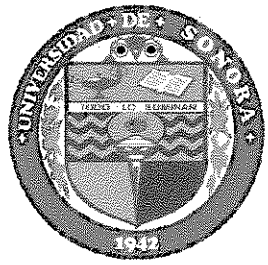
UNIVERSIDAD DE SONORA



"El saber de mis hijos
hará mi grandeza"

Departamento de Ingeniería Industrial

"SISTEMA DE INVENTARIOS **CAFFENIO**"

Café del Pacífico S.A de C.V

Iván Bernardo Noriega Cárdenas #204203637

Hermosillo, Son., a Miércoles 23 de Mayo del 2012

INDICE

1. INTRODUCCIÓN

Vivimos hoy en día en una sociedad en la que cada vez la tecnología se adentra más en nuestra forma de vivir, en la forma de cómo hacemos las cosas, de cómo nos comunicamos con las demás personas, por consiguiente esto ha repercutido en un cambio muy notorio en como interactuamos con las demás personas así como se ve reflejado en la mayoría de las diferentes actividades que realizamos en cualquier ambiente de nuestra vida, ya sea en un ambiente estudiantil, en el ambiente laboral y en cualquier otro situación que conlleve la sociabilización de las personas.

Las computadoras de escritorio, las portátiles, tabletspcs, blackberrys, etc. Cada vez estos sistemas de información están mas al alcance de nuestras manos sin importar edad, sexo, situación, etc., de echo ya hemos llegado a tal grado que sin estos dispositivos electrónicos no podríamos realizar muchas de nuestras actividades que llevamos a cabo.

Todos estos equipos "sistemas de información" como ya se mencionó en el párrafo anterior para siquiera puedan funcionar es de vital importancia el uso de software para su funcionamiento. Es por eso que en este escrito se pretende presentar la realización de un sistema de información que se realizó en la empresa Café del Pacífico en el cual dicho sistema guardara toda la información de las computadoras que hay en toda la empresa, en pocas palabras "un sistema de inventarios". Se explicará de forma detallada cómo funciona, si se logro lo que se pretendía y la experiencia que se adquirió durante la realización del proyecto.

1.1 DESCRIPCIÓN DEL ÁREA DE LA INSTITUCIÓN

El proyecto se realizó en el área de Tecnologías de Información de la empresa " Café del Pacífico S.A de C.V con el fin de facilitar las actividades de captura de información de los equipos de cómputo en el área de Soporte Técnico.

El departamento de tecnologías de la información de la compañía "Café del Pacífico" cuenta con dos tipos de áreas de sistemas, uno que es de soporte a los drives "Caffenios" y el otro que es a las sucursales de toda la república y a las oficinas administrativas, en el segundo es en donde nos enfocaremos.

Este departamento es el encargado de ofrecer los servicios de soporte de sistemas en oficinas administrativas, así como lo mencionamos anteriormente, en las 12 sucursales con la que cuenta el café. En dicho departamento cuenta con cinco personas en las cuales son las siguientes:

- Gerente de Sistemas.
- Programador en Genexux.
- Programador en C# y PDAS.
- Programador en C#.
- Soporte Técnico.

El gerente de sistemas como cualquier otro gerente es el responsable de todo el área de TI y encargado de que se realicen todos los proyectos del área. Él es el responsable de las actividades de planeación, coordinación, mantenimiento y entrega de resultados de proyectos de informática. Dentro de los cuales se encuentran: Programas, telefonía IP, cableado estructurado, servidores, correo, internet, etc.

El programador en Genexus se encarga de dar soporte principalmente a los programas que desarrolla, estos accesos a su vez son agregados en la pantalla prin-

cipal del sistema ERP " Microsoft Great Plains", aparte de desarrollar software también le da soporte al sistema por si algunos usuarios tienen problemas al contabilizar, al perder facturas, etc. Se les da soporte.

El programador en C# y PDAS es el encargado de asignar las PDAS a los usuarios como en este caso que son los asesores de Oxxo, y de darles soporte a dichos usuarios. También se encarga de desarrollar software tanto de las PDAS como a usuarios en las oficinas administrativas.

El programador en C# así como al igual que el otro programador se encarga de desarrollar software para las oficinas administrativas y también de los drives.

El encargado de Soporte Técnico le da soporte a todas las computadoras de las oficinas administrativas y a sucursales. Se encarga de que la red esté en perfecto funcionamiento, del correo de los Caffenios y de todo el soporte a equipos ya sea impresoras, computadoras de escritorio, laptops, blackberrys, teléfonos, etc.

1.2 JUSTIFICACIÓN

Este proyecto se hace con el propósito de tener un mayor control en los gastos de los equipos y sus respectivos controles de mantenimiento. También serviría para mantener la información más organizada y de una manera más segura ya que antes no se contaba la información con seguridad.

Este sistema de inventarios serviría mucho para el control de los gastos de cada usuario ya que muchos de los usuarios que solicitan accesorios computacionales ya sea por daño o porque así lo requieran. También ayudaría mucho para las fechas en que se compra el equipo y sea para tener en cuenta cuando se le va cambiar equipo o igual para las garantías algo muy importante dentro del soporte y las fechas de los mantenimientos preventivos.

Es necesario contar con un sistema de inventarios debido al crecimiento que está teniendo la empresa ya que cada vez está recabando mas información de los equipos y por consiguiente a la hora de capturara esta información se esta haciendo más difícil de manejar con un simple procesador de textos, esto ayudaría mucho a la hora de consultar la información.

1.3 OBJETIVOS DEL PROYECTO

Objetivo General:

Eficientar la manera de recabar la información de los equipos de cómputo en la empresa.

Objetivos Específicos:

- Tener un mayor control de los gastos de los equipos.
- Consulta de información de los equipos para realizar las siguientes actividades: Mantenimiento de los equipos, cambios de equipos (cuando le toca cambiar equipo), garantías de los equipos y/o accesorios (por si viene un defecto en el equipo), consulta en general.

1.4 PROBLEMAS PLANTEADOS

- Se detectó una deficiencia en el manejo de información en el inventario de las computadoras de Caffenio.- Se refiere a que la información esta guardada de una manera muy anticuada como en este caso una hoja de Excel. Lo que se requiere aquí es un sistema práctico en donde se pueda realizar búsquedas rápidas principalmente, avisos que el sistema indique como por ejemplo que te indique las fechas de mantenimiento de equipo, etc.
- Inconsistencia en los datos.- Esto se puede prestar a que los datos que se van capturando se capturen de una forma incorrecta y que se pueda prestar a malentendidos, confusiones, etc.
- Poca organización en el manejo de la información de los equipos de cómputo.- Muchos de los gastos realizados en accesorios computacionales no se sabe para quién fue, igual es ineficiente manejar una base de datos de los equipos en una simple hoja de Excel.

1.5 ALCANCES Y LIMITACIONES EN LA SOLUCIÓN DE LOS PROBLEMAS

A continuación se describen los alcances y limitaciones que se encontraron a lo largo del proceso de solución de problemas dentro de la empresa.

Alcances:

- Se realizó la instalación del software en la computadora en donde se captura el inventario de las computadoras. La base de datos se creó y se instaló en el servidor principal de la empresa para que ahí se guarde toda la información así como las demás bases de datos y sirve para que también a la hora de hacer el respaldo semanal de todas las bases de datos se guarde y tengamos un resguardo del mismo.
- La información está más organizada y estructurada para la consulta de esta.
- Se eficientó más el proceso de búsqueda de información de las computadoras de la empresa. A la hora de hacer las consultas se hace de una manera mas rápida y concisa.

Limitaciones:

- La única limitación que vemos aquí es que se pueda perder la base de datos del servidor ya sea que el disco duro del servidor falle o que por un fallo de luz se dañe el disco duro. En caso de que la información del servidor se perdiera se llegaría a perder el trabajo realizado entre 1 a 5 días dependiendo, pero igual no es lo mínimo que se perdería. Esto es teniendo en cuenta que se realizó el respaldo semanal.

Universidad de Sonora
Departamento de Ingeniería Industrial
Coordinación de Prácticas Profesionales

FPP-3
Formato de Liberación de la Práctica Profesional

Hermosillo, Sonora, 29 (DÍA) 05 (MES) 2012 (AÑO)

Con mi carácter de Tutor de Prácticas Profesionales, hago constar que:
I.- El alumno(a) Iván Bernardo Noriega Cárdenas, de la carrera de Ingeniería en Sistemas de Información con número de expediente: 204203637, ha cumplido con la entrega oportuna de:

a.- Los reportes de avances periódicos de su práctica profesional.
b.- El informe técnico del proyecto realizado.

II.- He corroborado que los contenidos y tiempos de los reportes de avances están acordes con lo planeado en los anexos del formato de inscripción FPP-1, y que los contenidos y forma del informe técnico satisfacen los requerimientos especificados en la normatividad actual.

III.- El número de horas acumuladas de práctica profesional, de acuerdo a los reportes de avances, es de 340

Por lo anteriormente expuesto, no tengo inconveniente alguno en dar por liberado(a), al (la) alumno(a), anteriormente referido(a), del cumplimiento de la práctica profesional:

[X] TOTALMENTE, y evaluarlo(a) con 20 créditos cumplidos.

IV.- Debido a que el alumno no pudo terminar su práctica profesional en la empresa asignada, con base en sus reportes, y dado que ha acumulado [ ] horas de práctica, no tengo inconveniente alguno en dar por liberado(a), al (la) alumno(a), anteriormente referido(a), del cumplimiento de la práctica profesional:

[ ] PARCIALMENTE, y evaluarlo(a) con los siguientes créditos:

Con número [ ] Con letra [ ]

MOTIVOS POR NO HABER TERMINADO CON LA PRÁCTICA PROFESIONAL

ATENTAMENTE:

Dr. Alonso Pérez Soltero
Tutor de Prácticas Profesionales

Vo. Bo.

Dr. Mario Barceló Valenzuela
Coordinador de Prácticas Profesionales

Original: Coordinación de Prácticas Profesionales
C.c.p.: Alumno
C.c.p.: Tutor de Prácticas Profesionales

2 FUNDAMENTO TEÓRICO

C# (pronunciado *si sharp* en inglés) es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

El nombre C Sharp fue inspirado por la notación musical, donde '#' (sostenido, en inglés *sharp*) indica que la nota (*C* es la nota do en inglés) es un semitono más alta, sugiriendo que C# es superior a C/C++. Además, el signo '#' viene de cuatro '+' pegados.[1]

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Ya existe un compilador implementado que provee el marco Mono - DotGNU, el cual genera programas para distintas plataformas como Windows, Unix y GNU/Linux.

Historia

Durante el desarrollo de la plataforma .NET, las bibliotecas de clases fueron escritas originalmente usando un sistema de código gestionado llamado Simple Managed C (SMC). En enero de 1999, Anders Hejlsberg formó un equipo con la misión de desarrollar un nuevo lenguaje de programación llamado Cool (C orientado a objetos). Este nombre tuvo que ser cambiado debido a problemas de marca, pa-

sando a llamarse C#. La biblioteca de clases de la plataforma .NET fue migrada entonces al nuevo lenguaje.

Hejlsberg lideró el proyecto de desarrollo de C#. Anteriormente, ya participó en el desarrollo de otros lenguajes como Turbo Pascal, J++ y Embarcadero Delphi.

Tipos de datos

C# contiene dos categorías generales de tipos de datos integrados: **tipos de valor** y **tipos de referencia**. El término **tipo de valor** indica que esos tipos contienen directamente sus valores.

Tipos para definir números enteros:

Los tipos de coma flotante pueden representar números con componentes fraccionales. Existen dos clases de tipos de coma flotante: **float** y **double**. El tipo **double** es el más utilizado porque muchas funciones matemáticas de la biblioteca de clases de C# usan valores **double**. Quizá, el tipo de coma flotante más interesante de C# es **decimal**, dirigido al uso de cálculos monetarios. La aritmética de coma flotante normal está sujeta a una variedad de errores de redondeo cuando se aplica a valores decimales. El tipo **decimal** elimina estos errores y puede representar hasta 28 lugares decimales.

Los caracteres en C# no tienen un tamaño de 8 bits como en otros muchos lenguajes de programación, sino que usa un tamaño de 16 bits llamado Unicode al cual se le llama char. No existen conversiones automáticas de tipo entero a char.

Literales

En ocasiones, resulta más sencillo usar un sistema numérico en base 16 en lugar de 10, para tal caso C# permite especificar números enteros en formato hexade-

cimal, y se define anteponiendo 0x, por ejemplo: 0xFF, que equivale a 255 en decimal.

C# tiene caracteres denominados *secuencias de escape* para facilitar la escritura con el teclado de símbolos que carecen de representación visual.

C#, al igual que C++, define un tipo de cadena de caracteres. Dentro de la cadena de caracteres se pueden usar secuencias de escape. Una cadena de caracteres puede iniciarse con el símbolo @ seguido por una cadena entre comillas ("), en tal caso, las secuencias de escape no tienen efecto, y además la cadena puede ocupar dos o más líneas.

Variables

Las variables son identificadores asociados a valores. Se declaran indicando el tipo de dato que almacenará y su identificador.

Un identificador puede:

- empezar por "_".
- contener caracteres Unicode en mayúsculas y minúsculas (sensible a mayúsculas y minúsculas).

Un identificador no puede:

- empezar por un número.
- empezar por un símbolo, ni aunque sea una palabra clave.
- contener más de 511 caracteres.

Constantes

Las constantes son valores inmutables, y por tanto no se pueden cambiar. Cuando se declara una constante con la palabra clave const, también se debe asignar el valor. Tras esto, la constante queda bloqueada y no se puede cambiar. Son implícitamente estáticas (static).

Instrucciones de control

- Las instrucciones if-else, for, while, do-while, switch, return, break, continue son, básicamente, iguales que en C, C++ y Java.
- La instrucción foreach, al igual que en Java, realiza un ciclo a través de los elementos de una matriz o colección. En este ciclo se recorre la colección y la variable recibe un elemento de dicha colección en cada iteración.
- La instrucción goto se sigue utilizando en C# a pesar de la polémica sobre su uso.

Métodos

- Todo método debe ser parte de una clase, no existen métodos globales (funciones).
- Por defecto, los parámetros se pasan por valor. (Nótese que las listas y otras colecciones son variables *por referencia* (referencias al espacio reservado para esa lista en la pila) y que se pasa por valor al método la referencia , pero el espacio reservado para la lista es común, por lo que si elimina un elemento lo hace también de la original)
- El modificador ref fuerza a pasar los parámetros por referencia en vez de pasarlos por valor.
- El modificador out es similar al modificador ref, con la diferencia de que el método debe asignar un valor al parámetro antes de que el método finalice.

- Cuando ref y out modifican un parámetro de referencia, la propia referencia se pasa por referencia.
- El modificador params sirve para definir un número variable de argumentos los cuales se implementan como una matriz.
- Un método debe tener como máximo un único parámetro params y éste debe ser el último.
- Un método puede devolver cualquier tipo de dato, incluyendo tipos de clase.
- Ya que en C# las matrices se implementan como objetos, un método también puede devolver una matriz (algo que se diferencia de C++ en que las matrices no son válidas como tipos de valores devueltos).
- C# implementa *sobrecarga de métodos*, dos o más métodos pueden tener el mismo nombre siempre y cuando se diferencien por sus parámetros.
- El método Main es un método especial al cual se refiere el punto de partida del programa.

Clases y objetos

- Una variable de objeto de cierta clase no almacena los valores del objeto sino su referencia (al igual que Java).
- El operador de asignación no copia los valores de un objeto, sino la referencia al mismo (al igual que Java).
- Un constructor tiene el mismo nombre que su clase y es sintácticamente similar a un método.
- Un constructor no devuelve ningún valor (ni siquiera void).
- Al igual que los métodos, los constructores también pueden ser sobrecargados.
- Si no se especifica un constructor en una clase, se usa uno por defecto que consiste en asignar a todas las variables el valor 0, null o false según corresponda.

- Cuando un objeto no es referenciado por ninguna variable, el recolector de basura ejecuta el destructor de dicha clase y libera la memoria utilizada.
- El destructor de una clase no se llama cuando un objeto sale del ámbito.
- Todos los destructores se llamarán antes de que finalice un programa.
- La palabra clave this es una referencia al mismo objeto en el cual se usa.
- La palabra clave base es una referencia a la clase padre del objeto en la que se usa (por defecto, Object).
- La palabra clave static hace que un miembro pertenezca a una clase en vez de pertenecer a objetos de dicha clase. Se puede tener acceso a dicho miembro antes de que se cree cualquier objeto de su clase y sin referencias a un objeto.
- Un método static no tiene una referencia this.
- Un método static puede llamar sólo a otros métodos static.
- Un método static sólo debe tener acceso directamente a datos static.
- Un constructor static se usa para inicializar atributos que se aplican a una clase en lugar de aplicarse a una instancia.
- C# permite la sobrecarga de operadores (+, -, *, etc.) con la palabra clave operator.
- Al comparar objetos (==, !=, <, >, <=, >=) se comprueba si hacen referencia al mismo objeto.

Cadenas de caracteres

- El tipo de dato para las cadenas de caracteres es string.
- Realmente la palabra clave string es un alias de la clase System.String de la plataforma .NET.
- En C# las cadenas son objetos y no una matriz de caracteres; aun así, se puede obtener un carácter arbitrario de una cadena por medio de su índice (mas no modificarlo).

- Las cadenas son inmutables, una vez creadas no se pueden modificar, solo se pueden copiar total o parcialmente.
- El operador == determina si dos referencias hacen referencia al mismo objeto, pero al usar dicho operador con dos variables tipo string se prueba la igualdad del contenido de las cadenas y no su referencia. Sin embargo, con el resto de los operadores relacionales, como < y >=, sí se comparan las referencias.
- Se pueden concatenar (unir) dos cadenas mediante el operador +.
- Las cadenas se pueden usar en las instrucciones switch.

Compiladores

En la actualidad existen los siguientes compiladores para el lenguaje C#:

- Microsoft .NET Framework 2.0 (SDK) incluye un compilador de C#, pero no un IDE.
- Microsoft Visual Studio, IDE por excelencia de este lenguaje.
- SharpDevelop, IDE libre para C# bajo licencia GNU LGPL, con una interfaz muy similar a Microsoft Visual Studio.
- Mono, es una implementación con licencia GNU GPL de todo el entorno .NET desarrollado por Novell. Como parte de esta implementación se incluye un compilador de C#.
- Delphi 2006, de Borland Software Corporation.
- DotGNU Portable.NET, de la Free Software Foundation.
- Compilar, compilador en línea.

Metas del diseño del lenguaje

El estándar ECMA-334 lista las siguientes metas en el diseño para C#:

- Lenguaje de programación orientado a objetos simple, moderno y de propósito general.
- Inclusión de principios de ingeniería de software tales como revisión estricta de los tipos de datos, revisión de límites de vectores, detección de intentos de usar variables no inicializadas, y recolección de basura automática.
- Capacidad para desarrollar componentes de software que se puedan usar en ambientes distribuidos.
- Portabilidad del código fuente.
- Fácil migración del programador al nuevo lenguaje, especialmente para programadores familiarizados con C, C++ y Java.
- Soporte para internacionalización.
- Adecuación para escribir aplicaciones de cualquier tamaño: desde las más grandes y sofisticadas como sistemas operativos hasta las más pequeñas funciones.
- Aplicaciones económicas en cuanto a memoria y procesado.

3 PROCEDIMIENTOS EMPLEADOS Y ACTIVIDADES DESARROLLADAS

Requerimientos para la creación del sistema:

- Instalación del Equipo de Cómputo:

Para la instalación del sistema de inventarios no se requiere de una super computadora sino más bien de una computadora standard, en éste caso se escogió una computadora con el cual cuenta con las siguientes características:

- Pentium dual Core de 2.40ghz de procesador
- 2gb de memoria ram
- Una capacidad de almacenamiento muy bueno de 320 gb.

Por lo visto es una computadora muy potente para lo que requerimos así que no va ver problemas por insuficiencia de recursos. Ya que la computadora tenía mucho datos no correspondientes a dicho proyecto se optó por formatear el equipo, en este caso se le instalo el sistema operativo Windows 7 ya que es lo último que se está usando, esto con el fin de que el usuario este más al día y que se adapte a lo nuevo.

- Instalación del Software:

Para poder realizar este proyecto se requiero de dos herramientas básicas para la programación de sistemas de información:

- Microsoft Visual Studio 8 Version 9.0
- Microsoft SQL Server 2005

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de

programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQLy ANSI SQL. Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son *Oracle*, *PostgreSQL* o *MySQL*.

ACTIVIDAD 1: Planteamiento del Problema

Caffenio es una empresa que hoy en día esta en crecimiento y por lo tanto los equipos de computo y usuarios del mismo pues lógicamente también van a la alza. Anteriormente la persona responsable del inventario de las computadoras capturaba la información a través de un Excel, es la forma más común y fácil de cómo capturar la información de los equipos, pero aquí más bien lo que se necesita como toda empresa en crecimiento es de un sistema de información que sirva para manipular la información de una forma más accesible, rápido y cómodo.

ACTIVIDAD 2: Desarrollo del Sistema

Como lo comentamos anteriormente este es un sistema desarrollado en Visual Studio 8 con programación C# que es lo que se está usando hoy en día en las empresas, aunado a esto para el manejo de datos se uso como comúnmente se utiliza en la programación en C# es el programa SQL Server Management Studio Express.

ACTIVIDAD 3: Presentación del Sistema

A continuación se explicará de forma breve la interface del sistema con sus respectivas pantallas a forma de manual.

Pantalla PRINCIPAL

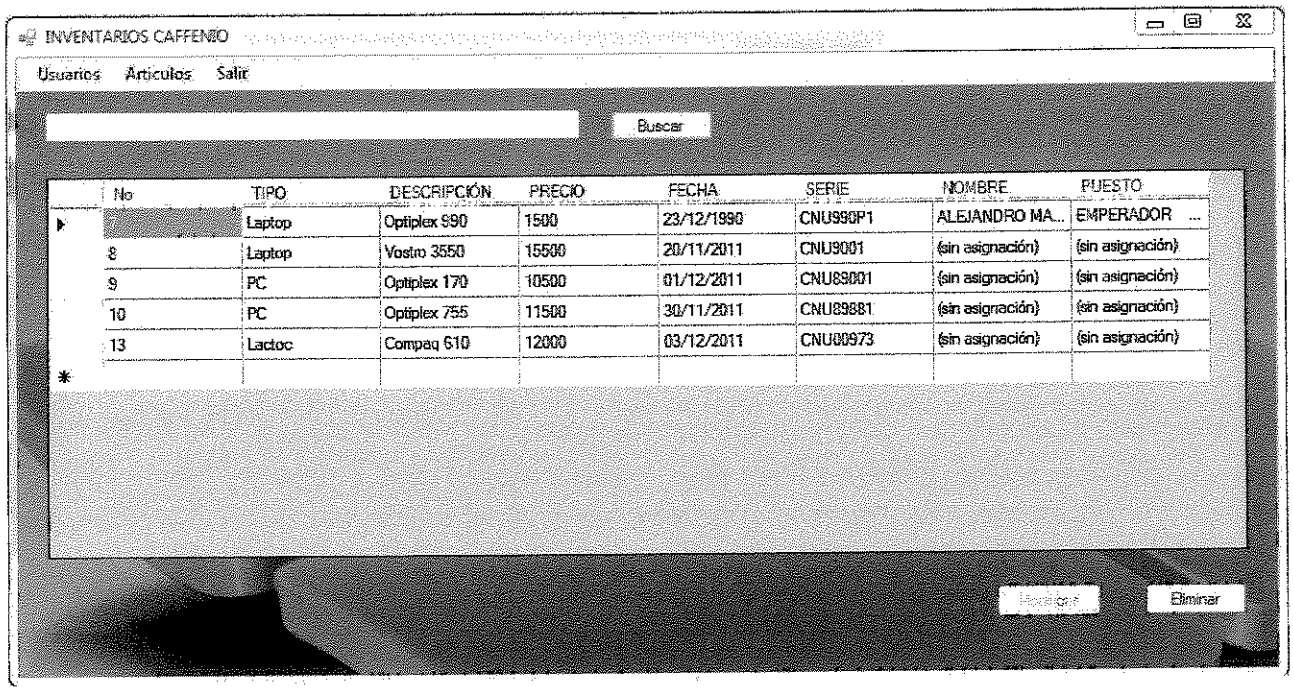


Figura1. Pantalla Principal

En la figura 1 se muestra el menú principal del sistema con su botón de búsqueda por artículo representado en un data grid la información que se quiere mostrar, en este caso un artículo. La búsqueda de información del artículo se puede hacer por tipo, descripción, fecha y serie. Se pueden modificar o eliminar los artículos seleccionando el campo con un click en el data grid y posteriormente dar click en el botón que se desee.

Pantalla de AGREGAR USUARIOS

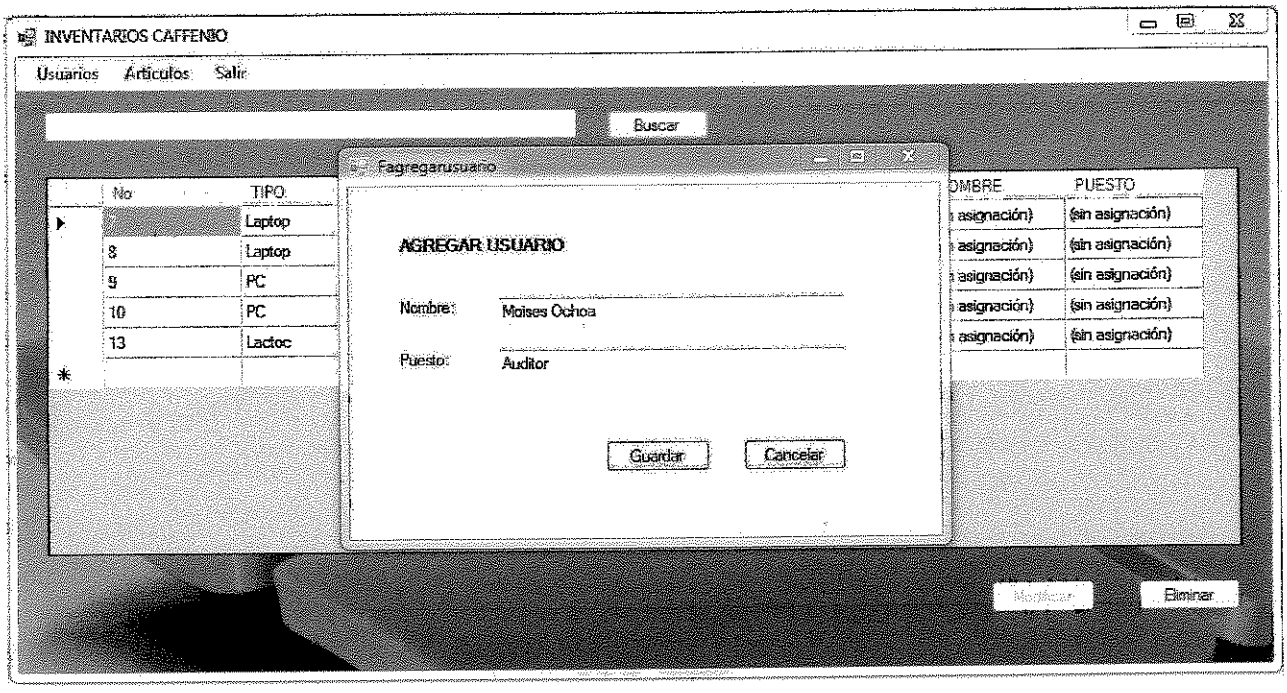


Figura 2. Captura de nuevos usuarios

En la figura 2 se encuentra el menú de Usuarios en el submenú "Agregar". Aquí se van a crear los usuarios a los que se les asignará los artículos. Como se ven pantalla es muy sencillo nomas se debe de poner el nombre y el puesto y le damos click en guardar y listo.

Pantalla MODIFICAR/ELIMINAR USUARIOS

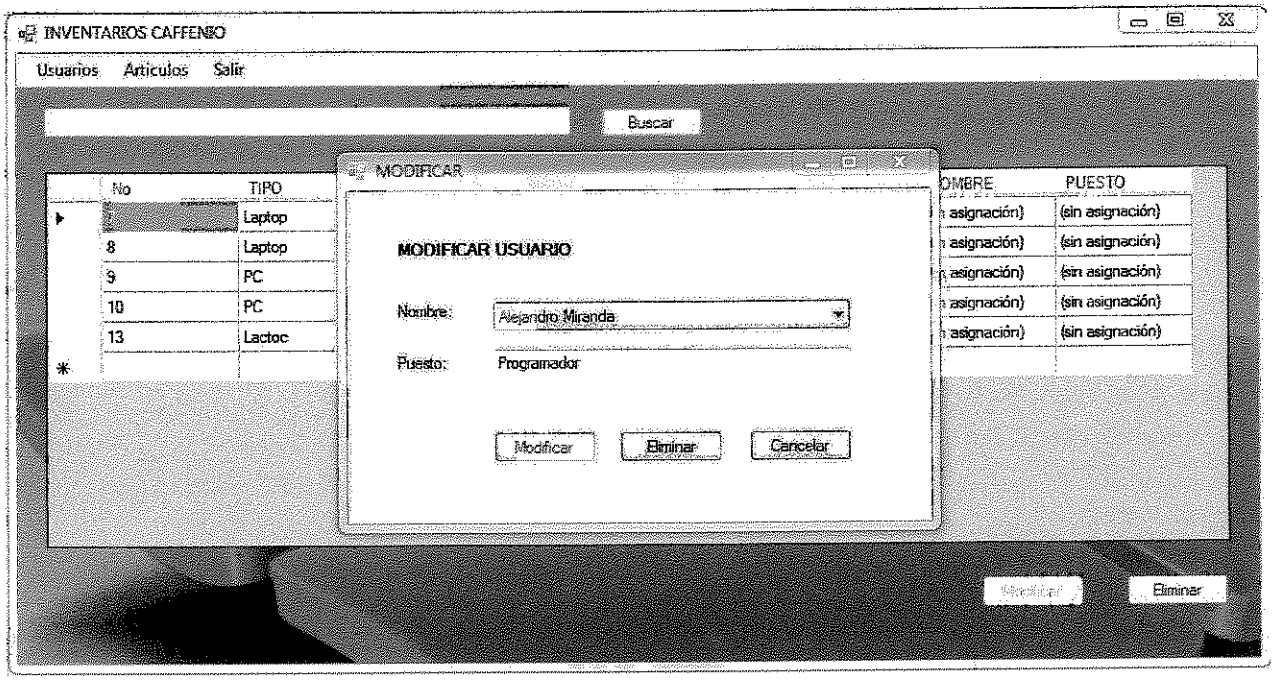


Figura 3. Modificación y/o eliminación de usuarios

En la figura 3 nos muestra el menú de Usuarios en el submenú "Modificar/Eliminar". Esta pantalla sirve para modificar el puesto de un Usuario o eliminar el usuario en su totalidad. El Usuario se escoge a través de un combobox así como se muestra en la imagen

Pantalla AGREGAR ARTICULOS

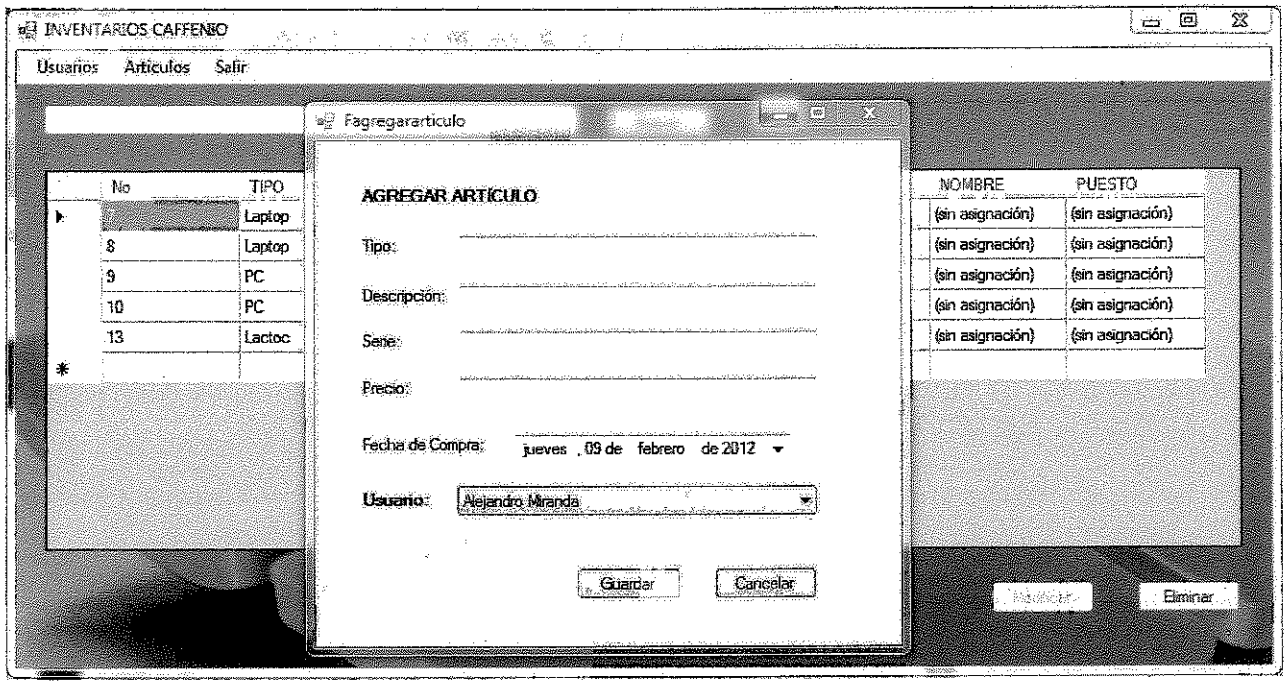


Figura 4. Captura de nuevos artículos

En la figura 4 se encuentra el menú de Artículos en el submenú "Agregar". Como se ve en pantalla para agregar un artículo nos pide el Tipo de artículo (laptop, pc, blackberry, accesorio computacional, etc.), luego Descripción, el número de Serie del artículo, el Precio, la Fecha de Compra y por último a que Usuario se le va asignar.

Pantalla MODIFICAR ARTÍCULO

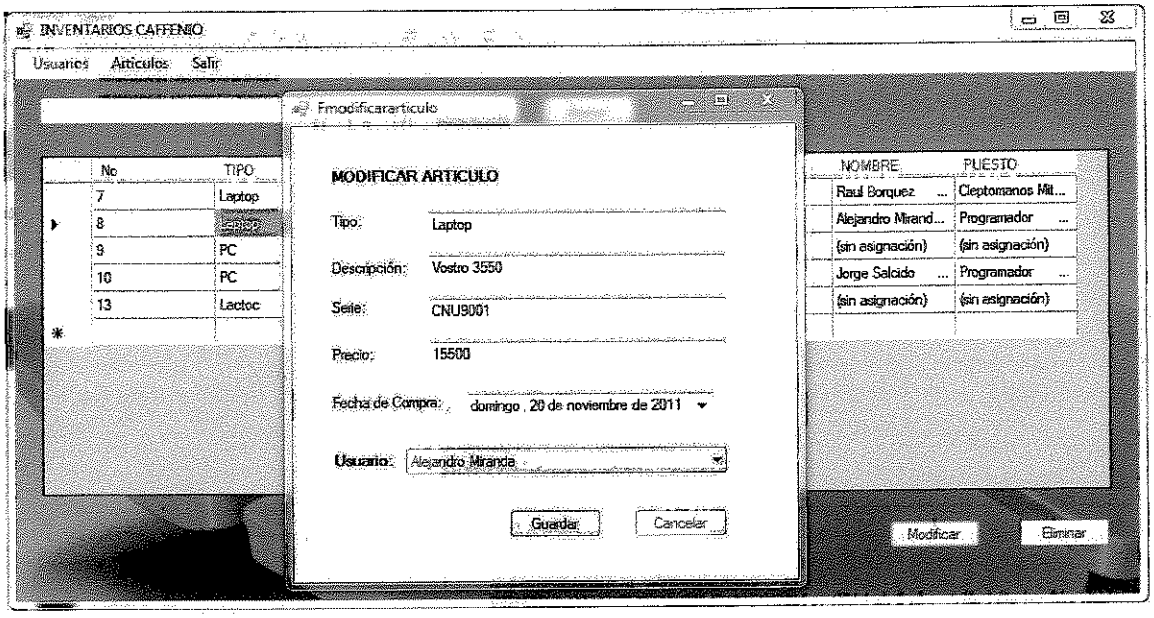


Figura 5. Modificación de artículos

Para modificar un artículo en existencia, así como lo muestra la figura 5, lo que se debe de hacer como lo hemos dicho anteriormente es seleccionar dicho articulo dentro del data grid y darle el botón de modificar al momento de hacer eso, esto aparecerá facilitando la modificación de todos los campos incluso del usuario.

RESULTADOS OBTENIDOS

Al haber instalado el programa de "Inventarios Caffenio" se ha notado una mayor eficiencia en el manejo de la información de las computadoras, esto se refiere a que la información se esta capturando de una manera mas rápida y organizada.

Al momento de hacer consultas en el sistema, el sistema te desglosa lo que realmente se esta buscando con solo teclear unos números o letras y por consiguiente presionar simplemente un botón de buscar, no como en el caso de la hoja de Excel que te limitaba de búsquedas específicas. Ahora sí es más cómodo el hacer las búsquedas de la información que se ocupa ya sea por clientes, por el número de serie, por el puesto, nombre, etc. En el caso de búsqueda por fecha, da la certeza de ver en que fecha se compró la computadora y eso me serviría para varias acciones como: hacer válida alguna garantía que se ocupe porque así ya se sabría la fecha de compra, con la búsqueda de serial igual me sirve para levantar los reportes de garantía, las bajas de computadoras, cambio de computadoras a usuarios, etc.

En general este sistema ayuda a eficientar y ahorrar tiempo en el área de soporte de sistemas para sus habituales procesos cotidianos.

CONCLUSIONES Y RECOMENDACIONES

Este sistema de información en realidad es muy básico, pero de igual manera es lo que se requería en el área de soporte para la captura de las máquinas. La programación no fue muy extensa al igual que la base de datos, pero se cumplió con lo que se propuso.

Se aprendió mucho durante este desarrollo del proyecto ya que no se contaba con mucha experiencia en programación, me costó mucho trabajo pero a final de cuentas todo salió bien.

El sistema está trabajando bien sin ningún problema, es todo lo que se requería para soluciones de problemas muy básicos de soporte más que nada en las fechas de compra de las computadoras.

Me gustaría que en algún futuro se siguiera tomando en cuenta el sistema en cuanto a mejoras del mismo y ver nuevas implementaciones en el sistema. Estaría muy bien más adelante combinarlo con el sistema que se tiene de PDAS y que una persona de soporte aparte se encargara del mismo.

RETROALIMENTACIÓN

Durante la estancia profesional se tuvo que enfrentar a un cambio radical del modo de ver y hacer las cosas, esto ya no nomás es un simple proyecto de escuela sino mas bien un trabajo profesional y serio que se le hizo a la empresa.

Esto requirió un cambio de actitud en la forma de hacer las cosas, de compromiso con la empresa, de prometer con lo que se definió en un principio y hacerlo valer, cumplir con las tareas en el tiempo indicado.

Me tuve que enfrentar a la tarea de ser constante con el proyecto, si no se sabía hacer una cosa investigar hasta encontrar la forma de hacerlo, hasta que quedara bien porque aquí ya no te puedes equivocar así como en la escuela, las cosas quedan bien o no se presentan.

Una de las desventajas que se podría decir que tuve es la forma de hacer las cosas ya que desde un principio no tenía los conocimientos de programación, así que me di a la tarea de investigar y lograr comprender como hacer el sistema.

A continuación se describirán las fortalezas y debilidades, posteriormente las oportunidades y recomendaciones.

Fortalezas:

- Habilidad para detectar el problema.- Desde un principio se detectó lo que se tenía que hacer y cómo resolverlo.
- Habilidad para investigar dicho problema.- No solo se detectó el problema sino que se buscó e investigó la manera de cómo resolverlo.
- Actitud para la resolución del problema.- buena actitud con responsable, siempre de buena manera para conseguir el óptimo resultado.

Debilidades:

- No saber programar.- lo que significa que se tuvo que estudiar, investigar y tomar asesoría con personas del rango de la programación para así resolver el problema.

Oportunidades y Recomendaciones:

Como alumno de la universidad de sonora y el hecho de haber tenido el privilegio de estudiar en tan dichosa y prestigiosa universidad, me permito decir que durante la estancia profesional encontré un área de oportunidad que para mi personalmente tiene gran relevancia para esta carrera de Ingeniería en Sistemas de Información; dicha área de oportunidad que encontré fue lo que es la programación.

Mi recomendación para futuros egresados de la Universidad de Sonora sería que le pusieran más empeño en la programación, que le dediquen más, que consigan mejores maestros, etc. Para mi forma de ver la programación es la base de la carrera y es en lo que mejor es remunerado como profesional, es como si quisieras ser arquitecto y no poder ser capaz de hacer casas, un ingeniero en sistemas las casas son los sistemas que es lo que debería de ser capaz de hacer.

Si el alumno de plano no quisiera programar la otra alternativa sería que la universidad reforzara el nivel en el área de redes, que se dieran por completo los ciscos, me hubiera gustado que hubiera un clase que se llamara "cliente y servidor" y "redes distribuidos", etc. No sé cómo esté ahorita las clases en la carrera, pero a mí si me tocó muy pobre el área de redes, llevamos un cisco 1 de muy bajo nivel y el cisco 2 que llevé no se apegó nada al temario, los demás ciscos no los llevé.

La verdad, como ya lo había comentado, sí me hubiera gustado salir egresado con un buen nivel de conocimiento en redes.

ANEXOS

En el siguiente apartado se va explicar detenidamente la estructura del programa en términos de Visual Studio. El programa cuenta con 5 formas y 2 filas de código que son los siguientes:

- Form1 (Form Application)
- Fagregarusuario. (Form Application)
- Fmodificarusuario (Form Application)
- Fagregararticulo (Form Application)
- Fmodificararticulo (Form Application)
- Variables. (File Code)
- Funciones (File Code)

La **"Form1"** es la ventana principal del sistema en donde se encuentra la barra de menú con los módulos de Usuarios y Artículos. Más abajo se encuentra un textbox con su respectivo botón para las búsqueda de artículos, mas abajo un data grid donde mostrara la búsqueda de artículos y con sus respectivos botones de modificar el articulo y el botón de eliminar artículo. Su respectivo código sería:

```
namespace Inventarios
{
    public partial class Form1 : Form
    {
        public static int id_Articulo = 0;


        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            buscar(false);
        }
```

```
        private    void    agregarToolStripMenuItem1_Click(object    sender,
EventArgs e)
        {
            Fagregararticulo agregararticulo = new Fagregararticulo();
            agregararticulo.Show();
        }

        private    void    agregarToolStripMenuItem_Click(object    sender,
EventArgs e)
        {
            Fagregarusuario agregarusuario = new Fagregarusuario();
            agregarusuario.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Fmodificararticulo        modificararticulo        =        new
Fmodificararticulo();
            modificararticulo.Show();
        }

        private    void    modificarToolStripMenuItem_Click(object    sender,
EventArgs e)
        {
            Fmodificarusuario modificarusuario = new Fmodificarusuario();
            modificarusuario.Show();
        }


        //PARA SALIR DEL SISTEMA

        private     void     salirToolStripMenuItem_Click(object     sender,
EventArgs e)
        {
            Application.Exit();
        }


        //PARA HACER LAS BÚSQUEDAS CON BOTON "Buscar"

        private void textBox1_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                buscar(false);
            }
       }


        //SELECCIONAR UN REGISTRO EN EL DATAGRID Y GUARDARLO EN UNA
VARIABLE PUBLICA Id_Articulo

        private      void      dataGridView1_MouseClick(object      sender,
MouseEventArgs e)
        {
            try
            {
```

```
                if (dataGridView1.Rows.Count > 0)
                {
                    if (e.Button == MouseButtons.Left)
                    {
                        button2.Enabled = true;
                        id_Articulo                                    =
Convert.ToInt32(dataGridView1.Rows[dataGridView1.CurrentRow.Index].Cells[
0].Value);
                    }
                }
            }
            catch
            {
                button2.Enabled = false;
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            buscar(true);
        }


        // Para Buscar Articulos con el boton "Buscar"

        private void buscar(bool opcion)
        {
            DataSet ds = new DataSet();
            string buscar;
            if (opcion == true)
                buscar = "SELECT a.idArticulo AS No, a.Tipo AS TIPO,
a.Descripcion AS DESCRIPCIÓN, a.Precio AS PRECIO, a.Fecha AS FECHA,
a.Serie AS SERIE, u.Nombre AS NOMBRE, u.Puesto AS PUESTO FROM Articulos
a, Usuarios u WHERE a.idUsuario = u.idUsuario UNION SELECT a.idArticulo
AS No, a.Tipo AS TIPO, a.Descripcion AS DESCRIPCIÓN, a.Precio AS PRECIO,
a.Fecha AS FECHA, a.Serie AS SERIE, '(sin asignación)' AS NOMBRE, '(sin
asignación)' AS PUESTO FROM Articulos a WHERE a.idUsuario is NULL OR
a.idUsuario = 0";
            else
                buscar = "SELECT a.idArticulo AS No, a.Tipo AS TIPO,
a.Descripcion AS DESCRIPCIÓN, a.Precio AS PRECIO, a.Fecha AS FECHA,
a.Serie AS SERIE, u.Nombre AS NOMBRE, u.Puesto AS PUESTO FROM Articulos
a, Usuarios u WHERE a.idUsuario = u.idUsuario AND (Descripcion like '%" +
textBox1.Text + "%' OR Tipo like '%" + textBox1.Text + "%') UNION SELECT
a.idArticulo AS No, a.Tipo AS TIPO, a.Descripcion AS DESCRIPCIÓN,
a.Precio AS PRECIO, a.Fecha AS FECHA, a.Serie AS SERIE, '(sin
asignación)' AS NOMBRE, '(sin asignación)' AS PUESTO FROM Articulos a
WHERE (a.idUsuario is NULL OR a.idUsuario = 0) AND (Descripcion like '%"
+ textBox1.Text + "%' OR Tipo like '%" + textBox1.Text + "%')";
            Variables.conexion_servidor                    =                new
SqlConnection(Variables.cadena_conexion);
            Variables.conexion_servidor.Open();
            Variables.adaptador      =      new      SqlDataAdapter(buscar,
Variables.conexion_servidor);
            Variables.adaptador.Fill(ds);
            dataGridView1.DataSource = ds.Tables[0];
        }
```

```
        //PARA ELIMINAR UN ARTÍCULO DEL DATA GRI

        private void button3_Click(object sender, EventArgs e)
        {
            ABC METODO_ABC = new ABC();
            METODO_ABC.ELIMINAR(Form1.id_Articulo, 0, true);
        }
    }
}
```

La Forma **"Fagregarusuario"** es la ventana en donde se van agregar los artículos y su respectivo código sería:

```
namespace Inventarios
{
    public partial class Fagregarusuario : Form
    {
        public Fagregarusuario()
        {
            InitializeComponent();
        }

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

// PARA AGREGAR USUARIO

        private void button1_Click(object sender, EventArgs e)
        {
            string insertar = "INSERT INTO Usuarios (Nombre, Puesto)
VALUES ('"+textBox1.Text+"','"+textBox2.Text+"')";

            Variables.conexion_servidor = new
SqlConnection(Variables.cadena_conexion);
            Variables.conexion_servidor.Open();
            Variables.comando =
Variables.conexion_servidor.CreateCommand();
            Variables.comando.CommandText = insertar;
            Variables.comando.ExecuteNonQuery();

            Variables.conexion_servidor.Close();

            MessageBox.Show("USUARIO AGREGADO");
        }
    }
}
```

La forma **"Fmodificarusuario"** es la ventana en donde se pueden modificar y eliminar usuarios. Su código sería:

```
namespace Inventarios
{
    public partial class Fmodificarusuario : Form
    {
        public Fmodificarusuario()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        // PARA MOSTRAR LOS USUARIOS EN EL COMBOBOX

        private void Fmodificarusuario_Load(object sender, EventArgs e)
        {
            LlenarDatos();

            if (comboBox1.Items.Count < 1)
            {
                button2.Enabled = false;
                button3.Enabled = false;
            }
        }

        //PARA MOSTRAR EL PUESTO DEL USUARIO AL SELECCIONARLO

        private    void    comboBox1_SelectedIndexChanged(object    sender,
EventArgs e)
        {
            string leer = "SELECT rTrim(Puesto) AS Puesto FROM Usuarios
WHERE idUsuario = " + Convert.ToString(comboBox1.SelectedValue) + "";

            Variables.conexion_servidor                =                new
SqlConnection(Variables.cadena_conexion);
            Variables.conexion_servidor.Open();
            Variables.comando                                            =
Variables.conexion_servidor.CreateCommand();
            Variables.comando.CommandText = leer;
            Variables.lectura = Variables.comando.ExecuteReader();
            while (Variables.lectura.Read())
            {
                textBox2.Text                                            =
Convert.ToString(Variables.lectura["Puesto"]);
            }


            Variables.conexion_servidor.Close();
        }
```

```
        //MODIFICAR EL PUESTO DEL USUARIO SEGUN EL COMBO BOX

        private void button2_Click(object sender, EventArgs e)

        {
            string insertar = "UPDATE Usuarios SET Nombre = '" +
comboBox1.Text + "', Puesto = '" + textBox2.Text + "' WHERE idUsuario = "
+ Convert.ToString(comboBox1.SelectedValue) + "";

            Variables.conexion_servidor = new
SqlConnection(Variables.cadena_conexion);
            Variables.conexion_servidor.Open();
            Variables.comando =
Variables.conexion_servidor.CreateCommand();
            Variables.comando.CommandText = insertar;
            Variables.comando.ExecuteNonQuery();

            Variables.conexion_servidor.Close();
            MessageBox.Show("USUARIO MODIFICADO");
        }

        //ELIMINAR USUARIOS

        private void button3_Click(object sender, EventArgs e)
        {
            ABC METODO_ABC = new ABC();
            METODO_ABC.ELIMINAR(0,
Convert.ToInt32(comboBox1.SelectedValue), false);
            LlenarDatos();
        }

        public void LlenarDatos()
        {
            DataSet ds = new DataSet();
            string buscar = "SELECT idUsuario AS ID, rTrim(Nombre) AS N
FROM Usuarios";
            Variables.conexion_servidor = new
SqlConnection(Variables.cadena_conexion);
            Variables.conexion_servidor.Open();
            Variables.adaptador = new SqlDataAdapter(buscar,
Variables.conexion_servidor);
            Variables.adaptador.Fill(ds);
            comboBox1.DataSource = ds.Tables[0];
            Variables.conexion_servidor.Close();
        }
    }
}
```

La **"Fagregarartículo"** aquí es donde se agregará los artículos y a qué usuario se le destinará también. Su código sería:

```
namespace Inventarios
{
    public partial class Fagregararticulo : Form
    {
```

```
        public Fagregararticulo()
        {
            InitializeComponent();
        }

        //GUARDAR ARTÍCULOS

        private void button1_Click(object sender, EventArgs e)
        {
            string insertar = "INSERT INTO Articulos (Tipo, Descripcion,
Serie, Precio, Fecha, idUsuario) VALUES ('" + TBFagregarTipo.Text + "','"
+ TBFagregarDescripcion.Text + "','" + TBFagregarSerie.Text + "','" +
TBagregarPrecio.Text + "','" + dateTimePicker1.Text + "', " +
Convert.ToInt32(comboBox1.SelectedValue) + ")";

            Variables.conexion_servidor = new
SqlConnection(Variables.cadena_conexion);
            Variables.conexion_servidor.Open();
            Variables.comando =
Variables.conexion_servidor.CreateCommand();
            Variables.comando.CommandText = insertar;
            Variables.comando.ExecuteNonQuery();
            Variables.conexion_servidor.Close();
            MessageBox.Show("ARTICULO AGREGADO");
        }

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

        //MOSTRAR LOS  USUARIOS EN EL COMBOBOX

        private void Fagregararticulo_Load(object sender, EventArgs e)
        {
            DataSet ds = new DataSet();
            string buscar = "SELECT idUsuario AS ID, rTrim(Nombre) AS N
FROM Usuarios";
            Variables.conexion_servidor = new
SqlConnection(Variables.cadena_conexion);
            Variables.conexion_servidor.Open();
            Variables.adaptador = new SqlDataAdapter(buscar,
Variables.conexion_servidor);
            Variables.adaptador.Fill(ds);
            DataRow Fila = ds.Tables[0].NewRow();
            Fila["ID"] = "0";
            Fila["N"] = "(ninguno)";
            ds.Tables[0].Rows.Add(Fila);
            comboBox1.DataSource = ds.Tables[0];
            Variables.conexion_servidor.Close();
        }
    }
}
```

La forma **"Fmodificararticulo"** como dice su nombre es donde se modificará el artículo en caso de requerirlo. Su código:

```
namespace Inventarios
{
    public partial class Fmodificararticulo : Form
    {
        public Fmodificararticulo()
        {
            InitializeComponent();
        }

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


        private void Fmodificararticulo_Load(object sender, EventArgs e)
        {


            //MOSTRAR LOS USUARIOS EN EL COMBOBOX

            DataSet ds = new DataSet();
            string buscar = "SELECT idUsuario AS ID, rTrim(Nombre) AS N
FROM Usuarios";
            Variables.conexion_servidor                =                new
SqlConnection(Variables.cadena_conexion);
            Variables.conexion_servidor.Open();
            Variables.adaptador      =      new      SqlDataAdapter(buscar,
Variables.conexion_servidor);
            Variables.adaptador.Fill(ds);
            DataRow Fila = ds.Tables[0].NewRow();
            Fila["ID"] = "0";
            Fila["N"] = "(ninguno)";
            ds.Tables[0].Rows.Add(Fila);
            comboBox1.DataSource = ds.Tables[0];
            Variables.conexion_servidor.Close();

            //MOSTRAR LOS DATOS EN LOS CAMPOS TEXTBOXES

            string leer = "SELECT rTrim(Tipo) AS T, rTrim(Descripcion) AS
D, rTrim(Serie) AS S, Fecha AS F, rTrim(Precio) AS P, idUsuario AS U FROM
Articulos WHERE idArticulo = " + Form1.id_Articulo + "";

            Variables.conexion_servidor                =                new
SqlConnection(Variables.cadena_conexion);
            Variables.conexion_servidor.Open();
            Variables.comando                                           =
Variables.conexion_servidor.CreateCommand();
            Variables.comando.CommandText = leer;
            Variables.lectura = Variables.comando.ExecuteReader();
            while (Variables.lectura.Read())
            {
```

```
                textBox1.Text = Convert.ToString(Variables.lectura["T"]);
                textBox2.Text = Convert.ToString(Variables.lectura["D"]);
                textBox3.Text = Convert.ToString(Variables.lectura["S"]);
                dateTimePicker1.Text                                    =
Convert.ToString(Variables.lectura["F"]);
                textBox5.Text = Convert.ToString(Variables.lectura["P"]);
                comboBox1.SelectedValue                                 =
Convert.ToString(Variables.lectura["U"]);
            }
            Variables.conexion_servidor.Close();
        }

        //GUARDAR LOS DATOS

        private void button1_Click(object sender, EventArgs e)
        {
            DateTime fecha = Convert.ToDateTime(dateTimePicker1.Text);

            string insertar = "UPDATE Articulos SET Tipo = '" +
textBox1.Text + "', Descripcion = '" + textBox2.Text + "', Serie = '" +
textBox3.Text + "', Precio = " + textBox5.Text + ", Fecha = '" +
fecha.ToString("dd/MM/yyyy HH:mm:ss") + "', idUsuario = " +
Convert.ToInt32(comboBox1.SelectedValue) + " WHERE idArticulo = " +
Form1.id_Articulo + "";

            Variables.conexion_servidor                 =                 new
SqlConnection(Variables.cadena_conexion);
            Variables.conexion_servidor.Open();
            Variables.comando                                             =
Variables.conexion_servidor.CreateCommand();
            Variables.comando.CommandText = insertar;
            Variables.comando.ExecuteNonQuery();
            Variables.conexion_servidor.Close();
            MessageBox.Show("ARTICULO MODIFICADO");
        }
    }
}
```

La clase **"Variables"** es un archivo de código en donde se escribió código para llamar a la función de la cadena de conexión de la base de datos. Su código sería:

```
public class Variables
{
    public static string cadena_conexion = @"Data Source=LIMON-
PC\SQLEXPRESS;Initial Catalog=InventarioBD;Integrated Security=True";
    public static SqlConnection conexion_servidor;
    public static SqlCommand comando;
    public static SqlDataReader lectura;
    public static SqlDataAdapter adaptador;
}
```

La clase **"FUNCIONES"** es otro archivo de código para hacer la llamada de funciones, en éste caso se hizo dos tipos de funciones que son para la eliminación de un artículo o eliminación de un usuario. Esto es con el fin de ahorrar código y de hacer la programación de una manera más efectiva y eficiente. Su código:

```
public class ABC
{
    /// <summary>
    /// SIRVE PARA ELIMINAR UN ARTICULO O UN USUARIO. Si se desea
eliminar un articulo, teclee el idarticulo,
    /// idUsuario = 0 y opcion = true
    /// </summary>
    public void ELIMINAR(int idArticulo, int idUsuario, bool opcion)
    {
        string eliminar;
        if (opcion == true)
            eliminar = "DELETE Articulos WHERE idArticulo = " +
idArticulo + "";
        else
            eliminar = "DELETE Usuarios WHERE idUsuario = " + idUsuario +
"";

        Variables.conexion_servidor = new
SqlConnection(Variables.cadena_conexion);
        Variables.conexion_servidor.Open();
        Variables.comando = Variables.conexion_servidor.CreateCommand();
        Variables.comando.CommandText = eliminar;
        Variables.comando.ExecuteNonQuery();

        if (opcion == false)
        {
            Variables.comando.CommandText = "UPDATE Articulos SET
idUsuario = 0 WHERE idUsuario = " + idUsuario + "";
            Variables.comando.ExecuteNonQuery();
        }

        Variables.conexion_servidor.Close();
        MessageBox.Show("¡ELIMINADO!");
    }
}
```

Con respecto a la base de datos consta de dos tablas una que se llama Artículos y otra de Usuarios, a continuación se presentara de que consta la tabla de Artículos:

- idArticulo (PK, int, No NULL) : Es la llave principal, es un entero y no acepta caracteres nulos.

- Tipo (nchar(10), No NULL) : Variable nchar de 10 caracteres, no acepta valores nulos.
- Descripcion (nvarchar(20), No NULL) : Variable de tipo nvarchar de 20 caracteres, no acepta valores nulos.
- Serie (nvarchar(10), NULL) : Variable tipo nvarchar de 10 caracteres, si acepta valores nulos.
- Precio (int, NULL) : Variable de tipo entero y acepta valores nulos
- Fecha (datetime, NULL) : Variable de tipo fecha aceptando valores nulos
- idUsuario (int, NULL) : Variable de tipo relación con la tabla de usuarios, de tipo entero y acepta valores nulos.

Ahora veremos como esta constituido la tabla de Usuarios:

- idUsuario (PK, int, No NULL) : Llave principal, variable tipo entero y no acepta valores nulos.
- Nombre (char(50), No NULL) : Variable tipo char de 50 caracteres, no acepta valores nulos.
- Puesto (char(30), No NULL) : Variable tipo char de 30 caracteres, no acepta valores nulos.

REFERENCIAS BIBLIOGRÁFICAS

- Microsoft, Introduction to C# Programming with Microsoft .NET, 2609A, EUA, 05/2002.
- Daniel Solis, Illustrated C# 2008: C# presented clearly, concisely, and visually, New York, 2008
- El guille*: **http://www.elguille.info/NET/cursoCSharpErik/index.htm***