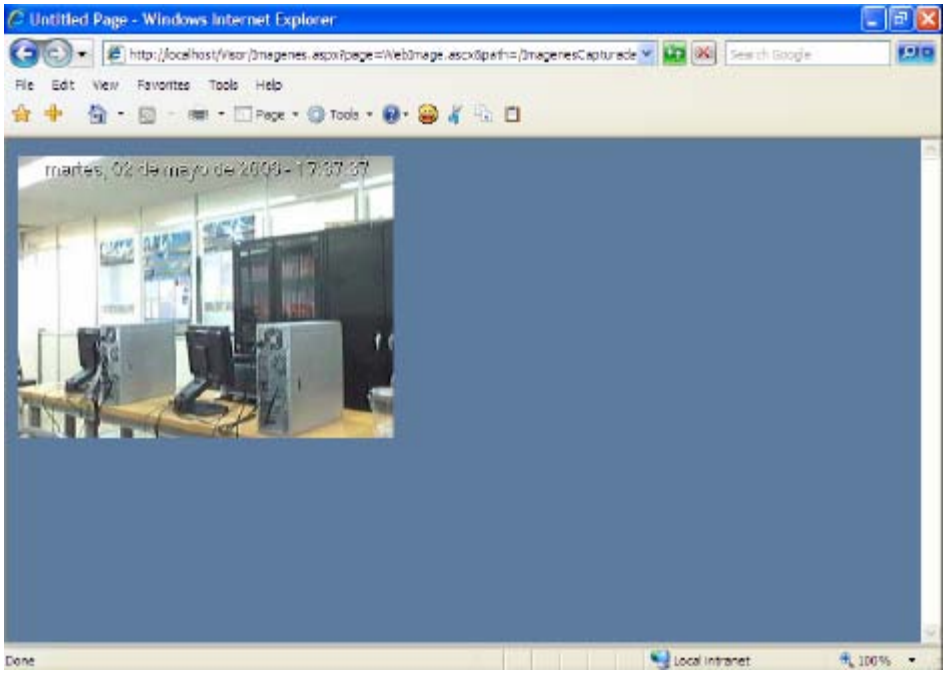

Fig.2.11 Al dar clic en alguna de las imágenes thumbnail, se muestra la imagen original.

Las clases `HtmlTools` y `ImageTools` son las que realizan el procesamiento de las imágenes. `HtmlTools` crea una tabla de imágenes con sus respectivas ligas, e `ImageTools` se encarga de cambiar el tamaño de las imágenes, guardarlas y leerlas desde la carpeta `\thumbs`.

II.2. Sistema de entrada y salida de datos por puerto paralelo

Mediante el Sistema de Seguridad Remoto, se controlan de manera remota dos cerraduras eléctricas, con lo cuál los administradores del Laboratorio controlarán a través de Internet el acceso a las instalaciones. Para poder controlar estas cerraduras, se necesita hacer uso de alguno de los puertos de la computadora para poder enviar e ingresar datos desde la computadora. Se eligió utilizar el puerto paralelo debido a su relativa facilidad para ser utilizado, ya que no es necesario utilizar un UART para serializar los datos, como se necesitaría para transmitirlos a través del puerto serie.

Para enviar los datos a través del puerto paralelo, se utilizó el archivo inpout32.DLL, que se consiguió de la página http://www.logix4u.net/ de manera gratuita. Windows XP no permite controlar directamente el puerto paralelo, ya que al conectar un periférico en alguno de los puertos de la computadora XP automáticamente trata de conseguir e instalar algún driver incluido dentro del sistema operativo para el dispositivo conectado, no permitiendo así que el programador controle directamente el puerto paralelo a través de algún lenguaje de programación. Debido a esto, se tuvo que utilizar dicho archivo DLL. La función de este archivo, se muestra en el siguiente diagrama de flujo, obtenido en la página mencionada anteriormente.

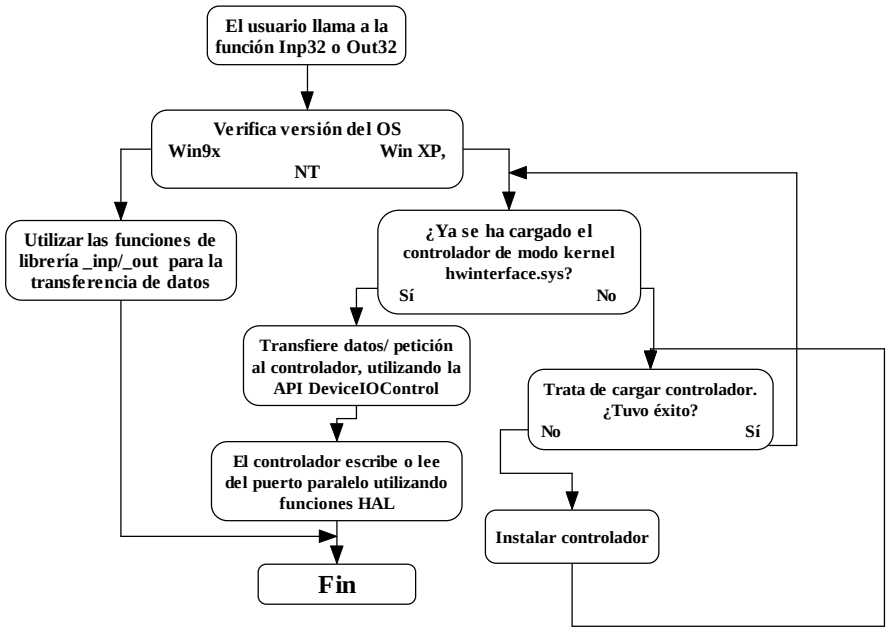

Fig. 2.12 Diagrama de flujo del archivo inpout.dll. Si la plataforma es WIN NT o XP, carga el driver que controla la transferencia de datos por el puerto paralelo. [4]

Este archivo .dll se importa en el proyecto mediante la siguiente clase, la cual especifica la dirección del archivo, y las funciones que sirven como puntos de entrada:

```
public class PortAccess
{
    [DllImport("./inpout32.dll", EntryPoint = "Out32")]
    public static extern void Output(int address, int v);
    // Add the following two lines into the "PortInterop.cs" file
    [DllImport("./inpout32.dll", EntryPoint = "Inp32")]
    public static extern int Input(int address);
}
```

Out32 escribe datos en el registro que se le indique, mientras que Inp32 lee los datos que se encuentren en el registro indicado.

Por ejemplo, `PortAccess.Output(888, 001);` envía el dato 00000001 al registro de datos en LPT1, cuya dirección 0x378 en decimal es igual a 888.

Para leer datos, primero se debe de habilitar el modo bidireccional del puerto, lo cual se logra al mandar el dato 32 al registro de control, de la siguiente forma:

```
PortAccess.Output(890, 32);
```

La dirección del registro de control en LPT1 0x37a es igual a 890 en decimal.

Después de esto, se pueden leer los datos en el registro de datos mediante:

```
PortAccess.Input(888);
```

En la página ASP.NET principal Default.aspx (Fig 2.5), se utilizan dos botones, uno para abrir la cerradura eléctrica de la puerta principal del laboratorio, y otro botón para controlar la cerradura de la puerta donde se encuentra el servidor. Cuando se carga la página por primera vez, se manda a traves del puerto paralelo un ‘0’ lógico a las cerraduras, con el fin de que éstas permanezcan cerradas. Cuando se aprieta por primera vez alguno de los botones para abrir las cerraduras, se mandan 5 volts a través del puerto, los cuales sirven como señal para poder activar las cerraduras. Debido a que estos 5 volts permanecen en el registro de datos del puerto paralelo, hay que cambiar este voltaje a 0 volts para que así se pueda volver a abrir la puerta. Cuando se aprieta el botón “abrir cerradura *x*”, se leen mediante `PortAccess.Input` los datos que se encuentran en el registro de datos del puerto paralelo. Sí anteriormente se habian mandado 5 volts para abrir las puertas, se estará leyendo 001 o 010, mediante un “case” se “limpiará” el registro de datos del puerto paralelo mandando 0 volts:

```
case 001:
                {
```

```
                    PortAccess.Output(888, 000);
                    Label1.Text = "Por favor, vuelva a apretar el botón
para abrir la puerta principal";
                }

case 010
      {
                ……
      }
etc….
```

Como el registro de datos del puerto paralelo ahora se encuentra con 0 volts, entonces ya se pueden volver a abrir las cerraduras mandando 5 volts a través del puerto paralelo. En la página principal (Fig. 2.5), el botón que controla la cerradura mostrará el siguiente mensaje: "*Por favor, vuelva a apretar el botón para abrir la cerradura*".

Por lo tanto, la primera vez que se accede a la página principal, y se aprieta alguno de los botones para abrir las cerraduras, sólo basta con apretar estos botones una sola vez para abrir la cerradura correspondiente. Si se desea volver a abrir esa misma cerradura, se tendrá que apretar dos veces el botón correspondiente, la primera vez para "limpiar" el puerto paralelo mandándole 0 volts, y la segunda vez para mandar los 5 volts que abrirán la cerradura. Cuando por ejemplo, se aprieta el botón "*Abrir puerta principal*", y después se aprieta el botón "*Abrir puerta del servidor*", no hay necesidad de apretar dos veces el mismo botón, ya que al apretar alguno de los dos botones y después el otro, automáticamnte se limpia el pin de datos D*x* del primer botón. LA ÚNICA VEZ QUE SE TENDRÁ QUE APRETAR DOS VECES EL MISMO BOTÓN ES CUANDO SE TRATA DE ABRIR DOS VECES SEGUIDAS LA MISMA CERRADURA. De cualquier manera, el botón desplegará un texto indicando si es necesario apretarlo una vez más o no.

II.3 Sistema de cerraduras.

II.3.1 Parte mecánica del sistema de cerraduras.

El sistema de cerraduras permite activar mediante el puerto paralelo, dos solenoides, los cuales atrancan las dos puertas corredizas del laboratorio Microsoft Research, la puerta principal y la puerta del servidor, sirviendo así como una protección adicional en caso de que alguien logré forzar las cerraduras con las que actualmente cuenta el laboratorio.

El solenoide (Fig. 2.13) consiste de una bobina, la cual al ser energizada con una corriente alterna de 120 V, se comporta como un imán, para así poder atraer un pequeño cilindro situado entre la bobina. Este cilindro servirá para atrancar la puerta, debido a que ésta es corrediza. Mediante la fuerza de gravedad, el cilindro caerá cuando no se energiza el solenoide, impidiendo que la puerta se deslice, y

subirá cuando el solenoide sea energizado, permitiendo así que la puerta se deslice libremente. A continuación se muestran algunas imágenes de dicho solenoide.

Fig. 2.13 Vista superior del solenoide

Fig. 2.14 Vista lateral del solenoide sin energizar

Fig. 2.15 Vista del solenoide al energizarse.

Para que estos solenoides puedan atrancar las puertas, se instaló una pieza de aluminio a lo largo de la parte superior de las puertas. Dicha pieza de aluminio tiene una perforación del tamaño del cilindro del solenoide, con el fin de que cuando la puerta se encuentre en la posición de cerrada, y el solenoide no esté energizado, el cilindro caiga dentro de la perforación de la pieza de aluminio, atrancando así a toda la puerta (Fig. 2.16 y Fig. 2.17). Cuando la puerta se encuentre abierta, no importa que el cilindro caiga, ya que se deslizará a lo largo de la pieza de aluminio, permitiendo así que la puerta esté abierta.

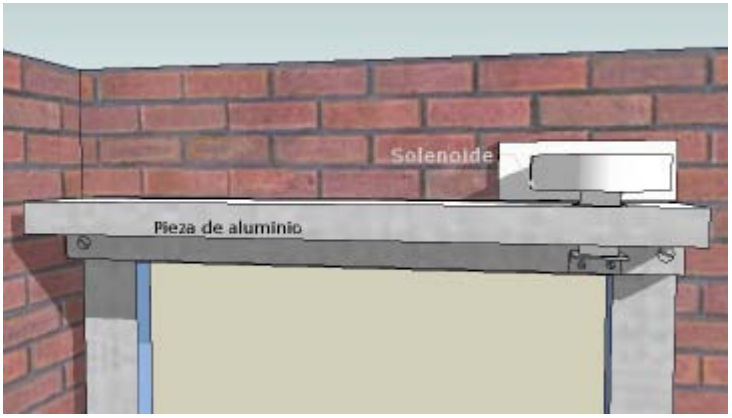

Fig. 2.16 Solenoide atrancando la parte superior de la puerta corrediza

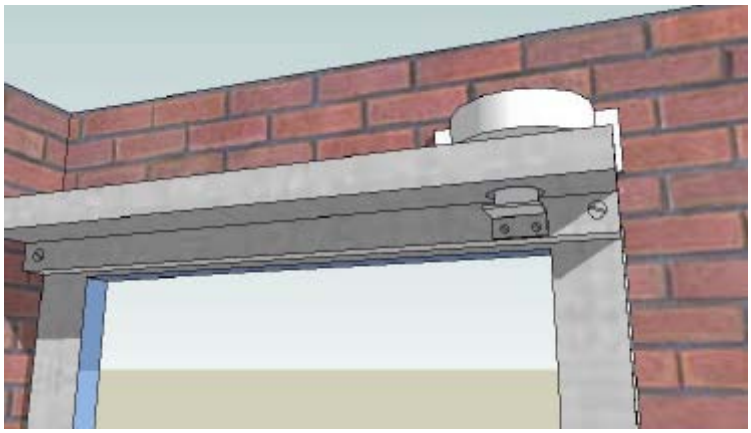

Fig. 2.17 Vista de la pieza de aluminio instalada en la parte superior de la puerta corrediza.

Fig. 2.18 Vista general de la puerta corrediza.

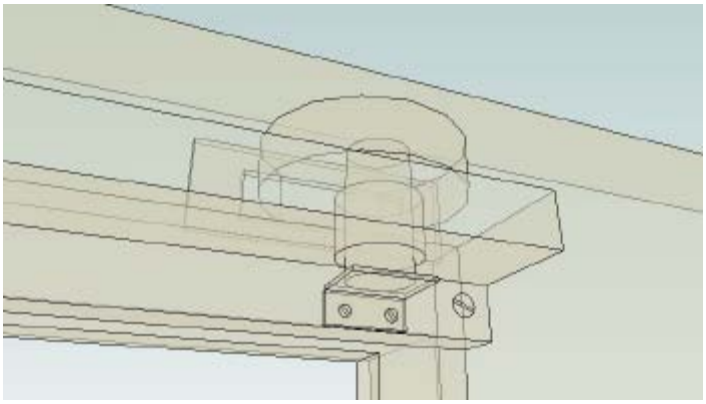

Fig. 2.19 El cilindro del solenoide cae dentro de la perforación de la placa de aluminio, atrancando así la puerta corrediza.

Fig. 2.20 La puerta se puede deslizar libremente aún cuando el solenoide no esté activado, mientras éste no se encuentre por encima de la perforación de la pieza de aluminio.

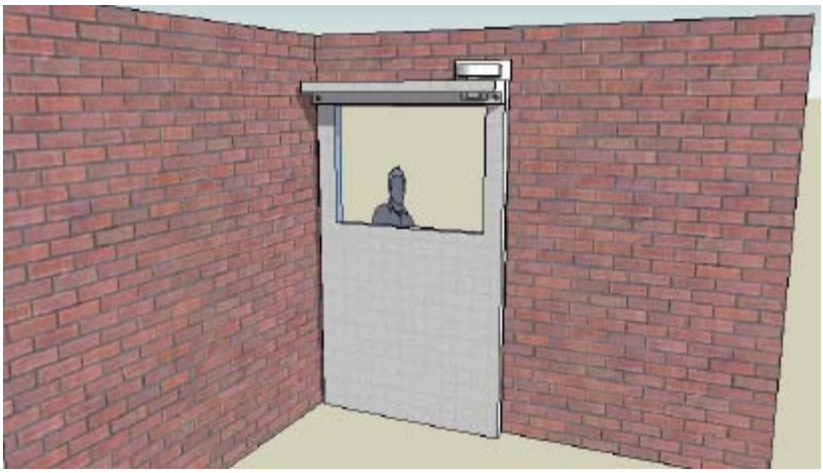

Fig. 2.21 Al no activarse el solenoide, la puerta permanece atrancada.

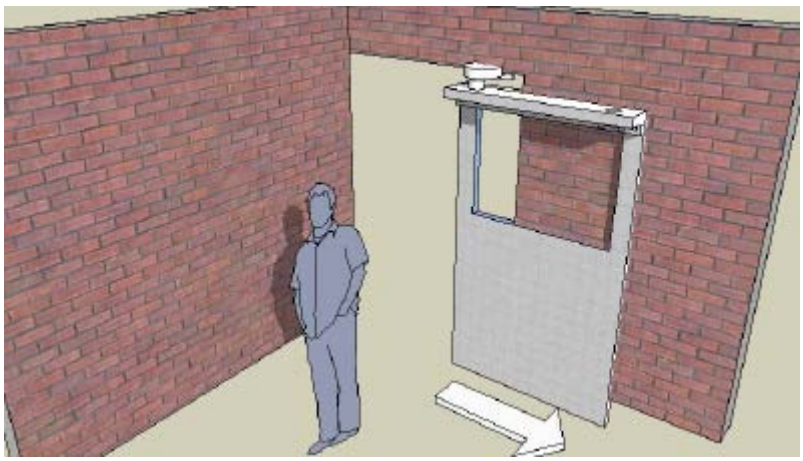

Fig. 2.22 Al activarse el solenoide, éste no atranca más a la puerta corrediza, por lo tanto ésta puede deslizarse libremente

Como se puede ver en las imágenes anteriores, se puede controlar el acceso al laboratorio atrancando las puertas con los solenoides. Estos solenoides son activados con una corriente de 120 V, la cual es tomada directamente de un contacto de la instalación eléctrica del laboratorio. Sin embargo, estos solenoides no pueden permanecer energizados durante mucho tiempo, debido al calor extremo que es generado dentro de la bobina. Aproximadamente los solenoides pueden estar activados máximo 30 segundos, en caso de permanecer por más tiempo activados, la bobina se derrite y el solenoide queda inservible. Para evitar esto, los solenoides sólo podrán permanecer energizados durante un tiempo determinado. El tiempo máximo es de 15 segundos, el cual puede ser ajustado mediante el circuito electrónico que se describe más adelante.

El paso de esta corriente eléctrica de 120 volts es controlado mediante el puerto paralelo de la computadora que hospeda la aplicación Web ASP.NET. Para controlar el paso de la corriente hacia el solenoide, se utilizó un *relay* o relevador, el cual sirve como *switch* que se activa al mandar un “1” lógico a través del puerto paralelo de la computadora. Sin embargo, debido a que se desea que los solenoides se desactiven automáticamente después de unos cuantos segundos, en este caso 6 segundos, y debido a que el puerto paralelo no puede activar directamente al *relay*, se diseñó un circuito electrónico el cual permite que los 5 volts del pin de datos del puerto paralelo puedan controlar el flujo de la corriente alterna que alimenta a los solenoides.

II.3.2 Circuito electrónico para activar las cerraduras.

Se diseñó el siguiente circuito electrónico con las siguientes finalidades:

- Como medida de protección para el puerto paralelo de la computadora, ya que resulta sumamente peligroso conectarlo directamente a algún componente electrónico, debido a que puede haber una sobredemanda de corriente, se podría causar algún corto circuito, podrían producirse picos en el voltaje, etc., los cuales podrían ocasionar no sólo que el puerto paralelo quede inservible, sino que incluso podrían descomponer la tarjeta de I/O, o incluso quemar la tarjeta madre de la computadora. Para evitar todo esto, es necesario aislar al puerto paralelo mediante un optocoplador.
- Los relays que sirven como switch para controlar el paso de la corriente alterna hacia los solenoides, se activan con 5V pero demandan una corriente de 130 mA, la cual es imposible de obtener con el puerto paralelo, el cual sólo proporciona 26 mA. Debido a esto, se decidió utilizar una fuente de poder externa, que pueda proporcionar la corriente demandada por los relays, y que además sirva como alimentación de todos los componentes electrónicos que son utilizados en el circuito.
- Debido a que se requiere que los solenoides no permanezcan energizados por mucho tiempo, ya que si se activan por más de 30 segundos, simplemente se quemarían. Para lograr esto, se implementó un timer que automáticamente apaga los relays después de unos cuantos segundos.

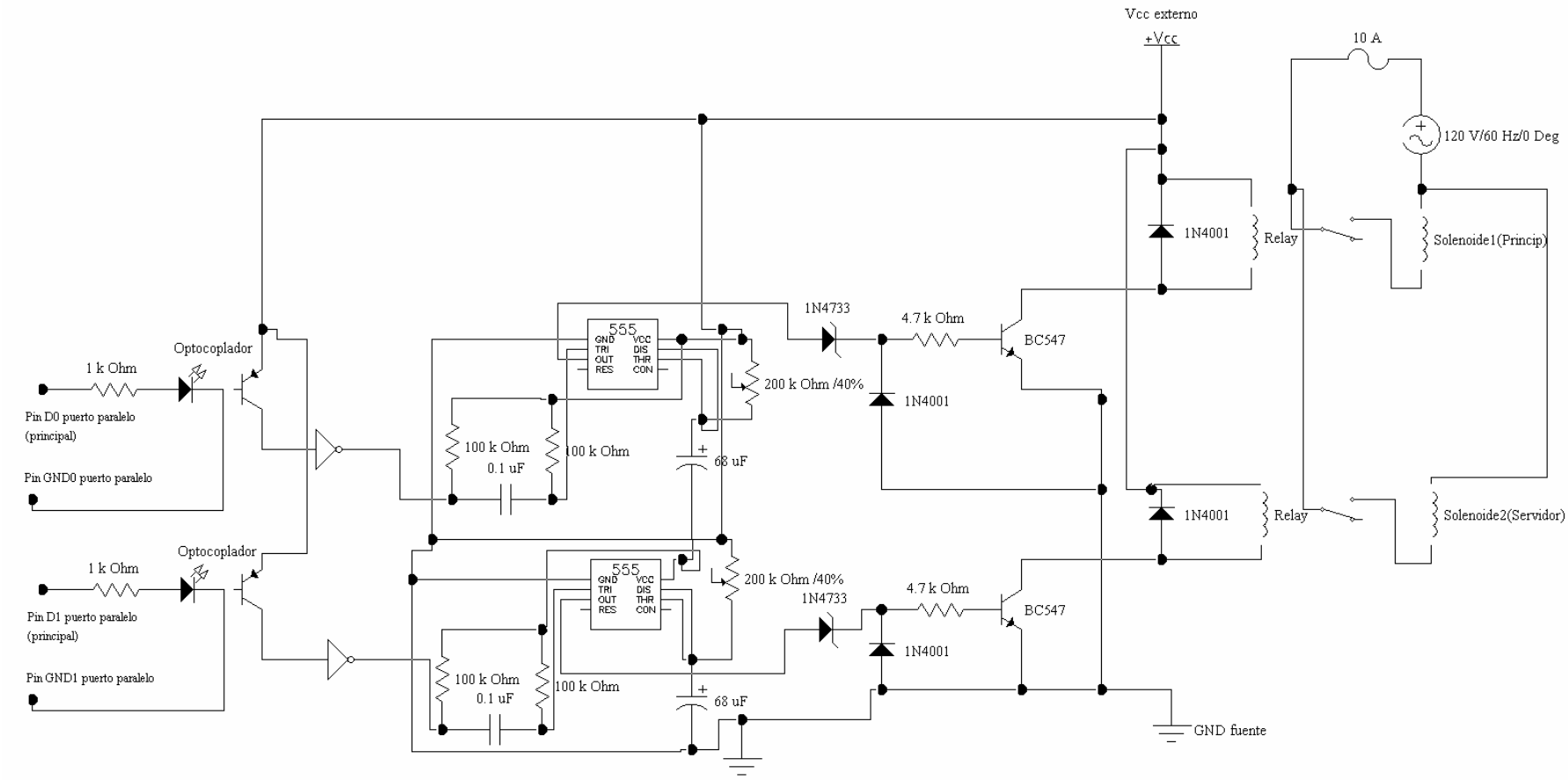

Fig. 2.23 Esquema general del circuito electrónico para activar las cerraduras.

La fuente externa para alimentar el circuito que se utilizó es un eliminador universal AC-DC, que proporciona voltajes en un rango de 1.5 VDC hasta 12 VDC, y que proporciona una corriente máxima de 500mA. Debido a que este eliminador no proporciona un voltaje de 5 VDC, se optó por ajustarlo para que proporcione 6 VDC, y conectar esta salida a un regulador 7805, cuya salida es de exactamente 5 VDC, la cual servirá para alimentar al circuito.

Los pines 2 y 3 del puerto paralelo (D0 y D1), son los que controlan a los relays del circuito, y los pines 24 y 25, sirven como tierras. Dichos pines del puerto paralelo son optoaislados del resto del circuito mediante un optocoplador 4N32. La resistencia de 1K? entre el pin del puerto paralelo y el 4N32 sirve para limitar la corriente demandada por el 4N32.

Las salidas de los optocopladores se conectan a un inversor 7404, el cual permite activar los relays mediante un "1" lógico del puerto paralelo. Ya que el C.I. LM555 se activa con un flanco de bajada, es necesario invertir la salida del puerto paralelo, así, si se mandan 5 V por el puerto paralelo, pasarán por el inversor 7404 y llegarán al LM555 como un flanco de bajada.

Se implementó un arreglo de resistencias y capacitores conectados al LM555 con el fin de que cuando a éste le llegue un flanco de bajada, dé una salida de 5 V durante *n* segundos. El número de segundos que el 555 está activo se calcula mediante la siguiente ecuación:

$$t = 1.1*R*C$$

Donde R= valor de la resistencia en ?
C= valor del capacitor en µF

En lugar de utilizar una resistencia fija, se utilizó una resistencia variable de 220 k? , para así poder ajustar manualmente el tiempo que se desea que el timer LM555 esté activado. Por lo tanto, el tiempo máximo que el timer puede estar activado es de:

t= 1.1*(220k?)*(68µF)= 16.456 seg.

$$t \approx 16.5 seg$$

Como se consideró que este tiempo es demasiado, se optó por calibrar la resistencia variable en aproximadamente 80 K? , para que así:

$$t = 1.1*(80K\Omega)*(68mF) = 5.984 \text{ seg.}$$

La salida del timer 555 pasa por un diodo Zener IN4733 a 6.8V y ½W, con el fin de rectificar y regular el voltaje que sale del 555, evitando así que algún voltaje se regrese al

LM555. Se eligió este diodo de 6.8V debido a que el voltaje de polarización del diodo Zener es:

$$V_p = V_z + \frac{1}{3}V_z$$

$$V_p = 5V + \left(\frac{1}{3}\right)5V = 6.665\,\mathrm{V}$$

La salida del diodo Zener llega a una resistencia de 4.7K? , para limitar la corriente que se le suministra al transistor BC547.

El transistor funciona como un switch para controlar la corriente que llega al relay, o sea, la corriente proveniente de la fuente externa, en este caso el eliminador universal. Cuando el timer 555 no está activado, 0 V le llegan al transistor BC547, con lo cual se encuentra en estado de corte, no dejando fluir la corriente del colector hacia el emisor. Cuando al transistor le llegue un voltaje mayor a 0.7 V (en nuestro caso 5 V de la salida del LM555), se pondrá en estado de saturación, con lo cual dejará pasar la corriente de la fuente externa a través del relay, y fluir del colector del BC547 al emisor, el cual se encuentra conectado a GND, cerrando así el circuito.

El diodo IN4001 que va del emisor del BC547 a la resistencia de 4.7 k? sirve como protección para los picos de voltaje y para eliminar voltajes negativos.

Debido a que la bobina del relay posee una gran inductancia, cuando la corriente se corta, se genera dentro de la bobina una gran cantidad de picos de voltaje. Por lo tanto se conectó un diodo rectificador IN4001 en paralelo con la bobina del relay, con el fin de evitar que algún pico de voltaje llegue al transistor, dejándolo así inservible, e incluso dichos picos de voltaje podrían llegar y dañar a los otros componentes del circuito.

Por último, el relay utilizado fue un Sun Hold RAS 0510, el cual funciona con 5 VDC y demanda 72 mA de corriente.

Fig. 2.24 Relay Sun Hold RAS 0510. [19]

Como se puede ver en la imagen (Fig. 2.24), este relay tiene 5 patas, dos patas son los extremos de la bobina que funciona a 5 VDC y 3 patas conectadas a una placa que pueden

transmitir 120 VAC, y una corriente máxima de 10A. Cuando la bobina se energiza, se genera un campo electromagnético alrededor de ella, el cual mueve la placa conectada en la pata 4 hacia la pata 5 del relay, cerrando así el circuito por el cual fluirá la corriente alterna de 120 V. Así, el circuito permanecerá cerrado mientras la placa haga contacto con la pata 5 del relay, al dejar de suministrarle 5 VDC a la bobina, la placa regresará a su estado original, o sea, a la pata 3 del relay, abriendo así el circuito.

Como se puede observar en el esquema general del circuito (Fig. 2.23), el solenoide que atranca las puertas corredizas está conectado en un extremo directamente al toma corriente de las instalaciones del laboratorio, y el otro extremo del solenoide a la pata 4 del relay. La pata 5 del relay se conectará al toma corriente del laboratorio, habiendo un fusible de 120 V y 10 A de por medio.

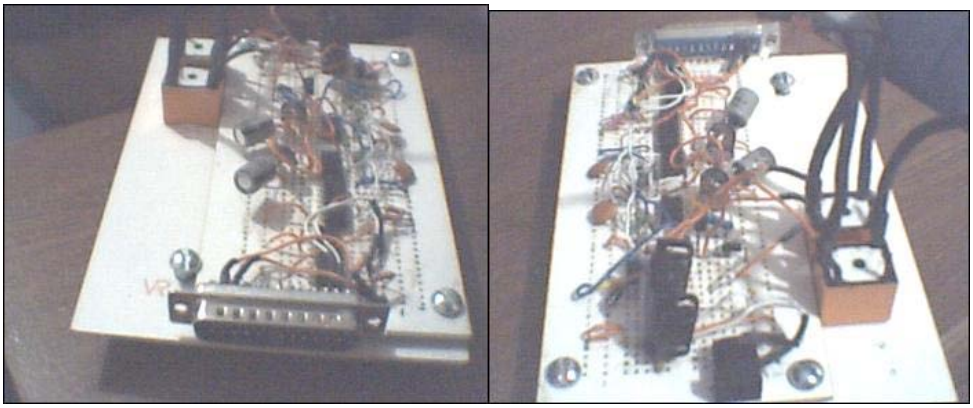

Fig. 2.25 Imágenes del circuito electrónico físico.

II.4. Sistema de interfaz vía Internet.

La página ASP.NET se hospeda en una computadora del Laboratorio Microsoft Research, la cual servirá como nuestro servidor. De esta manera, dicha página podrá ser accedida desde cualquier parte del mundo, a través de Internet. Así, la página se podrá acceder escribiendo:

http://*IP_SERVIDOR*/SegLab/Login.aspx

Donde SegLab es el nombre del fólder que contiene a la página ASP.NET, y Login.aspx (Fig. 2.1) es la página de inicio donde los administradores ingresan sus credenciales.

El servidor de páginas de Internet que se utilizó para el Sistema de Seguridad Remoto, es Microsoft Internet Information Server, el cual es un conjunto de servicios basados en Internet para servidores que utilizan Microsoft Windows. Actualmente es el segundo servidor Web más popular, sólo por detrás del servidor HTTP Apache.

Los servicios que IIS ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS. Este servicio convierte a una computadora 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.

IIS viene incluido dentro del paquete de instalación del sistema operativo Windows XP, sin embargo, IIS no se instala por default al instalar Windows XP. Hay que instalar IIS manualmente una vez que el sistema operativo ha sido instalado correctamente. Para instalar IIS hay que realizar los siguientes pasos:

- En Windows XP, ir a Inicio-> Panel de Control-> Agregar o Quitar Programas
- En Agregar o Quitar Programas, dar clic en Agregar o Quitar Componentes de Windows.
- En la ventana que aparece enseguida, en el cuadro de componentes, buscar Servicios de Internet Information Server (IIS), y dar clic en su casilla de verificación correspondiente.
- Dar clic en siguiente.

Aparece el programa de instalación, el cual pedirá que se inserte el disco de instalación del sistema operativo Windows XP.

Una vez instalado Internet Information Server (IIS), se creará una carpeta llamada Inetpub en la unidad C:

C:\Inetpub

Dentro de Inetpub, hay una carpeta llamada wwwroot. Dentro de esta carpeta es donde se deberán colocar las páginas ASP.NET que quieran ser publicadas a Internet. Por ejemplo, para poder acceder al Sistema de Seguridad Remoto, sólo hay que copiar la carpeta que contiene las páginas ASP.NET en C:\Inetpub\wwwroot:

C:\Inetpub\wwwroot\SegLab

Después, hay que dar de alta en el IIS, nuestra página Web, para eso:

- Ir a Inicio->Panel de Control-> Rendimiento y mantenimiento-> Herramientas administrativas
- Dar doble clic en Servicios de Internet Information Server
- En la ventana emergente (Fig. 2.26), expandir el árbol de Servicios de Internet Information Server, y localizar la carpeta que se copió a wwwroot

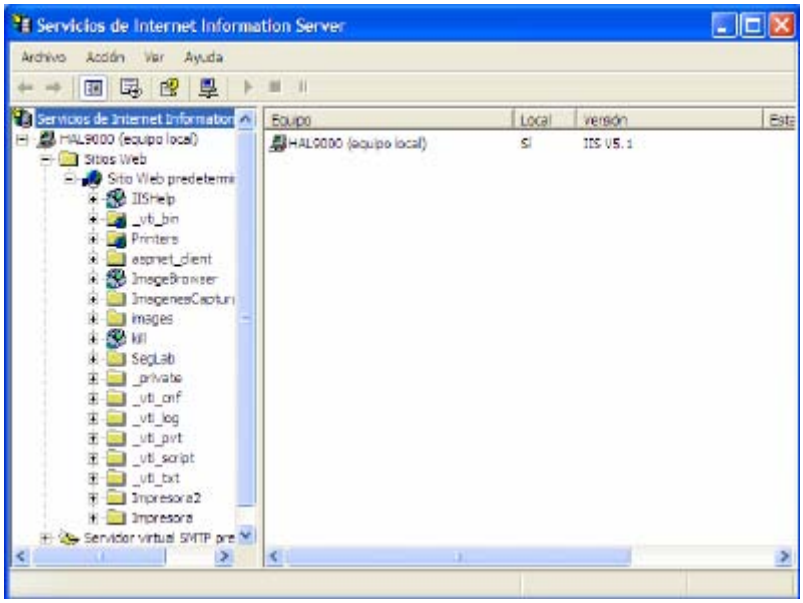

Fig. 2.26 Pantalla de administración de Internet Information Server (IIS).

- Dar clic derecho sobre la carpeta, y elegir propiedades.
- En la ventana que emerge (Fig.2.27), localizar configuración de aplicación y dar clic en Crear.

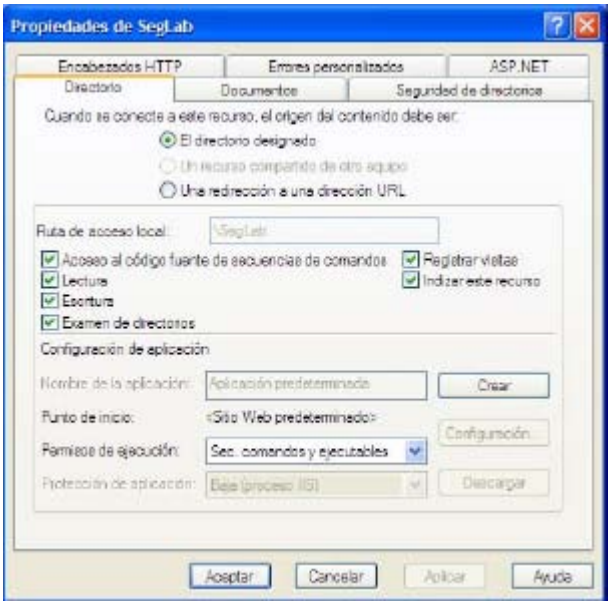

Fig. 2.27 Pantalla de configuración de la página ASP.NET

- Dar clic en Aceptar.

Después de esto, sólo basta con abrir el explorador de Internet, y escribir en la barra de direcciones http://*IP_SERVIDOR*/SegLab/Login.aspx, e IIS enviará una copia de la página para ser mostrada en el explorador de Internet.

Cabe señalar que no se necesita iniciar sesión en el servidor para que IIS acepte peticiones de exploradores Web. Tan sólo basta con que el servidor esté prendido para que las páginas hospedadas en IIS puedan ser accedidas a través de Internet.

II.5. Sistema de detección de movimiento.

El Sistema de Detección de Movimiento, es una aplicación desarrollada en Visual C#. Ésta aplicación se creó con el fin de detectar movimiento en el Laboratorio Microsoft Research mientras los administradores del laboratorio no se encuentren presentes. Cuando la aplicación detecta movimiento, guarda imágenes en el disco duro del momento en que se detectó el movimiento, para su posterior análisis por alguno de los administradores del laboratorio.

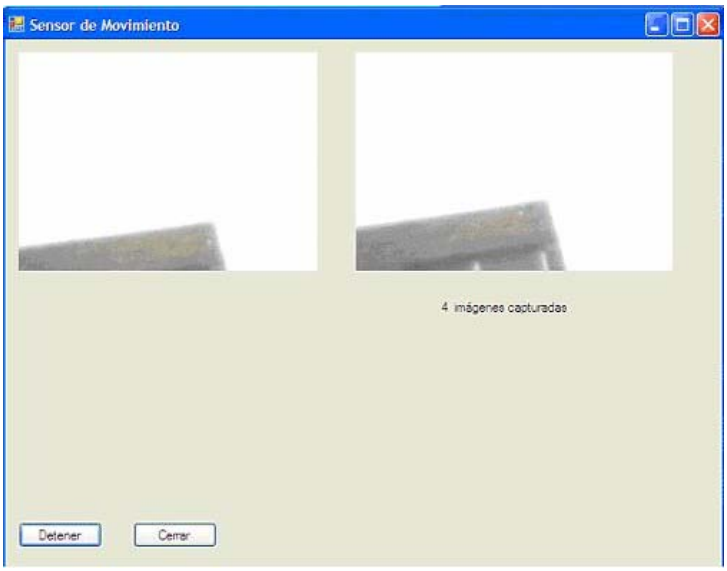

Fig. 2.28 Interfaz del Detector de Movimiento.

El Detector de Movimiento utiliza la webcam instalada en el servidor, con el propósito de tomar imágenes "panorámicas" del laboratorio. Primero la webcam toma una primera imagen del laboratorio (imagen #1), después de 5 segundos la webcam vuelve a tomar otra imagen del laboratorio (imagen #2), y entonces el detector de movimiento compara el cambio de los tres colores básicos (RGB: Rojo-Verde-Azul) de cada uno de los píxeles que forman la imagen. Si existe una diferencia de los colores de los píxeles de la imagen #1 y la imagen #2 mayor a 86% (`PorcentajeActual = 86;`), el programa guarda en el disco duro del servidor la imagen #2, agregando a la imagen la fecha y hora en que fue tomada.

Cuando termina el proceso de comparación, el programa espera 5 segundos, después de los cuales la primera imagen (imagen #1) es eliminada de memoria, la imagen #2 pasa a ser ahora la imagen #1, y la webcam vuelve a tomar una nueva imagen del laboratorio, la cual se convierte en la nueva imagen #2, y el proceso de comparación se vuelve a repetir.

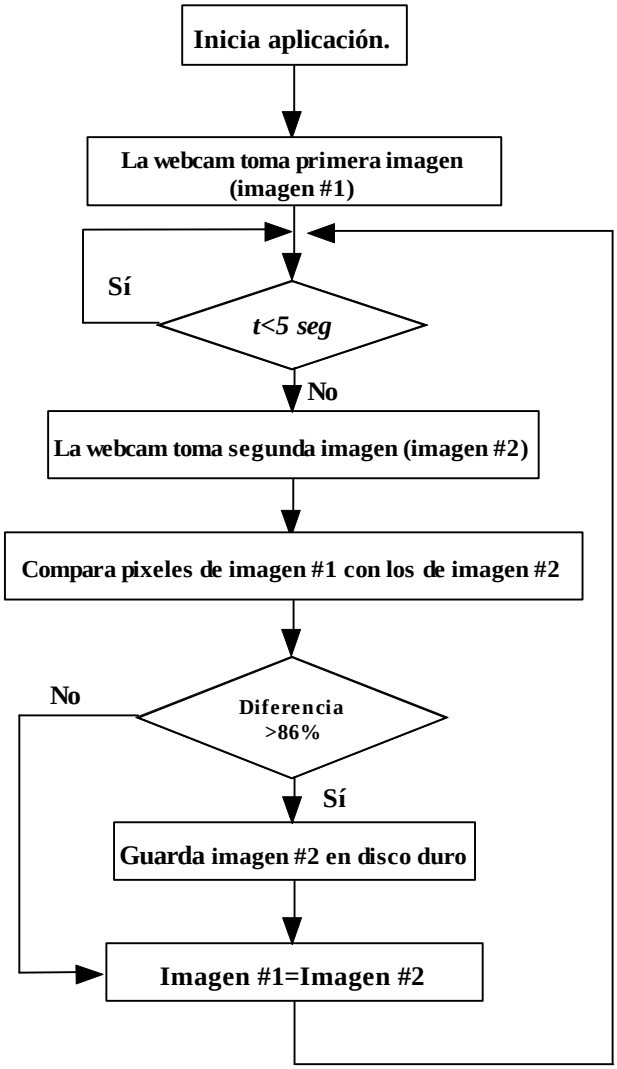

Fig. 2.29 Diagrama de flujo del detector de movimiento.

Se creó una forma, la cual contiene dos controles pictureBox, el primero para mostrar la imagen actual, y el segundo pictureBox para mostrar la imagen que se guardó en disco duro si se detectó movimiento. Se utilizó el archivo CamServer.dll mencionado con anterioridad, con el fin de interactuar con el driver de la Webcam y así poder obtener la imagen. Debido a que este archivo ya había sido registrado con anterioridad para la página Default.aspx (Fig. 2.5), se hizo una referencia a él desde Visual Studio 2005 de la misma manera que se describe en el capítulo II: Propuesta de sistema de seguridad, Pág.45, Fig. 2.7.

Se utilizó un control Label que indica cuántas imágenes se han guardado en disco duro, y dos botones, uno para detener el Detector de Movimiento, y otro para cerrar completamente la aplicación. Además, se utilizó un control Timer dentro de la forma principal. Este Timer se configuró con intervalos de 5000 ms, con lo cual cada 5 segundos se estará tomando una imagen nueva con la webcam y procesando las imágenes de manera cíclica. `timer1_Tick(object sender, EventArgs e){` *Código para procesar las imágenes…* `} ,` es

el evento que emerge cuando han elapsado los 5 segundos del Timer, y entonces se ejecuta el código que procesa las imágenes.

Mediante `Bitmap bmp = new Bitmap(ms); Bitmap bmp2 = new Bitmap(ms2); Bitmap diferente = new Bitmap(bmp.Width, bmp.Height);`, se toman las dos fotos necesarias, se crea una copia de `bmp` llamada `diferente` y posteriormente se llama al método `ImageProcessing` de la clase ImageProcessing.cs, la cual procesará a las dos imágenes capturadas: `ImageProcessing imageProcessing = new ImageProcessing(bmp, bmp2, diferente);`.

Por último, si existe diferencia entre las dos imágenes, se guarda la última imagen capturada por la webcam en el folder ImagenesCapturadas, en la unidad C:\ del servidor del Sistema de Seguridad Remoto, con el nombre “imagen+*hora*.jpg”:

```
bmp2.Save(@"C:\ImagenesCapturadas\imagen" + hora, icf[1], encps);
```

CAPITULO III. DESARROLLO, IMPLEMENTACIÓN E INSTALACIÓN DEL SISTEMA DE SEGURIDAD REMOTO.

III. 1. Instalación del Sistema de Seguridad Remoto en el Laboratorio Microsoft Research.

Para instalar el sistema dentro del laboratorio, se necesitó tender en el laboratorio cable de corriente de calibre 16, voltaje máximo de 600 VAC, para poder energizar los solenoides (Fig. 2.13) que actuarán como cerraduras. Se necesitó 27 mts. de este cable, suficientes para conectar los solenoides hasta el circuito que los controlará (Fig. 2.23), y que estará conectado al puerto paralelo del servidor.

Se utilizó un eliminador universal AC-DC, que proporciona voltajes en un rango de 1.5 VDC hasta 12 VDC, y que proporciona una corriente máxima de 500mA, para poder alimentar el circuito que controla a los solenoides (Fig. 2.23). También se compró un cable DB25 macho-hembra, el cual se conectó al puerto paralelo del servidor, y al conector DB25 del circuito de control (Fig. 2.23). Se conectó la pata 4 de los relays y uno de los conectores de los solenoides al toma corriente del laboratorio mediante un cable de extensión doméstica. La conexión de estos cables se muestra en el esquema de la Fig. 2.23.

Las piezas de aluminio que se encuentran colocadas en la parte superior de las puertas corredizas (Fig. 2.16 y 2.17) se obtuvieron de pedazos de aluminio que sobraron cuando se hicieron las puertas corredizas del laboratorio. Lo único que se necesitó fue cortar estas piezas a la medida con una segueta, y perforarle los hoyos con un taladro. Se utilizaron pigas para fijar estas piezas y los solenoides a las puertas.

La PC en la cual está hospedada la aplicación ASP.NET del Sistema de seguridad Remoto, y que servirá como servidor del sistema, es una computadora Compaq con procesador Pentium 3, 512 MB de RAM. La computadora, con IP 132.248.59.38, tiene salida a Internet mediante un cable UTP conectado de la tarjeta LAN de la PC al switch de la red del Laboratorio de Computación. La webcam Logitech que se instaló en el servidor del Sistema de Seguridad, pertenece al Laboratorio Microsoft Research, por lo cual no fue necesario comprar una nueva.

Debido a que la aplicación se creó en una PC con las mismas características que las de la máquina servidor (otra computadora similar del laboratorio), no hubo problemas de incompatibilidad con el sistema operativo o con IIS. El sistema operativo de la máquina es Windows XP Professional Service Pack 2. La versión de IIS es v5.1, y la aplicación Web se instaló con una cuenta que tiene privilegios de administrador.

Con respecto a la aplicación del Detector de Movimiento, Visual Studio 2005 tiene una opción para publicar proyectos, con lo cual, después de ser compiladas las aplicaciones, se puede generar un archivo instalador .msi (Fig. 3.1).

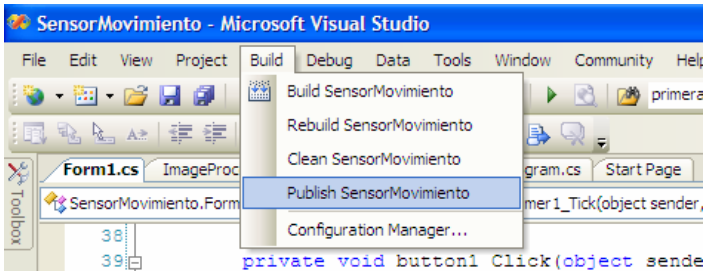

Fig. 3.1 Dentro del menú Build, seleccionar Publish NombreProjecto para crear un instalador de la aplicación.

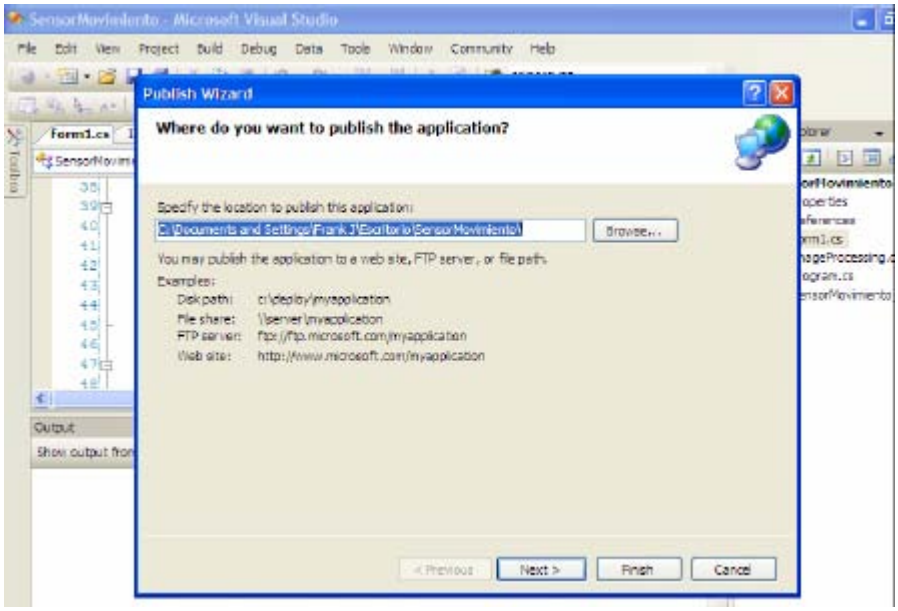

Fig. 3.2 Un wizard nos ayuda a crear el archivo instalador .msi de la aplicación, el cual entre otras cosas nos pregunta donde guardar el archivo .msi, si éste se instalará desde un CD-ROM o desde Internet, si este buscará regularmente actualizaciones en alguna dirección de Internet, etc.

Con este archivo fácilmente se pudo instalar en la máquina servidor la aplicación Detector de Movimiento, sólo hay que dar clic en el instalador creado para que comience la instalación en el sistema de manera automática.

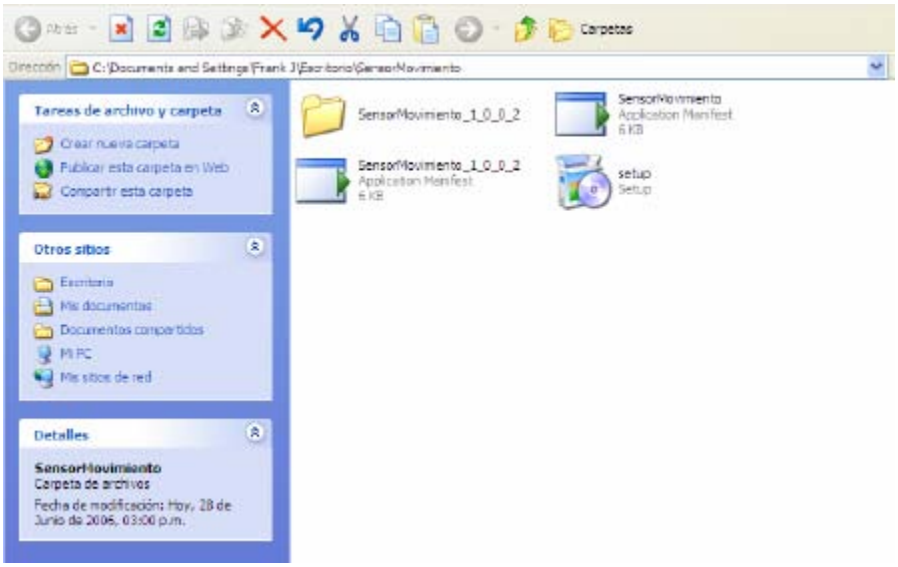

Fig. 3.3 Archivos creados mediante la opción Publish de Visual Studio 2005.
El archivo setup es el .msi que inicia la instalación de la aplicación,
los demás son archivos de metadatos, o manifiestos de aplicación
como son conocidos en .NET.

Por último, se registró el archivo CamServer.dll (con el cual se pueden tomar imágenes a través de la webcam) en la máquina servidor, tal como se describe en el capítulo II.1 Sistema de monitoreo con Webcam -> Página principal., (Fig. 2.6). Lo único que hay que mencionar es que para registrar este archivo hay que iniciar sesión en la máquina servidor con una cuenta con privilegios de administrador.

III.2. Implantación del Sistema de Seguridad Remoto en otras instalaciones.

En teoría, instalar el sitio Web en una máquina diferente a la creada es sumamente fácil. Dentro del menú de Microsoft Visual Studio 2005, existe una opción para copiar el proyecto hacia otra computadora en red, o hacia un dispositivo de almacenamiento externo. Con esta herramienta, sólo hay que apretar dentro de la ventana Website->Copy Web Site (Fig. 3.4), el botón Connect to.., elegir la carpeta o sitio Web remoto a donde se quiere copiar la aplicación Web, y mover los archivos de la ventana Source Website a Remote Website (Fig. 3.5). Visual Studio 2005 automáticamente copia los archivos a través de la red local o de Internet de manera automática. Como el proyecto se creó como aplicación Web IIS, no hace falta darlo de alta en el servidor IIS de la máquina a la que se copió. Hay que mencionar que la máquina servidor, deberá tener instalado Microsoft SQL Server 2005 Express Edition, para poder manejar la base de datos de autenticación ASPNETDB.MDF. Esta versión de SQL Server 2005, Microsoft la proporciona de manera gratuita a través de su sitio de Internet. Además, Visual Studio 2005 instala automáticamente esta versión de SQL Server.

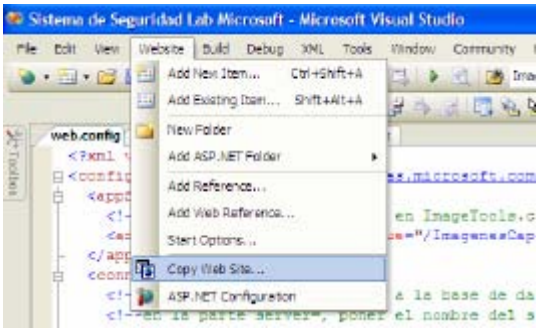

Fig.3.4 En el menú Website, Copy Web Site nos permite copiar la aplicación Web a una computadora remota.

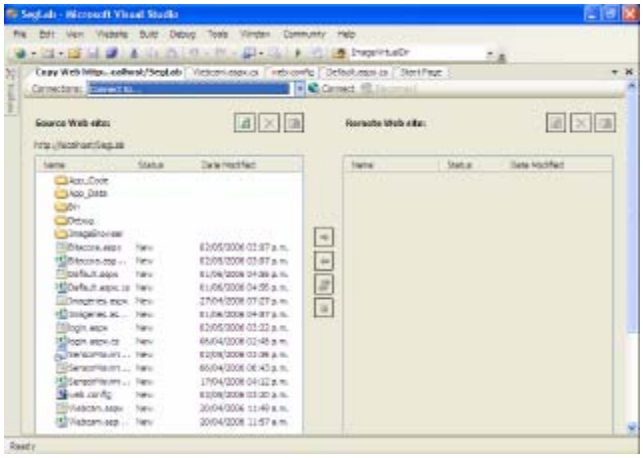

Fig. 3.5 Para empezar a copiar la aplicación Web, elegir en la ventana de la izquierda los archivos que se desea copiar, y apretar el botón → para copiarlos al equipo remoto. La ventana derecha muestra los archivos en la computadora remota. Para eliminar archivos en la máquina remota, elegirlos en la ventana derecha y oprimir el botón ←.

Otra forma de copiar la aplicación sería la de simplemente colocar la carpeta SegLab, que contiene la aplicación ASP.NET del Sistema de Seguridad Remoto, dentro de la carpeta C:\Inetpub\wwwroot de la computadora en la cual se desea instalar (Asumiendo que IIS ya se encuentra instalado en la computadora, para información sobre cómo instalarlo, referirse al capítulo II.4. Sistema de interfaz vía Internet., Pág. 63).

Después, hay que dar de alta la aplicación en el IIS de la máquina, mediante la consola de Servicios de Internet Information Server. Para más información de cómo dar de alta la aplicación Web en IIS, referirse al capítulo II.4. Sistema de interfaz vía Internet, Pág. 63.

Posteriormente hay que registrar el archivo CamServer.dll, el cual se encuentra en la carpeta \COM Component, que se encuentra dentro de la carpeta de la aplicación Web *C:\Inetpub\wwwroot\SegLab\COM Component*. El archivo se registra a través de la consola

de línea de comandos, cambiando de directorio hasta llegar a *C:\Inetpub\wwwroot\SegLab\COM Component* y escribiendo regsvr32 CamServer.dll. Para más información, referirse al capítulo II.1.1 Página principal., Pág. 44.

Por último, crear una carpeta llamada ImagenesCapturadas dentro de wwwroot:

C:\Inetpub\wwwroot\ImagenesCapturadas

Esta carpeta contendrá las imágenes capturadas por la aplicación Web, y por el Detector de Movimiento.

Sin embargo, esta manera de copiar la aplicación Web parece no funcionar bien. Al ser copiada a otro sistema, la aplicación Web no puede conectarse a la base de datos que contiene logins y passwords encriptados de los administradores. Esta base de datos, llamada ASPNETDB.MDF, se encuentra dentro de la carpeta ~\SegLab\App_Data, y es utilizada por la página Login.aspx (Fig. 2.1) para autenticar a los usuarios que quieran acceder al Sistema de Seguridad Remoto.

Debido a esto, para poder copiar la aplicación Web ASP.NET desde la máquina donde se desarrolló a la máquina que actuará como servidor, se optó por crear un proyecto Web nuevo con Visual Studio 2005. Se nombró igual que el proyecto original (SegLab). Esta aplicación Web se creó como aplicación IIS dentro del directorio C:\Inetpub\wwwroot. Por último, mediante ASP.NET Configuration Tool, se crearon las cuentas de acceso, roles y permisos para los administradores. Cuando se terminaron de crear las cuentas, se creó automáticamente la base de datos ASPNETDB.MDF. Posteriormente se agregó al proyecto una página llamada Login.aspx, se le agregó un control Login, y se configuró como su URL de destino a la página Default.aspx.

Se probó la página Login.aspx, la cual como fue creada en la máquina servidor, ya no tuvo problema en acceder la base de datos ASPNETDB.MDF, y validó los logins y passwords de manera correcta.

Ya que la parte de autenticación funcionaba, se prosiguió a agregar al proyecto de la máquina servidor los archivos del proyecto original. El proyecto original se copió de la máquina donde se desarrolló a una memoria flash. Posteriormente, como los dos proyectos (el original y el que se desarrolló en la máquina servidor) tienen el mismo nombre, no se tuvo que modificar el código original de ninguno de los archivos del proyecto original.

Mediante Add Existing Item (Fig. 3.6) se agregó uno por uno todos los archivos del proyecto original al proyecto de la máquina servidor. Los archivos Login.aspx, Default.aspx y web.config que se crearon en el nuevo proyecto, se remplazaron con los archivos Login.aspx, Default.aspx y web.config del proyecto original. EL ÚNICO ARCHIVO QUE NO SE REMPLAZÓ EN EL PROYECTO SEGLAB CREADO EN LA MÁQUINA DONDE SE DESEA ESTÉ LA APLICACIÓN WEB (LA MÁQUINA SERVIDOR), FUE LA BASE DE DATOS ASPNETDB.MDF. Este archivo (que es la

causa de que no se haya podido copiar la aplicación Web de la forma mencionada anteriormente), se dejó intacto en el nuevo proyecto.

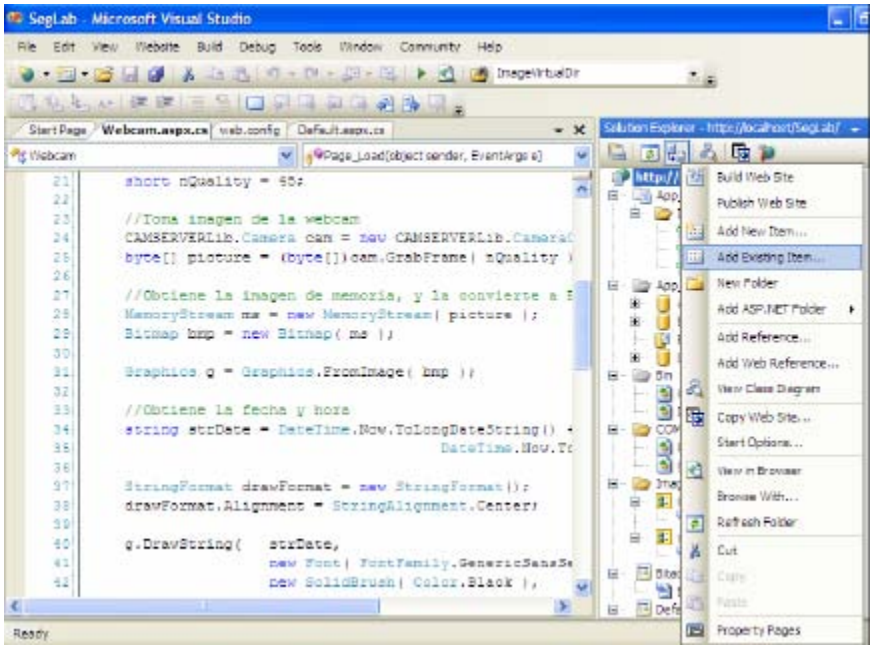

Fig. 3.6 Mediante Add Existing Item, se puede agregar archivos existentes al nuevo proyecto, por lo cual no es necesario volver a crear completamente la aplicación Web.

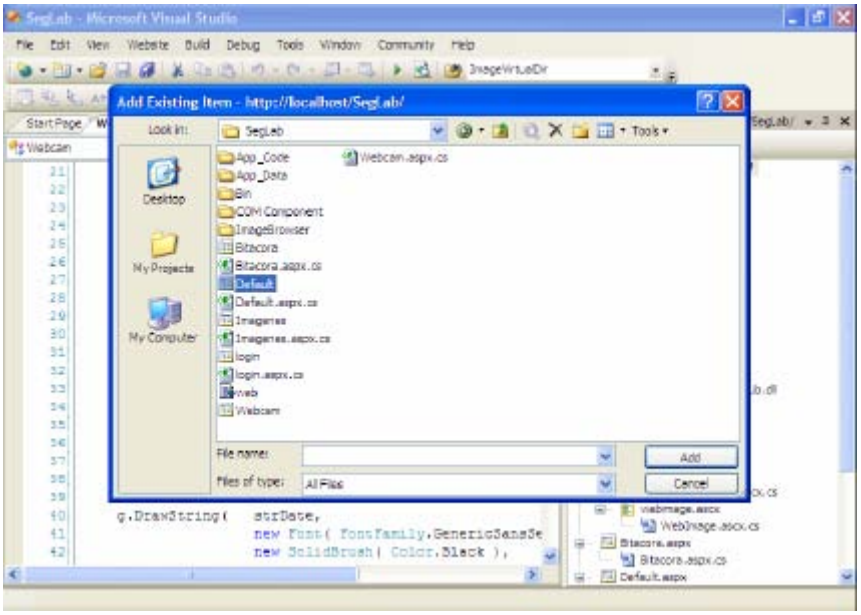

Fig. 3.6 Como el proyecto original se transportó mediante una memoria flash, hay que buscar la carpeta SegLab en nuestra memoria flash, y agregar uno por uno todos los archivos dentro de la carpeta, con excepción del archivo ASPNETDB.MDF que se encuentra dentro de App_Data.

Se agregaron todos los archivos del proyecto original en el proyecto nuevo de forma que quedaran dentro de una estructura mostrada en la figura 3.7:

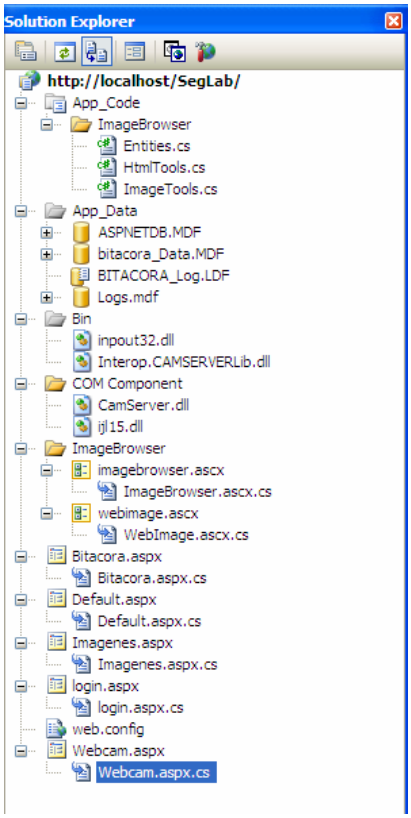

Fig. 3.7. Nuestro proyecto final (el que se encontrará en la máquina servidor) deberá contener, tal como se muestra en la imagen, todos los archivos del proyecto original que se agregaron con Add Existing Item. Recordar que el archivo ASPNETDB.MDF fue el único que no se modificó en el nuevo proyecto.

El archivo Interop.CAMSERVERLib.dll lo agrega automáticamente Visual Studio 2005 al añadir al proyecto una referencia al archivo CamServer.dll, suponiendo que previamente ya se ha registrado CamServer.dll en la máquina servidor (Para saber como registrar CamServer.dll en la máquina y como agregar al proyecto una referencia a este archivo, favor de referirse al capítulo II.1.1 Página principal, Fig. 2.6 y 2.7, en las páginas 44 y 45).

En resumen:

- Con Visual Studio 2005 instalado en la máquina que será nuestro servidor, se creó un nuevo Proyecto de Sitio Web llamado SegLab, dentro del directorio IIS.
- Se agregó al proyecto una página Web llamada login.aspx y se le agregó un control ASP.NET login. Se configuró el URL de destino del control login como Default.aspx (página creada automáticamente cuando se creó el proyecto).

- Se crearon, mediante ASP.NET Configuration Tool, un rol llamado administrador, se eligió el método de autenticación de Internet, y se creó la cuenta de administrador, añadiéndolo al rol de *administrador*.
- Después de que se comprobó que la página login.aspx realizaba la autenticación de manera correcta, se agregó uno por uno todos los archivos del Sistema de Seguridad Remoto, con excepción de la base de datos ASPNETDB.MDF. Se remplazaron también los archivos Default.aspx y Login.aspx por los del Sistema de Seguridad.
- Se construyó el proyecto para cerciorarse que todo funcionaba bien.

Después de esto, se pudo acceder sin problemas el Sistema de Seguridad Remoto desde cualquier otra máquina en la red, mediante:

http:// 132.248.59.38/SegLab/Login.aspx

Donde 132.248.59.38 es la IP del servidor.

Por último, para instalar el Detector de Movimiento, se utilizó el instalador .msi que se creó al compilar el proyecto del Detector de Movimiento. Los pasos para crear el archivo .msi se describen en el capítulo III. 1. Instalación del Sistema de Seguridad Remoto en el Laboratorio Microsoft Research, página 70.

Los solenoides utilizados en el Sistema de Seguridad Remoto, se pueden conseguir en lugares donde venden cableado eléctrico. Las piezas de aluminio que se colocaron en la parte superior de las puertas pueden ser creadas conforme al diseño mostrado en las figuras 2.16 y 2.17, Pág. 55.

El cable DB25 macho-hembra puede conseguirse en tiendas de electrónica, o se puede fabricar de manera casera mediante dos conectores DB25 macho y hembra, y un poco de cable plano. El cableado que se tendió en el Laboratorio Microsoft Research se realizó con cable para corriente calibre 18, el cual puede ser conseguido en ferreterías.

CAPITULO IV. PRUEBAS Y RESULTADOS

Después de haber instalado los solenoides dentro del Laboratorio Microsoft Research y haber tendido todo el cableado necesario, se procedió a probar el Sistema de Seguridad Remoto. Se observó que cuando se trataba de acceder a la página ASP.NET de inicio de sesión *Login.aspx* y no se introducía el login o password correcto, no se podía acceder al Sistema de Seguridad Remoto. Además, si se trataba de acceder directamente a la página *Default.aspx* sin haber iniciado sesión, o a otra página o archivo del Sistema de Seguridad que no fuera *Login.aspx*, automáticamente se nos redireccionaba a la página de inicio *Login.aspx*. Por lo tanto, se demostró que los métodos de autenticación implementados funcionan de manera correcta.

No se detectó ningún problema en el funcionamiento de los solenoides. Se ingresó al Sistema de Seguridad Remoto desde una computadora a través de Internet. Se pudieron activar de manera inmediata los solenoides. Se observó satisfactoriamente que los solenoides sólo permanecen activados durante 6 segundos, después de los cuales automáticamente se desactivaron. Durante estos 6 segundos las puertas pudieron ser abiertas libremente. Si después de los 6 segundos las puertas permanecían abiertas, éstas podían deslizarse libremente hasta que se encontraban en la posición de cerrada, en donde los solenoides las atrancaron, con lo cual estos solenoides se tenían que activar de nuevo a través de la página ASP.NET del Sistema de Seguridad Remoto, para poder abrir las puertas.

Además, se pudo acceder al Sistema de Seguridad Remoto a través de un PDA IPaq, el cual tiene acceso a Internet a través de la red inalámbrica del edificio Valdés Vallejo. Se pudo observar que la página ASP.NET sigue teniendo la misma funcionalidad que si se hubiera accedido con una PC normal. El único problema que se detectó es que el código JavaScript que refresca cada 5 segundos la imagen de la webcam dentro de la página *Default.aspx*, no se ejecuta correctamente en el explorador Web del PDA. Por lo tanto se tiene que actualizar la imagen manualmente. Todas las demás partes de la página ASP.NET funcionaron correctamente.

Con esto, cuando alguno de los administradores del laboratorio desee ingresar a las instalaciones, sólo necesitará tener a su alcance un PDA conectado a la red inalámbrica del edificio Valdés Vallejo, y en el momento en que se encuentre frente a las puertas del laboratorio, podrá abrir las puertas con su PDA en mano.

Por último, la aplicación Detector de Movimiento funcionó sin problemas. Se ejecutó durante varia horas, y no se detectaron problemas de funcionalidad. El único problema de la aplicación es que no funciona de noche cuando las luces están apagadas, debido a que la webcam utilizada no posee visión nocturna o *Night Vision*.

CONCLUSIONES

Se pudo crear de manera satisfactoria el Sistema de Seguridad para el Laboratorio Microsoft Research. Para ello, se creó la aplicación Web ASP.NET que a través de Internet puede ejecutar el código necesario para transmitir datos a través del puerto paralelo del servidor, y que también puede interactuar con la webcam para realizar el monitoreo. Se desarrolló un circuito que sirve como "puente" entre los solenoides y el puerto paralelo, de manera que éstos puedan interactuar de manera segura. Este circuito se desarrolló también con el propósito de limitar el tiempo que los solenoides permanecen activados. Debido a algunas limitaciones de ASP.NET, el detector de movimiento se desarrolló como una aplicación Windows con Visual C#, la cual sólo se ejecuta de manera local en la computadora que aloja la página ASP.NET, haciendo uso de la misma webcam que utiliza la página ASP.NET.

El Sistema de Seguridad Remoto puede ser mejorado de manera sustancial. Por ejemplo, debido a que parte de las puertas del Laboratorio Microsoft Research son de vidrio, alguna persona podría romper los vidrios y fácilmente entrar al laboratorio. Se podrían remplazar estas puertas por unas que fueran más seguras, como podrían ser unas puertas de metal. También se podría utilizar chapas eléctricas especiales en lugar de los solenoides. Estas chapas eléctricas no pudieron ser utilizadas durante el desarrollo de esta tesis, debido a su alto costo, cuyos precios cotizados en Internet oscilan entre los $75 y $100 USD. [20]

En el futuro, se podría agregar mayor funcionalidad al Sistema de Seguridad Remoto, como por ejemplo, conseguir una webcam con visión nocturna para que se pudiera monitorear el laboratorio de noche. También se podría desarrollar un arreglo de motores y engranes, o pistones hidráulicos conectados a las puertas corredizas, con el fin de que se pudiera controlar el movimiento de las puertas directamente con la aplicación Web desarrollada a lo largo de esta tesis. Estos motores o pistones podrían ser controlados a través del puerto paralelo de la computadora, de la misma manera que se controlan los solenoides.

Se podría agregar movimiento a la webcam, para así poder moverla desde Internet. Para esto sería necesario montar la webcam sobre un servo motor, el cual movería la webcam en determinados grados, instalar sensores ópticos o de otro tipo alrededor de la webcam con el fin de detectar la posición en la que se encuentre la webcam, y posiblemente un microcontrolador o PIC, que recibiera instrucciones a través del puerto paralelo, las procesara y moviera el servo motor, así como procesar la información de los sensores y mandarlos al servidor a través del puerto paralelo.

También se podría instalar un termómetro electrónico dentro del laboratorio, como por ejemplo un circuito LM35 o algún otro termómetro electrónico conectado a un microcontrolador o PIC, con el fin de detectar y mostrar la temperatura del laboratorio en la página ASP.NET del Sistema de Seguridad Remoto. Con esto se podrían evitar fallas en el equipo de cómputo del laboratorio debido al calor extremo que pudiera generarse dentro de este laboratorio.

Por último, se constató que mediante la tecnología Microsoft .NET se optimiza considerablemente el tiempo de desarrollo de la aplicación, ya que con .NET el código que fue necesario escribir fue notablemente menor que el que hubiera sido necesario de haber utilizado otro lenguaje de programación similar, como por ejemplo Java y JSP.

La facilidad con la que se pueden establecer conexiones con bases de datos, sus cajas de herramientas o *Toolboxes*, la interfaz gráfica de su IDE, la tecnología Intellisense, y sobre todo ASP.NET 2.0, permitieron que el código escrito con la tecnología .NET se haya reducido en aproximadamente 60%, tal como Microsoft lo promete, y sobre todo, que el tiempo de desarrollo se haya reducido de manera increíble.

ANEXOS

Código fuente de la aplicación Web del Sistema de Seguridad Remoto.

Login.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Login.aspx.cs" Inherits="Login" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Inicie sesión</title>
</head>
<body bgcolor="#5d7b9d">
    <form id="form1" runat="server">
    <div>
        <asp:Login ID="Login1" runat="server" BackColor="#F7F6F3" BorderColor="#E6E2D8"
BorderPadding="4"
            BorderStyle="Solid" BorderWidth="1px" DestinationPageUrl="~/Default.aspx"
DisplayRememberMe="False"
            FailureText="Contraseña incorrecta. Intente de nuevo." Font-Names="Verdana"
Font-Size="0.8em"
            ForeColor="#333333" Height="240px" InstructionText="Introduzca su login y
password para ingresar al sistema."
            LoginButtonText="Ingresar" PasswordRequiredErrorMessage="Password necesario."
            Style="z-index: 100; left: 157px; position: absolute; top: 134px"
TitleText="Inicie Sesión."
            UserNameLabelText="Login:" UserNameRequiredErrorMessage="Login necesario."
Width="366px" FailureAction="RedirectToLoginPage">
            <TitleTextStyle BackColor="#5D7B9D" Font-Bold="True" Font-Size="0.9em"
ForeColor="White" />
            <InstructionTextStyle Font-Italic="True" ForeColor="Black" />
            <TextBoxStyle Font-Size="0.8em" />
            <LoginButtonStyle BackColor="#FFFBFF" BorderColor="#CCCCCC" BorderStyle="Solid"
BorderWidth="1px"
                Font-Names="Verdana" Font-Size="0.8em" ForeColor="#284775" />
        </asp:Login>
        <asp:Label ID="Label1" runat="server" BackColor="#F7F6F3" BorderStyle="Outset" Font-
Bold="False"
            Font-Overline="True" Font-Size="Larger" Style="z-index: 102; left: 67px;
position: absolute;
            top: 76px" Text="Bienvenido al Sistema de Seguridad Remoto del Laboratorio
Microsoft Research" Width="613px"></asp:Label>

    </div>
    </form>
</body>
</html>
```

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs"
Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Bienvenido al Sistema de Seguridad del Laboratorio de Microsoft</title>
```

```
  <script language="JavaScript" type="text/javascript">
<!--
var camImage = 'Webcam.aspx';
var refreshIntervalSeconds = 5;

var secondsLeft = refreshIntervalSeconds;

function startClock() {
  if (secondsLeft > 0) {
    if (secondsLeft < 5) {
      document.webCam.timer.value = secondsLeft;
    } else {
      document.webCam.timer.value = secondsLeft;
    }
    secondsLeft = secondsLeft - 1
    timerID = setTimeout('startClock()', 1000);
  } else {
  date = new Date();
  imageNumber = date.getTime();
  document.webCamImage.src = camImage + '?' + imageNumber;
  secondsLeft = refreshIntervalSeconds;
  startClock();
  }
}


// -->
</script>

</head>

<body onLoad="startClock()" bgcolor="#5d7b9d">

    <form id="form1" runat="server">
    <div>
    <img src="Webcam.aspx" name="webCamImage" alt="web cam" title="web cam" width="320"
height="240" border="0" style="z-index: 108; left: 85px; position: absolute; top: 94px"
id="IMG1" language="javascript" onclick="return IMG1_onclick()" runat="server">
    <asp:LoginStatus ID="LoginStatus1" runat="server" ForeColor="White"
LogoutAction="RedirectToLoginPage"
            LogoutText="Cerrar Sesión" OnLoggingOut="LoginStatus1_LoggingOut" Style="z-
index: 100;
            left: 269px; position: absolute; top: 10px" Font-Size="Medium" />
    <asp:Button ID="Button1" runat="server" Height="35px" Style="z-index: 101; left: 438px;
            position: absolute; top: 113px" Width="358px" OnClick="Button1_Click" Font-
Size="X-Small" Text="Abrir Puerta Principal" />
        <asp:Button ID="Button2" runat="server" Height="36px" OnClick="Button2_Click"
Style="z-index: 102;
            left: 438px; position: absolute; top: 160px" Text="Abrir Puerta Del Servidor"
Width="355px" Font-Size="X-Small" />
        <asp:Label ID="Label1" runat="server" Font-Size="Small" ForeColor="White"
Height="82px"
            Style="z-index: 103; left: 438px; position: absolute; top: 23px" Text="La puerta
se cerrará automáticamente después de n segundos"
            Width="196px"></asp:Label>
         
       <asp:LoginName ID="LoginName1" runat="server" BackColor="#5D7B9D" ForeColor="Black"
            Style="z-index: 104; left: 55px; position: absolute; top: 14px" Width="170px"
ToolTip="Usuario actual" Font-Size="Larger" />
           <asp:SqlDataSource ID="SqlDataSource1"
            runat="server" ConnectionString="<%$ ConnectionStrings:LogsConnectionString %>"
            SelectCommand="SELECT DISTINCT * FROM [Logs] ORDER BY [horain] DESC"
OldValuesParameterFormatString="original_{0}" InsertCommand="INSERT INTO Logs(usuario, ip,
hostname, horain, horaout) VALUES (@usuario, @ip, @host, @horain, @horaout)"
UpdateCommand="UPDATE Logs SET horaout = @horaout WHERE (horain = @horain) AND (usuario =
@usuario)">
            <UpdateParameters>
                <asp:SessionParameter Name="usuario" SessionField="usuario" />
                <asp:SessionParameter Name="ip" SessionField="ip" />
```

```
                <asp:SessionParameter Name="host" SessionField="host" />
                <asp:SessionParameter Name="horain" SessionField="horain" />
                <asp:SessionParameter Name="horaout" SessionField="horaout" />
            </UpdateParameters>
            <InsertParameters>
                <asp:SessionParameter Name="usuario" SessionField="usuario" />
                <asp:SessionParameter Name="ip" SessionField="ip" />
                <asp:SessionParameter Name="host" SessionField="host" />
                <asp:SessionParameter Name="horain" SessionField="horain" />
                <asp:SessionParameter Name="horaout" SessionField="horaout" />
            </InsertParameters>
        </asp:SqlDataSource>
                   <br />
          <br />
         
         <input id="Button3" type="button" value="Registro de entradas"
onClick="window.open('./Bitacora.aspx','poup','toolbar=yes,status=yes,scrollbars=yes,_resiza
ble=yes,width=720,height=440')" style="z-index: 109; left: 438px; position: absolute; top:
242px; width: 189px; height: 22px;" runat="server" />              
   
         <asp:Button ID="Button5" runat="server" OnClick="Button5_Click" Style="z-index:
106;
             left: 438px; position: absolute; top: 276px" Text="Guardar imagen"
Width="189px" Height="22px" />
         
                    <input id="Button7" type="button" value="Visor de imágenes"
onClick="window.open('./Imagenes.aspx','poup','toolbar=yes,status=yes,scrollbars=yes,_resiza
ble=yes,width=720,height=440')" style="z-index: 110; left: 438px; position: absolute; top:
308px; width: 189px; height: 22px;" runat="server"  />            

          <br />
          <br />
    </div>
    </form>

    <form name="webCam" action="#">
      <span style="font-size: 14pt"><br />
          <br />

          <br />
          <br />
          <br />
          <br />
          <br />
          <br />
          <br />
           <br />
          <br />
          <br />
          <br />
                        La imagen se actualizará en:
</span>
        <input type="text" name="timer"  size="2" style="background-color: #5d7b9d; color:
white;">
        <span style="font-size: 14pt">segundos. 
        </span>
</form>
</body>
</html>
```

Default.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
```

```
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Runtime.InteropServices;
using System.Threading;
using System.IO;
using System.IO.Ports;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;

public partial class _Default : System.Web.UI.Page
{
    public string user;
    public string timein;
    public string timeout;
    public string ip;
    public string proceso;
    private CAMSERVERLib.Camera cam = new CAMSERVERLib.CameraClass();
    short nQuality = 45;
    private string hora = "";

    public class PortAccess
    {
        [DllImport(@"C:\Inetpub\wwwroot\SegLab\Bin\inpout32.dll", EntryPoint = "Out32")]
        public static extern void Output(int address, int v);
        [DllImport(@"C:\Inetpub\wwwroot\SegLab\Bin\inpout32.dll", EntryPoint = "Inp32")]
        public static extern int Input(int address);
    }


 protected void Page_Load(object sender, EventArgs e)
    {

            if (!IsPostBack)
            {

                Session["usuario"] = User.Identity.Name.ToString();
                Session["horain"] = Server.HtmlEncode(DateTime.Now.ToString());
                Session["ip"] = Server.HtmlEncode(Request.UserHostAddress);
                Session["host"] = Server.HtmlEncode(Request.UserHostName);
                Session["horaout"] = "";
                SqlDataSource1.Insert();
                PortAccess.Output(888, 000);
            }

        #region roleadmin
        if (Roles.IsUserInRole("administrador") == false)
        {

                Button1.Visible = false;
                Button2.Visible = false;
                Button3.Visible = false;
                Label1.Visible = false;
                Button5.Visible = false;
                Button7.Visible = false;

        }

       #endregion

    }


    protected void Button1_Click(object sender, EventArgs e)
    {
```

```
        PortAccess.Output(890, 32);
        int readdata1 = (int)PortAccess.Input(888);
        //(0,cerradura servidor,cerradura principal)
        switch (readdata1)
        {
            case 000: //cerradura principal cerrada, servidor cerrada
                {
                    PortAccess.Output(888, 001);
                    Button1.Text = "Abrir Puerta Principal";
                    Button2.Text = "Abrir Puerta Del Servidor";
                    break;
                }
            case 001: //cerradura principal abierta, servidor cerrada
                {
                    PortAccess.Output(888, 000);

                    Button1.Text = "Por favor, vuelva a oprimir este botón para abrir la
puerta";
                    Button2.Text = "Abrir Puerta Del Servidor";
                    break;
                }
            case 010: //cerradura principal cerrada, servidor abierta
                {
                    PortAccess.Output(888, 001);
                    Button1.Text = "Abrir Puerta Principal";
                    Button2.Text = "Abrir Puerta Del Servidor";
                    break;
                }
            case 011:  //cerradura principal abierta, servidor abierta
                {
                    PortAccess.Output(888, 000);

                    Button1.Text = "Por favor, vuelva a oprimir este botón para abrir la
puerta";
                    Button2.Text = "Abrir Puerta Del Servidor";
                    break;
                }
            default:
                {
                    PortAccess.Output(888, 000);

                    Button1.Text = "Abrir Puerta Principal";
                    Button2.Text = "Abrir Puerta Del Servidor";
                    break;
                }
        }
    }
    protected void LoginStatus1_LoggingOut(object sender, LoginCancelEventArgs e)
    {


        Session["horaout"] = Server.HtmlEncode(DateTime.Now.ToString());
        SqlDataSource1.Update();

    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        PortAccess.Output(890, 32);
        int readdata1 = (int)PortAccess.Input(888);
        //(0,cerradura servidor,cerradura principal)
        switch (readdata1)
        {
            case 000:  //cerradura principal cerrada, servidor cerrada
                {
                    PortAccess.Output(888, 010);
                    Button1.Text = "Abrir Puerta Principal";
                    Button2.Text = "Abrir Puerta Del Servidor";
                    break;
                }
```

```
            case 001: //cerradura principal abierta, servidor cerrada
                {
                    PortAccess.Output(888, 010);

                    Button1.Text = "Abrir Puerta Principal";
                    Button2.Text = "Abrir Puerta Del Servidor";
                    break;
                }
            case 010: //cerradura principal cerrada, servidor abierta
                {


                    PortAccess.Output(888, 000);

                    Button2.Text = "Por favor, vuelva a oprimir este botón para abrir la
puerta";
                    Button1.Text = "Abrir Puerta Principal";
                    break;
                }
            case 011:  //cerradura principal abierta, servidor abierta
                {
                    PortAccess.Output(888, 000);

                    Button2.Text = "Por favor, vuelva a oprimir este botón para abrir la
puerta";
                    Button1.Text = "Abrir Puerta Principal";
                    break;
                }
            default:
                {
                    PortAccess.Output(888, 000);

                    Button1.Text = "Abrir Puerta Principal";
                    Button2.Text = "Abrir Puerta Del Servidor";
                    break;
                }
        }
    }


    protected void SqlDataSource1_Selecting(object sender, SqlDataSourceSelectingEventArgs
e)
    {

    }


    protected void Button5_Click(object sender, EventArgs e)
    {
        byte[] imagen = (byte[])cam.GrabFrame(short.MaxValue);
        MemoryStream ms = new MemoryStream(imagen);
        Bitmap bmp = new Bitmap(ms);
        #region Graphics

        Graphics g = Graphics.FromImage(bmp);

        string strDate = DateTime.Now.ToLongDateString() + " - " +
                                            DateTime.Now.ToLongTimeString();

        StringFormat drawFormat = new StringFormat();
        drawFormat.Alignment = StringAlignment.Center;

        g.DrawString(strDate,
                        new Font(FontFamily.GenericSansSerif, 12),
                        new SolidBrush(Color.Black),
                        new RectangleF(1, 1, 320, 240),
                        drawFormat
                    );

        g.DrawString(strDate,
```

```
                        new Font(FontFamily.GenericSansSerif, 12),
                        new SolidBrush(Color.White),
                        new RectangleF(0, 0, 320, 240),
                        drawFormat
                    );

        //codecs
        ImageCodecInfo[] icf = ImageCodecInfo.GetImageEncoders();

        EncoderParameters encps = new EncoderParameters(1);
        EncoderParameter encp = new EncoderParameter(System.Drawing.Imaging.Encoder.Quality,
                                                     (long)nQuality);


        encps.Param[0] = encp;

        #endregion
        hora = System.DateTime.Now.Millisecond.ToString() + ".jpeg";
        bmp.Save(@"C:\Inetpub\wwwroot\ImagenesCapturadas\imagen" + hora, icf[1], encps);

    }


    protected void Button7_ServerClick(object sender, EventArgs e)
    {

    }
}
```

Webcam.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Webcam.aspx.cs" Inherits="Webcam" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    </div>
    </form>
</body>
</html>
```

Webcam.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
```

```
using System.IO;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;

public partial class Webcam : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    //Calidad de compresión JPEG
    short nQuality = 45;

    //Toma imagen de la webcam
    CAMSERVERLib.Camera cam = new CAMSERVERLib.CameraClass();
    byte[] picture = (byte[])cam.GrabFrame( nQuality );

    //Obtiene la imagen de memoria, y la convierte a Bitmap.
    MemoryStream ms = new MemoryStream( picture );
    Bitmap bmp = new Bitmap( ms );

    Graphics g = Graphics.FromImage( bmp );

    //Obtiene la fecha y hora
    string strDate = DateTime.Now.ToLongDateString() + " - " +
                                        DateTime.Now.ToLongTimeString();

       StringFormat drawFormat = new StringFormat();
    drawFormat.Alignment = StringAlignment.Center;


    g.DrawString(   strDate,
                    new Font( FontFamily.GenericSansSerif, 12 ),
                    new SolidBrush( Color.Black ),
                    new RectangleF( 1,1,320,240 ),
                    drawFormat
                );

    //Inserta la fecha dentro de la imagen
    g.DrawString(   strDate,
                    new Font( FontFamily.GenericSansSerif, 12 ),
                    new SolidBrush( Color.White ),
                    new RectangleF( 0,0,320,240 ),
                    drawFormat
                );

    //codecs
    ImageCodecInfo[] icf = ImageCodecInfo.GetImageEncoders();
    EncoderParameters encps = new EncoderParameters( 1 );
    EncoderParameter encp = new EncoderParameter( System.Drawing.Imaging.Encoder.Quality,
                                                  (long) nQuality );

    encps.Param[0] = encp;

    //manda a memoria la imagen procesada
    bmp.Save( Response.OutputStream, icf[1], encps );

    //Libera los recursos
    g.Dispose();
    bmp.Dispose();
}
    }
```

Bitacora.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Bitacora.aspx.cs"
Inherits="Bitacora" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body bgcolor="#5b7b9d">
    <form id="form1" runat="server">
    <div>
         </div>
           
        <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$
ConnectionStrings:LogsSeguridad_DataConnectionString %>"
            SelectCommand="SELECT DISTINCT * FROM [Logs] ORDER BY [Horain]"
DeleteCommand="DELETE FROM Logs">
        </asp:SqlDataSource>
        <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Style="z-index: 100;
            left: 412px; position: absolute; top: 67px" Text="Borrar log" />
         
        <asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
CellPadding="4"
            DataKeyNames="Username,Horain" DataSourceID="SqlDataSource1" ForeColor="#333333"
            GridLines="None" Height="196px" Style="z-index: 102; left: 64px; position:
absolute;
            top: 140px" Width="516px" AllowSorting="True">
            <FooterStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
            <Columns>
                <asp:BoundField DataField="Username" HeaderText="Username" ReadOnly="True"
SortExpression="Username" />
                <asp:BoundField DataField="IP" HeaderText="IP" SortExpression="IP" />
                <asp:BoundField DataField="Hostname" HeaderText="Hostname"
SortExpression="Hostname" />
                <asp:BoundField DataField="Horain" HeaderText="Horain" ReadOnly="True"
SortExpression="Horain" />
                <asp:BoundField DataField="Horaout" HeaderText="Horaout"
SortExpression="Horaout" />
            </Columns>
            <RowStyle BackColor="#F7F6F3" ForeColor="#333333" />
            <EditRowStyle BackColor="#999999" />
            <SelectedRowStyle BackColor="#E2DED6" Font-Bold="True" ForeColor="#333333" />
            <PagerStyle BackColor="#284775" ForeColor="White" HorizontalAlign="Center" />
            <HeaderStyle BackColor="#5D7B9D" Font-Bold="True" ForeColor="White" />
            <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
        </asp:GridView>
    </form>
</body>
</html>
```

Bitacora.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.IO;

public partial class Bitacora : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
```

```
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        SqlDataSource1.Delete();
    }
}
```

Imagenes.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Imagenes.aspx.cs"
Inherits="Web.Imagenes" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body bgcolor="#5d7b9d">
    <form id="form1" runat="server">
    <div>
              <asp:Panel id="ImagePanel" runat="server" Width="100%"></asp:Panel>

    </div>
    </form>
</body>
</html>
```

Imagenes.aspx.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.ComponentModel;
using System.Drawing;
using System.Web.SessionState;
using ImageBrowser.Entities;
using System.IO;

namespace Web
{
    public partial class Imagenes : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.Panel MainPanel;

        protected void Page_Load(object sender, EventArgs e)
        {

            string page = Request.ApplicationPath + @"/ImageBrowser/ImageBrowser.ascx";

            if (Request["page"] != null && Request["page"].Length > 0)
                page = Request.ApplicationPath + @"/ImageBrowser/" + Request["page"];

            ImagePanel.Controls.Add(new UserControl().LoadControl(page));



        }
    }
```

```
}
```

imagebrowser.ascx

```
<%@ Control Language="c#" Inherits="ImageBrowser.Controls.ImageBrowser"
CodeFile="ImageBrowser.ascx.cs" %>
<P><asp:panel id="ImageBrowserPanel" runat="server" EnableViewState="False"></asp:panel></P>
```

imagebrowser.ascx.cs

```
namespace ImageBrowser.Controls
{
	using System;
	using System.Data;
	using System.Drawing;
	using System.Web;
	using System.Web.UI.WebControls;
	using System.Web.UI.HtmlControls;
	using Entities;

	/// <summary>
	///		Summary description for ImageBrowser.
	/// </summary>
	public partial  class ImageBrowser : System.Web.UI.UserControl
	{

		protected void Page_Load(object sender, System.EventArgs e)
		{
	    string path = ImageTools.GetPath(Request["path"]);

	     DirectoryWrapper data = new DirectoryWrapper(path);


	     ImageBrowserPanel.Controls.Add(HtmlTools.RenderImageTable(7, 0, data,
Request.Path + "?page=" + "WebImage.ascx&path="));

	}


		#region Web Form Designer generated code
		override protected void OnInit(EventArgs e)
		{
			//
			// CODEGEN: This call is required by the ASP.NET Web Form Designer.
			//
			InitializeComponent();
			base.OnInit(e);
		}

		///		Required method for Designer support - do not modify
		///		the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{

		}
		#endregion
	}
}
```

webimage.ascx

```
<%@ Control language="c#" Inherits="ImageBrowser.WebImage" enableViewState="False"
CodeFile="WebImage.ascx.cs" %>
<asp:HyperLink id="image" runat="server" Target=_blank>Imagen</asp:HyperLink>
```

webimage.ascx.cs

```
namespace ImageBrowser.Controls
{
	using System;
	using System.Data;
	using System.Drawing;
	using System.Web;
	using System.Web.UI.WebControls;
	using System.Web.UI.HtmlControls;
	using Entities;

	/// <summary>
	///		Summary description for ImageBrowser.
	/// </summary>
	public partial  class ImageBrowser : System.Web.UI.UserControl
	{

		protected void Page_Load(object sender, System.EventArgs e)
		{
	    string path = ImageTools.GetPath(Request["path"]);

	    DirectoryWrapper data = new DirectoryWrapper(path);

	    ImageBrowserPanel.Controls.Add(HtmlTools.RenderImageTable(7, 0, data,
Request.Path + "?page=" + "WebImage.ascx&path="));

	 }


		#region Web Form Designer generated code
		override protected void OnInit(EventArgs e)
		{
			//
			// CODEGEN: This call is required by the ASP.NET Web Form Designer.
			//
			InitializeComponent();
			base.OnInit(e);
		}

		///		Required method for Designer support - do not modify
		///		the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{

		}
		#endregion
	}
}
```

Entities.cs

```
using System;
using System.Drawing;
using System.IO;
using System.Drawing.Imaging;
using System.Collections;

namespace ImageBrowser.Entities
{
	/// <summary>
	/// Información del contenido del directorio
	/// </summary>
	public class DirectoryWrapper
```

```
	{
		private ArrayList directories =  new ArrayList();
		private ArrayList images =  new ArrayList();
		private string directory;

		/// <summary>
		///
		/// </summary>
		/// <param name="dir">directorio con el contenido</param>
		public DirectoryWrapper(string dir)
		{
			directory = dir;


			foreach ( string s in
Directory.GetDirectories(ImageTools.RootDirectory + "/" + directory) )
			{
				string[] path = s.Replace( "\\", "/" ).Split('/');

				if ( path[path.Length - 1] != "thumbs" && path[path.Length -
1] != "webpics" && path[path.Length - 1][0] != '_' )
				{

	directories.Add(ImageTools.GetSubDirectoryWrapper(directory + "/" + path[path.Length
- 1]));
				}
			}

			//agrega imagenes
			foreach ( string s in Directory.GetFiles(ImageTools.RootDirectory +
"/" + directory) )
			{
				if (  s[0] != '_' )
				{
					string extension = null;
					if (s.IndexOf(".") > 0)
					{
						string[] parts = s.Split('.');
						extension = parts[parts.Length - 1];
					}
					if ( extension == null ) continue;

					extension = extension.ToLower();

					if ( extension == "jpg" ||
						extension == "png" ||
			  extension == "jpeg" ||
						extension == "gif" )
					{
						string[] path = s.Replace(@"\","/").Split('/');
						images.Add(ImageTools.GetImageWrapper(directory
+ "/" + path[path.Length - 1]));
					}
				}
			}
		}

		/// <summary>

		/// </summary>
		public ArrayList Directories
		{
			get
			{
				return directories;
			}
		}

		/// <summary>
```

```
		/// </summary>
		public ArrayList Images
		{
			get
			{
				return images;
			}
		}

		/// <summary>

		/// </summary>
		public string Name
		{
			get
			{
				string[] paths = directory.Replace(@"\","/").Split('/');
				return paths[paths.Length - 1];
			}
		}


		public string Blurb
		{
			get
			{
				FileStream fs = null;
				try
				{

					string[] dirs = directory.Replace(@"\","/").Split('/');
					string name = dirs[dirs.Length - 1];

					if ( name == null || name.Length == 0 ) name = "Home";

					if ( File.Exists(ImageTools.RootDirectory + "/" +
directory + "/" + name + ".txt" ) )
					{
						fs = File.OpenRead(ImageTools.RootDirectory +
"/" + directory + "/" + name + ".txt" );

						byte[] b = new Byte[fs.Length];
						fs.Read(b,0,(int)fs.Length);
						fs.Close();
						return
System.Text.ASCIIEncoding.ASCII.GetString(b);
					}
				}
				catch(Exception)
				{
					if ( fs != null && fs.CanRead ) fs.Close();
				}
				return "";
			}
		}
	}


	public class SubDirectoryWrapper
	{
		private string directory;
		private string name;
		private string defaultImage;

		public SubDirectoryWrapper(string dir, string defaultFolderImage)
		{
			directory = dir;
			defaultImage = defaultFolderImage;

			string[] dirs = directory.Replace(@"\","/").Split('/');
```

```
                name = dirs[dirs.Length - 1];
        }

        public string Name
        {
                get
                {
                        return name;
                }
        }


        public string Src
        {
                get
                {
                        string parent = ImageTools.RootDirectory + "/" + directory;
                        string[] files = Directory.GetFiles( parent, name + @".*");

                        if ( files.Length > 0 )
                        {
                                string file = null;

                                foreach ( string s in files )
                                {
                                        string extension = null;
                                        if (s.IndexOf(".") > 0)
                                        {
                                                string[] parts = s.Split('.');
                                                extension = parts[parts.Length - 1];
                                        }
                                        if ( extension == null ) continue;

                                        extension = extension.ToLower();

                                        if ( extension == "jpg" ||
                                                extension == "png" ||
                                extension == "jpeg" ||
                                                extension == "gif" )
                                        {
                                                string[] filepath =
s.Replace(@"\","/").Split('/');
                                                file = directory + "/" +
filepath[filepath.Length - 1];
                                                break;
                                        }
                                }
                                if ( file != null )
                                {
                                        string[] filename =
file.Replace(@"\",@"/").Split('/');

                                        string thumb =
string.Join("/",filename,0,filename.Length - 1) + "/thumbs/" + filename[filename.Length -
1];

                                        ImageTools.CreateImage(file,thumb,100);
                                        return ImageTools.VirtualDirectory + thumb;
                                }
                        }
                        return defaultImage;
                }
        }

        public string HREF
        {
                get
                {
                        return ImageTools.VirtualDirectory + directory;
                }
```

```
		}

	}


	public class ImageWrapper
	{
		private string file;
		private string name;

		public ImageWrapper(string fileName)
		{
			file = fileName;
			string[] dirs = file.Replace(@"\","/").Split('/');
			name = dirs[dirs.Length - 1];

		}


		public string Name
		{
			get
			{
				return name;
			}
		}


		public string WebImageHref
		{
			get
			{
				string[] name = file.Split('/');

				string thumb = string.Join("/",name,0,name.Length - 1) +
"/webpics/" + name[name.Length - 1];


	ImageTools.CreateImage(ImageTools.GetPath(file),ImageTools.GetPath(thumb),640);


				return ImageTools.VirtualDirectory +
ImageTools.GetPath(thumb);
			}
		}


		public string FullImageHref
		{
			get
			{
				return ImageTools.VirtualDirectory + ImageTools.GetPath(file);
			}
		}


		public string ThumbHref
		{
			get
			{
				string[] name = file.Split('/');

				string thumb = string.Join("/",name,0,name.Length - 1) +
"/thumbs/" + name[name.Length - 1];


	ImageTools.CreateImage(ImageTools.GetPath(file),ImageTools.GetPath(thumb),100);

				return ImageTools.VirtualDirectory +
ImageTools.GetPath(thumb);
```

```
                        }
                }
        }
}
```

HtmlTools.cs

```
using System;
using System.Web.UI.WebControls;
using System.Drawing;

using ImageBrowser.Entities;

namespace ImageBrowser
{

        public class HtmlTools
        {

                private HtmlTools(){}


                public static Table RenderImageTable(int x, int y, DirectoryWrapper data,
string url)
                {
                        Table table = new Table();
                        table.Width = Unit.Percentage(100);

                        TableRow tr = null;

                        foreach ( ImageWrapper image in data.Images )
                        {
                                if ( tr == null ) tr = new TableRow();

                                HyperLink h = new HyperLink();
                                h.ImageUrl = image.ThumbHref;
                                h.NavigateUrl = url + image.FullImageHref;
                                h.Text = image.Name;


                                TableCell td = new TableCell();
                                td.Controls.Add(h);
                                tr.Cells.Add(td);

                                if ( tr.Cells.Count == x )
                                {
                                        table.Rows.Add(tr);
                                        tr = null;
                                }
                        }

                        if ( tr != null ) table.Rows.Add(tr);

                        return table;
                }


                public static Literal BR
                {
                        get
                        {
                                Literal l = new Literal();
                                l.Text = "<BR>";
                                return l;
                        }
                }


                public static Literal HR
```

```
		{
			get
			{
				Literal l = new Literal();
				l.Text = "<HR>";
				return l;
			}
		}
	}
}
```

ImageTools.cs

```
using System;
using ImageBrowser.Entities;
using System.IO;
using System.Drawing;
using System.Drawing.Imaging;
using System.Web;

namespace ImageBrowser
{
	public class ImageTools
	{
		private ImageTools(){}

		public static string VirtualDirectory =
System.Configuration.ConfigurationSettings.AppSettings["ImageVirtualDir"];

		public static string RootDirectory =
HttpContext.Current.Server.MapPath(VirtualDirectory);

		public static SubDirectoryWrapper GetSubDirectoryWrapper(string dir)
		{
			return new SubDirectoryWrapper(dir,
HttpContext.Current.Request.ApplicationPath + "/folder.jpeg");
		}

		public static ImageWrapper GetImageWrapper(string file)
		{
			return new ImageWrapper(file);
		}

		public static string GetPath(string queryPath)
		{
			string path = "";
            if (queryPath != null && queryPath.Length > 0)
            {
                path = queryPath.Replace(@"\", "/");
            }
            if (path.StartsWith(VirtualDirectory))
            {
                path = path.Substring(VirtualDirectory.Length, path.Length -
VirtualDirectory.Length);
            }
			return path;
		}

		public static string CreateImage( string src, string dest, int width )
		{
```

```
                        string description = null;

                        try
                        {
                                if ( File.Exists(RootDirectory + "/" + dest) ) return
description;

                                string path = Directory.GetParent(RootDirectory + "/" +
dest).FullName;

                                if ( ! Directory.Exists(path))
Directory.CreateDirectory(path);


                                Image image = Image.FromFile(RootDirectory + "/" + src);

                                int y =
(int)(((double)((double)width/(double)image.Size.Width)) * (double)image.Size.Height);

                                Image thumb;


                 if (width > image.Width)
                 {
                     thumb = image;
                 }
                 else if (width > 200)
                 {
                     thumb = Resize(new Bitmap(image), width, y, (bool)(width > 200));
                 }
                 else
                 {

                     Image.GetThumbnailImageAbort myCallback = new
Image.GetThumbnailImageAbort(ThumbnailCallback);
                     thumb = image.GetThumbnailImage(width, y, myCallback, IntPtr.Zero);
                 }
                     thumb.Save(RootDirectory + "/" + dest, ImageFormat.Jpeg);
                       }
                       catch(Exception){}

                       return description;
                }


                public static bool ThumbnailCallback()
                {
                        return false;
                }


                public static Bitmap Resize(Bitmap b, int nWidth, int nHeight, bool
bBilinear)
                {
                        //Bitmap bTemp = (Bitmap)b.Clone();
                        Bitmap bTemp = b;

                        b = new Bitmap(nWidth, nHeight, bTemp.PixelFormat);

                        double nXFactor = (double)bTemp.Width/(double)nWidth;
                        double nYFactor = (double)bTemp.Height/(double)nHeight;

                        if (bBilinear)
                        {
                                double fraction_x, fraction_y, one_minus_x, one_minus_y;
                                int ceil_x, ceil_y, floor_x, floor_y;
                                Color c1 = new Color();
                                Color c2 = new Color();
                                Color c3 = new Color();
                                Color c4 = new Color();
```

```
                        byte red, green, blue;

                        byte b1, b2;

                        for (int x = 0; x < b.Width; ++x)
                                for (int y = 0; y < b.Height; ++y)
                                {
                                        // Setup

                                        floor_x = (int)Math.Floor(x * nXFactor);
                                        floor_y = (int)Math.Floor(y * nYFactor);
                                        ceil_x = floor_x + 1;
                                        if (ceil_x >= bTemp.Width) ceil_x = floor_x;
                                        ceil_y = floor_y + 1;
                                        if (ceil_y >= bTemp.Height) ceil_y = floor_y;
                                        fraction_x = x * nXFactor - floor_x;
                                        fraction_y = y * nYFactor - floor_y;
                                        one_minus_x = 1.0 - fraction_x;
                                        one_minus_y = 1.0 - fraction_y;

                                        c1 = bTemp.GetPixel(floor_x, floor_y);
                                        c2 = bTemp.GetPixel(ceil_x, floor_y);
                                        c3 = bTemp.GetPixel(floor_x, ceil_y);
                                        c4 = bTemp.GetPixel(ceil_x, ceil_y);

                                        // Blue
                                        b1 = (byte)(one_minus_x * c1.B + fraction_x *
c2.B);

                                        b2 = (byte)(one_minus_x * c3.B + fraction_x *
c4.B);

                                        blue = (byte)(one_minus_y * (double)(b1) +
fraction_y * (double)(b2));

                                        // Green
                                        b1 = (byte)(one_minus_x * c1.G + fraction_x *
c2.G);

                                        b2 = (byte)(one_minus_x * c3.G + fraction_x *
c4.G);

                                        green = (byte)(one_minus_y * (double)(b1) +
fraction_y * (double)(b2));

                                        // Red
                                        b1 = (byte)(one_minus_x * c1.R + fraction_x *
c2.R);

                                        b2 = (byte)(one_minus_x * c3.R + fraction_x *
c4.R);

                                        red = (byte)(one_minus_y * (double)(b1) +
fraction_y * (double)(b2));

                                        b.SetPixel(x,y,
System.Drawing.Color.FromArgb(255, red, green, blue));
                                }
                        }
                        else
                        {
                                for (int x = 0; x < b.Width; ++x)
                                        for (int y = 0; y < b.Height; ++y)
                                                b.SetPixel(x, y,
bTemp.GetPixel((int)(Math.Floor(x * nXFactor)),
                                                        (int)(Math.Floor(y * nYFactor))));
                        }

                        return b;
                }
```

```
        }
}
```

web.config

```
<?xml version="1.0"?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
        <appSettings>
    <!--ImageVirtualDir se utiliza en ImageTools.cs-->
                <add key="ImageVirtualDir" value="/ImagenesCapturadas"/>
        </appSettings>
        <connectionStrings>
    <!--En caso de que la conexión a la base de datos no funcione, descomentar el siguiente
código y-->
    <!--en la parte server=, poner el nombre del sistema en que se instala la aplicación-->
    <!--<remove name="LocalSqlServer" />
    <add name="LocalSqlServer" connectionString="server=HAL9000;database=aspnetdb;integrated
security=sspi;" />-->
    <add name="LogsConnectionString" connectionString="Data
Source=.\SQLEXPRESS;AttachDbFilename=C:\Inetpub\wwwroot\SegLab\App_Data\Logs.mdf;Integrated
Security=True;Connect Timeout=30;User Instance=True" providerName="System.Data.SqlClient"/>
        </connectionStrings>
        <system.web>
                <authentication mode="Forms"/>
                <roleManager enabled="true"/>
                <compilation debug="false"/>
                <authorization>
                        <deny users="?"/>
                </authorization>
                <sessionState timeout="10"> <!--Expira sesión después de 10 mins-->
                </sessionState>
        </system.web>
</configuration>
```

Código fuente de la aplicación Detector de Movimiento.

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;
using System.Globalization;
using ImgProcessing;
using System.Drawing.Drawing2D;
using System.Drawing.Imaging;

namespace SensorMovimiento
{
    public partial class Form1 : Form
    {
        private Int32 Porcentaje;
        private Int32 PorcentajeActual = 86; //Diferencia 86% entre pixeles de las
        private bool Primera = true;         //dos imágenes
        private string hora = "";
        private CAMSERVERLib.Camera cam = new CAMSERVERLib.CameraClass();
        private int contador;
        public short nQuality;
        public Form1()
        {
            InitializeComponent();


            button1.Enabled = true;
            timer1.Enabled = true;
            contador = 0;
            label1.Text = contador.ToString() + "  imágenes capturadas";
            nQuality = 45;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.button1.Enabled = false;
            this.timer1.Enabled = false;
            Primera = true;

        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            if (!Primera)
            {


                byte[] imagen1 = (byte[])cam.GrabFrame(short.MaxValue);

                byte[] imagen2 = (byte[])cam.GrabFrame(short.MaxValue);

                MemoryStream ms = new MemoryStream(imagen1);
                MemoryStream ms2 = new MemoryStream(imagen2);
                Bitmap bmp = new Bitmap(ms);
                Bitmap bmp2 = new Bitmap(ms2);
                Bitmap diferente = new Bitmap(bmp.Width, bmp.Height);
                //call ImgProc
```

```
                ImageProcessing imageProcessing = new ImageProcessing(bmp, bmp2, diferente);
                imageProcessing.CompareUnsafeFaster(out Porcentaje);
                Porcentaje = (Int32)Porcentaje * 100 / (diferente.Width * diferente.Height);


                if ((Porcentaje >= PorcentajeActual))
                {

                    hora = System.DateTime.Now.Millisecond.ToString() + ".jpg";

                    Graphics g = Graphics.FromImage(bmp2);

                    string strDate = DateTime.Now.ToLongDateString() + " - " +
                                                        DateTime.Now.ToLongTimeString();

                    StringFormat drawFormat = new StringFormat();
                    drawFormat.Alignment = StringAlignment.Center;

                    g.DrawString(strDate,
                                    new Font(FontFamily.GenericSansSerif, 12),
                                    new SolidBrush(Color.Black),
                                    new RectangleF(1, 1, 320, 240),
                                    drawFormat
                                );

                    g.DrawString(strDate,
                                    new Font(FontFamily.GenericSansSerif, 12),
                                    new SolidBrush(Color.White),
                                    new RectangleF(0, 0, 320, 240),
                                    drawFormat
                                );

                    //Obtener Códecs
                    ImageCodecInfo[] icf = ImageCodecInfo.GetImageEncoders();

                    EncoderParameters encps = new EncoderParameters(1);
                    EncoderParameter encp = new
EncoderParameter(System.Drawing.Imaging.Encoder.Quality,
                                                                    (long)nQuality);

                    encps.Param[0] = encp;

                      //Guarda imagen en carpeta ImagenesCapturadas
                    bmp2.Save(@"C:\Inetpub\wwwroot\ImagenesCapturadas\imagen" + hora,
icf[1], encps);
                    ++contador;
                    label1.Text = contador.ToString() + "  imágenes capturadas";

                }
                else
                {


                }

                pictureBox2.Image=bmp;
                pictureBox3.Image=bmp2;

                Application.DoEvents();
            }
            else
            {
                byte[] imagen = (byte[])cam.GrabFrame(short.MaxValue);
                MemoryStream ms = new MemoryStream(imagen);
                Bitmap bmp = new Bitmap(ms);
                pictureBox2.Image=bmp;
                Primera = false;

            }
        }
```

```
        private void pictureBox1_Click(object sender, EventArgs e)
        {

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void pictureBox1_Click_1(object sender, EventArgs e)
        {

        }

        private void pictureBox2_Click(object sender, EventArgs e)
        {

        }

        private void pictureBox3_Click(object sender, EventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void pictureBox4_Click(object sender, EventArgs e)
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

ImageProcessing.cs

```
using System;
using System.Drawing;
using System.Drawing.Imaging;


namespace ImgProcessing
{
        /// <summary>
        /// Summary description for ImageProcessing.
        /// </summary>
        public unsafe sealed class ImageProcessing
        {
                Bitmap flag,flag2,flag3;
                int width,width2,width3;
                BitmapData bitmapData = null,bitmapData2= null,bitmapData3= null;
                Byte* pBase = null,pBase2=null,pBase3=null;

                #region ImageProcessing Constructor
                public ImageProcessing(Bitmap picOld,Bitmap picNew,Bitmap target)
                {
                        this.flag=picOld;
```

```
            this.flag2=picNew;
            this.flag3=target;
        }
        public ImageProcessing(Bitmap source,Bitmap target)
        {
            this.flag=source;
            this.flag2=target;
        }
        #endregion

        #region internal methods
        #region PixelSize
        public Point PixelSize
        {
            get
            {
                GraphicsUnit unit = GraphicsUnit.Pixel;
                RectangleF bounds = flag.GetBounds(ref unit);

                return new Point((int) bounds.Width, (int) bounds.Height);
            }
        }

        public Point PixelSize2
        {
            get
            {
                GraphicsUnit unit = GraphicsUnit.Pixel;
                RectangleF bounds = flag2.GetBounds(ref unit);

                return new Point((int) bounds.Width, (int) bounds.Height);
            }
        }
        public Point PixelSize3
        {
            get
            {
                GraphicsUnit unit = GraphicsUnit.Pixel;
                RectangleF bounds = flag3.GetBounds(ref unit);

                return new Point((int) bounds.Width, (int) bounds.Height);
            }
        }
        #endregion

        #region PixelAt
        public Pixel* PixelAt(int x, int y)
        {
            return (Pixel*) (pBase + y * width + x * sizeof(Pixel));
        }
        public Pixel* PixelAt2(int x, int y)
        {
            return (Pixel*) (pBase2 + y * width2 + x * sizeof(Pixel));
        }
        public Pixel* PixelAt3(int x, int y)
        {
            return (Pixel*) (pBase3 + y * width3 + x * sizeof(Pixel));
        }

        #endregion

        #region LockBitMap
        public void LockBitmap()
        {
            GraphicsUnit unit = GraphicsUnit.Pixel;
            RectangleF boundsF = flag.GetBounds(ref unit);
            Rectangle bounds = new Rectangle((int) boundsF.X,
                (int) boundsF.Y,
                (int) boundsF.Width,
                (int) boundsF.Height);
```

```
                        width = (int) boundsF.Width * sizeof(Pixel);
                        if (width % 4 != 0)
                        {
                                width = 4 * (width / 4 + 1);
                        }

                        bitmapData =
                                flag.LockBits(bounds, ImageLockMode.ReadWrite,
PixelFormat.Format24bppRgb);

                        pBase = (Byte*) bitmapData.Scan0.ToPointer();

                }
                public void UnlockBitmap()
                {
                        flag.UnlockBits(bitmapData);
                        bitmapData = null;
                        pBase = null;

                }
                public void LockBitmap2()
                {
                        GraphicsUnit unit = GraphicsUnit.Pixel;
                        RectangleF boundsF = flag2.GetBounds(ref unit);
                        Rectangle bounds = new Rectangle((int) boundsF.X,
                                (int) boundsF.Y,
                                (int) boundsF.Width,
                                (int) boundsF.Height);


                        width2 = (int) boundsF.Width * sizeof(Pixel);
                        if (width2 % 4 != 0)
                        {
                                width2 = 4 * (width2 / 4 + 1);
                        }

                        bitmapData2 =
                                flag2.LockBits(bounds, ImageLockMode.ReadWrite,
PixelFormat.Format24bppRgb);

                        pBase2 = (Byte*) bitmapData2.Scan0.ToPointer();
                }

                public void UnlockBitmap2()
                {
                        flag2.UnlockBits(bitmapData2);
                        bitmapData2= null;
                        pBase2= null;
                }
                public void LockBitmap3()
                {
                        GraphicsUnit unit = GraphicsUnit.Pixel;
                        RectangleF boundsF = flag3.GetBounds(ref unit);
                        Rectangle bounds = new Rectangle((int) boundsF.X,
                                (int) boundsF.Y,
                                (int) boundsF.Width,
                                (int) boundsF.Height);


                        width3 = (int) boundsF.Width * sizeof(Pixel);
                        if (width3 % 4 != 0)
                        {
                                width3 = 4 * (width3 / 4 + 1);
                        }

                        bitmapData3 =
                                flag3.LockBits(bounds, ImageLockMode.ReadWrite,
PixelFormat.Format24bppRgb);
```

```
                pBase3 = (Byte*) bitmapData3.Scan0.ToPointer();
            }

            public void UnlockBitmap3()
            {
                flag3.UnlockBits(bitmapData3);
                bitmapData3= null;
                pBase3= null;
            }
            #endregion

            #region Save
            public void Save(string filename)
            {
                flag3.Save(filename, ImageFormat.Jpeg);
            }
            #endregion

            #region Dispose
            public void Dispose()
            {
                flag=null;
                flag2=null;
                flag3=null;
            }
            #endregion

            public Bitmap bitmap
            {
                get
                {
                    return(this.flag3);
                }
            }


            #endregion

            #region Magic code for CompareUnsafeFaster
            public void CompareUnsafeFaster(out Int32 percent)
            {
                Point size = PixelSize;
        percent=0;
                LockBitmap();
                LockBitmap2();
                LockBitmap3();

                for (int y = 0; y < size.Y; y++)
                {
                    Pixel* pPixel = PixelAt(0, y);
                    Pixel* pPixel2 = PixelAt2(0, y);
                    Pixel* pPixel3 = PixelAt3(0, y);
                    for (int x = 0; x < size.X; x++)
                    {
                        if(!(pPixel->green==pPixel2->green))//((pPixel-
>red==pPixel2->red)))//&&(pPixel->green==pPixel2->green)&&(pPixel->blue==pPixel2->blue)))
                        {
                            pPixel3->red = pPixel2->red;
                            pPixel3->green = pPixel2->green;
                            pPixel3->blue = pPixel2->blue;
                            percent++;
                        }
                        pPixel++;
                        pPixel2++;
                        pPixel3++;
                    }
                }
                UnlockBitmap3();
                UnlockBitmap2();
```

```
                UnlockBitmap();
        }

                #endregion

                #region Magic code for ComplementUnsafeFaster
                public void ComplementUnsafeFaster()
                {
                        Point size = PixelSize;

                        LockBitmap();
                        LockBitmap2();


                        for (int y = 0; y < size.Y; y++)
                        {
                                Pixel* pPixel = PixelAt(0, y);
                                Pixel* pPixel2 = PixelAt2(0, y);

                                for (int x = 0; x < size.X; x++)
                                {

                                        pPixel2->red =(byte)(255-(int)pPixel->red);
                                        pPixel2->green = (byte)(255-(int)pPixel->green);
                                        pPixel2->blue = (byte)(255-(int)pPixel->blue);

                                        pPixel++;
                                        pPixel2++;

                                }
                        }
                        UnlockBitmap2();
                        UnlockBitmap();
                        flag3=new Bitmap(flag2);

                }

                #endregion

                #region Magic code for ColorBallUnsafeFaster
                public void ColorBallUnsafeFaster(double red,double green,double blue)
                {
                        Point size = PixelSize;
                        double fr, fg, fb;
                        if(red<0)
                        {
                                fr=red/100+1;
                        }
                        else
                        {
                                fr=red/100;
                        }
                        if(green<0)
                        {
                                fg=green/100+1;
                        }
                        else
                        {
                                fg=green/100;
                        }
                        if(blue<0)
                        {
                                fb=blue/100+1;
                        }
                        else
                        {
                                fb=blue/100;
                        }
                        LockBitmap();
                        LockBitmap2();
```

```
                        for (int y = 0; y < size.Y; y++)
                        {
                                Pixel* pPixel = PixelAt(0, y);
                                Pixel* pPixel2 = PixelAt2(0, y);

                                for (int x = 0; x < size.X; x++)
                                {

                                        if(  red < 0 )
                                        {
                                                pPixel2->red =(byte)((int)pPixel->red * fr);
                                        }
                                        else
                                        {
                                                pPixel2->red = (byte)((int)pPixel->red + (255 -
(int)pPixel->red) * fr);
                                        }
                                        if(  green < 0 )
                                        {
                                                pPixel2->green = (byte)((int)pPixel->green *
fg);
                                        }
                                        else
                                        {
                                                pPixel2->green = (byte)((int)pPixel->green +
(255 - (int)pPixel->green) * fg);
                                        }
                                        if(  blue < 0 )
                                        {
                                                pPixel2->blue =(byte)((int) pPixel->blue * fb);
                                        }
                                        else
                                        {
                                                pPixel2->blue= (byte)((int)pPixel->blue + (255
- (int)pPixel->blue) * fb);
                                        }

                                        pPixel++;
                                        pPixel2++;

                                }
                        }
                        UnlockBitmap2();
                        UnlockBitmap();
                        flag3=new Bitmap(flag2);

                }

                #endregion

                #region Magic code for Brightness
                public void Brightness(double brightness)
                {
                        Point size = PixelSize;
                        double f;
                        if(brightness<0)
                        {
                                f=brightness/100+1;
                        }
                        else
                        {
                                f=brightness/100;
                        }

                        LockBitmap();
                        LockBitmap2();
```

```
                        for (int y = 0; y < size.Y; y++)
                        {
                                Pixel* pPixel = PixelAt(0, y);
                                Pixel* pPixel2 = PixelAt2(0, y);

                                for (int x = 0; x < size.X; x++)
                                {
                                        if(  brightness < 0 )
                                        {
                                                pPixel2->red =(byte)( (int)pPixel->red * f);
                                                pPixel2->green =(byte)( (int)pPixel->green *
f);
                                                pPixel2->blue =(byte)( (int)pPixel->blue * f);
                                        }
                                        else
                                        {
                                                pPixel2->green = (byte)((int)pPixel->red + (255
- (int)pPixel->red) * f);
                                                pPixel2->green = (byte)((int)pPixel->green +
(255 - (int)pPixel->green) * f);
                                                pPixel2->blue= (byte)((int)pPixel->blue + (255
- (int)pPixel->blue) * f);
                                        }

                                        pPixel++;
                                        pPixel2++;

                                }
                        }
                        UnlockBitmap2();
                        UnlockBitmap();
                        flag3=new Bitmap(flag2);

                }

                #endregion

                #region MakeGreyUnsafeFaster
                public void MakeGreyUnsafeFaster()
                {
                        Point size = PixelSize;

                        LockBitmap();
                        LockBitmap2();

                        for (int y = 0; y < size.Y; y++)
                        {
                                Pixel* pPixel = PixelAt(0, y);
                                Pixel* pPixel2 = PixelAt2(0, y);
                                for (int x = 0; x < size.X; x++)
                                {
                                        byte value = (byte) ((pPixel->red + pPixel->green +
pPixel->blue) / 3);
                                        pPixel2->red =  value;
                                        pPixel2->green = value;
                                        pPixel2->blue = value;
                                        pPixel++;
                                        pPixel2++;
                                }
                        }
                        UnlockBitmap2();
                        UnlockBitmap();
                        flag3=new Bitmap(flag2);
                }
                #endregion

        }

        #region Pixel struct
        public struct Pixel
```

```
        {
                public byte blue;
                public byte green;
                public byte red;
        }
        #endregion
}
```

GLOSARIO

Beacon: Onda portadora de baja potencia generada por un transmisor auxiliar. Puede no estar modulada para pruebas de propagación, o modulada para telemetría.

Bug: Error, fallo, falta, o imperfección de un programa de computadora que no le permite trabajar correctamente o produce un resultado erróneo. Los bugs son ocasionados por errores cometidos dentro del código fuente o durante el diseño.

Firmware: Instrucciones permanentes y datos programados directamente en los circuitos de una memoria ROM con el propósito de controlar la operación de una computadora o unidad de disco.

Hash: Algoritmo que traduce un conjunto de bits a otro de manera que el algoritmo produce siempre el mismo resultado, o hash, para el mismo mensaje, y es en teoría imposible reconstruir el mensaje a partir del hash resultante. Dos mensajes diferentes no pueden producir el mismo hash.

Internet Information Server (IIS): Servidor Web de Microsoft que corre bajo plataformas Windows NT. ISS viene integrado dentro de Windows NT 4.0; debido a que IIS está estrechamente integrado dentro del sistema operativo, es relativamente fácil de administrar.

Information Technology (IT): Rama de la ingeniería referente a la utilización de computadoras y telecomunicaciones para recuperar, almacenar y transmitir información.

Java Virtual Machine (JVM): Software responsable de ejecutar programas en Java. Una nueva JVM es iniciada cada vez que se abre algún programa en Java. Se le llama máquina virtual debido a que es software que emula a una computadora física. Los programas en Java son creados para correr en esta máquina virtual, permitiéndoles correr en cualquier máquina física que tenga instalado JVM.

Open Source: En general, se refiere a cualquier programa cuyo código fuente está disponible para su utilización y modificación por los usuarios o desarrolladores si lo desean. Este software es usualmente desarrollado como colaboración pública y está disponible gratuitamente.

Packet: Unidad fundamental para transmisión de información en todas las redes de cómputo modernas. Un packet consiste de los datos que se quiere transmitir, más cierta información de control, incluyendo la dirección de destino.

Píxel: Abreviación de Picture Element. Unidad Básica de la cuál una imagen de computadora o video están formados. Esencialmente es un punto con un color determinado y un valor de brillo. Entre más píxeles posea la imagen, mayor es su resolución.

Remote Procedure Call (RPC): Mecanismo de comunicación que permite a un proceso Unix comunicarse con otro proceso Unix. Estos procesos de comunicación pueden ocurrir en diferentes computadoras dentro de una red.

Roaming: Término general en telecomunicaciones inalámbricas que se refiere al entendimiento del servicio de conectividad dentro de una red diferente a la red donde la estación está registrada.

Runtime: Cuando un programa se está ejecutando, se dice que se encuentra en tiempo de ejecución o *"runtime"*. El término es comúnmente utilizado por desarrolladores de software para especificar cuando ocurren errores en un programa. Un error de tiempo de ejecución o *"runtime error"* es un error que surge cuando el programa se está ejecutando. Por ejemplo, si un programa da como resultado 2+2=5000, eso sería un error de tiempo de ejecución. De igual manera, cuando un programa empieza a consumir cantidades excesivas de memoria del sistema, también es un *"runtime error"*.

Sandbox: Mecanismo de seguridad para ejecutar programas de manera segura. Es frecuentemente utilizado para ejecutar código que aún no se han probado, o programas creados por terceros o de usuarios no confiables.

Server Side: Referente a alguna aplicación o componente de la aplicación que se ejecute en el servidor en lugar del cliente.

Servicios Web: Un servicio Web es cualquier pieza de software que pueda ser accedida a través de Internet y que utilice un sistema estándar de mensajes XML. Las partes interesadas en el servicio deben tener algún mecanismo simple para localizar el servicio y localizar su interfaz pública. El más prominente directorio de Servicios Web está actualmente disponible vía UDDI, o Universal Description, Discovery and Integration.

SOAP: Estándar para el intercambio de mensajes basados en XML a través de una red de cómputo, usualmente utilizando HTTP.

Stylus: Dispositivo para seleccionar con forma de pluma para asistentes personales digitales pen-based.

UDDI: Protocolo basado en XML que provee un directorio distribuido que permite a negocios enlistarse en Internet y descubrir otros servicios. Similar a un número de teléfono, los negocios pueden enlistarse por nombre, producto, locación, o por los servicios Web que ofrecen.

Widget: Objeto utilizado para contener datos y presentar una interfaz al usuario. Un widget es una combinación de estados y procedimientos. Cada widget es miembro de una clase, la cual contiene los procedimientos y estructuras de datos comunes para todos los widgets de esa clase. Una instancia del widget contiene los procedimientos y estructuras de datos particulares de ese widget. Cada clase de widgets típicamente provee el comportamiento general asociado con un modo particular de interacción con el usuario.

Wrapper: Objeto creado para proveer una interfaz estilo objeto a algún otro tipo de datos. Los objetos Number y Boolean son ejemplos de objetos wrapper.

Extensible Markup Language (XML): XML es una forma flexible de crear formatos de información comunes e intercambiar tanto al formato, como la información a través del World Wide Web, Intranets, etc. XML es una recomendación formal del World Wide Web Consortium (W3C). XML es algo similar al Hypertext Markup Language (HTML).

BIBLIOGRAFÍA

[1] MSDN Library for Visual Studio 2005
[2] www.microsoft.com
[3] www.msdn.com
[4] http://www.logix4u.net
[5] Microsoft Encarta 2005
[6] msdn.microsoft.com/vstudio
[7] asp.net
[8] msdn.microsoft.com/vcsharp
[9] en.wikipedia.org/wiki/C_sharp
[10] en.wikipedia.org/wiki/ASP.NET
[11] en.wikipedia.org/wiki/Microsoft_Visual_Studio
[12] www.howstuffworks.com/parallel-port.htm
[13] www.codeproject.com
[14] computer.howstuffworks.com/wireless-network.htm
[15] en.wikipedia.org/wiki/WiFi
[16] en.wikipedia.org/wiki/Personal_Digital_Assistant
[17] www.howstuffworks.com/pda.htm
[18] microsoft.fi-b.unam.mx/lmsr/infraestructura.htm
[19] www.epanorama.net/ parallel_output.html
[20] www.smarthome.com/5192b.html