UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

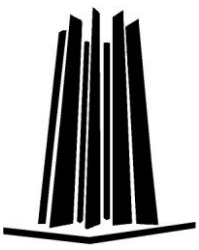
FACULTAD DE ESTUDIOS SUPERIORES
ARAGÓN

"SISTEMA DE ADMINISTRACIÓN DE CENTROS DE CÓMPUTO"

T R A B A J O E S C R I T O
EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL
QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN
P R E S E N T A:

MARIO MINORU TACHIKA OHARA

ASESOR: M. EN I. ARCELIA BERNAL DÍAZ



MÉXICO, 2008.

Contenido

Temario

Índice de Figuras

Anexos

Introducción

Actualmente la computadora se ha vuelto una herramienta de trabajo indispensable para todas las empresas, tanto públicas como privadas, lo que ha llevado a la necesidad de implementar redes de cómputo locales y por lo tanto tener un área de sistemas que ofrezca la infraestructura y el soporte necesarios para que los empleados de la empresa puedan desempeñar sus funciones.

El presente trabajo está enfocado hacia una institución gubernamental, la Auditoría Superior de la Federación, la cual posee una Dirección General de Sistemas, que se encarga de otorgar la infraestructura tecnológica necesaria para que la institución pueda cumplir con sus objetivos; esta Dirección General posee tres Direcciones de Área: la Dirección de Desarrollo de Sistemas, que se encarga del desarrollo, implementación y mantenimiento de sistemas administrativos internos; la Dirección de Mantenimiento, que se encarga de dar soporte técnico y mantenimiento al equipo de cómputo de los usuarios finales; y la Dirección de Tecnología, que se encarga de implementar y mantener toda la infraestructura física y lógica de la red, así como de administrar y operar los centros de cómputo.

Una subdirección perteneciente a la Dirección de Tecnología es la Subdirección de Tecnologías de Información, que es la responsable de administrar y operar los centros de cómputo así como los servicios que en sus equipos se implementen, tales como correo electrónico o acceso a Internet. Esta subdirección también tiene a su resguardo el software institucional, el cual se proporciona al personal de la Dirección de Mantenimiento cuando lo requiere.

El propósito de este trabajo es diseñar, desarrollar e implementar en un sistema de cómputo el manejo de dos procesos administrativos de la Subdirección de Tecnologías de Información: el control de software, y el inventario físico de los equipos en los centros de cómputo.

El desarrollo de este sistema tiene como objetivos: mantener un inventario actualizado del hardware y software existente, e implementar controles adecuados para cuando se proporciona software al personal de la Dirección General de Sistemas, o cuando hay movimientos de equipos por mantenimientos, actualizaciones o reubicaciones.

La organización, que en este caso es la Dirección General de Sistemas, contará con una herramienta de trabajo eficaz para el control de los centros de cómputo institucionales.

El sistema de información está diseñado para llevar un control preciso de los dos procesos administrativos mencionados anteriormente (control de software e inventario de equipos), los cuales se muestran en el anexo 1, en forma de diagramas conceptuales de los módulos que integran el sistema, así como las operaciones que efectúa cada módulo.

El propósito del módulo de software es llevar a cabo un control preciso del software perteneciente a la Subdirección de Tecnologías de Información, manteniendo también un control de préstamo del software, así como de las licencias permitidas. Para esto se apoyará en la generación de dos bitácoras, una para los movimientos del software, y otra para los préstamos al personal.

El módulo de hardware tiene como objetivos registrar las características de los servidores pertenecientes a los centros de cómputo de la Auditoría Superior de la Federación, así como su ubicación, y llevar un control preciso de movimientos que tengan, como mantenimientos, mejoras al hardware o cambios de ubicación.

Por lo tanto, el presente trabajo consta de 5 capítulos, organizados de la siguiente manera:

1. Antecedentes: Se realiza una breve descripción del análisis de requerimientos para el desarrollo del sistema.
2. Análisis y Diseño del Sistema de Administración de Centros de Cómputo: Se elaboran a detalle los módulos principales del sistema y se define la estructura de la base de datos.
3. Desarrollo del Sistema: Se elabora la aplicación que utiliza la base de datos.
4. Resultados: Se muestra el funcionamiento del sistema ya implementado.
5. Conclusiones.

1. Antecedentes

La Subdirección de Tecnologías de Información es la responsable de administrar la infraestrucutra física de la red de cómputo instalada en la Auditoría Superior de la Federación, por lo tanto, es la que se encarga de llevar el control del cableado estructurado, administrar los nodos de red, administrar los equipos, servicios y aplicaciones que se implementen dentro de los centros de cómputo, respaldar la información almacenada en los equipos de dichos centros de cómputo, proporcionar el servicio de acceso a internet y correo electrónico, administrar el software adquirido para la institución, definir el software que se instalará en los equipos de los usuarios finales, definir políticas de seguridad para el uso de los equipos de cómputo de los usuarios finales, y estar actualizado en cuanto a las tecnologías que involucran las actividades anteriores.

La administración del software institucional consiste en llevar un inventario del software adquirido, y el control del mismo para evitar que se utilicen programas no autorizados, o se exceda el número de licencias. El registro de estos medios (CDs o DVDs) actualmente se lleva a cabo en una hoja de cálculo, la cual ya está al límite de su capacidad debido al constante aumento de medios adquiridos. Además, el control del software actualmente no se lleva, por lo que se dificulta el rastreo y oportuna entrega de los medios. Estos dos factores han llevado a la necesidad de elaborar un sistema de información que permita el inventario de software entregado a la Subdirección por parte del proveedor, y además controle los medios registrando el software solicitado por el personal de la Dirección General.

Para poder llevar a acabo la administración de los equipos en los centros de cómputo se necesita contar con un inventario del equipo físico asignado a los mismos y tener un control preciso sobre él, lo que llevó a la integración de un módulo dentro del sistema que registre y actualice esta información para que esté a disposición del personal de la Dirección General de Sistemas.

2. Análisis

2.1 Consideraciones necesarias para el diseño del sistema

A continuación se mencionan algunas consideraciones para el diseño de los módulos principales del sistema: Software y Hardware.

2.1.1 Software

Para que el control del software se pueda lleva a cabo de forma precisa, se deben tomar en cuenta las siguientes consideraciones para el préstamo de software, así como para el personal que lo solicita:

1. Se generará un formato de préstamo, el cual firmará el solicitante. Este formato tendrá la fecha en la que se solicitó, y la fecha en la que deberá ser devuelto.
2. El préstamo solo se hará por un día.
3. No hay límite para las extensiones de préstamo, pero serán por períodos de un día, generando los formatos que sean necesarios.
4. Si hay más de una copia del software, sólo se prestará una copia del mismo
5. En el caso de software con un número limitado de licencias, se deberán registrar los datos del usuario final y la fecha de la instalación.
6. Solo estarán registrados empleados activos de la Dirección General (no incluye personal de servicio social ni prácticas profesionales).
7. El solicitante puede estar registrado en el sistema, pero puede no estar autorizado para pedir software.

En base a estas consideraciones se diseñó el diagrama conceptual correspondiente al software (anexo 1), el cual muestra dos componentes

principales: préstamos y reportes, siendo *préstamos* la parte del sistema que lleva el control del software, teniendo cuatro operaciones:

1. Alta de Préstamo
2. Devolución de Préstamo
3. Extensión de Préstamo
4. Reasignación de licencias

El diagrama de actividades, el modelo de la organización y los diagramas de casos de uso correspondientes a esta parte del sistema estarán enfocados a la explicación de estas cuatro operaciones.

2.1.2 Hardware

Al igual que en el módulo de software, para poder llevar un control preciso de este inventario es necesario definir algunas consideraciones para el registro de esta información:

1. Solo cubrirá equipos de cómputo, no considera equipos de otro tipo que puedan estar dentro del centro de cómputo.
2. El mantenimiento correctivo solo aplica para los equipos que tengan garantía, o a los que se les pueda dar servicio sin costo.

Al igual que en el módulo de software, a partir de estas consideraciones se diseñó el diagrama conceptual correspondiente al hardware (anexo 1), que también muestra dos componentes principales: movimientos y reportes, siendo *movimientos* la parte del sistema que lleva el control del inventario de hardware, teniendo cuatro operaciones:

1. Mantenimiento preventivo
2. Mantenimiento correctivo

El diagrama de actividades, el modelo de la organización, y los diagramas de casos de uso correspondientes a esta parte del sistema estarán enfocados a la explicación de estas cinco operaciones.

2.2 Diagrama de Actividades

El Diagrama de Actividades es el resultado de realizar un estudio de la organización en cada una de las áreas, y estudiando éstas desde la perspectiva de los distintos tipos de usuarios que intervienen. Los procesos son actividades que tienen entradas y salidas, que pueden ser ejecutados por personas o máquinas, y pueden ser manuales o automáticos. Para el propósito del presente trabajo, se plantean diagramas de actividades correspondientes a los módulos de préstamos de software, y movimientos de hardware, en los cuales se puede observar que las áreas funcionales, es decir, las áreas involucradas en las distintas operaciones que se están analizando, son las que conforman la Dirección General de Sistemas. Estos diagramas se muestran en el anexo 2.

2.3 Modelo de la Organización

El modelo de la organización es una representación gráfica de los procesos que maneja el sistema de información, para lo que se utilizan diagramas de estructura*, los cuales tienen 4 componentes básicos:

1. Entidades: representadas por un cuadro con esquinas rectas (figura 2.1).

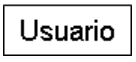


Fig. 2.1 Ejemplo de una Entidad

* Consultar las referencias [4], [5] y [8] de *Bibliografía y Referencias*

2. Procesos: representados por un cuadro con esquinas redondeadas, con una separación horizontal (figura 2.2).

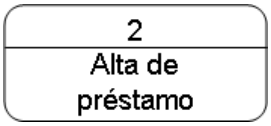


Fig. 2.2 Ejemplo de un proceso

3. Flujos: representada por una flecha.

4. Almacenes de Datos: representados por un cuadro con esquinas redondeadas, con una separación vertical (figura 2.3).

Fig. 2.3 Ejemplo de un Almacén de Datos

El modelo de la organización, al igual que el diagrama de actividades (capítulo 2.2.), está dividido de acuerdo a dos procesos: préstamos de software, y movimientos de hardware.

2.3.1 Software

La representación del diagrama conceptual correspondiente al módulo de préstamos de software (ver anexo 1) en un diagrama de estructura se muestra en la figura 2.4.

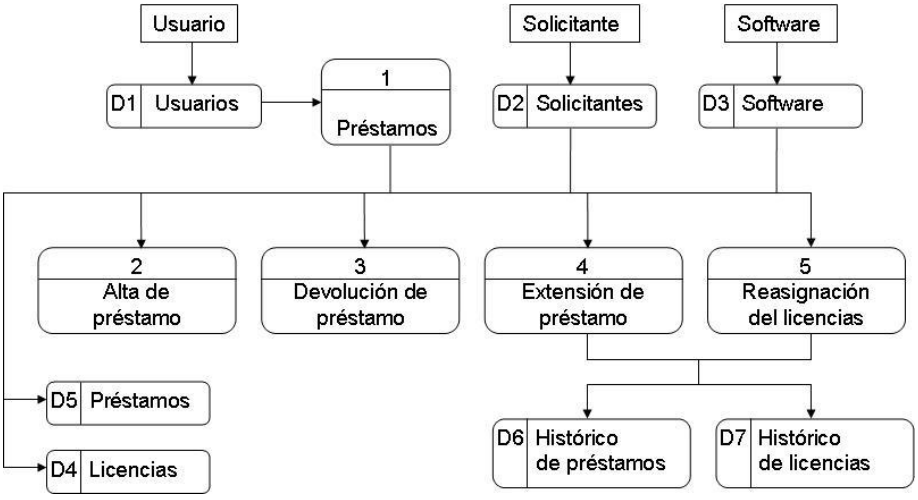


Fig. 2.4 Diagrama de estructura de préstamo de software

Para esta parte del sistema se identifican tres entidades principales:

- Software: El software que se va a inventariar y registrar de préstamo.
- Usuario: La persona que registra el software, así como los préstamos.
- Solicitante: La persona que necesita el software para préstamo.

Para elaborar los diagramas de estructura de datos, se identifican 7 almacenes de datos:

- Usuarios: Los usuarios que trabajan con el sistema.
- Solicitantes: Los usuarios que necesitan un software.
- Software: El inventario del software.
- Préstamos: El control de los préstamos.
- Histórico de Préstamos: Registros anteriores relacionados con un mismo préstamo.
- Licencias: Datos de usuario final con software que tenga número limitado de licencias.
- Histórico de Licencias: Registro histórico relacionado al control de licencias.

Alta de préstamo

La figura 2.5 muestra el diagrama de estructura para el registro de un préstamo de software.

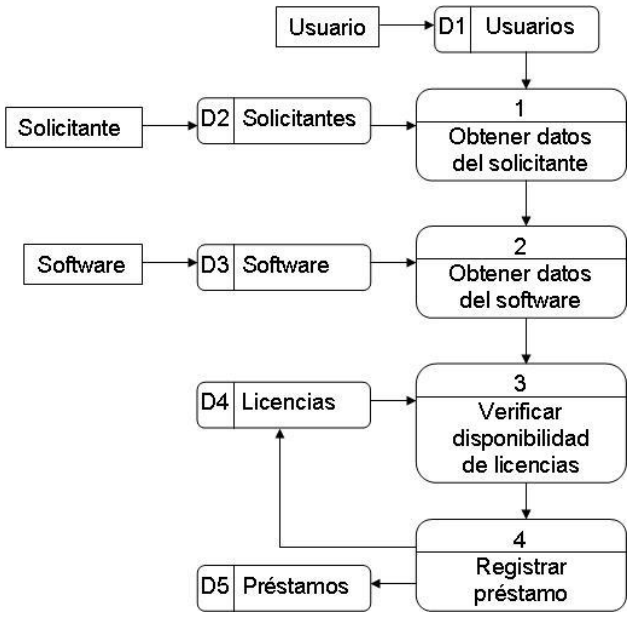


Fig 2.5 Diagrama de estructura de alta de préstamo de software

Los pasos para registrar un préstamo de software son:

1. El usuario ingresa su clave de acceso al sistema, se valida en el catálogo de usuarios (D1) y obtiene del catálogo de solicitantes (D2) los datos del empleado que necesita el software.
2. Obtiene del catálogo de software (D3) los datos del medio que necesita, así como el número de licencias disponibles.
3. Verifica en el almacén de datos de Licencias (D4) si ya ha sido asignado el total de licencias disponibles.
4. Se genera un registro en el almacén de datos de Préstamos (D5) y, en caso de que haya licencias disponibles, se genera un nuevo registro en el almacén de datos de Licencias (D4).

Devolución de Préstamo

La figura 2.6 representa el proceso de devolución de préstamo en un diagrama de estructura.

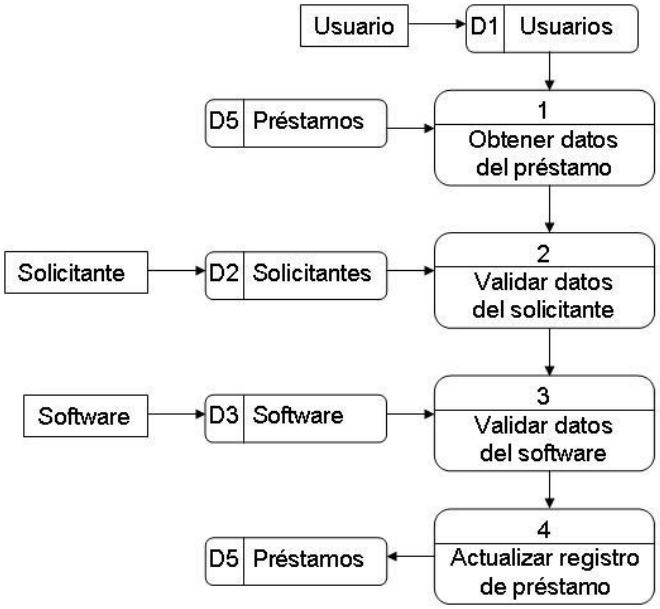


Fig. 2.6 Diagrama de estructura de devolución de software

Los pasos para devolver un software son:

1. El usuario ingresa su clave de acceso al sistema, se valida en el catálogo de usuarios (D1) y obtiene los datos del préstamo del almacén de datos correspondiente (D5).
2. Se validan los datos del solcitante (D2).
3. Se validan los datos del software (D3).
4. Se actualiza el registro correspondiente en el almacén de datos (D5).

Extensión de Préstamo

La figura 2.7 muestra el diagrama de estructura correspondiente a una extensión de préstamo.

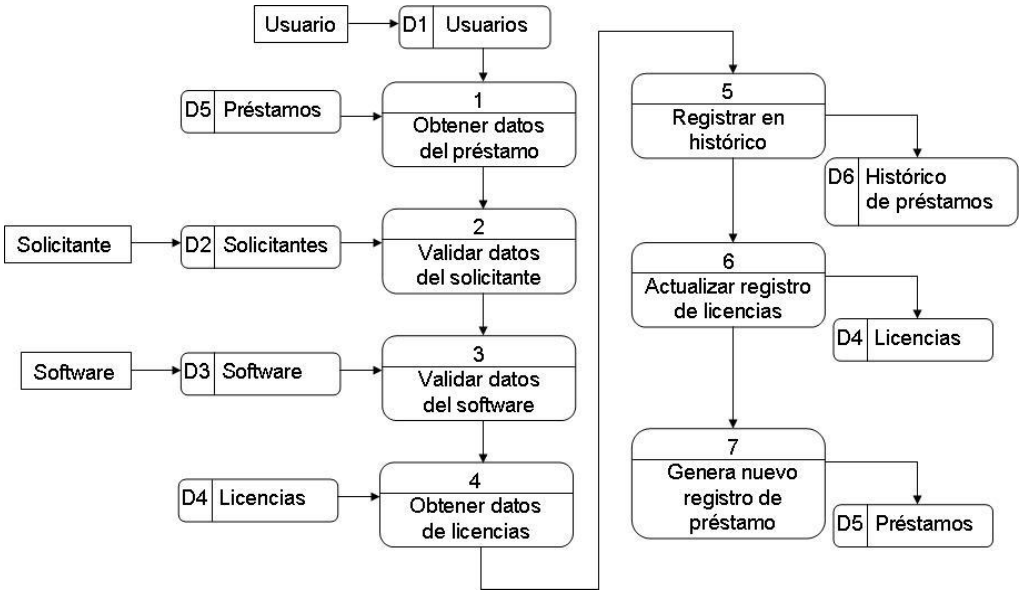


Fig. 2.7 Diagrama de estructura de extensión de préstamo de software

Los pasos para registrar una extensión de préstamo son:

1. El usuario ingresa su clave de acceso al sistema, se valida en el catálogo de usuarios (D1) y obtiene los datos del préstamo del almacén de datos correspondiente (D5).
2. Se validan los datos del solicitante (D2).
3. Se validan los datos del software (D3).
4. Si el software tiene un número limitado de licencias, se obtienen la información del almacén de datos correspondiente (D4).
5. Se genera un registro histórico con los datos anteriores (D6).
6. Si el software tiene un número limitado de licencias, se actualizan los registros en el almacén de datos correspondiente (D4).
7. Se genera un nuevo registro en el almacén de datos de Préstamos (D5).

Reasignación de Licencias

El diagrama de estructutra de una reasignación de licencia se muestra en la figura 2.8.

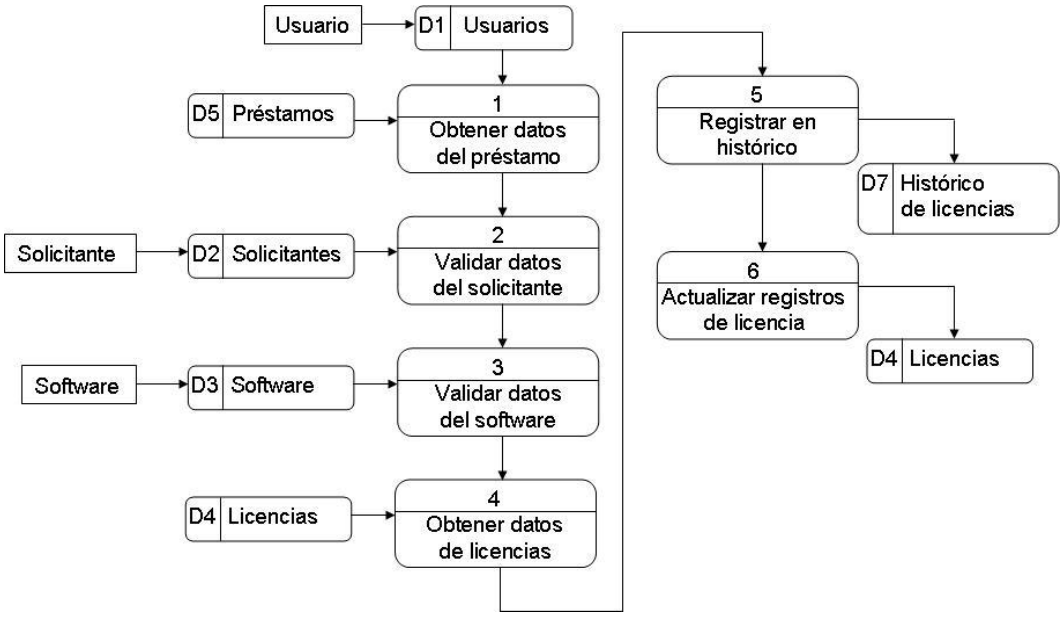


Fig. 2.8 Diagrama de estructura de reasignación de licencia de software

Los pasos para registrar una reasignación de licencias son:

1. El usuario ingresa su clave de acceso al sistema, se valida en el catálogo de usuarios (D1) y obtiene los datos del préstamo del almacén de datos correspondiente (D5).
2. Se validan los datos del solicitante (D2).
3. Se validan los datos del software (D3).
4. Si el software tiene un número limitado de licencias, se obtienen la información del almacén de datos correspondiente (D4).
5. Se genera un registro histórico con los datos anteriores (D7).
6. Se actualiza la información en el almacén de datos de licencias (D4).

2.3.2 Hardware

La figura 2.9 muestra el diagrama de estructura del módulo de movimientos de hardware, de acuerdo al diagrama conceptual (anexo 1).

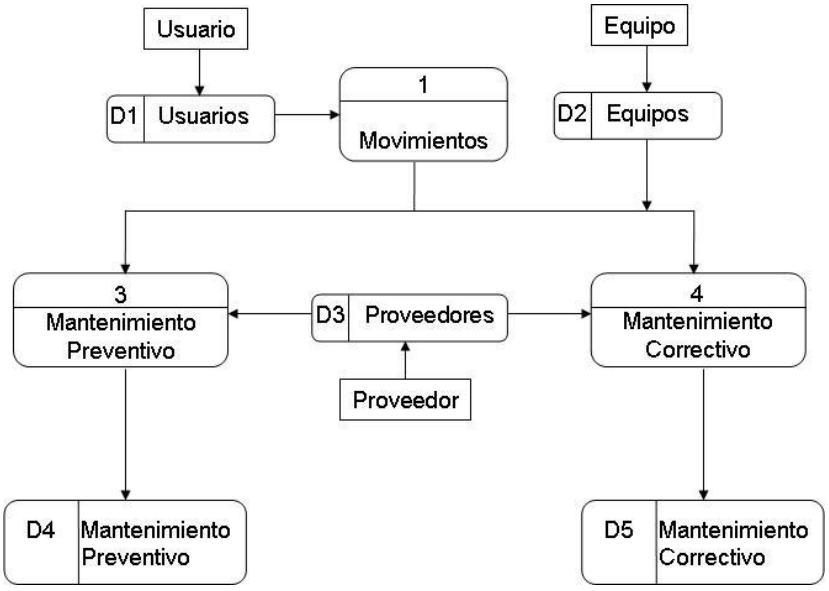


Fig. 2.9 Diagrama de estructura de movimientos en hardware

Para esta parte del sistema se identifican tres entidades principales:

- Usuario: La persona que registra los movimientos.
- Equipo: El equipo sobre el que se efectúa algún movmiento.
- Proveedor: La empresa que realiza el mantenimiento del equipo.

Para elaborar los diagramas de estructura de datos, se identifican 5 almacenes de datos:

- Usuarios: Los usuarios que trabajan con el sistema.
- Equipos: Los equipos pertenecientes al Centro de Cómputo.
- Proveedores: Las empresas que efectúan el mantenimiento de los equipos.

- Mantenimiento Preventivo: Registro de operaciones de mantenimiento preventivo a los equipos.
- Mantenimiento Correctivo: Registro de operaciones de mantenimiento correctivo a los equipos.

Mantenimiento Preventivo

El diagrama de estructura que respresenta el proceso de registro de un mantenimiento preventivo se muestra en la figura 2.10.

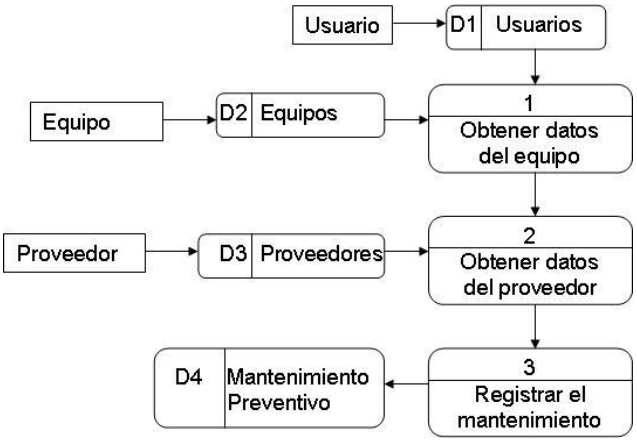


Fig. 2.10 Diagrama de estructura de mantenimiento preventivo de hardware

Los pasos para registrar un mantenimiento preventivo son:

1. El usuario ingresa su clave de acceso al sistema, se valida en el catálogo de usuarios (D1) y obtiene los datos del equipo al que se le va a dar mantenimiento (D8).
2. Se validan los datos del proveedor (D9).
3. Se registra la información en el almacén de datos de mantenimiento preventivo (D14).

Mantenimiento Correctivo

La figura 2.11 muestra el diagrama de estructura para el registro de un mantenimiento correctivo.

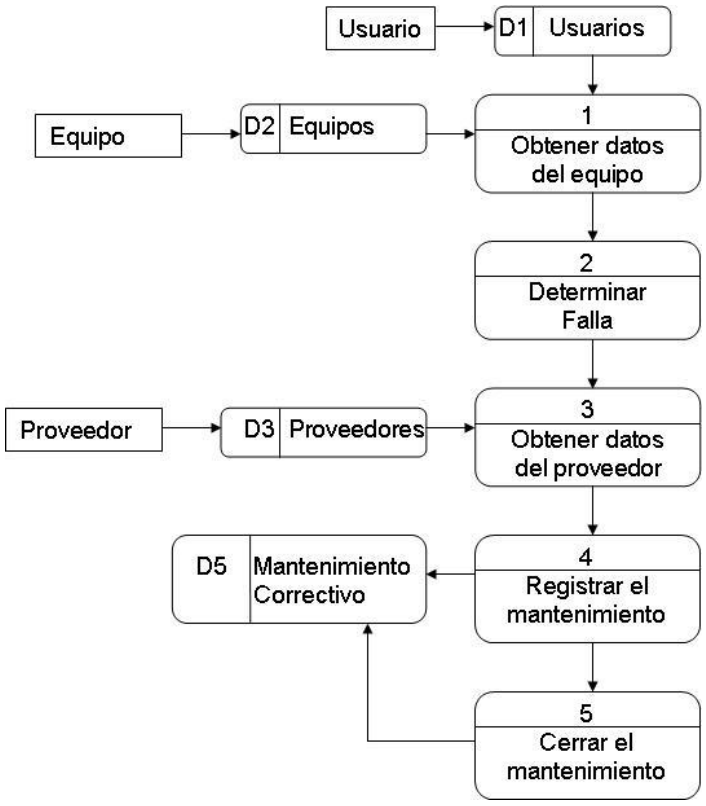


Fig. 2.11 Diagrama de estructura de mantenimiento correctivo de hardware

Los pasos para registrar un mantenimiento correctivo son:

1. El usuario ingresa su clave de acceso al sistema, se valida en el catálogo de usuarios (D1) y obtiene los datos del equipo al que se le va a dar mantenimiento (D8).
2. Se determina la falla del equipo.
3. Se validan los datos del proveedor (D9).
4. Se registra la información en el almacén de datos de mantenimiento correctivo (D15).

5. Al cerrar el mantenimiento, se actualiza la información en el almacén de datos de mantenimiento correctivo (D15).

2.4 Diagramas de Casos de Uso

La técnica de casos de uso se utiliza para capturar información de cómo un sistema o negocio trabaja, o de cómo se desea que trabaje. No pertenece estrictamente al enfoque orientado a objeto, es una técnica para captura de requisitos. Cada Caso de Uso puede estar definido por: texto que lo describe, secuencia de pasos (flujo de eventos) ejecutados dentro del caso de uso, precondiciones y postcondiciones para que el caso de uso comience o termine, o una mezcla de las anteriores.

Los casos de uso, al igual que el modelo de la organización (capítulo 2.3) y el diagrama de actividades (capítulo 2.2), están divididos de acuerdo a los procesos de préstamos de software y movimientos de hardware.

2.4.1 Software

La figura 2.12 muestra el caso de uso que representa el diagrama conceptual del módulo de préstamos de software (anexo 1).

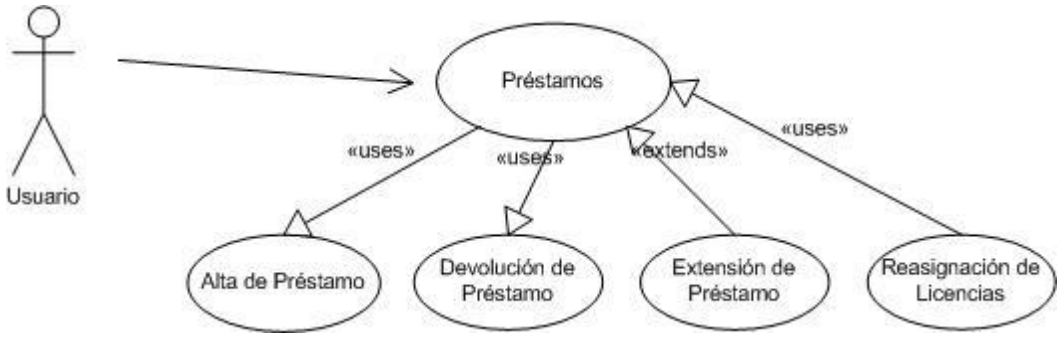


Fig. 2.12 Caso de uso de préstamos de software

La figura 2.13 muestra el caso de uso para registrar un préstamo:

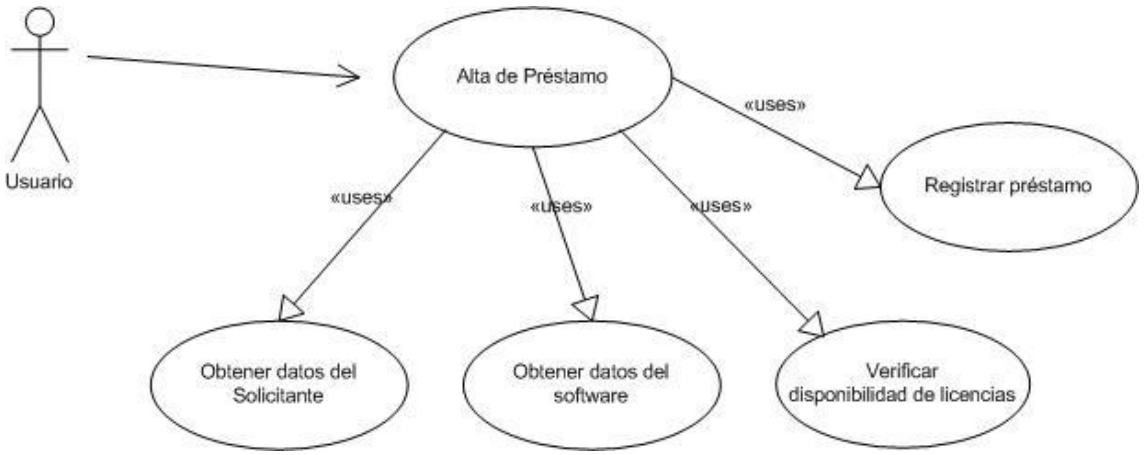


Fig. 2.13 Caso de uso de alta de préstamo de software

El caso de uso para una devolución de préstamo es la figura 2.14:

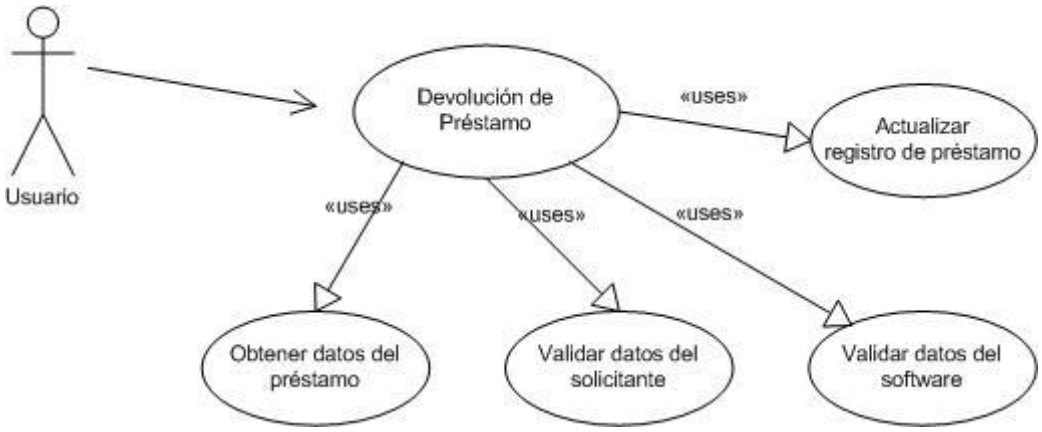


Fig. 2.14 Caso de uso de devolución de software

La figura 2.15 es el caso de uso para una extensión de préstamo:

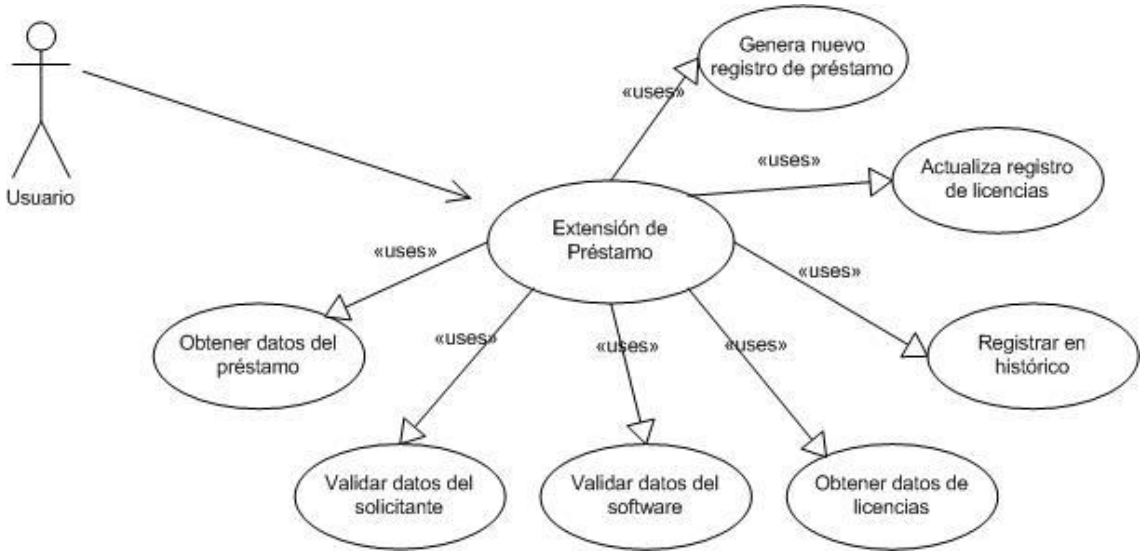


Fig. 2.15 Caso de uso de extensión de préstamo de software

El caso de uso para una reasignación de licencia se muestra en la figura 2.16:

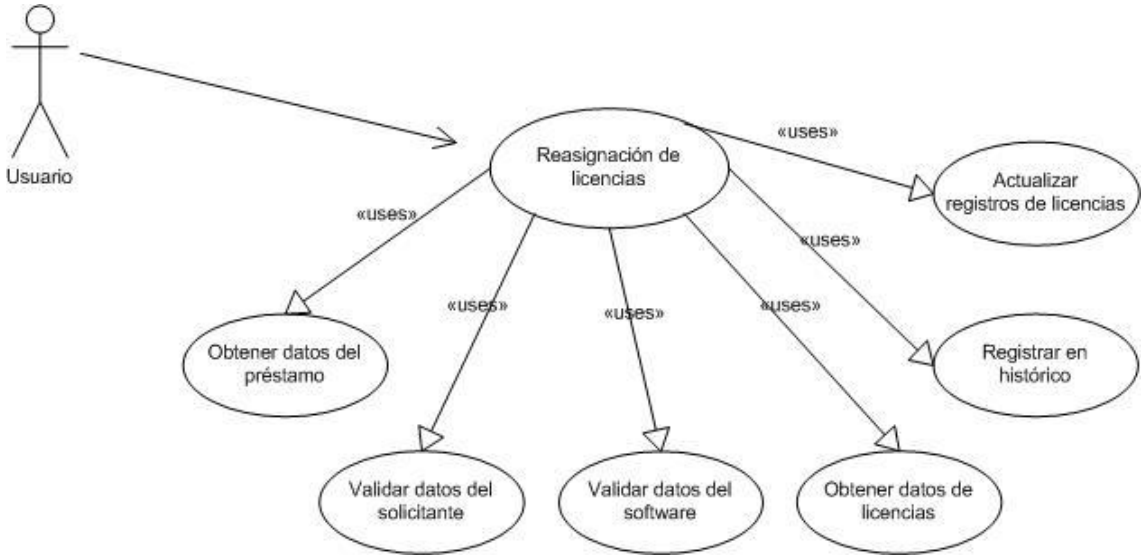


Fig. 2.16 Caso de uso de reasignación de licencia de software

2.4.2 Hardware

El caso de uso del módulo de movimientos de hardware (anexo 1) se muestra en la figura 2.17:

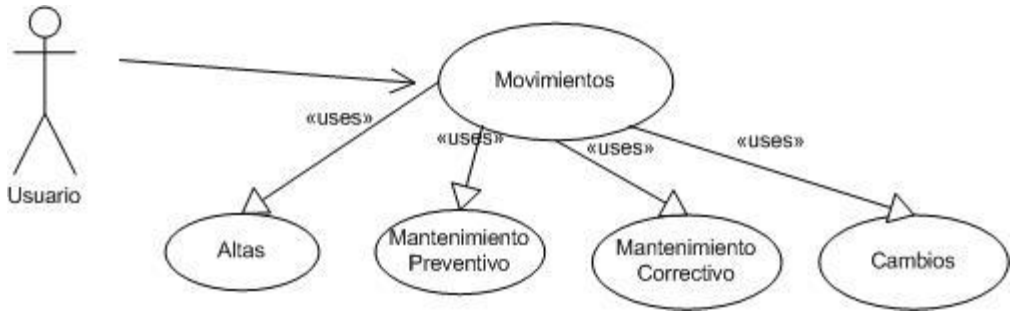


Fig. 2.17 Caso de uso de movimientos de hardware

La figura 2.18 muestra el caso de uso correspondiente a un mantenimiento preventivo:

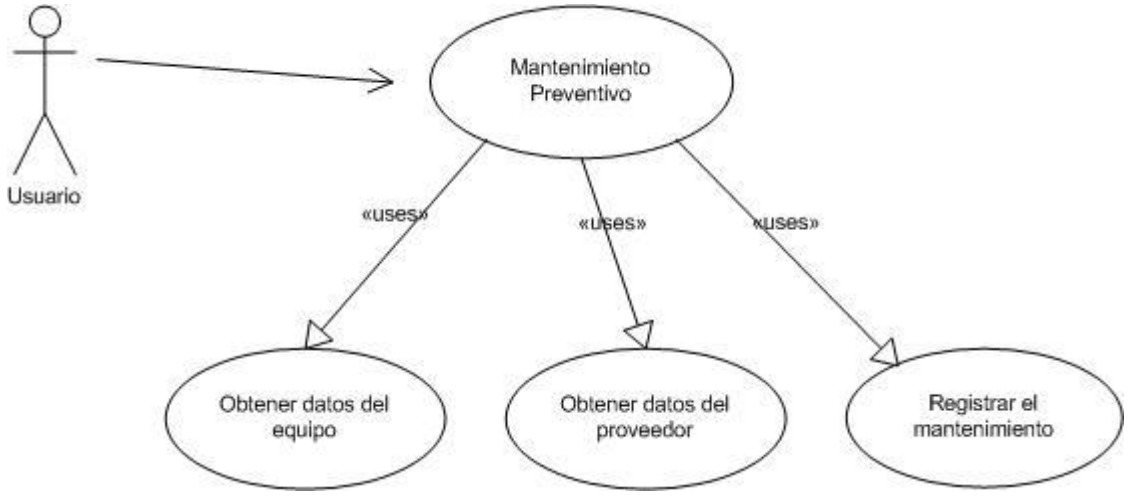


Fig. 2.18 Caso de uso de mantenimiento preventivo de hardware

El caso de uso para un mantenimiento correctivo se puede apreciar en la figura 2.19:

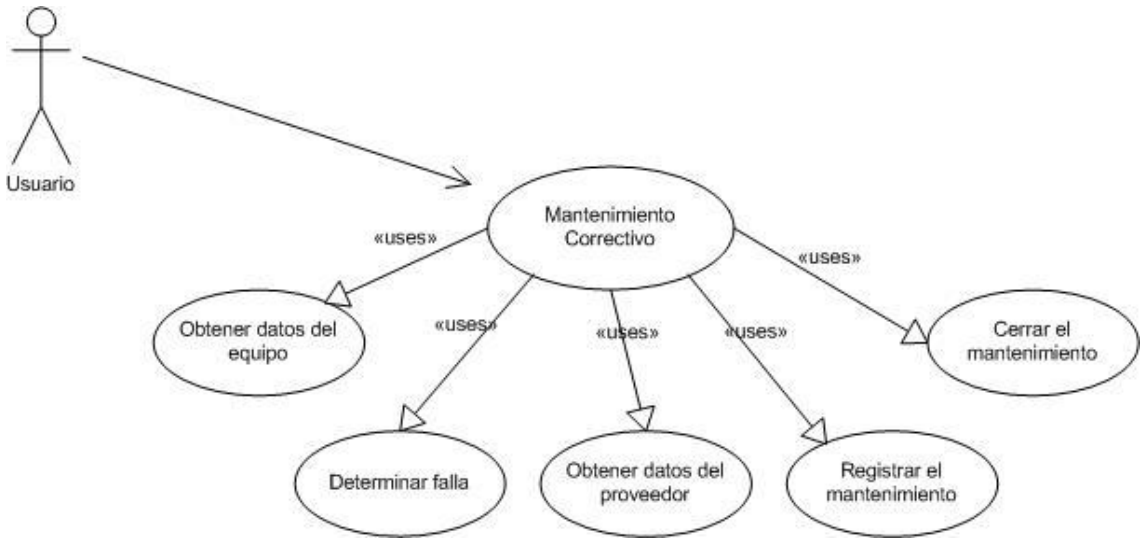


Fig. 2.19 Caso de uso de mantenimiento correctivo de hardware

2.5 Diagrama Entidad – Relación

El diagrama entidad – relación se muestra en el anexo 3.

2.6 Diccionario de Datos

El diccionario de datos es un conjunto de metadatos que contiene las características lógicas de los datos que se van a utilizar en el sistema.

Un metadato es una descripción estructurada disponible para ayudar a localizar objetos, conteniendo información para ayudar a identificar, descubrir, valorar y administrar los objetos descritos.

A continuación se muestra el diccionario de datos elaborado para este sistema.

Nombre	Alias	Tipo	Tamaño
Identificador de perfil de usuario	c_perfil	char	2
Descripción del perfil de usuario	d_perfil	nvarchar	20
Número de Empleado del personal que opera el sistema	c_usuario	smallint	
Nombre(s)	d_nombre	nvarchar	30
Apellido Paterno	d_apaterno	nvarchar	40
Apellido Materno	d_amaterno	nvarchar	40
Fecha de alta	f_alta	datetime	
Fecha de baja	f_baja	datetime	
Clave de baja	c_baja	char	3
Contraseña del usuario que opera el sistema	c_password	varbinary	100
Número de empleado del personal usuario del sistema	c_solicitante	smallint	
Identificador de usuario autorizado	b_autorizado	bit	
Observaciones o comentarios	d_comentarios	nvarchar	200
Usuario que da de baja el registro	c_usuariobaja	smallint	5
Identificador de medio magnético	c_medio	smallint	5
Identificador del software	c_software	smallint	5
Descripción del software	d_medio	nvarchar	50
Número de parte del medio	d_parte	nvarchar	10
Tipo de software	c_tipo_prod	char	2
Tipo de medio magnético	d_tipo_medio	nvarchar	5
Identificador de producto con número de llave requerido	b_llave	bit	
Plataforma (PC, Mac, Linux, etc)	d_plataforma	nvarchar	20
Mes de elaboración del software	d_mes	smallint	
Año de elaboración del software	d_aa	smallint	

Nombre	Alias	Tipo	Tamaño
Idioma del producto	d_idioma	nvarchar	10
Número de copias del software	n_copias	smallint	
Número de licencias adquiridas	n_licencias	smallint	
Descripción del motivo de baja	d_baja	nvarchar	100
Clasificación del motivo de baja	d_tipo_mot	nvarchar	20
Identificador de registro activo o no activo	b_activo	bit	
Clave del tipo de software (Aplicación, Sistema, etc.)	c_tipo_prod	char	2
Descripción del tipo de software	d_tipo_prod	nvarchar	20
Folio asignado al préstamo	c_folio_prestamo	int	
Número de empleado que registra la devolución del software	c_usuariodevolucion	smallint	
Fecha de devolución del software	f_devolución	datetime	
Número de licencia asignada	c_licencia	smallint	
Número de empleado que tiene la licencia asignada	c_usuario_final	smallint	
Identificador de registro histórico	c_con	int	
Número de empleado que genera el registro histórico	c_usuariohis	smallint	
Fecha de alta del registro histórico	f_altahis	datetime	
Número de inventario interno de equipos	c_inv	smallint	
Marca del equipo	d_marca	nvarchar	20
Modelo del equipo	d_modelo	nvarchar	40
Número de parte del equipo	d_parte	nvarchar	40
Año de Adquisición del equipo	d_aa	smallint	
Número de procesadores del equipo	n_procesadores	smallint	

Nombre	Alias	Tipo	Tamaño
Descripción de los procesadores del equipo	d_procesadores	nvarchar	30
Tamaño de memoria RAM del equipo	d_ram	nvarchar	10
Identificador del arreglo de discos	c_arreglo	smallint	
Número de discos duros	n_discos	smallint	
Tamaño de los discos duros	d_discos	nvarchar	20
Tipo de arreglo formado	d_arreglo	nvarchar	20
Número de tarjetas de red	n_tarjetas_red	smallint	
Descripción de las tarjetas de red	d_tarjetas_red	nvarchar	30
Sistema Operativo instalado	d_sis_ope	nvarchar	40
Identificador interno del proveedor de servicio	c_proveedor	smallint	
Nombre del proveedor de servicio	d_proveedor	nvarchar	50
Identificador de registro de mantenimiento preventivo o correctivo	c_reg	int	
Folio de orden de mantenimiento preventivo o correctivo	c_folio	nvarchar	10
Descripción de la falla del equipo	d_falla	nvarchar	100
No. de empleado que cierra el mantenimiento correctivo	c_usuariocierre	smallint	
Fecha de cierre del mantenimiento correctivo	f_cierre	datetime	
Observaciones al cierre del mantenimiento correctivo	d_obs_cierre	nvarchar	100
Ubicación del equipo	d_ubicacion	nvarchar	50

2.7 Script para creación de base de datos

Debido a políticas internas de la Dirección General de Sistemas se utilizó Microsoft SQL Server para que maneje la base de datos de esta aplicación.

A continuación se muestra el script de la base de datos, el cual crea la base de datos, las tablas que utiliza el sistema, sus llaves primarias y las referencias entre tablas:

```
/*********************************************************************/
/***  Script para la creación de la base de datos correspondien-  ***/
/***  te al Sistema de Administración de Centros de Cómputo.      ***/
/***  Contiene las instrucciones necesarias para la creación de   ***/
/***  la base de datos, las tablas que utiliza el sistema, sus    ***/
/***  llaves primarias y las referencias entre tablas.            ***/
/***                                                              ***/
/***                       Elaborado por:                         ***/
/***                                       Mario M. Tachika Ohara  ***/
/*********************************************************************/


/*********************************************************************/
/**  Creación de la base de datos                                    **/
/*********************************************************************/

USE master
GO

CREATE DATABASE AdminCC ON
(
   NAME = AdminCC_Data,
   FILENAME = 'C:\Bdatos\AdminCC\AdminCC_Data.mdf',
   SIZE = 10,
   MAXSIZE = 4096,
   FILEGROWTH = 1
)
```

```
LOG ON
(
   NAME = AdminCC_Log,
   FILENAME = 'C:\Bdatos\AdminCC\AdminCC_Data.ldf',
   SIZE = 10,
   MAXSIZE = 4096,
   FILEGROWTH = 1
)
GO


/*********************************************************************/
/**  Creación de tablas                                             **/
/*********************************************************************/

USE AdminCC
GO

CREATE TABLE [dbo].[cat_perfiles]
(
   [c_perfil] [char](2)      NOT NULL,
   [d_perfil] [nvarchar](20) NOT NULL,
   [f_alta]   [datetime]     NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[cat_usuarios]
(
   [c_usuario]  [smallint]     NOT NULL,
   [d_nombre]   [nvarchar](30) NOT NULL,
   [d_apaterno] [nvarchar](40) NOT NULL,
   [d_amaterno] [nvarchar](40) NOT NULL,
   [c_perfil]   [char](2)      NOT NULL,
   [f_alta]     [datetime]     NOT NULL,
   [f_baja]     [datetime]     NULL,
   [c_baja]     [char](3)      NULL
) ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[cat_passwords]
(
   [c_usuario]  [smallint]       NOT NULL,
   [c_password] [varbinary](100) NOT NULL,
   [f_alta]     [datetime]       NOT NULL,
   [f_baja]     [datetime]       NULL
)  ON [PRIMARY]
GO

CREATE TABLE [dbo].[cat_solicitantes]
(
   [c_solicitante] [smallint]      NOT NULL,
   [d_nombre]      [nvarchar](30)  NOT NULL,
   [d_apaterno]    [nvarchar](40)  NOT NULL,
   [d_amaterno]    [nvarchar](40)  NOT NULL,
   [b_autorizado]  [bit]           NOT NULL,
   [c_usuario]     [smallint]      NOT NULL,
   [f_alta]        [datetime]      NOT NULL,
   [d_comentarios] [nvarchar](200) NULL,
   [c_usuariobaja] [smallint]      NULL,
   [f_baja]        [datetime]      NULL,
   [c_baja]        [char](3)       NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[cat_software]
(
   [c_medio]       [smallint]      NOT NULL,
   [c_software]    [smallint]      NOT NULL,
   [d_medio]       [nvarchar](200) NOT NULL,
   [d_parte]       [nvarchar](10)  NOT NULL,
   [c_tipo_prod]   [char](2)       NOT NULL,
   [d_tipo_medio]  [nvarchar](5)   NOT NULL,
   [b_llave]       [bit]           NOT NULL,
   [d_plataforma]  [nvarchar](20)  NOT NULL,
   [d_mes]         [smallint]      NOT NULL,
   [d_aa]          [smallint]      NOT NULL,
```

```
   [d_idioma]      [nvarchar](10)  NOT NULL,
   [n_copias]      [smallint]      NOT NULL,
   [n_licencias]   [smallint]      NOT NULL,
   [c_usuario]     [smallint]      NOT NULL,
   [f_alta]        [datetime]      NOT NULL,
   [c_usuariobaja] [smallint]      NULL,
   [f_baja]        [datetime]      NULL,
   [c_baja]        [char](3)       NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[cat_mot_bajas]
(
   [c_baja]     [char](3)       NOT NULL,
   [d_baja]     [nvarchar](100) NOT NULL,
   [d_tipo_mot] [nvarchar](20)  NOT NULL,
   [b_activo]   [bit]           NOT NULL,
   [c_usuario]  [smallint]      NOT NULL,
   [f_alta]     [datetime]      NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[cat_tipos]
(
   [c_tipo_prod] [char](2)      NOT NULL,
   [d_tipo_prod] [nvarchar](20) NOT NULL,
   [b_activo]    [bit]          NOT NULL,
   [c_usuario]   [smallint]     NOT NULL,
   [f_alta]      [datetime]     NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tab_prestamos]
(
   [c_folio_prestamo]    [int]      NOT NULL,
   [c_medio]             [smallint] NOT NULL,
   [c_software]          [smallint] NOT NULL,
   [c_solicitante]       [smallint] NOT NULL,
```

```
   [c_usuario]           [smallint] NOT NULL,
   [f_alta]              [datetime] NOT NULL,
   [c_usuariodevolucion] [smallint] NULL,
   [f_devolucion]        [datetime] NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tab_licencias]
(
   [c_medio]          [smallint]     NOT NULL,
   [c_software]       [smallint]     NOT NULL,
   [c_licencia]       [smallint]     NOT NULL,
   [c_usuario_final]  [smallint]     NOT NULL,
   [d_nombre]         [nvarchar](30) NOT NULL,
   [d_apaterno]       [nvarchar](40) NOT NULL,
   [d_amaterno]       [nvarchar](40) NOT NULL,
   [c_folio_prestamo] [int]          NOT NULL,
   [c_usuario]        [smallint]     NOT NULL,
   [f_alta]           [datetime]     NOT NULL,
   [c_usuariobaja]    [smallint]     NULL,
   [f_baja]           [datetime]     NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[his_prestamos]
(
   [c_con]            [int]      NOT NULL,
   [c_folio_prestamo] [int]      NOT NULL,
   [c_medio]          [smallint] NOT NULL,
   [c_software]       [smallint] NOT NULL,
   [c_solicitante]    [smallint] NOT NULL,
   [c_usuario]        [smallint] NOT NULL,
   [f_alta]           [datetime] NOT NULL,
   [c_usuariohis]     [smallint] NOT NULL,
   [f_altahis]        [datetime] NOT NULL
) ON [PRIMARY]
GO
```

```
CREATE TABLE [dbo].[his_licencias]
(
   [c_con]            [int]          NOT NULL,
   [c_medio]          [smallint]     NOT NULL,
   [c_software]       [smallint]     NOT NULL,
   [c_licencia]       [smallint]     NOT NULL,
   [c_usuario_final]  [smallint]     NOT NULL,
   [d_nombre]         [nvarchar](30) NOT NULL,
   [d_apaterno]       [nvarchar](40) NOT NULL,
   [d_amaterno]       [nvarchar](40) NOT NULL,
   [c_folio_prestamo] [int]          NOT NULL,
   [c_usuario]        [smallint]     NOT NULL,
   [f_alta]           [datetime]     NOT NULL,
   [c_usuariohis]     [smallint]     NOT NULL,
   [f_altahis]        [datetime]     NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[cat_equipos]
(
   [c_inv]          [smallint]     NOT NULL,
   [d_marca]        [nvarchar](20) NOT NULL,
   [d_modelo]       [nvarchar](40) NOT NULL,
   [d_parte]        [nvarchar](40) NOT NULL,
   [d_aa]           [smallint]     NOT NULL,
   [c_usuario]      [smallint]     NOT NULL,
   [f_alta]         [datetime]     NOT NULL,
   [c_usuariobaja] [smallint],
   [f_baja]         [datetime],
   [c_baja]         [char](3)
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tab_det_equipos]
(
   [c_inv]           [smallint]     NOT NULL,
   [n_procesadores] [smallint]     NOT NULL,
   [d_procesadores] [nvarchar](30) NOT NULL,
```

```
   [d_ram]          [nvarchar](10) NOT NULL,
   [n_tarjetas_red] [smallint]     NOT NULL,
   [d_tarjetas_red] [nvarchar](30) NOT NULL,
   [d_sis_ope]      [nvarchar](40) NOT NULL,
   [c_usuario]      [smallint]     NOT NULL,
   [f_alta]         [datetime]     NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tab_det_discos_equipos]
(
   [c_inv]          [smallint]     NOT NULL,
   [c_arreglo]      [smallint]     NOT NULL,
   [n_discos]       [smallint]     NOT NULL,
   [d_discos]       [nvarchar](20) NOT NULL,
   [d_arreglo]      [nvarchar](20) NOT NULL,
   [c_usuario]      [smallint]     NOT NULL,
   [f_alta]         [datetime]     NOT NULL,
   [c_usuariobaja]  [smallint]     NULL,
   [f_baja]         [datetime]     NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[his_det_equipos]
(
   [c_con]          [int]          NOT NULL,
   [c_inv]          [smallint]     NOT NULL,
   [n_procesadores] [smallint]     NOT NULL,
   [d_procesadores] [nvarchar](30) NOT NULL,
   [d_ram]          [nvarchar](10) NOT NULL,
   [n_tarjetas_red] [smallint]     NOT NULL,
   [d_tarjetas_red] [nvarchar](30) NOT NULL,
   [d_sis_ope]      [nvarchar](40) NOT NULL,
   [c_usuario]      [smallint]     NOT NULL,
   [f_alta]         [datetime]     NOT NULL,
   [c_usuariohis]   [smallint]     NOT NULL,
   [f_altahis]      [datetime]     NOT NULL
) ON [PRIMARY]
```

```
GO

CREATE TABLE [dbo].[his_det_discos_equipos]
(
   [c_con]        [int]          NOT NULL,
   [c_inv]        [smallint]     NOT NULL,
   [c_arreglo]    [smallint]     NOT NULL,
   [n_discos]     [smallint]     NOT NULL,
   [d_discos]     [nvarchar](20) NOT NULL,
   [d_arreglo]    [nvarchar](20) NOT NULL,
   [c_usuario]    [smallint]     NOT NULL,
   [f_cap]        [datetime]     NOT NULL,
   [c_usuariohis] [smallint]     NOT NULL,
   [f_altahis]    [datetime]     NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[cat_proveedores]
(
   [c_proveedor]   [smallint]     NOT NULL,
   [d_proveedor]   [nvarchar](50) NOT NULL,
   [c_usuario]     [smallint]     NOT NULL,
   [f_alta]        [datetime]     NOT NULL,
   [c_usuariobaja] [smallint]     NULL,
   [f_baja]        [datetime]     NULL,
   [c_baja]        [char](3)      NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tab_mant_prev]
(
   [c_reg]       [int]          NOT NULL,
   [c_folio]     [nvarchar](10) NOT NULL,
   [c_inv]       [smallint]     NOT NULL,
   [c_proveedor] [smallint]     NOT NULL,
   [c_usuario]   [smallint]     NOT NULL,
   [f_alta]      [datetime]     NOT NULL
) ON [PRIMARY]
```

```
GO

CREATE TABLE [dbo].[tab_mant_corr]
(
   [c_reg]           [int]           NOT NULL,
   [c_folio]         [nvarchar](10)  NOT NULL,
   [c_inv]           [smallint]      NOT NULL,
   [c_proveedor]     [smallint]      NOT NULL,
   [d_falla]         [nvarchar](100) NOT NULL,
   [c_usuario]       [smallint]      NOT NULL,
   [f_alta]          [datetime]      NOT NULL,
   [c_usuariocierre] [smallint]      NULL,
   [f_cierre]        [datetime]      NULL,
   [d_obs_cierre]    [nvarchar](100) NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[tab_ubicaciones]
(
   [c_inv]       [smallint]     NOT NULL,
   [d_ubicacion] [nvarchar](50) NOT NULL,
   [c_usuario]   [smallint]     NOT NULL,
   [f_alta]      [datetime]     NOT NULL
) ON [PRIMARY]
GO

CREATE TABLE [dbo].[his_ubicaciones]
(
   [c_con]        [int]          NOT NULL,
   [c_inv]        [smallint]     NOT NULL,
   [d_ubicacion]  [nvarchar](50) NOT NULL,
   [c_usuario]    [smallint]     NOT NULL,
   [f_alta]       [datetime]     NOT NULL,
   [c_usuariohis] [smallint]     NOT NULL,
   [f_altahis]    [datetime]     NOT NULL
) ON [PRIMARY]
GO
```

```
/*****************************************************************/
/**  Creación de llaves primarias                                **/
/*****************************************************************/

ALTER TABLE [dbo].[cat_perfiles] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_perfil]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[cat_usuarios] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_usuario]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[cat_passwords] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_usuario],
      [c_password]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[cat_solicitantes] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_solicitante]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[cat_software] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
```

```
      [c_medio],
      [c_software]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[cat_mot_bajas] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_baja]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[cat_tipos] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_tipo_prod]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[tab_prestamos] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_folio_prestamo]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[tab_licencias] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_medio],
      [c_software],
      [c_licencia]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[his_prestamos] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
```

```
      [c_con]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[his_licencias] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_con]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[cat_equipos] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_inv]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[tab_det_equipos] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_inv]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[tab_det_discos_equipos] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_inv],
      [c_arreglo]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[his_det_equipos] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_con]
   ) ON [PRIMARY]
```

```
GO

ALTER TABLE [dbo].[his_det_discos_equipos] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_con]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[cat_proveedores] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_proveedor]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[tab_mant_prev] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_reg]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[tab_mant_corr] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_reg]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[tab_ubicaciones] WITH CHECK
   ADD PRIMARY KEY CLUSTERED
   (
      [c_inv]
   ) ON [PRIMARY]
GO

ALTER TABLE [dbo].[his_ubicaciones] WITH CHECK
```

```
   ADD PRIMARY KEY CLUSTERED
   (
      [c_con]
   ) ON [PRIMARY]
GO


/****************************************************************/
/**  Creación de llaves foráneas                                **/
/****************************************************************/

ALTER TABLE [dbo].[cat_usuarios] ADD
   CONSTRAINT [fk_usuarios_perfiles] FOREIGN KEY
   (
      [c_perfil]
   )
   REFERENCES [dbo].[cat_perfiles]
   (
      [c_perfil]
   ),
   CONSTRAINT [fk_usuarios_mot_bajas] FOREIGN KEY
   (
      [c_baja]
   )
   REFERENCES [dbo].[cat_mot_bajas]
   (
      [c_baja]
   )
GO

ALTER TABLE [dbo].[cat_passwords] ADD
   CONSTRAINT [fk_passwords_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
     [c_usuario]
```

```
   )
GO

ALTER TABLE [dbo].[cat_solicitantes] ADD
   CONSTRAINT [fk_solicitantes_mot_bajas] FOREIGN KEY
   (
      [c_baja]
   )
   REFERENCES [dbo].[cat_mot_bajas]
   (
      [c_baja]
   ),
   CONSTRAINT [fk_solicitantes_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   ),
   CONSTRAINT [fk_solicitantes_usuarios_baja] FOREIGN KEY
   (
      [c_usuariobaja]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
   [c_usuario]
   )
GO

ALTER TABLE [dbo].[cat_software] ADD
   CONSTRAINT [fk_software_tipos] FOREIGN KEY
   (
      [c_tipo_prod]
   )
   REFERENCES [dbo].[cat_tipos]
   (
      [c_tipo_prod]
```

```
   ),
   CONSTRAINT [fk_software_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   ),
   CONSTRAINT [fk_software_usuarios_baja] FOREIGN KEY
   (
      [c_usuariobaja]
   )
      REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[cat_mot_bajas] ADD
   CONSTRAINT [fk_mot_bajas_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[cat_tipos] ADD
   CONSTRAINT [fk_tipos_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
```

```
   )
GO

ALTER TABLE [dbo].[tab_prestamos] ADD
   CONSTRAINT [fk_prestamos_software] FOREIGN KEY
   (
      [c_medio],
      [c_software]
   )
   REFERENCES [dbo].[cat_software]
   (
      [c_medio],
      [c_software]
   ),
   CONSTRAINT [fk_prestamos_usuariosdevolucion] FOREIGN KEY
   (
      [c_usuariodevolucion]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   ),
   CONSTRAINT [fk_prestamos_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   ),
   CONSTRAINT [FK_prestamos_solicitantes] FOREIGN KEY
   (
      [c_solicitante]
   )
   REFERENCES [dbo].[cat_solicitantes]
   (
      [c_solicitante]
   )
```

```
GO

ALTER TABLE [dbo].[tab_licencias] ADD
   CONSTRAINT [fk_licencias_software] FOREIGN KEY
   (
      [c_medio],
      [c_software]
   )
   REFERENCES [dbo].[cat_software]
   (
      [c_medio],
      [c_software]
   ),
   CONSTRAINT [fk_licencias_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   ),
   CONSTRAINT [fk_licencias_usuarios_baja] FOREIGN KEY
   (
      [c_usuariobaja]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[his_prestamos] ADD
   CONSTRAINT [fk_hisprestamos_prestamos] FOREIGN KEY
   (
      [c_folio_prestamo]
   )
   REFERENCES [dbo].[tab_prestamos]
   (
```

```
      [c_folio_prestamo]
   ),
   CONSTRAINT [fk_hisprestamos_usuarios] FOREIGN KEY
   (
      [c_usuariohis]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[his_licencias] ADD
   CONSTRAINT [fk_hislicencias_licencias] FOREIGN KEY
   (
      [c_medio],
      [c_software],
      [c_licencia]
   )
   REFERENCES [dbo].[tab_licencias]
   (
      [c_medio],
      [c_software],
      [c_licencia]
   ),
   CONSTRAINT [fk_hislicencias_usuarios] FOREIGN KEY
   (
      [c_usuariohis]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[cat_equipos] ADD
   CONSTRAINT [fk_equipos_usuarios] FOREIGN KEY
   (
```

```
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   ),
   CONSTRAINT [fk_equipos_usuarios_baja] FOREIGN KEY
   (
      [c_baja]
   )
   REFERENCES [dbo].[cat_mot_bajas]
   (
      [c_baja]
   )
GO

ALTER TABLE [dbo].[tab_det_equipos] ADD
   CONSTRAINT [fk_detequipos_equipos] FOREIGN KEY
   (
      [c_inv]
   )
   REFERENCES [dbo].[cat_equipos]
   (
      [c_inv]
   ),
   CONSTRAINT [fk_detequipos_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[tab_det_discos_equipos] ADD
   CONSTRAINT [fk_detdiscosequipos_equipos] FOREIGN KEY
   (
```

```
      [c_inv]
   )
   REFERENCES [dbo].[cat_equipos]
   (
      [c_inv]
   ),
   CONSTRAINT [fk_detdiscosequipos_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[his_det_equipos] ADD
   CONSTRAINT [fk_hisdetequipos_equipos] FOREIGN KEY
   (
      [c_inv]
   )
   REFERENCES [dbo].[tab_det_equipos]
   (
      [c_inv]
   ),
   CONSTRAINT [fk_hisdetequipos_usuarios] FOREIGN KEY
   (
      [c_usuariohis]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[his_det_discos_equipos] ADD
   CONSTRAINT [fk_hisdetdiscosequipos_equipos] FOREIGN KEY
   (
```

```
      [c_inv],
      [c_arreglo]
   )
   REFERENCES [dbo].[tab_det_discos_equipos]
   (
      [c_inv],
      [c_arreglo]
   ),
   CONSTRAINT [fk_hisdetdiscosequipos_usuarios] FOREIGN KEY
   (
      [c_usuariohis]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[cat_proveedores] ADD
   CONSTRAINT [fk_proveedores_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   ),
   CONSTRAINT [fk_proveedores_usuarios_baja] FOREIGN KEY
   (
      [c_usuariobaja]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[tab_mant_prev] ADD
```

```
   CONSTRAINT [fk_mantprev_equipos] FOREIGN KEY
   (
      [c_inv]
   )
   REFERENCES [dbo].[cat_equipos]
   (
      [c_inv]
   ),
   CONSTRAINT [fk_mantprev_proveedores] FOREIGN KEY
   (
      [c_proveedor]
   )
   REFERENCES [dbo].[cat_proveedores]
   (
      [c_proveedor]
   ),
   CONSTRAINT [fk_mantprev_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[tab_mant_corr] ADD
   CONSTRAINT [fk_mantcorr_equipos] FOREIGN KEY
   (
      [c_inv]
   )
   REFERENCES [dbo].[cat_equipos]
   (
      [c_inv]
   ),
   CONSTRAINT [fk_mantcorr_proveedores] FOREIGN KEY
   (
      [c_proveedor]
```

```
   )
   REFERENCES [dbo].[cat_proveedores]
   (
      [c_proveedor]
   ),
   CONSTRAINT [fk_mantcorr_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   ),
   CONSTRAINT [fk_mantcorr_usuarios_cierre] FOREIGN KEY
   (
      [c_usuariocierre]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[tab_ubicaciones] ADD
   CONSTRAINT [fk_ubicaciones_equipos] FOREIGN KEY
   (
      [c_inv]
   )
   REFERENCES [dbo].[cat_equipos]
   (
      [c_inv]
   ),
   CONSTRAINT [fk_ubicaciones_usuarios] FOREIGN KEY
   (
      [c_usuario]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
```

```
      [c_usuario]
   )
GO

ALTER TABLE [dbo].[his_ubicaciones] ADD
   CONSTRAINT [fk_hisubicaciones_equipos] FOREIGN KEY
   (
      [c_inv]
   )
   REFERENCES [dbo].[cat_equipos]
   (
      [c_inv]
   ),
   CONSTRAINT [fk_hisubicaciones_usuarios] FOREIGN KEY
   (
      [c_usuariohis]
   )
   REFERENCES [dbo].[cat_usuarios]
   (
      [c_usuario]
   )
GO
```

3. Desarrollo del Sistema

3.1 Herramienta de desarrollo

El desarrollo de este sistema se decidió que fuera como aplicación web, para que de esta forma solo fuera necesario instalarlo en un servidor y que los usuarios del sistema lo utilicen por medio del navegador web, por lo que la herramienta que se utilizó para su desarrollo fué Microsoft Visual Studio 2005, y se programó en Visual Basic .Net. debido a que, al ser uno de los lenguajes más sencillos de utilizar, permiten el desarrollo de aplicaciones en corto tiempo, además de que también por políticas internas de la Dirección General de Sistemas es el lenguaje que se utiliza para los desarrollos de la institución.

3.2 Conexión a la base de datos

Para efectuar el acceso a la base de datos se define una conexión abierta a base de datos (ODBC, Open DataBase Conection), con la que se establece el control para todas las operaciones de lectura y escritura que efectúan los distintos programas del sistema.

La figura 3.1 muestra el ODBC definido para la elaboración de este trabajo, el cual se utilizará como ejemplo para explicar el manejo de las instrucciones utilizadas en el desarrollo del sistema.

Una vez establecido el ODBC, se utiliza la clase de Visual Basic *SQLConnection* para establecer el acceso a la base de datos dentro de los programas, con la siguiente sintaxis:

```
Dim cn As New SQLConnection
cn.ConnectionString =
```

```
   ConfigurationManager.ConnectionStrings("AdminCCConn").ConnectionString
cn.Open()
```

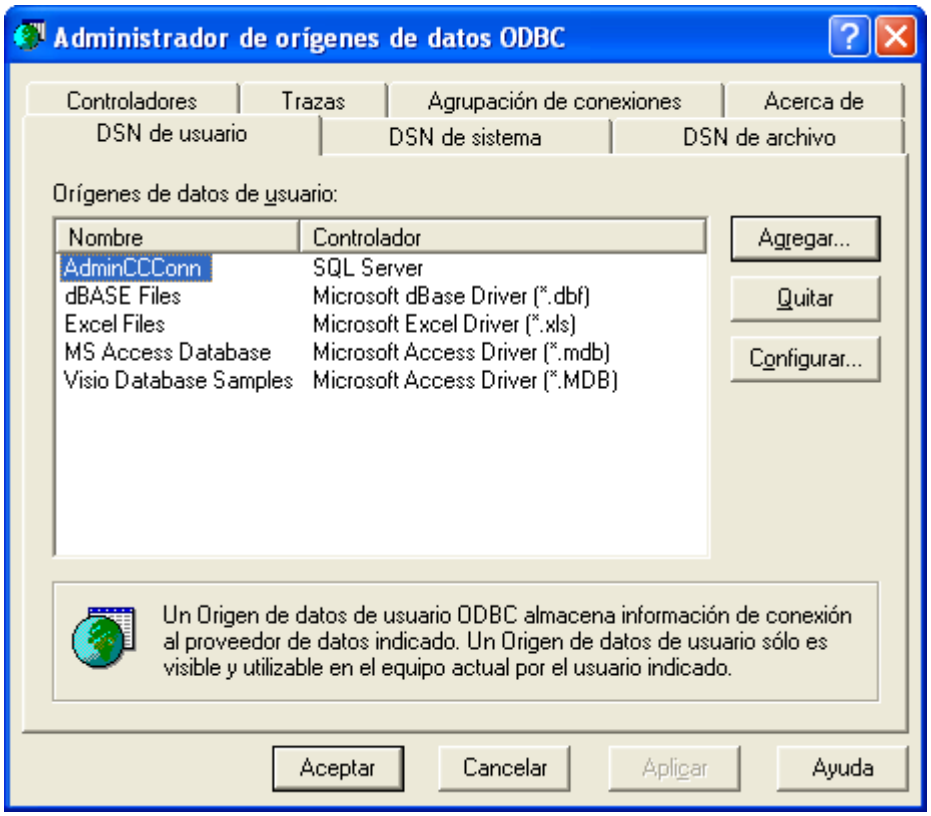


Fig. 3.1 Ventana principal para administración de ODBCs

Una vez efectuada la transacción, se debe cerrar la conexión para que no mantenga un recurso de base de datos utilizado, con la siguiente instrucción:

```
cn.Close()
```

Para efectuar consultas que devuelvan más de un registro, se utilizaron las clases SQLDataAdapter y DataSet como se muestran a continuación:

```
Dim daMotivos As New SqlDataAdapter("<instrucción select>",

ConfigurationManager.ConnectionStrings("AdminCCConn").ConnectionString)
Dim dsMotivos As New DataSet
daMotivos.Fill(dsMotivos)
```

Si la consulta solo devolvía un registro se utilizaron las clases *SQLCommand* y *SQLDataReader* de la siguiente forma:

```
Dim cmd As New SQLCommand
Dim dr As DataReader
cmd.Connection = cn
cmd.CommandText = "<instrucción select>"
dr = cmdSolicitante.ExecuteReader
```

Al igual que con la clase SQLConnection, la clase SQLDataReader debe cerrarse al terminar la transacción:

```
dr.Close()
```

Para efectuar instrucciones *insert*, *update* o *delete* se utiliza la clase *SQLDataSource* como se muestra a continuación:

```
Dim ds As New SqlDataSource
ds.ConnectionString =
   ConfigurationManager.ConnectionStrings("AdminCCConn").ConnectionString
ds.InsertCommand = "<instrucción insert>"
ds.UpdateCommand = "<instrucción update>"
ds.DeleteCommand = "<instrucción delete>"
Try
   ds.Insert()
   ds.Update()
   ds.Delete()
Catch ex As Exception
End Try
```

Para ejecutar un procedimiento almacenado (los procedimientos almacenados de esta aplicación se explican en el capítulo 3.6), se utilizaron dos métodos diferentes, dependiento del propósito del procedimiento.

Para el caso de procedimientos almacenados que efectúen operaciones de escritura o borrado sobre objetos, se utilizó la clase *SQLDataSource* en forma muy parecida al uso de instrucciones:

```
Dim ds As New SqlDataSource
ds.ConnectionString =
   ConfigurationManager.ConnectionStrings("AdminCCConn").ConnectionString
ds.DeleteCommand = "<nombre del procedimiento>"
ds.DeleteCommandType = SqlDataSourceCommandType.StoredProcedure
Try
   dsSolicitantes.Delete()
Catch ex As Exception
End Try
```

En el caso de procedimientos almacenados para consulta de datos, se utiliza la clase *SQLDataAdapter*, cuya sintaxis se muestra a continuación:

```
Dim daSolicitantes As New SqlDataAdapter("<nombre del procedimiento>", _

ConfigurationManager.ConnectionStrings("AdminCCConn").ConnectionString)
Dim dsSolicitantes As New DataSet
daSolicitantes.SelectCommand.CommandType = CommandType.StoredProcedure
daSolicitantes.SelectCommand.Parameters.Add("<parametro 1>",
SqlDbType.<tipo de dato>).Value = <valor>
daSolicitantes.SelectCommand.Parameters.Add("<parametro 2>",
SqlDbType.<tipo de dato>).Value = <valor>
.
.
daSolicitantes.SelectCommand.Parameters.Add("<parametro n>",
SqlDbType.<tipo de dato>).Value = <valor>
daSolicitantes.Fill(dsSolicitantes)
```

3.3 Módulo de Software

Se diseñó una estructura de menús que representa el diagrama conceptual mostrado en el anexo 1, la figura 3.2 muestra el menú correspondiente al diagrama del módulo de software, la figura 3.3 es el menú correspondiente al diagrama conceptual de préstamos de software, y la figura 3.4 muestra el menú correspondiente al diagrama conceptual de reportes de software.

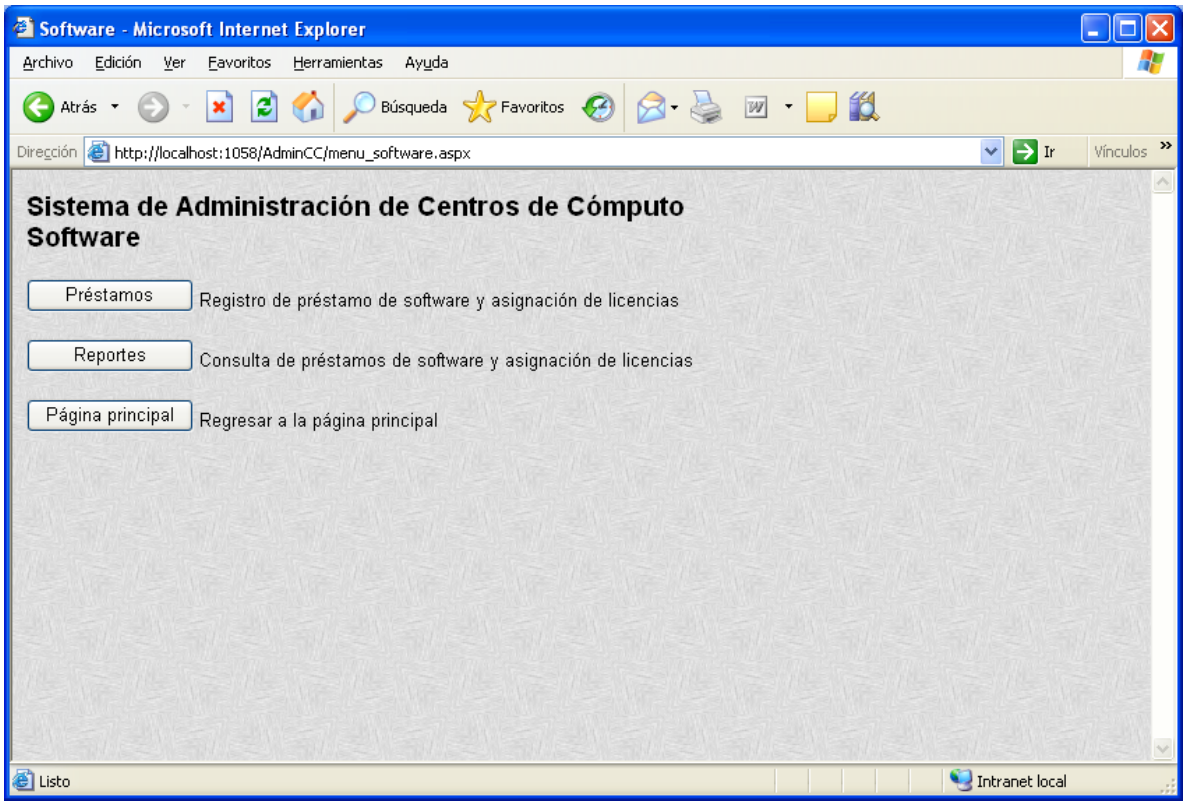


Fig. 3.2 Menú principal módulo de software

Al ejecutar cada una de las opciones mostradas en los menús se efectúan una serie de validaciones tanto de información previamente cargada en catálogos como de información introducida por el usuario, las cuales se mencionan más adelante.

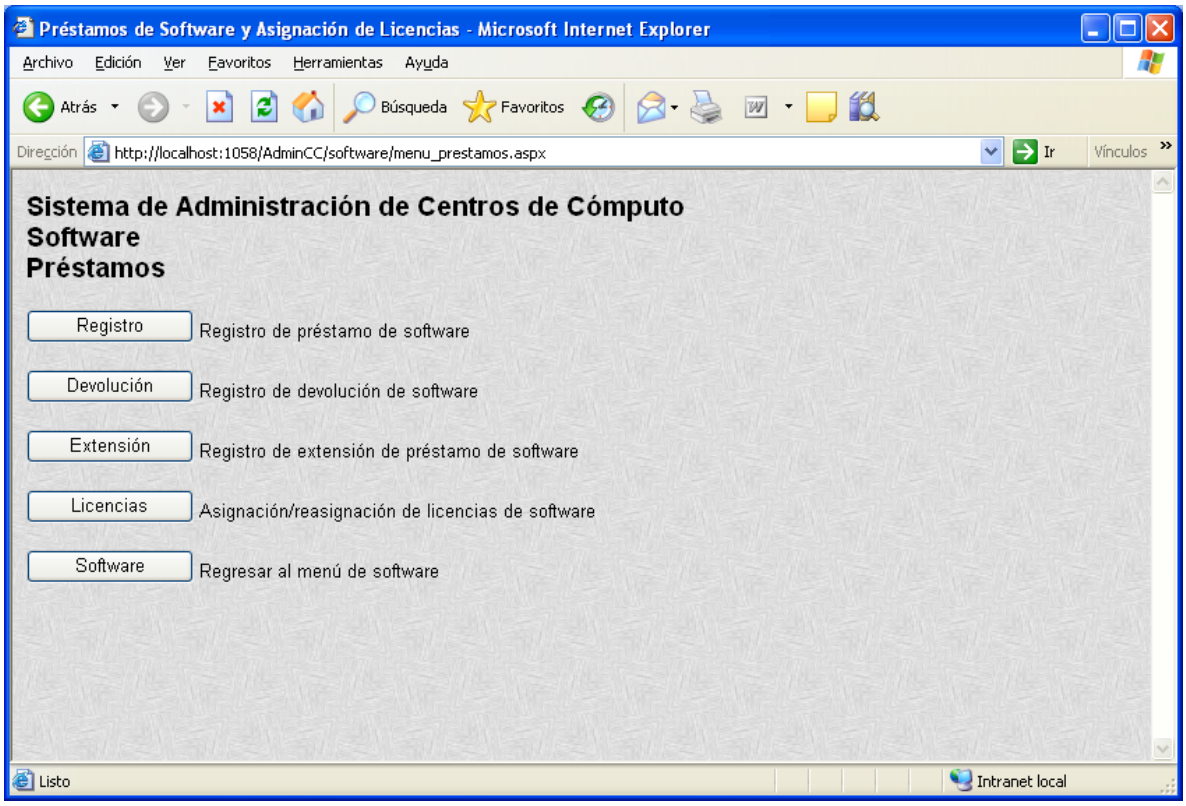


Fig. 3.3 Menú de préstamos

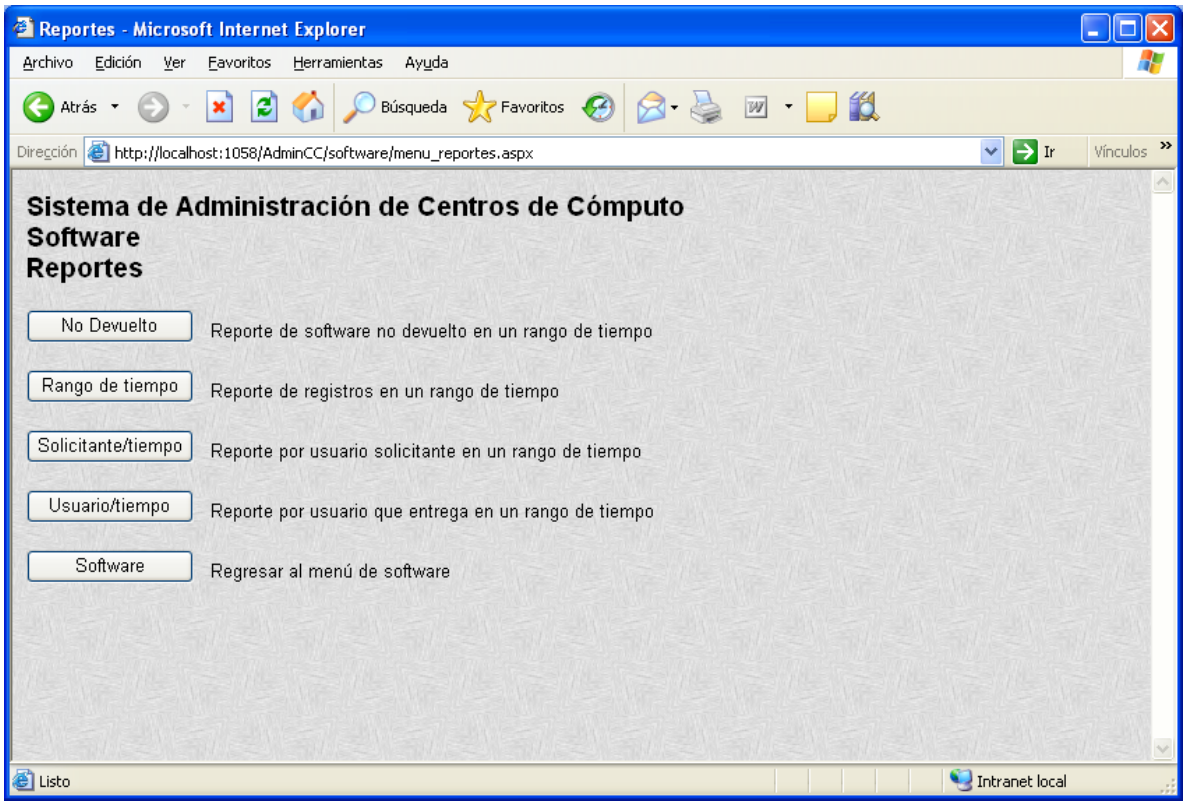


Fig. 3.4 Menú de reportes

3.3.1 Préstamos

Las validaciones de información para esta parte del sistema se mostrarán para cada uno de los procesos que la conforman.

Registro de préstamo

Para efectuar un registro de préstamo se realizan las siguientes validaciones:

1. Se carga el catálogo de solicitantes con estatus de autorizado y sin estar dados de baja, si no hay información que cumpla este requisito no se carga información y la lista aparece vacía.
2. Después de teclear el nombre del software, se efectúa una consulta en el catálogo de software y se muestran los resultados que cumplan con el criterio establecido para seleccionar un registro.
3. Al seleccionar un registro, primero se verifica si existe un préstamo no devuelto del software seleccionado, si no existe, se verifica si el número de copias es limitado o ilimitado (representado con el número 0), y si tiene un número limitado, se verifica que todavía haya licencias disponibles para su instalación, en cuyo caso se solicitan los datos del usuario tendrá el software instalado.
4. Al efectuar el registro del software, se generan los registros correspondientes tanto del préstamo como de la licencia asignada si es el caso, se muestra el número de folio y se imprime.

Devolución de préstamo

Las validaciones consideradas para efectuar una devolución de préstamo son:

1. Al teclear el número de folio para su devolución, se verifica que el número esté registrado en la base de datos.

2. Si está registrado, se comprueba que no haya sido devuelto con anterioridad.
3. Al registrar la devolución se actualiza el registro correspondiente del préstamo.

Extensión de préstamo

Para efectuar una extensión de préstamo se tomó en cuenta lo siguiente:

1. Al teclear el número de folio para extender el préstamo, se verifica que el número esté registrado en la base de datos.
2. Si está registrado, se comprueba que no haya sido devuelto con anterioridad.
3. Al registrar la extensión, se genera un nuevo registro en el histórico de préstamos, se actualiza el préstamo actual como devuelto, y se genera un nuevo registro de préstamo.
4. Si el software tiene licencias asignadas, se generan registros en el histórico de licencias con el folio actual y se actualizan los registros de licencias con el folio nuevo.

Reasignación de licencias

Las consideraciones para efectuar una reasignación de licencia son las siguientes:

1. Al teclear el número de folio para reasignar la licencia, se verifica que el número esté registrado en la base de datos.
2. Si está registrado, se comprueba que no haya sido devuelto con anterioridad.
3. Si no está devuelto, se verifica que tenga licencias asignadas.
4. Si tiene licencias asignadas, se muestran los datos en pantalla para seleccionar un registro.
5. Al seleccionar el registro, se solicitan los datos del nuevo usuario que tendrá instalado el software.

6. Al efectuar la reasignación, se genera un nuevo registro en el histórico de licencias, y se actualiza el registro de licencia correspondiente con los nuevos datos.

3.3.2 Reportes

En el caso de los reportes, si solicitan un rango de fechas, verifican que sea un rango válido y que no se exceda de la fecha actual, si solicitan algún dato adicional se carga el catálogo correspondiente para que el usuario seleccione un registro.

3.4 Módulo de Hardware

En este módulo también se diseñó una estructura de menús que representa el diagrama conceptual mostrado en el anexo 1. La figura 3.5 muestra el menú correspondiente al diagrama conceptual del módulo de hardware, el menú correspondiente al diagrama conceptual de movimientos de hardware se muestra en la figura 3.6, y la figura 3.7 muestra el menú correspondiente al diagrama conceptual del módulo de reportes de hardware.

En ese módulo también se efectúan una serie de validaciones tanto de información previamente cargada en catálogos como de información introducida por el usuario, las cuales a continuación se mencionan.

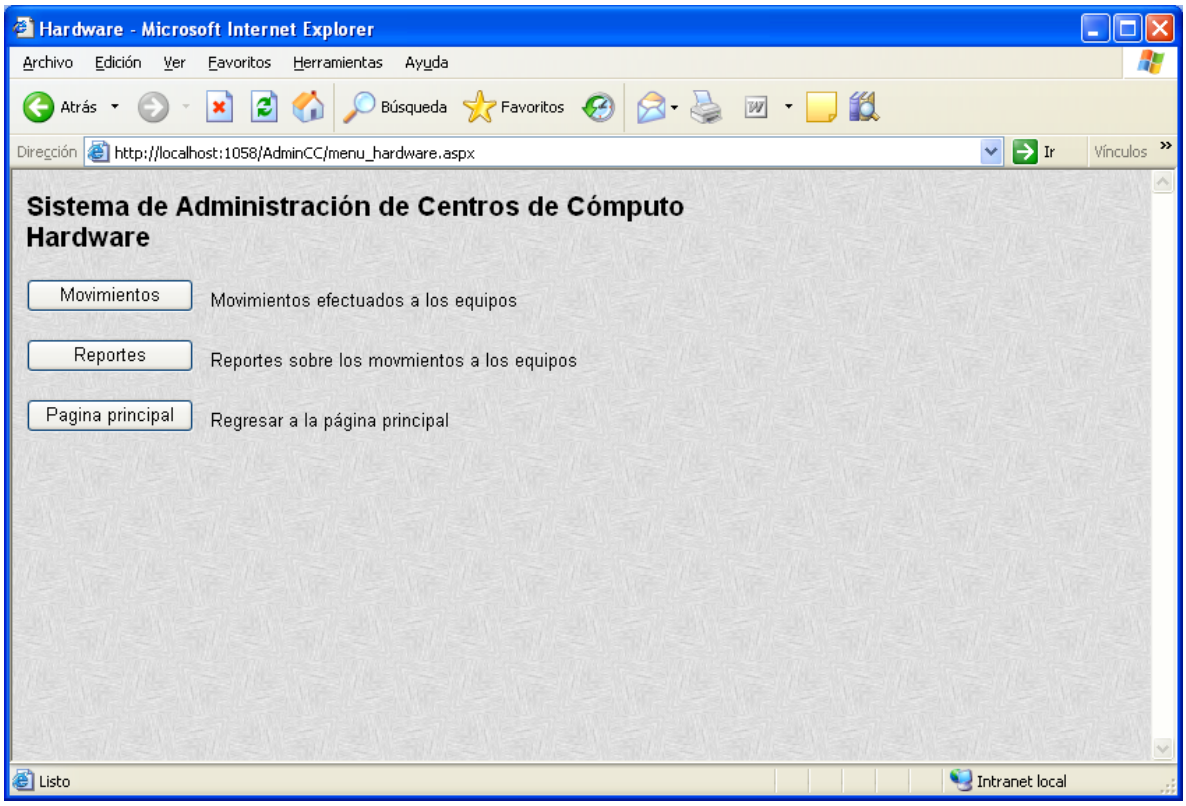


Fig. 3.5 Menú principal módulo de hardware

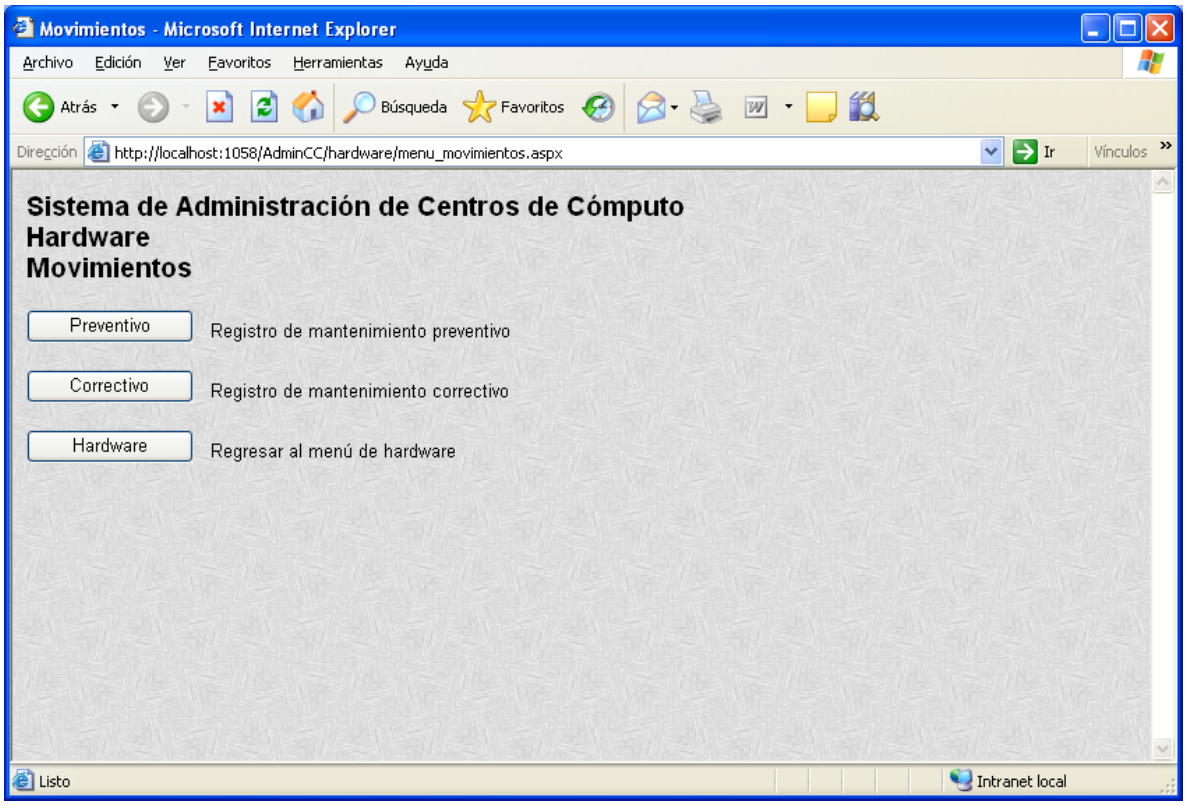


Fig. 3.6 Menú de movimientos

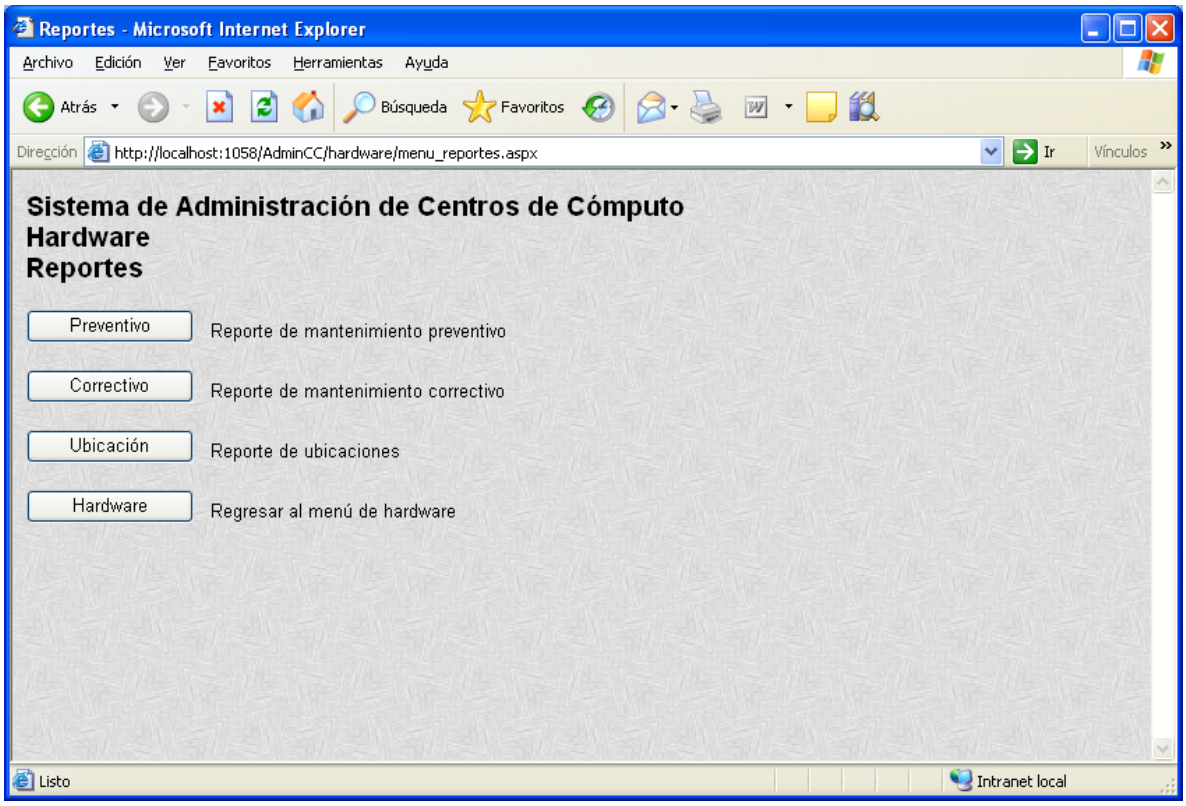


Fig. 3.7 Menú de reportes

3.4.1 Movimientos

Al igual que en el capítulo 3.3.1, se mencionan las validaciones efectuadas para cada programa que conforma esta parte del sistema.

Mantenimiento preventivo

Para registra un mantenimiento preventivo se valida lo siguiente:

1. Se carga el catálogo de proveedores sin estar dados de baja, si no hay información que cumpla este requisito no se carga información y la lista aparece vacía.
2. Al teclear el número de inventario del equipo se verifica que el número exista en la base de datos, y que corresponda a un equipo que no se haya dado de baja.

3. Al registrar el mantenimiento, se obtiene el número del registro siguiente y se guarda la información.

Mantenimiento correctivo

Las validaciones para registrar un mantenimiento correctivo son:

1. Se carga el catálogo de proveedores sin estar dados de baja, si no hay información que cumpla este requisito no se carga información y la lista aparece vacía.
2. Al teclear el número de inventario del equipo se verifica que el número exista en la base de datos, y que corresponda a un equipo que no se haya dado de baja.
3. Si el equipo no está dado de baja, se verifica si se tiene registro de algún mantenimiento efectuado con anterioridad.
4. Si no tiene un mantenimiento anterior, se solicita la falla y se habilita el registro.
5. Si tiene un mantenimiento anterior, se verifica si ya está cerrado.
6. Si ya está cerrado, se solicita la falla y se habilita el registro para un nuevo mantenimiento.
7. Si no está cerrado, se solicitan las observaciones de cierre y se habilita el cierre del mantenimiento.
8. Si se efectúa un registro nuevo, se obtiene el número del registro siguiente y se guarda la información
9. Si se efectúa un cierre, se actualizan los datos correspondientes.

3.4.2 Reportes

En este caso todos los reportes elaborados únicamente solicitan un rango de fechas, por lo que se verifica que sea un rango válido y que no se exceda de la fecha actual.

3.5 Seguridad en el sistema

Para evitar que cualquier persona pueda modificar los registros de la base de datos por medio de la aplicación, se implementó el acceso a la misma por medio del número de empleado y una contraseña, la cual se almacena en forma binaria dentro de la base de datos y se valida por medio del cifrado/descifrado de cadenas.

El método seleccionado para cifrar y descifrar las contraseñas se basa en algoritmos de llaves primarias. Típicamente, estos algoritmos de llaves primarias son conocidos como cifrado de bloques. Estos métodos utlizan bloques de 8 bytes cuando cifran datos. En su forma más simple, se cifra cada bloque de datos utlizando la misma llave, lo que significa que si existen bloques de datos idénticos su equivalente cifrado también será idéntico, lo que puede provocar que se descubra la llave de cifrado. Para resolver esto, se utiliza lo que se llama un vector de inicialización, el cual se mezcla con la llave para codificar los bloques de datos. Esto da como resultado que los bloques cifrados sean diferentes, aunque contengan la misma información.

Para el caso de .Net Framework, componente utilizado para el desarrollo de esta aplicación, se manejan los siguientes algoritmos de cifrado:

- DES: Data Encryption Standard, utiliza bloques de 8 bytes para cifrar datos. La llave para cifrar la información es la misma para descifrarla. Cuando se utiliza este algoritmo, se debe proveer una llave de 8 bytes, la cual se reduce a 7 bytes debido a que el octavo se utiliza para propósitos de paridad. DES cifra cada bloque de datos 16 veces, es decir, se utliza la llave para cifrar el bloque, se modifica ligeramente la llave y se vuelve a cifrar el bloque, hasta que se ha cifrado 16 veces.

- RC2: Rivest Chiper, fué diseñado como reemplazo para DES, y es 3 veces más rápido. También es un algoritmo de bloques de 8 bytes, sin embargo, RC2 puede manejar llaves tan grandes como la plataforma del software lo permita.

- TripleDES: Este algoritmo requiere una llave de 24 bytes, la cual es dividida en tres llaves de 8 bytes para cifrar cada bloque 3 veces. Esto significa que cada bloque se cifra 48 veces, lo cual provee un algoritmo de cifrado muy seguro, por lo que se tomó de base para cifrar las contraseñas de esta aplicación.

Debido a que al cifrar información los caracteres resultantes pueden ser imprimibles o no, no es posible mostrar la información cifrada en pantalla, por lo que se utilizaron arreglos de bytes para almacenar la información. La cadena de caracteres original se cifra y se almacena en un arreglo de bytes, y para descifrarla se toma del arreglo de bytes y se almacena en una cadena de caracteres.

Para poder almacenar este arreglo de bytes en la base de datos se utilizó un campo de tipo binario variable (varbynary), el cual acepta de 1 a 8000 bytes de información (mencionado en el diccionario de datos, capítulo 2.8). Al consultar la información en la base de datos solo menciona que el campo es binario, sin mostrar los caracteres que contiene (ver figura 3.8).

Para que el sistema pueda cifrar y descifrar las contraseñas se definió la clase pública *cls_tripledes*, la cual se muestra en el anexo 6. La rutina de cifrado se utiliza para establecer la contraseña del usuario y guardarla en la base de datos, mientras que la rutina de descifrado se utiliza para validar que la contraseña introducida para iniciar sesión corresponda a la almacenada dentro de la base.

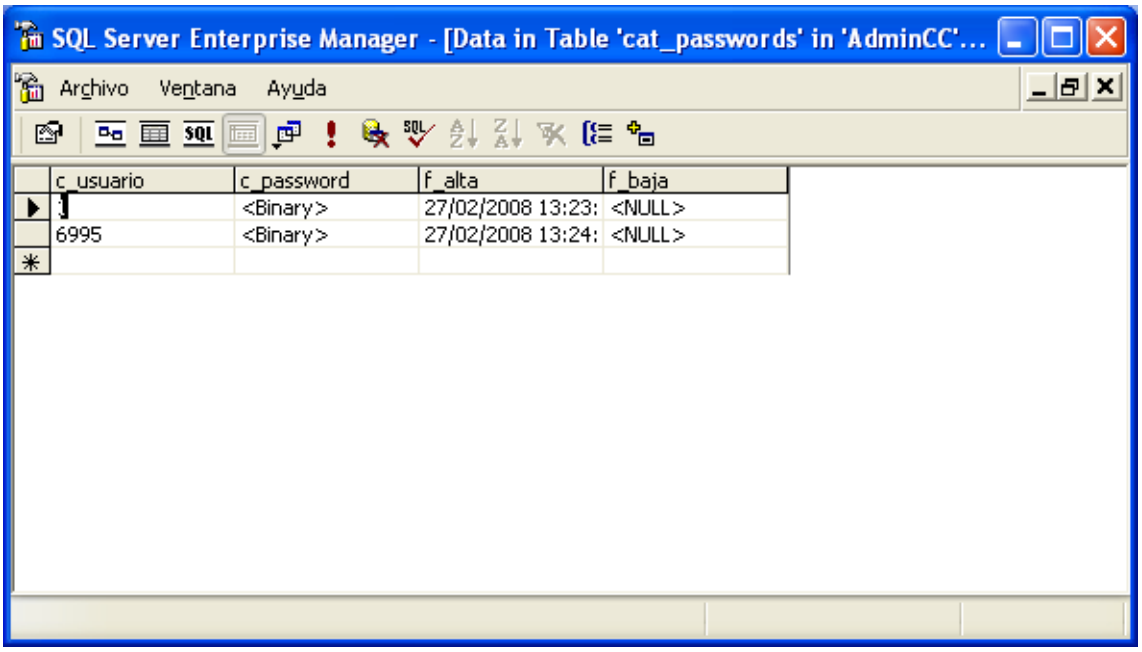


Fig. 3.8 Información cifrada en la base de datos

3.6 Procedimientos almacenados

Durante el desarrollo del sistema se detectó la necesidad de crear procedimientos almacenados en la base de datos para que las consultas de información se pudieran presentar en forma adecuada, ya que de esta forma se pueden manipular búsquedas de datos complejas dentro de la base y generar una tabla temporal con los resultados deseados, para que la aplicación la consulte. Los procedimientos elaborados para el sistema se muestran en el anexo 5.

4. Resultados

4.1 Carga inicial de datos

Para que se pueda utilizar los módulos de Software y Hardware es necesario cargar información en las tablas correspondientes a los catálogos del sistema. Debido a que la información correspondiente se encuentra en archivos de hojas de cálculo, se utilizaron los servicios de transformación de datos (DTS) integrados en SQL Server, los cuales son un conjunto de herramientas para importar, exportar y transformar datos. Un servicio de transformación de datos se crea como paquetes, lo cuales tienen tareas específicas para definir el trabajo que se va a efectuar, ya sea definiendo un origen o destino de los datos, o una conexión entre el origen y el destino de la información.

En el caso de esta aplicación, se crearon DTS para importar los datos de las hojas de cálculo en las tablas necesarias para la operación del sistema. La figura 4.1 muestra los DTS creados para este fin:

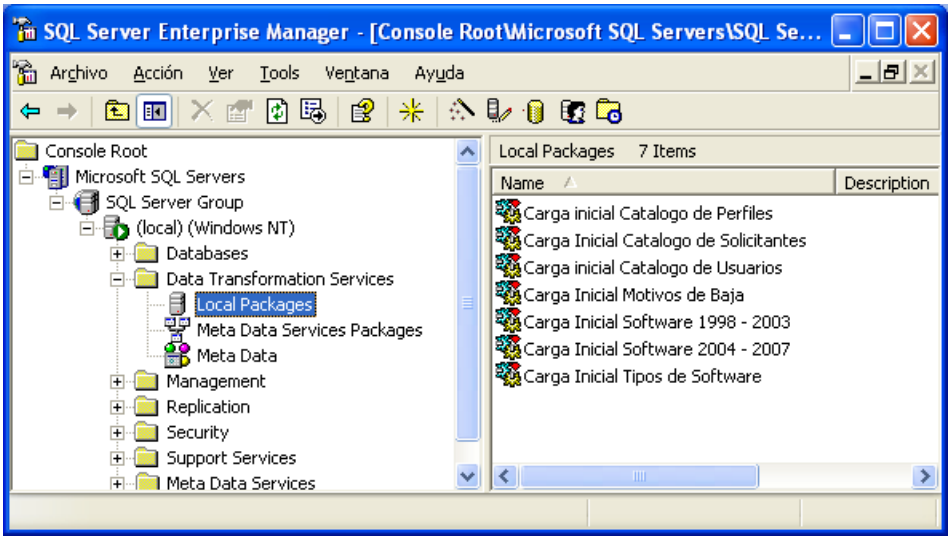


Fig. 4.1 Servicios de transformación de datos

Después de genear y ejecutar estos paquetes se efectuaron pruebas de los módulos correspondientes en el sistema, Administración y Catálogos (ver diagrama conceptual en el anexo 1). Como ejemplo, se muestra la operación del catálogo de motivos de baja.

Al seleccionar el catálogo de motivos de baja aparece la pantalla mostrada en la figura 4.2.

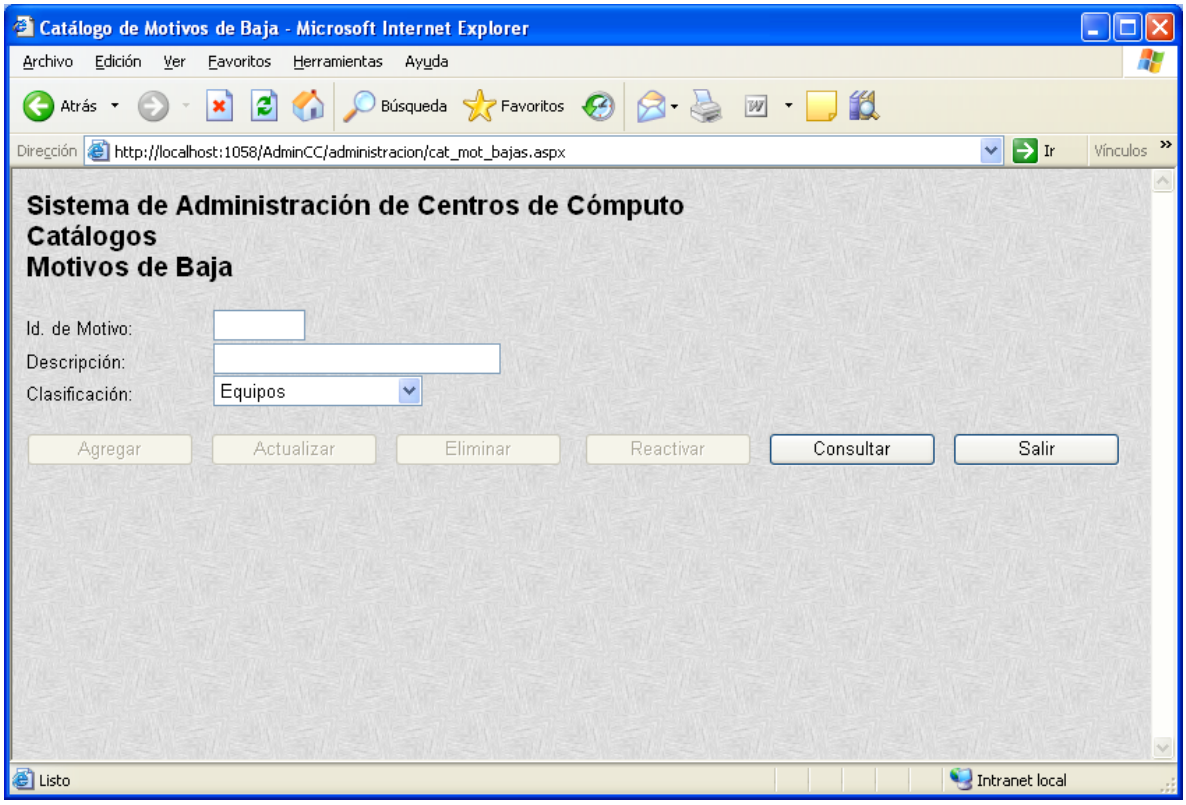


Fig. 4.2 Catálogo de motivos de baja

Cuando se teclea un identificador de motivo el sistema verifica si ya está registrado, si ese es el caso, muestra sus datos en pantalla y habilita los botones *Actualizar* y *Eliminar* (figura 4.3), de lo contrario habilita el botón *Agregar* (figura 4.4).

Si se desea agregar o actualizar un registro, y no se tiene toda la información requerida para efectuar la transacción, se muestra el mensaje de aviso de la figura 4.5.

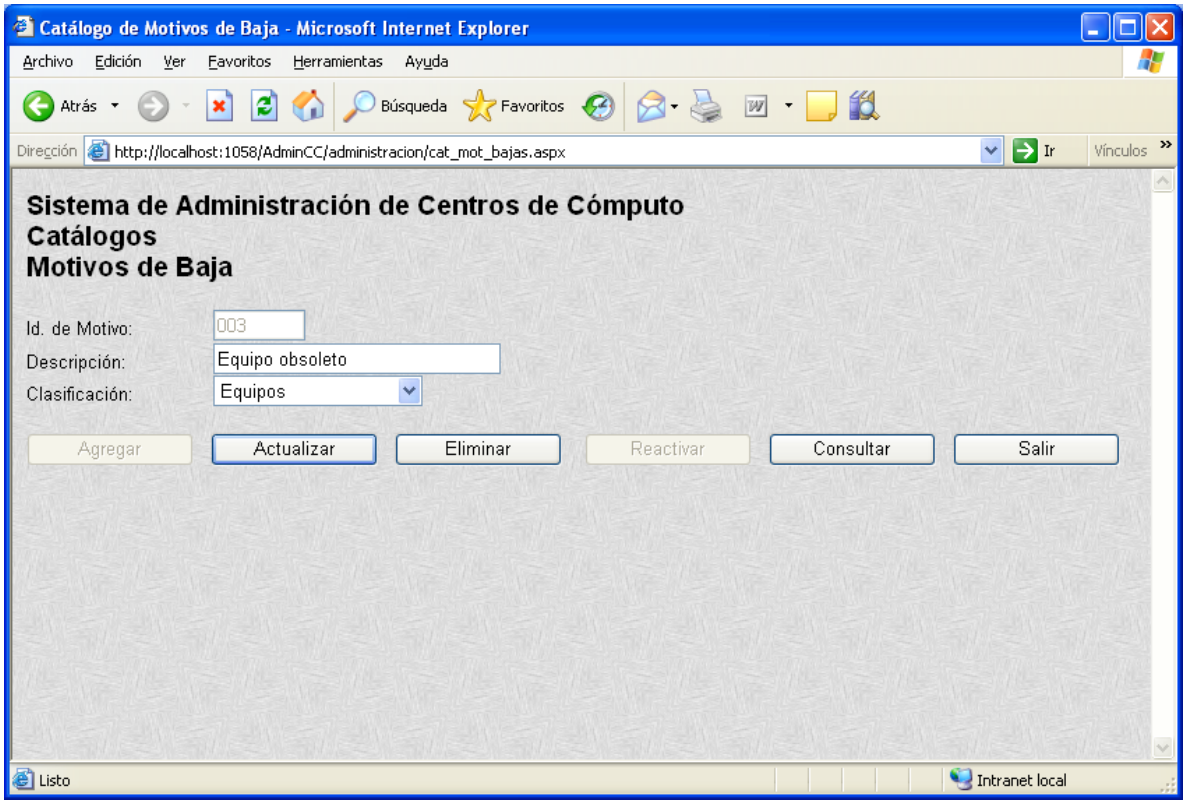


Fig. 4.3 Datos encontrados en la base de datos

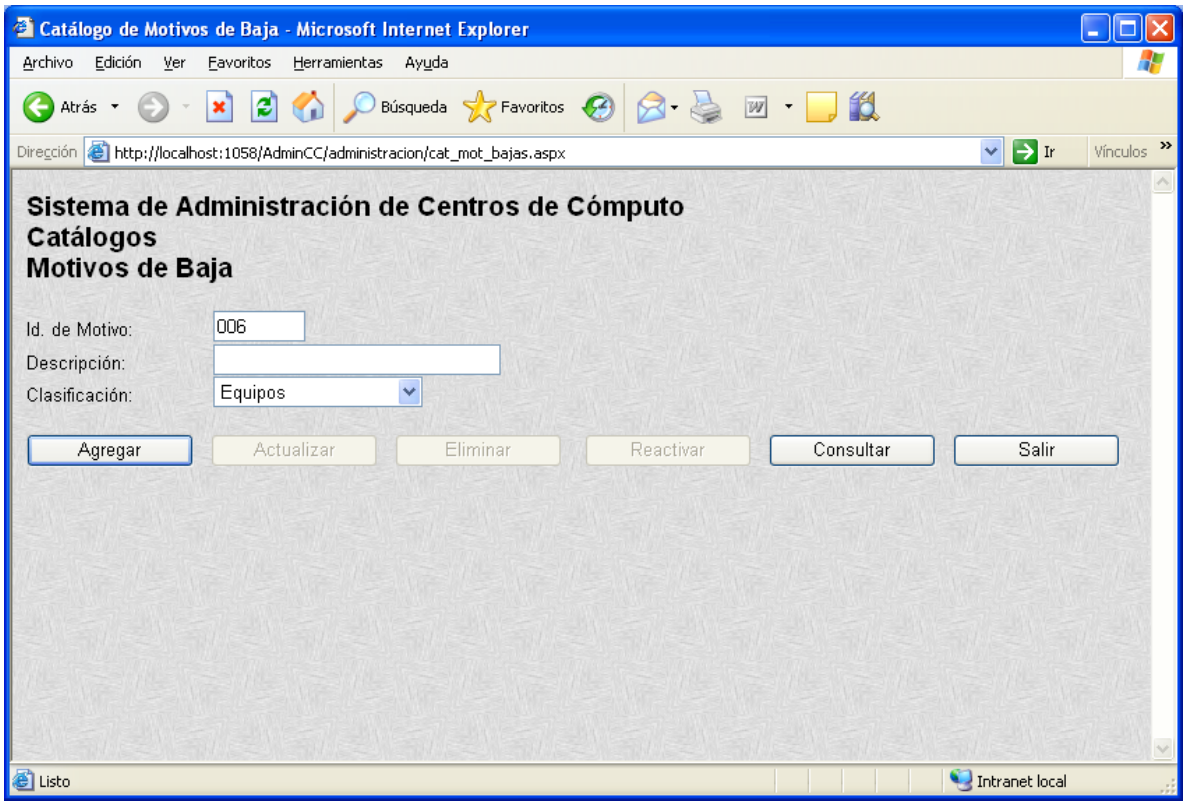


Fig. 4.4 Datos nuevos

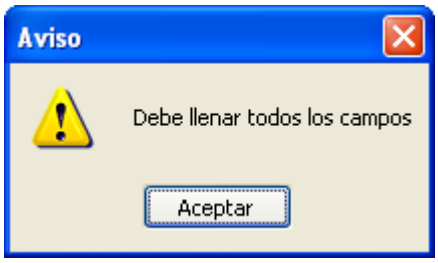


Fig. 4.5 Aviso de falta de información

Si se proporciona toda la información, se muestra un aviso de la figura 4.6 o 4.7 (dependiendo si se agregó o se actualizó un registro), indicando que la transacción fué efectuada.

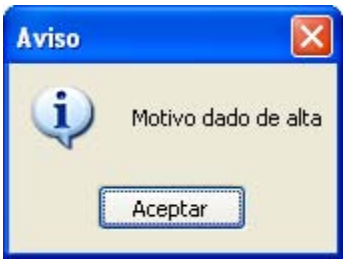


Fig. 4.6 Aviso de registro dado de alta

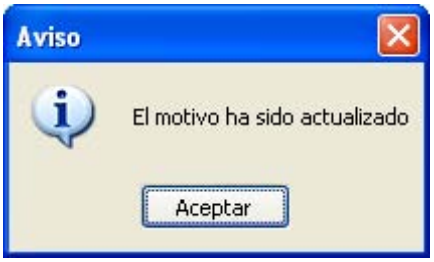


Fig. 4.7 Aviso de registro actualizado

Las operaciones para eliminar o reactivar un registro efectúan validaciones y muestran avisos de forma similar a los ya mostrados. Si se desea efectuar una consulta se inhabilitan todas las opciones hasta que se cancela la consulta, se muestran las opciones posibles para filtrar la consulta (figura 4.8) y al dar clic en el botón *Aceptar* se efectúa la búsqueda de información. El resultado de la búsqueda se muestra en forma de tabla, como se ve en la figura 4.9.

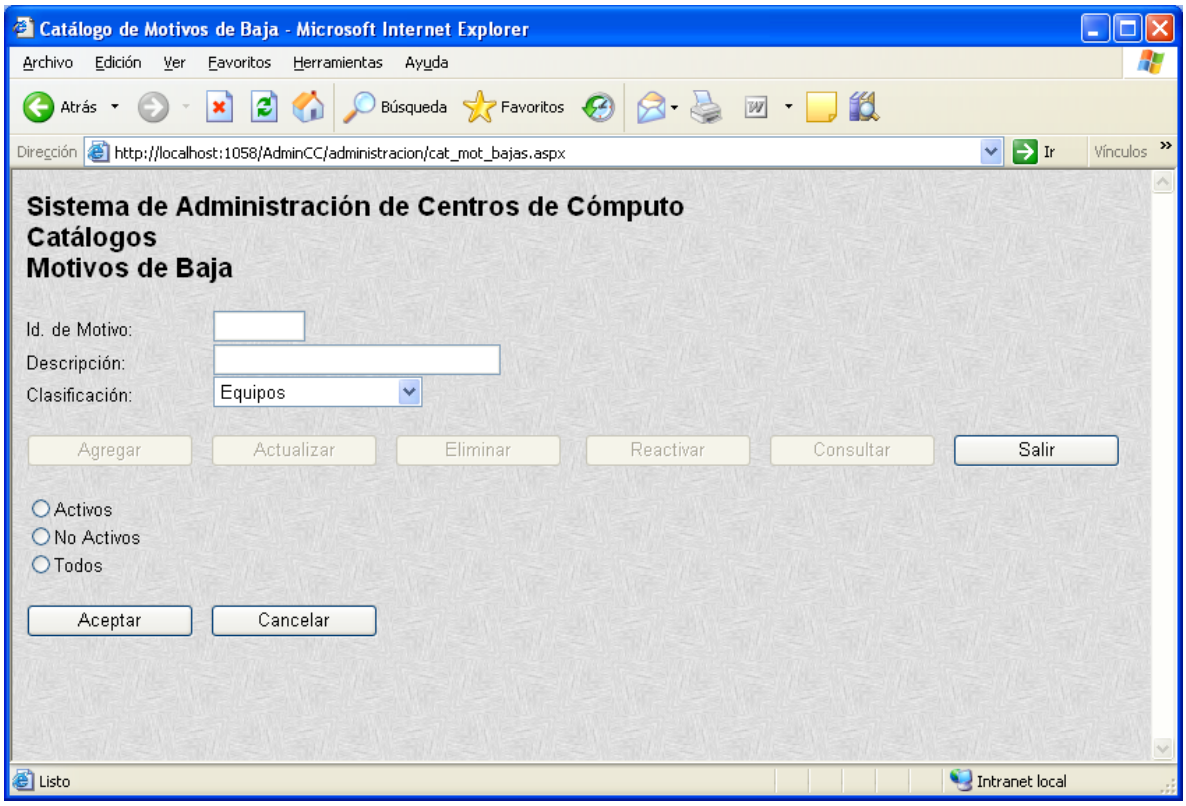


Fig. 4.8 Opciones de consulta

Al verificar esta información en la base de datos se puede comprobar que está almacenada, como lo muestra la figura 4.10.

Finalmente, el botón *Generar reporte* (ver figura 4.9) crea un archivo tipo pdf con el resultado de la consulta.

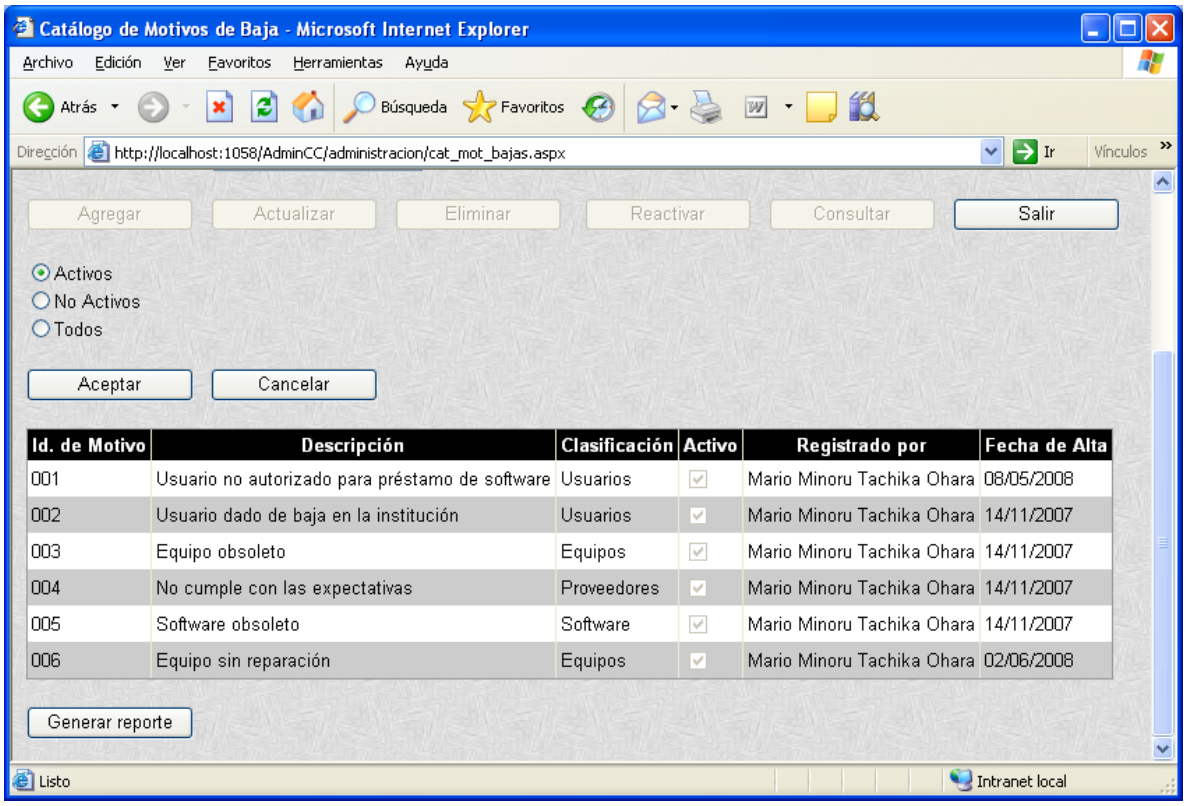


Fig. 4.9 Resultado de la consulta

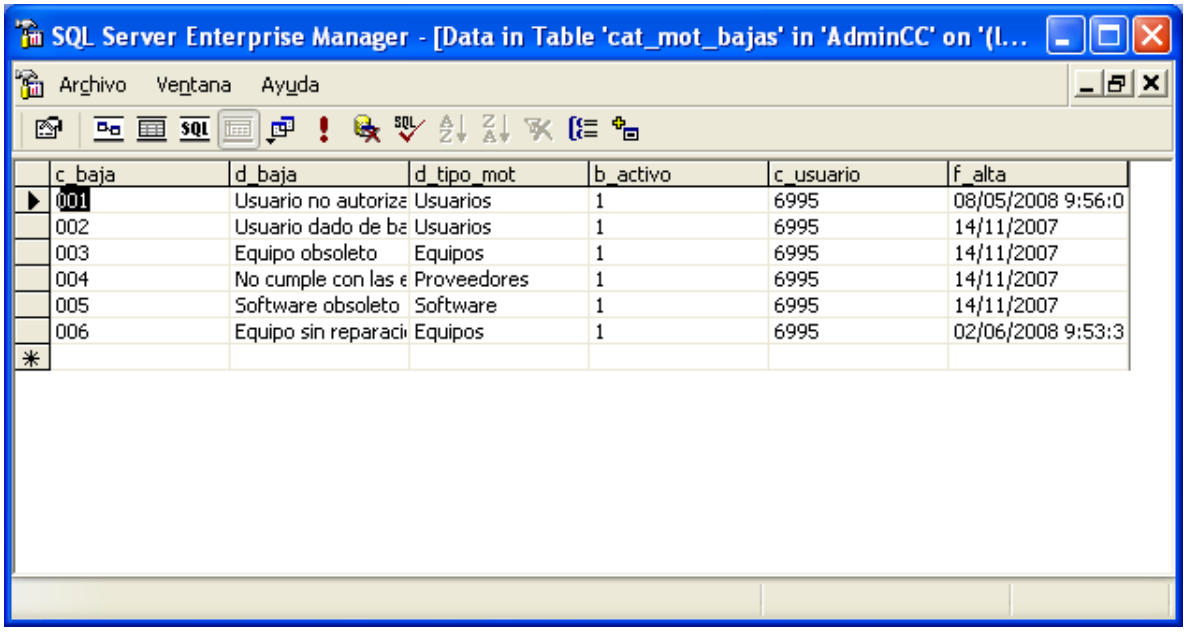


Fig. 4.10 Información contenida en la base de datos

4.2 Módulo de Software

Este módulo tiene dos opciones principales: *Préstamos* y *Reportes* (mencionados en el capítulo 3.3). Para mostrar el funcionamiento de estas opciones se toman como ejemplos el registro de un préstamo y la generación de un reporte por rango de tiempo.

4.2.1 Préstamos

Al seleccionar la opción correspondiente al registro de un préstamo aparece la pantalla mostrada en la figura 4.11.

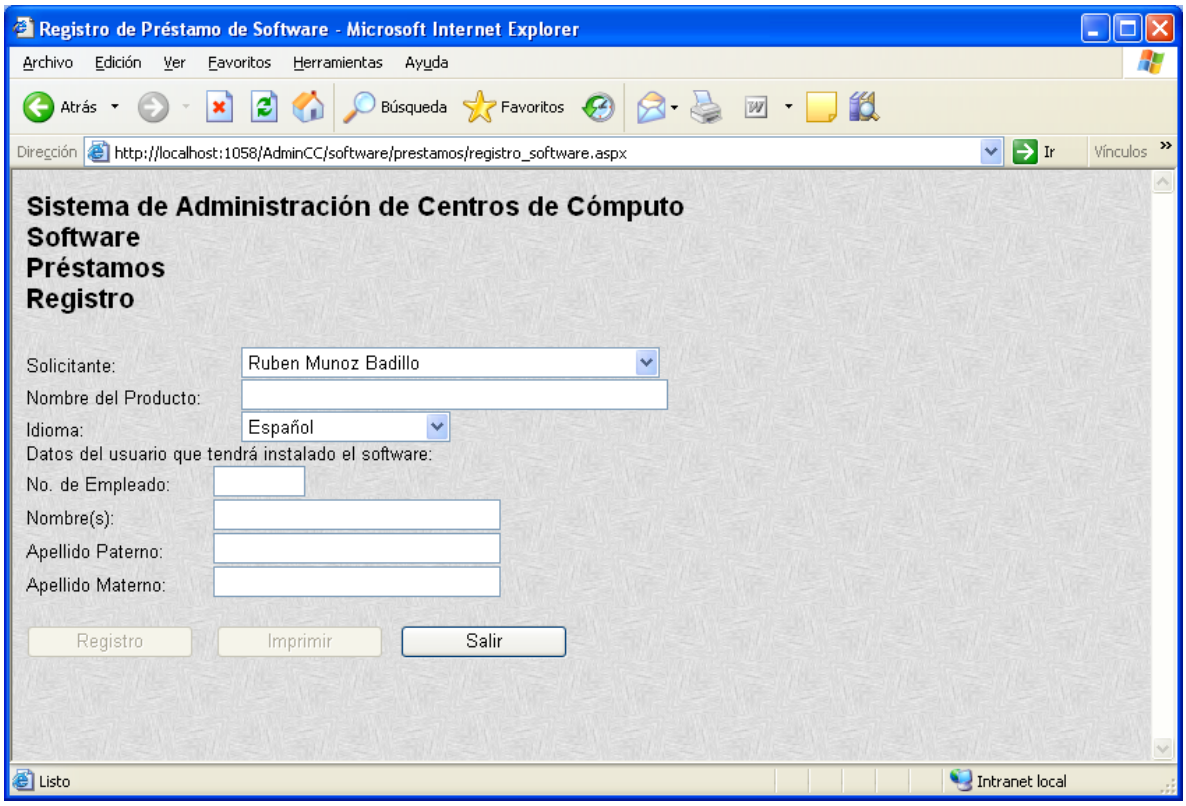


Fig. 4.11 Registro de préstamo

No es necesario escribir el nombre completo del producto, ya que el sistema busca los registros que contengan las palabras que son tecleadas (figura 4.12), y si se cambia el idioma la búsqueda se actualiza automáticamente.

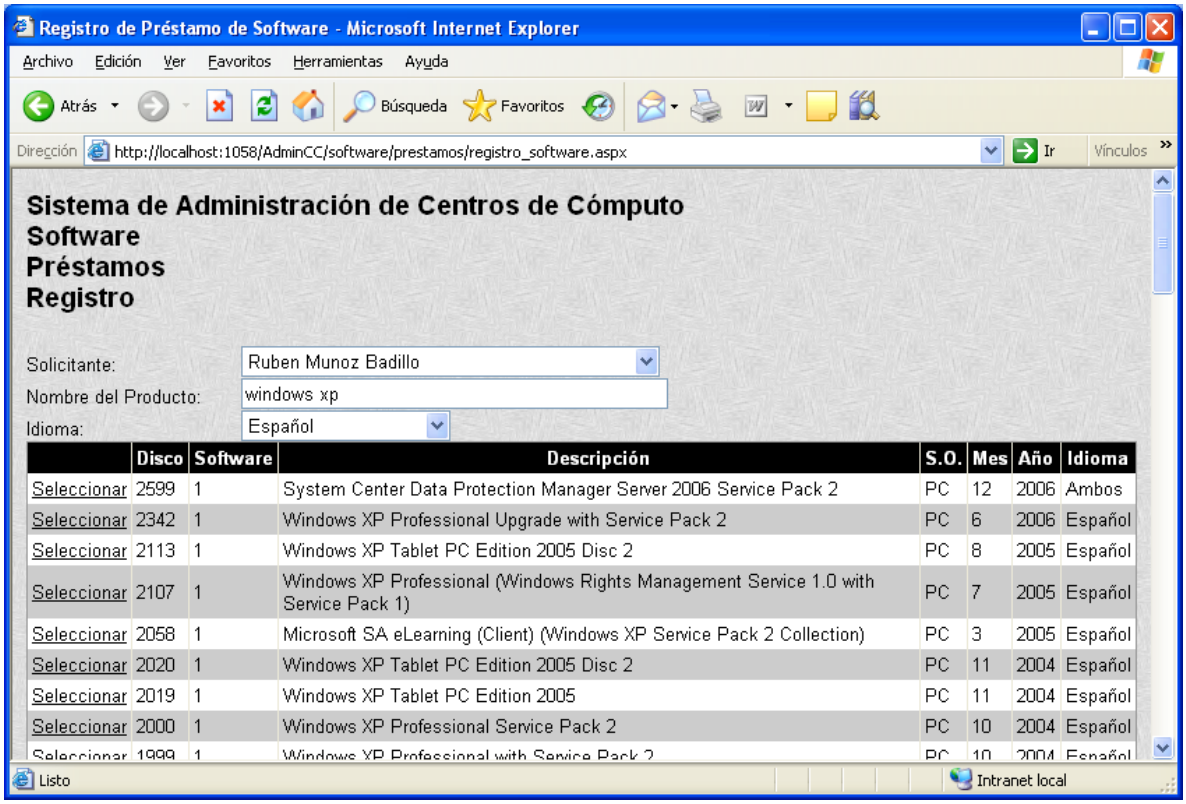


Fig. 4.12 Búsqueda de software

Una vez seleccionado el software, si no tiene un número limitado de licencias (como en este ejemplo), se deshabilitan los campos correspondientes al usuario final, y se habilita el botón *Registro* (figura 4.13).

Al registrar el préstamo se le asigna un número de folio (figura 4.14), y para asegurar que se imprima el formato de préstamo se deshabilitan los botones *Registro* y *Salir* (figura 4.15).

Al dar clic en el botón *Imprimir*, aparece en una ventana nueva el folio para su impresión, como se ve en la figura 4.16.

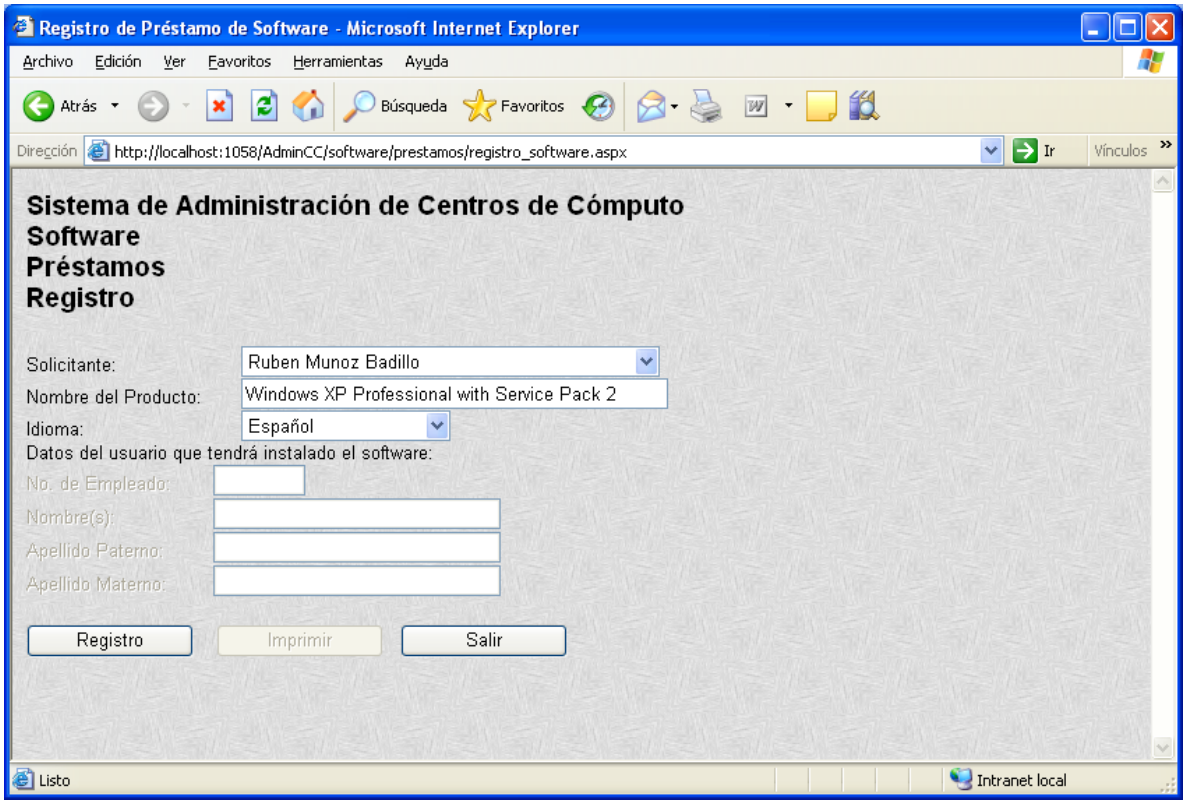


Fig. 4.13 Validación de licencias y botón de registro

Fig. 4.14 Número de folio asignado

Finalmente, Al verificar la base de datos se encuentran los registros correspondientes, como se muestra en la figura 4.17.

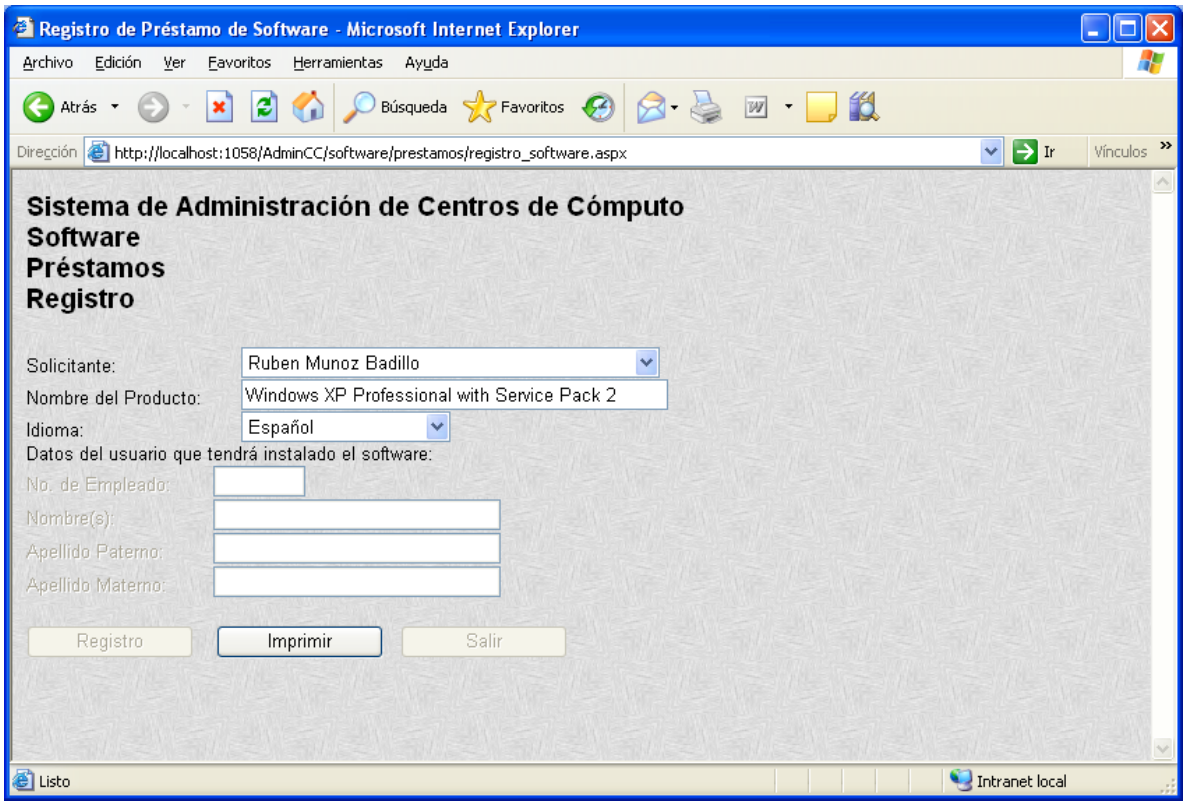


Fig. 4.15 Impresión de formato de préstamo

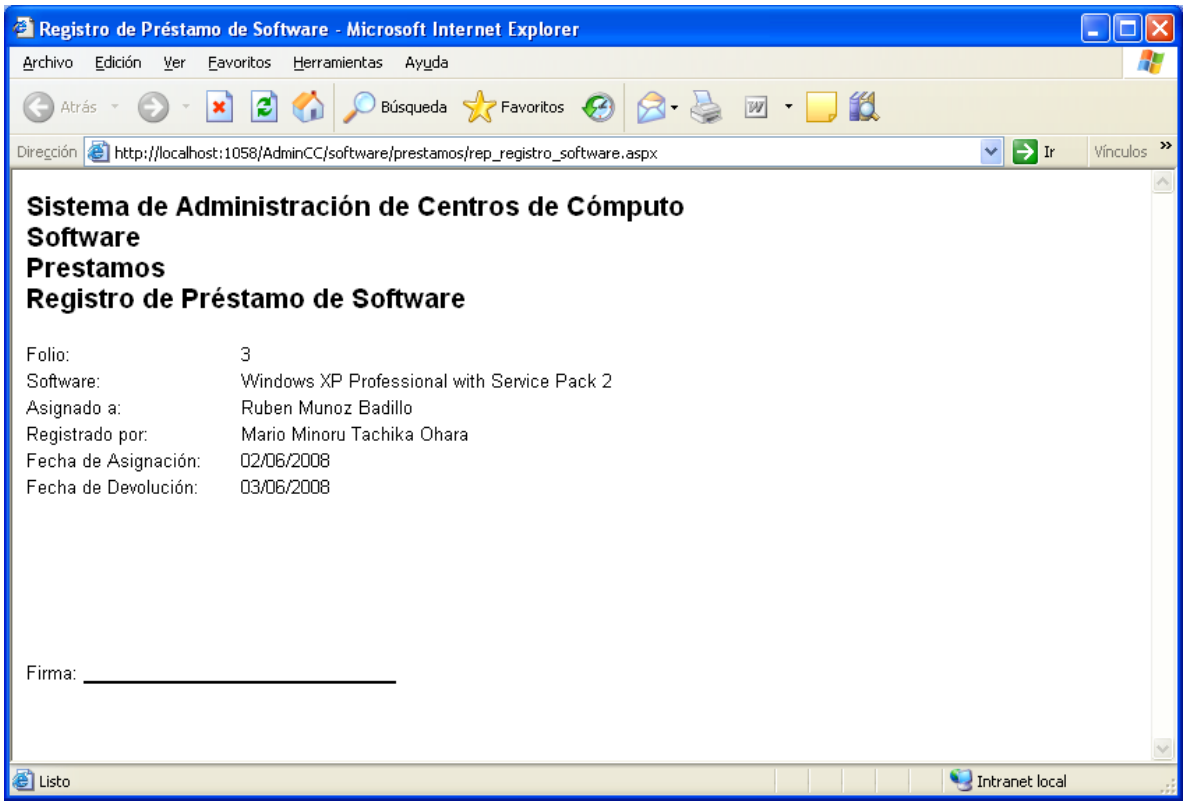


Fig. 4.16 Folio de préstamo

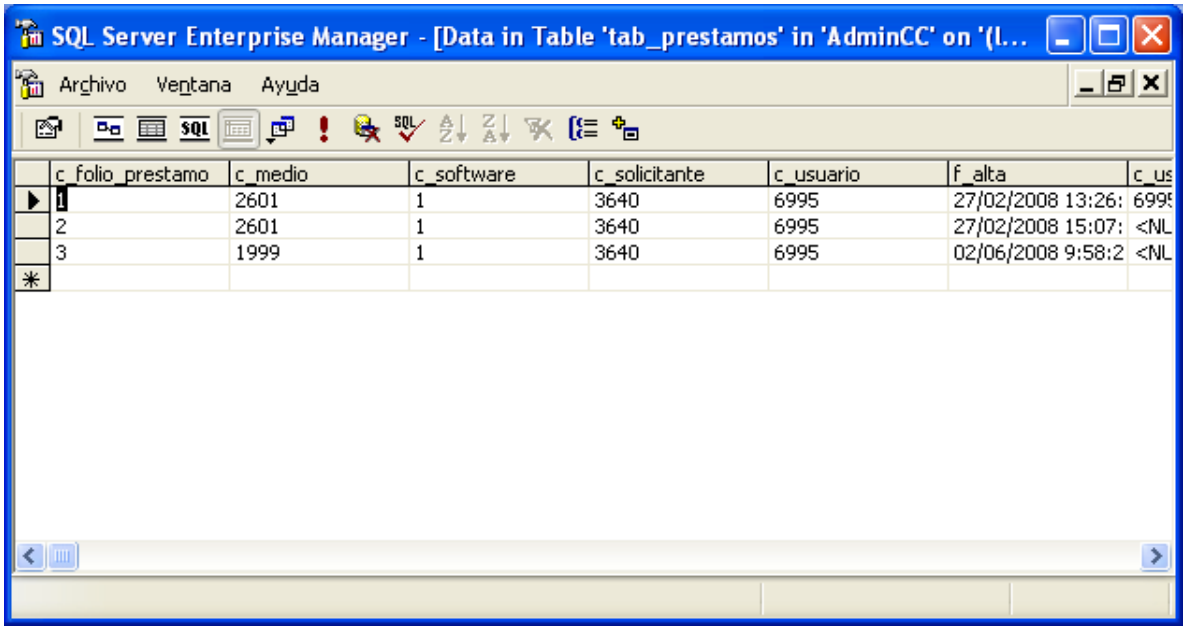


Fig. 4.17 Información contenida en la base de datos

4.2.2 Reportes

Los reportes efectúan consultas a la base de datos para mostrar la información en pantalla, y si se desea, generar un archivo pdf con el resultado de la consulta. Como ejemplo se muestra el reporte de préstamos por rango de tiempo (figura 4.18).

Se debe especificar el rango de tiempo que se desea consultar, el botón Calendario muestra fechas en pantalla para que se seleccione la deseada (figura 4.19).

Una vez seleccionado el rango de fechas se puede efectuar la consulta, se verifica que el rango de fechas sea válido y se muestran los resultados. Si se desea guardar esta consulta en un archivo pdf se utiliza el botón *Generar reporte*. (figura 4.20).

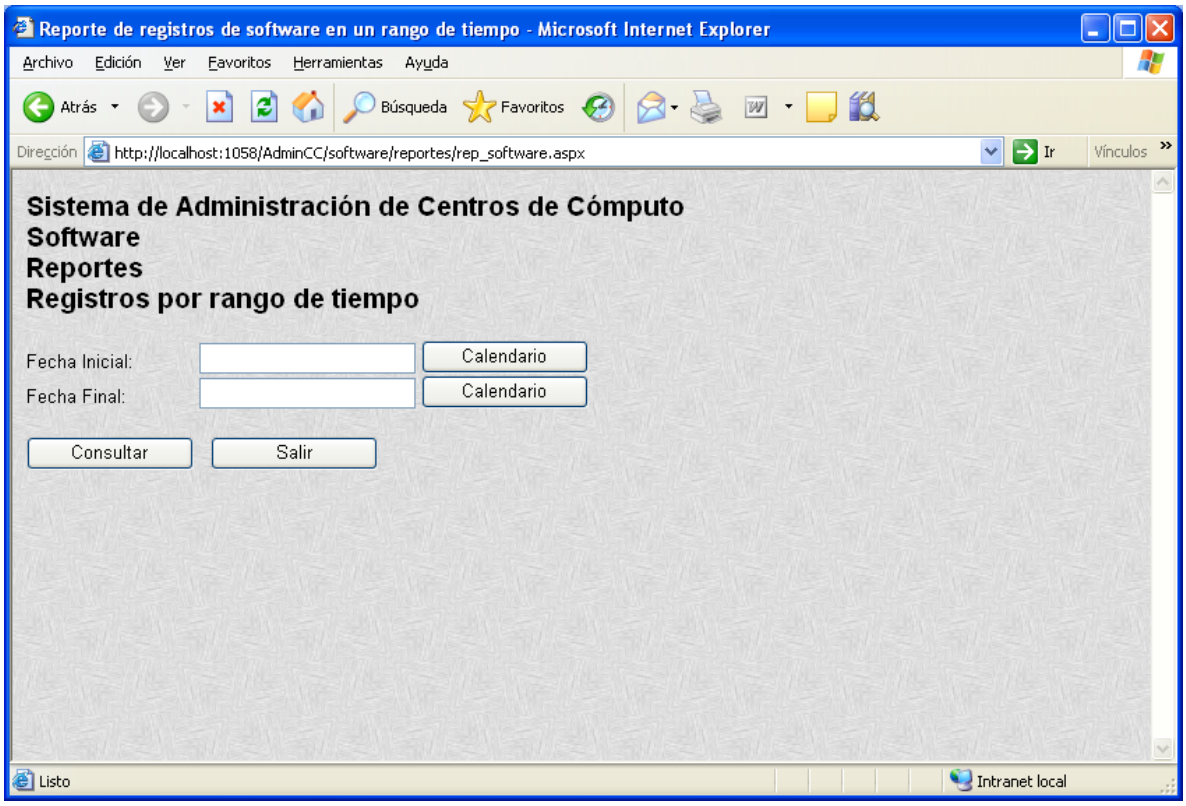


Fig. 4.18 Reporte de préstamos por rango de tiempo

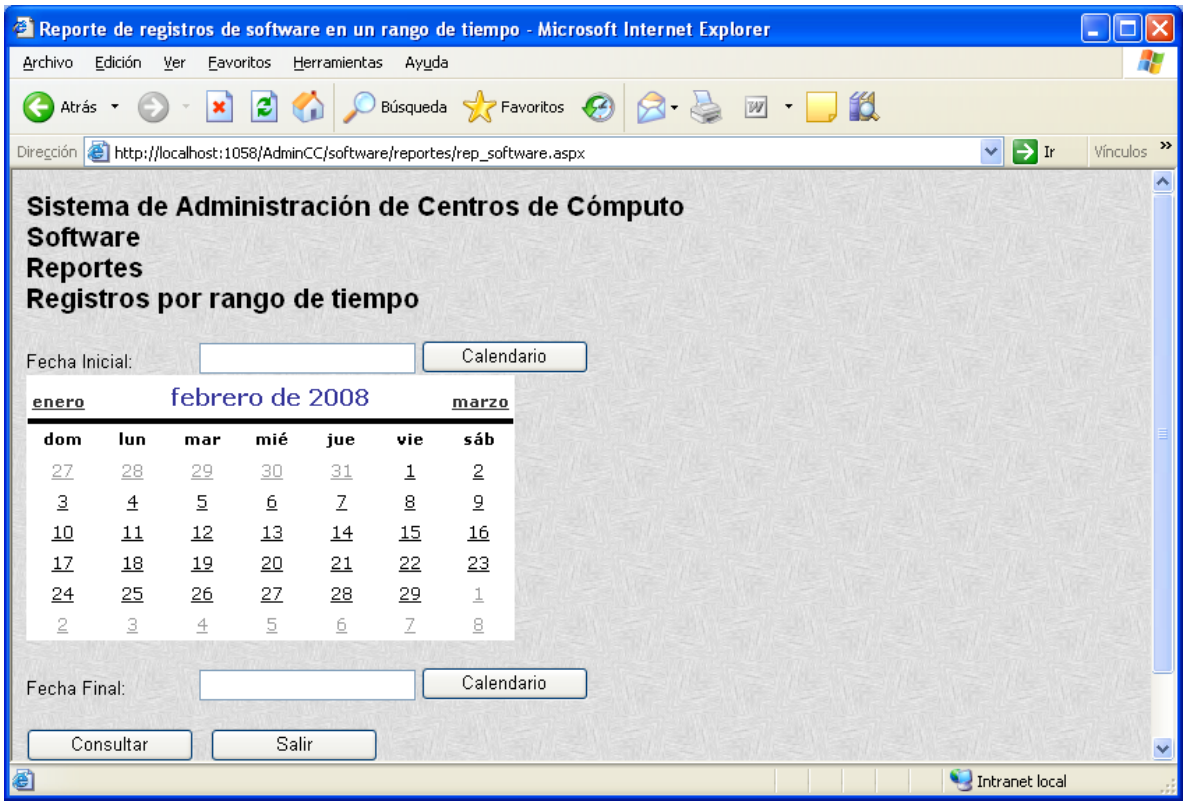


Fig. 4.19 Selección de fechas por medio del calendario

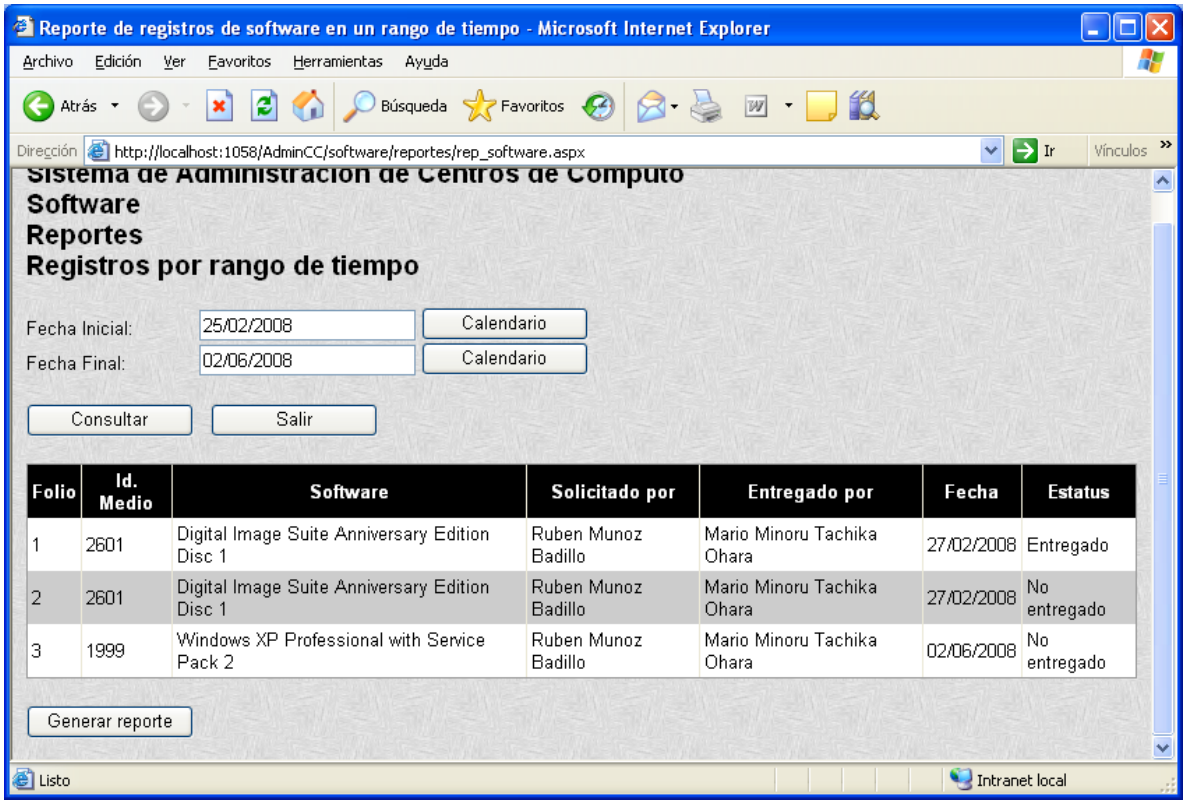


Fig. 4.20 Resultados de la consulta

4.3 Módulo de Hardware

Este módulo también tiene dos opciones principales: *Movimientos* y *Reportes* (ver capítulo 3.4). A continuación de mostrará un ejemplo para cada una de estas opciones.

4.3.1 Movimientos

Como ejemplo para este módulo se mostrará cómo se registra un mantenimiento correctivo. Al seleccionar la opción correspondiente aparece la pantalla mostrada en la figura 4.21.

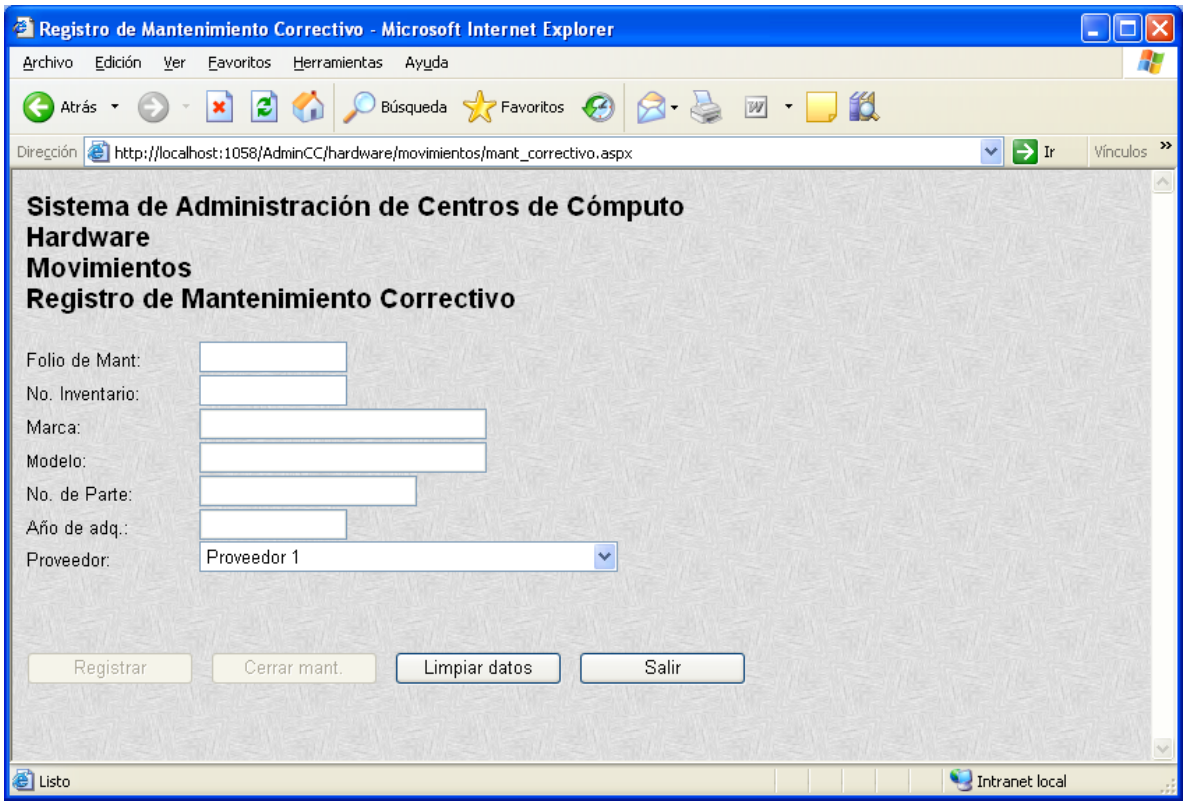


Fig. 4.21 Registro de mantenimiento correctivo

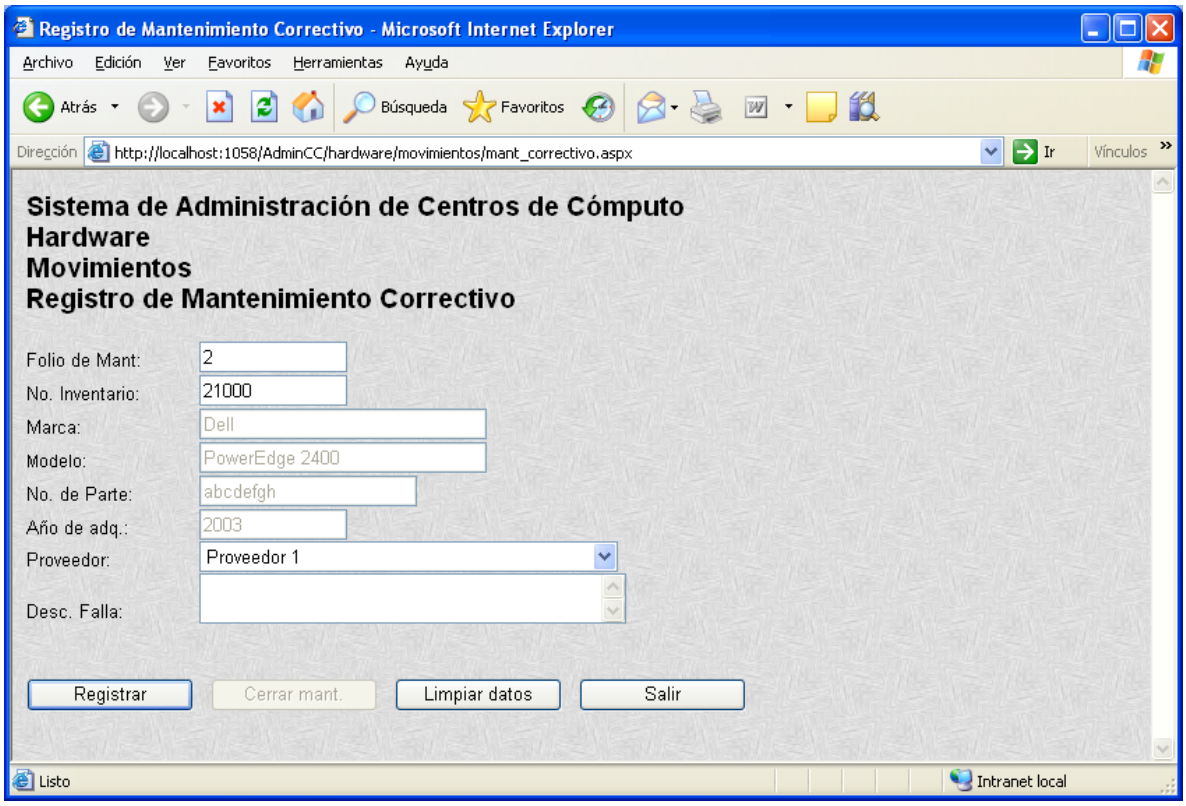


Fig. 4.22 Validación de reporte nuevo

Al escribir el número de inventario se efectúan las validaciones necesarias para determinar si es un equipo existente, que no esté dado de baja, y si tiene un reporte previo sin cerrar, al comprobar que se trata de un reporte nuevo se solicita la información de la falla y se habilita el botón de registro (figura 4.22).

Se debe teclear la información de la falla, en caso contrario si se trata de registrar el reporte aparece la indicación mostrada en la figura 4.23.

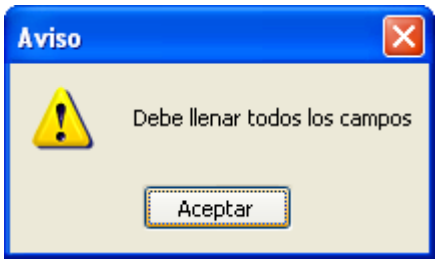


Fig. 4.23 Aviso de falta de información

Una vez que se introduce toda la información y se da clic en el botón *Registrar* se efectúa la transacción en la base de datos, al completarse aparece la indicación de que el registro fué generado (figura 4.24).

Fig. 4.24 Aviso de registro de datos

Si se verifica la base de datos, se encuentra el registro que se acaba de generar (figura 4.25).

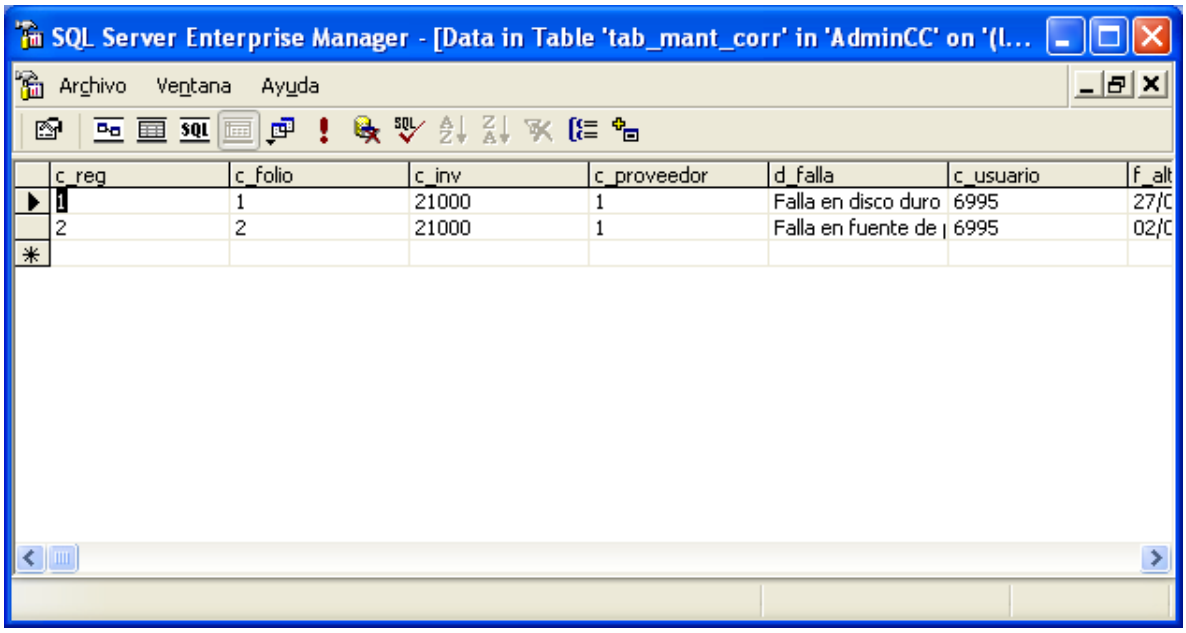


Fig. 4.25 Información contenida en la base de datos

4.3.2 Reportes

Para este módulo los reportes se efectúan por rango de fechas, el funcionamiento es similar al explicado en el módulo de software (capítulo 4.2.2).

5. Conclusiones

Este trabajo se efectuó como resultado de haber tomado el diplomado *"Diseño de Sistemas de Información orientado a Negocios con SQL Server y Oracle", tercera generación*, logrando desarrollar un sistema de información que cumple con el objetivo para el que fué diseñado: la administración de software y equipos en los centros de cómputo de la Auditoría Superior de la Federación.

Durante el desarrollo de este sistema se aplicaron los conocimientos adquiridos en este diplomado en las siguientes etapas:

- En la etapa de análisis y diseño, al elaborar el diagrama de actividades, el modelo de la organización, los diagramas de casos de uso, el diagrama entidad – relación y el script para creación de la base de datos.
- En la etapa de desarrollo del sistema, al implementar los procedimientos almacenados.
- En la etapa de resultados, al efectuar la carga inicial de datos.

Por otro lado, se logró profundizar en el conocimiento ya existente del manejador de bases de datos SQL Server, particularmente al elaborar la forma en la que se almacenan las contraseñas.

La implementación del sistema en Visual Basic .Net propició el aprendizaje de dicho lenguaje así como la adquisición del conocimiento necesario para efectuar la conexión a la base de datos desde esta herramienta, el cifrado de las contraseñas y la generación de archivos pdf desde los programas.

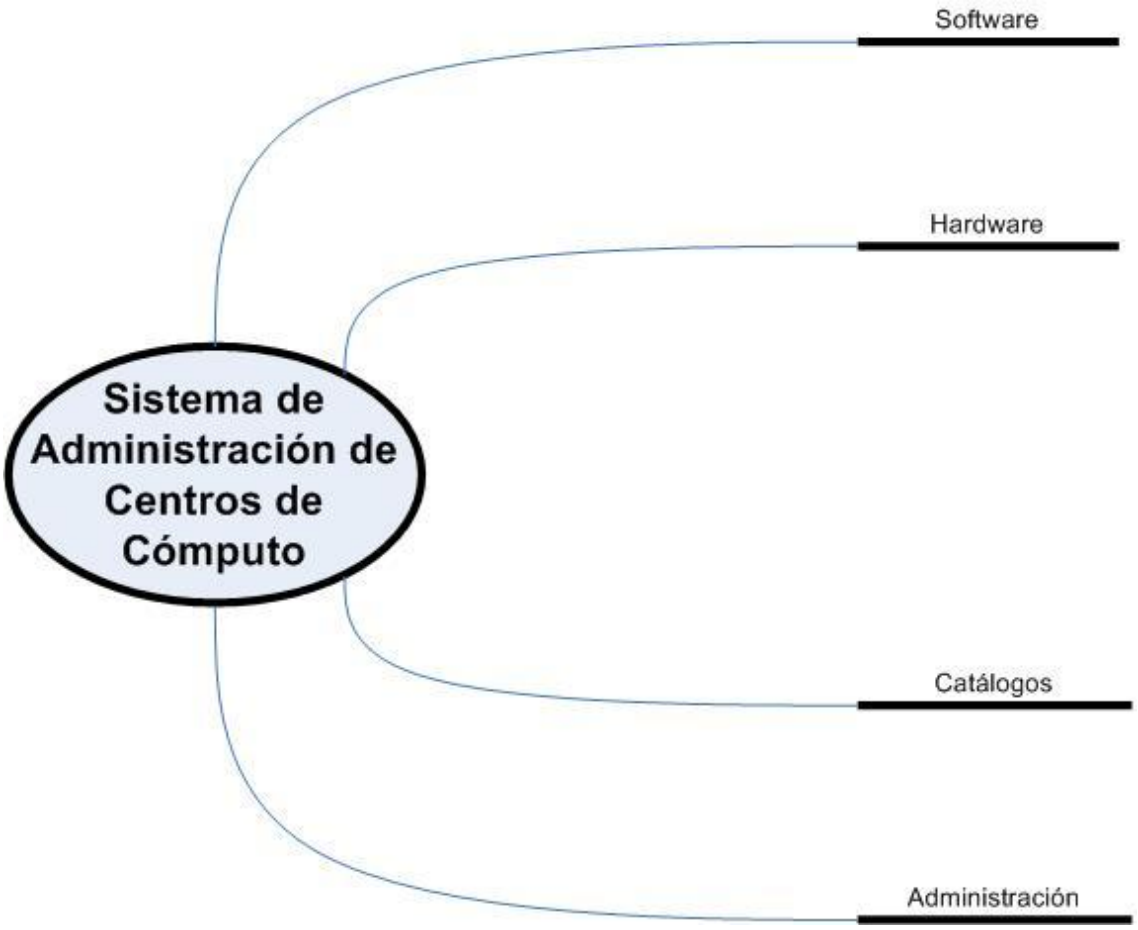
Además de la aplicación de los conocimientos adquiridos y/o profundizados en el Diplomado, en este trabajo se aplicaron conocimientos adquiridos durante el estudio de la carrera de Ingeniería en Computación, ya que las distintas etapas de análisis y diseño del sistema, y la habilidad para la programación del mismo, se

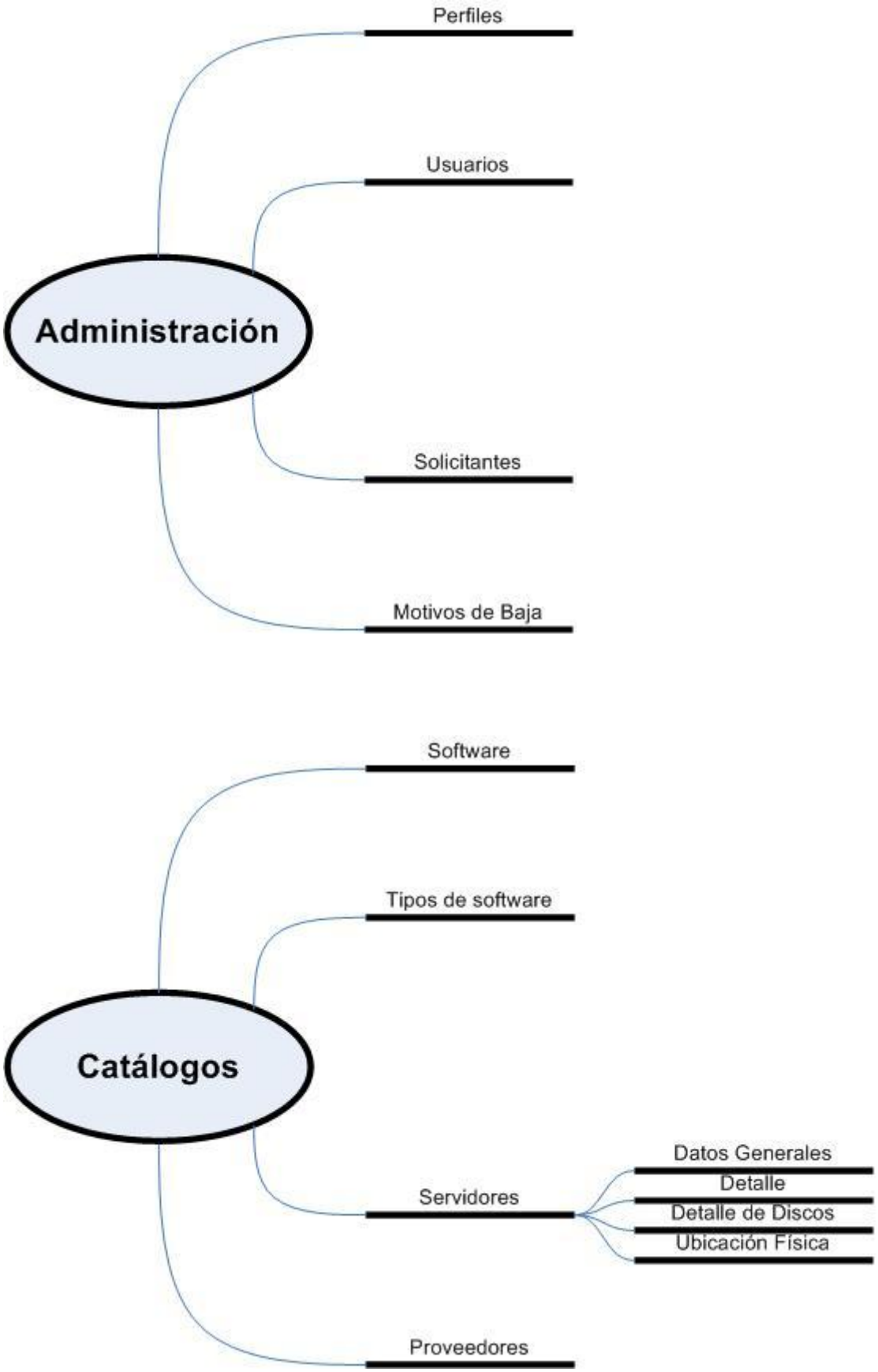
adquieren con la enseñanza de diversas materias impartidas a lo largo de la carrera.

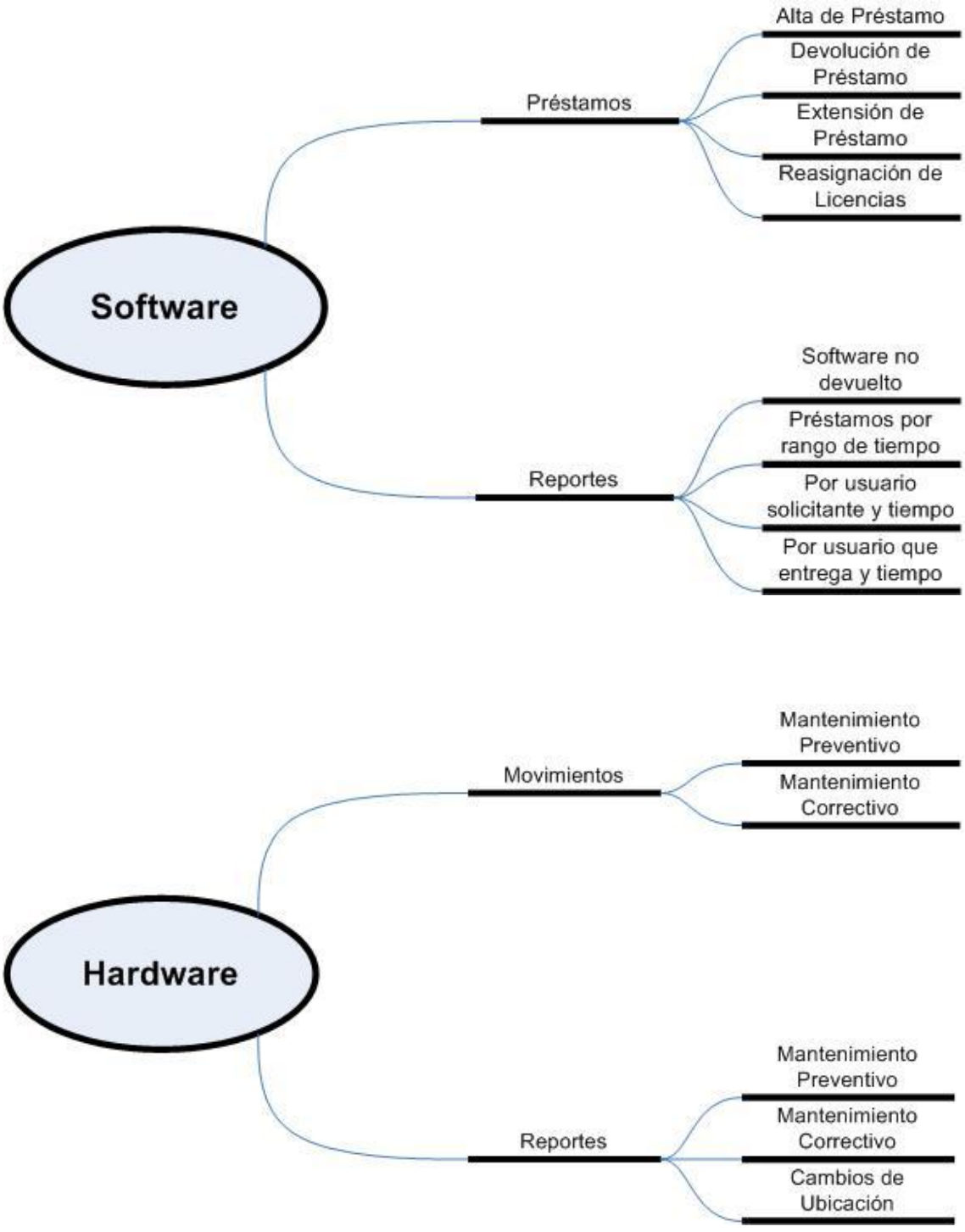
Este trabajo no solamente ayudó a adquirir nuevos conocimientos técnicos, sino que también sivrió para comprender mejor algunos procedimientos internos relacionados al puesto manejado en la Dirección General de Sistemas de la Auditoría Superior de la Federación, lo que también contribuye al desarrollo personal y profesional, esperando en un futuro no solo aplicarlo en esta institución, sino en las siguientes oportunidades laborales que se puedan presentar.

En general los resultados obtenidos en el desarrollo de este sistema son satisfactorios, ya que se obtuvo una herramienta con la que se puede llevar el control tanto del software instalado en los equipos de la institución, como del inventario y mantenimiento de equipos físicos en los centros de cómputo. Sin embargo, debido a que únicamente abarca software y equipos de cómputo, es susceptible de mejora, por ejemplo, agregando inventario y control de equipos como ruteadores o firewalls, por lo que espero que este trabajo sirva de base para que futuras generaciones lo aprovechen y desarrollen un mejor sistema.

Anexo 1. Diagrama conceptual a nivel operacional


Sistema de Administración de Centros de Cómputo
Software
Hardware
Catálogos
Administración


Perfiles
Usuarios
Administración
Solicitantes
Motivos de Baja
Software
Tipos de software
Catálogos
Datos Generales
Detalle
Servidores
Detalle de Discos
Ubicación Física
Proveedores



Diagramas hechos utilizando Microsoft Visio 2003 Professional

Anexo 2. Diagrama de actividades

Software

Área	Subárea	Operación	Actividad	Documentos que genera	Tiempo	Origen	Destino
DGS	DO DD DT	Alta de Préstamo	1. Obtener datos del solicitante 2. Obtener datos del software 3. Verificar disponibilidad de licencias 4. Registro de préstamo	Formato de préstamo, registro de préstamo, y registro de licencia (opcional)	5 min.	DGS	STI
DT	STI	Devolución de Préstamo	1. Obtener datos del préstamo 2. Validar datos del solicitante 3. Validar datos del software 4. Actualizar registro de préstamo	Registro de préstamo	5 min.	STI	DO DD DT

Área	Subárea	Operación	Actividad	Documentos que genera	Tiempo	Origen	Destino
DGS	DO DD DT	Extensión de Préstamo	1. Obtener datos del préstamo 2. Validar datos del solicitante 3. Validar datos del Software 4. Obtener datos de licencias 5. Registrar en histórico de préstamos 6. Actualizar datos de licencias 7. Generar nuevo registro de préstamo	Formato de préstamo, registro de préstamo, y registro de licencia (opcional)	5 min.	DGS	STI

Área	Subárea	Operación	Actividad	Documentos que genera	Tiempo	Origen	Destino
DGS	DO DD DT	Reasignación de Licencias	1. Obtener datos del préstamo 2. Validar datos del solicitante 3. Validar datos del Software 4. Obtener datos de licencias 5. Registrar en histórico de licencias 6. Actualizar datos de licencias	Formato de préstamo, y registro de licencia	5 min.	DGS	STI

Hardware

Área	Subárea	Operación	Actividad	Documentos que genera	Tiempo	Origen	Destino
DGS	DT	Mantenimiento Preventivo	1. Obtener datos del equipo 2. Obtener datos de la empresa 3. Registrar el mantenimiento	Registro de mantenimiento preventivo	Variable	DT	STI
DGS	DT	Mantenimiento Correctivo	1. Obtener datos del equipo 2. Determinar falla 3. Obtener datos de la empresa 4. Registrar solicitud de servicio 5. Cerrar solicitud de servicio	Registro de mantenimiento correctivo	Variable	DT	STI

DGS: Dirección General de Sistemas

DO: Dirección de Operaciones

DD: Dirección de Desarrollo

DT: Dirección de Tecnología

STI: Subdirección de Tecnologías de Información

Anexo 3. Diagrama Entidad – Relación

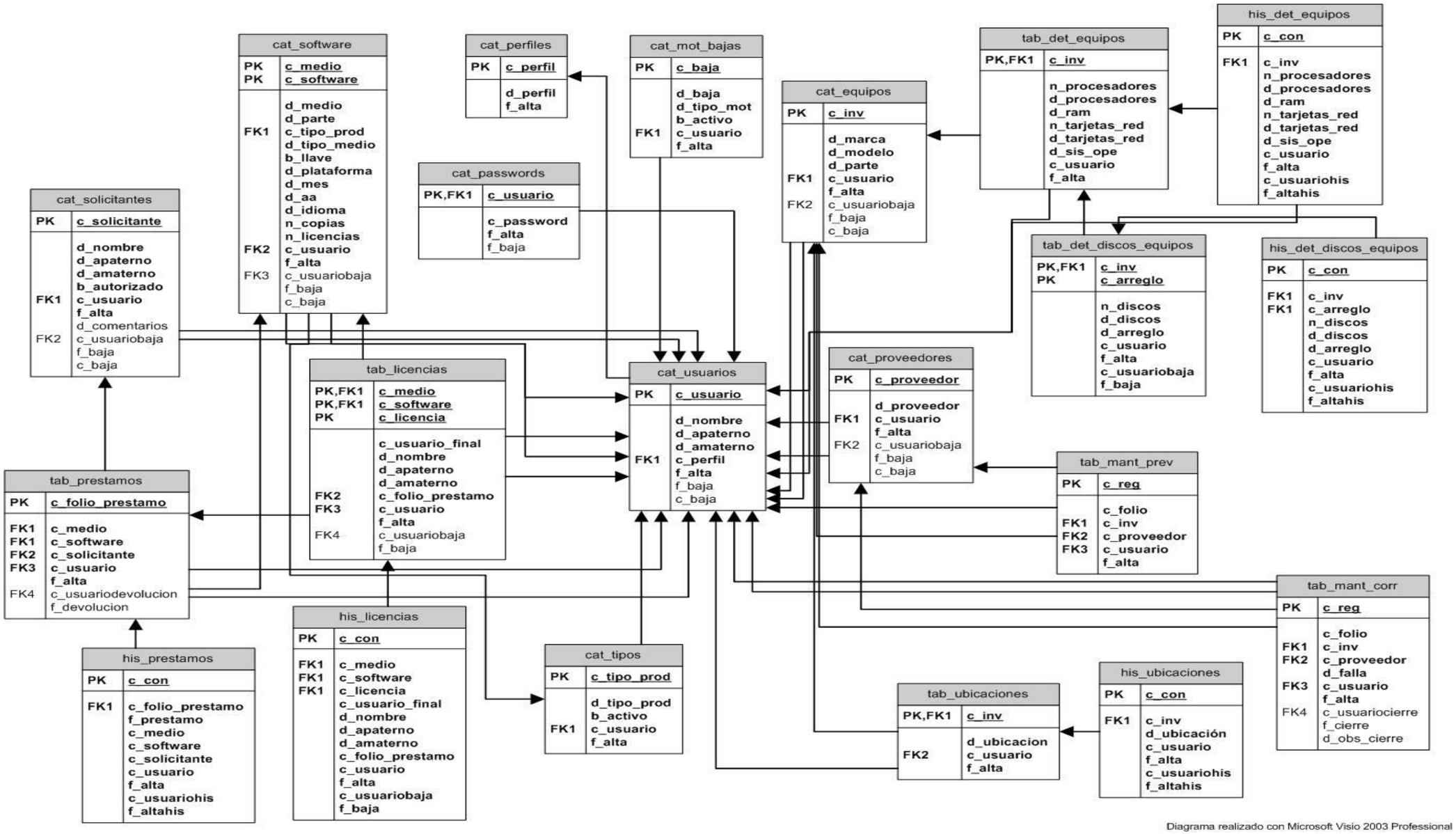


Diagrama realizado con Microsoft Visio 2003 Professional

Anexo 4. Estándares para nombres de objetos de la base de datos

Base de datos

- La base de datos tendrá un nombre asociado a su propósito.
- El nombre lógico del archivo de datos tendrá el mismo nombre que el de la base, añadíendole el sufijo *_Data*.
- El nombre físico del archivo de datos será igual al nombre lógico, añadiendo la extensón *.mdf*. La ubicación física estará en un directorio llamado *\Bdatos\<nombre base de datos>*, y dependerá de las unidades de disco con las que cuente el servidor.
- El tamaño inicial del archivo dependerá de las estimaciones de uso que se obtengan en el análisis del sistema.
- El tamaño máximo del archivo físico será 4096 (4Gb).
- El crecimiento del archivo dependerá de las estimaciones de uso que se obtengan en el análisis, y siempre se especificará en megabytes, no en porcentaje.

Ejemplo:

Si la base de datos se llama *AdminCC*, entonces los parámetros de creación son:

```
NAME = AdminSoftware_Data
FILENAME = 'C:\Bdatos\AdminSoftware\AdminCC_Data.mdf',
SIZE = 10,
MAXSIZE = 4096,
FILEGROWTH = 1
```

Tablas de la aplicación

Las tablas de la aplicación tendrán la siguiente nomenclatura:

<prefijo>_<nombre>

donde <*prefijo*> puede tener los siguientes valores:

- *cat* para tablas de catálogos
- *tab* para tablas de datos
- *his* para tablas de históricos

y <*nombre*> será un nombre descriptivo para la tabla, de ser posible en plural.

Ejemplos:

```
cat_usuarios
tab_prestamos
his_prestamos
```

Nombres de campos

Los campos de las tablas tendrán la siguiente nomenclatura:

<*tipo*>_<*nombre*>

donde <tipo> es una letra que indica el propósito del dato que se va a registrar:

- *c* indica que es una clave o id
- *d* indica que es un texto o descripción
- *n* indica que es un valor numérico
- *f* indica que es una fecha
- *b* indica un estatus de falso o verdadero

y <nombre> será un nombre descriptivo del campo, generalmente en singular.

Ejemplos:

```
c_usuario
d_solicitante
n_licencias
f_alta
b_llave
```

Procedimientos almacenados

Los procedimientos almacenados se nombrarán de la siguiente forma:

sp_<nombre>

donde <nombre> es un identificador que representa el motivo por el que el procedimiento es creado, si se utilizan varias palabras se escribirán sin espacios.

Ejemplo:

```
sp_consultausuarios
```

Anexo 5. Procedimientos almacenados

Consulta de Usuarios

```
CREATE PROCEDURE sp_consultausuarios
   @todos     BIT,
   @activos   BIT,
   @noactivos BIT
AS
DECLARE @cusuario     SMALLINT
DECLARE @dnombre      NVARCHAR(30)
DECLARE @dapaterno    NVARCHAR(40)
DECLARE @damaterno    NVARCHAR(40)
DECLARE @cperfil      CHAR(2)
DECLARE @falta        DATETIME
DECLARE @fbaja        DATETIME
DECLARE @cbaja        CHAR(3)
DECLARE @nomusuario   NVARCHAR(110)
DECLARE @dperfil      NVARCHAR(20)
DECLARE @dbaja        NVARCHAR(100)

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_usuarios')
   DROP TABLE temp_usuarios

CREATE TABLE temp_usuarios
(
   cusuario SMALLINT,
   dnombre  NVARCHAR(110),
   dperfil  NVARCHAR(20),
   falta    DATETIME,
   fbaja    DATETIME,
   dbaja    NVARCHAR(100)
)

DECLARE cur_usuarios CURSOR FOR
   SELECT *
   FROM cat_usuarios

OPEN cur_usuarios
```

```
FETCH NEXT FROM cur_usuarios INTO @cusuario, @dnombre, @dapaterno,
   @damaterno, @cperfil, @falta, @fbaja, @cbaja

WHILE @@FETCH_STATUS = 0
BEGIN
   SET @nomusuario = @dnombre + ' ' + @dapaterno + ' ' + @damaterno
   SELECT @dperfil = (SELECT d_perfil FROM cat_perfiles
                      WHERE c_perfil = @cperfil)
   IF @fbaja IS NULL
      BEGIN
         SET @dbaja = NULL
      END
     ELSE
      BEGIN
         SELECT @dbaja = (SELECT d_baja FROM cat_mot_bajas
                          WHERE c_baja = @cbaja)
      END
   INSERT INTO temp_usuarios VALUES(@cusuario, @nomusuario,
      @dperfil, @falta, @fbaja, @dbaja)
   FETCH NEXT FROM cur_usuarios INTO @cusuario, @dnombre,
   @dapaterno, @damaterno, @cperfil, @falta, @fbaja, @cbaja
END

CLOSE cur_usuarios
DEALLOCATE cur_usuarios

IF @todos = 1
   SELECT * FROM temp_usuarios
      ORDER BY cusuario
IF @activos = 1
   SELECT * FROM temp_usuarios
      WHERE fbaja IS NULL
      ORDER BY cusuario
IF @noactivos = 1
   SELECT * FROM temp_usuarios
      WHERE fbaja IS NOT NULL
      ORDER BY cusuario
```

```
RETURN

GO
```

Consulta de Proveedores

```
CREATE PROCEDURE sp_consultaproveedores
   @todos     BIT,
   @activos   BIT,
   @noactivos BIT
AS
DECLARE @cproveedor   SMALLINT
DECLARE @dproveedor   NVARCHAR(50)
DECLARE @cusuario     SMALLINT
DECLARE @falta        DATETIME
DECLARE @cusuariobaja SMALLINT
DECLARE @fbaja        DATETIME
DECLARE @cbaja        CHAR(3)
DECLARE @nombre       NVARCHAR(30)
DECLARE @apaterno     NVARCHAR(40)
DECLARE @amaterno     NVARCHAR(40)
DECLARE @nomusuario   NVARCHAR(110)
DECLARE @nombaja      NVARCHAR(110)
DECLARE @dbaja        NVARCHAR(100)

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_proveedores')
   DROP TABLE temp_proveedores

CREATE TABLE temp_proveedores
(
   cproveedor   SMALLINT,
   dproveedor   NVARCHAR(50),
   nomusuario   NVARCHAR(110),
   falta        DATETIME,
   nombaja      NVARCHAR(110),
```

```
   fbaja        DATETIME,
   dbaja        NVARCHAR(100),
)

DECLARE cur_proveedores CURSOR FOR
   SELECT *
   FROM cat_proveedores

OPEN cur_proveedores

FETCH NEXT FROM cur_proveedores INTO @cproveedor, @dproveedor,
   @cusuario, @falta, @cusuariobaja, @fbaja, @cbaja

WHILE @@FETCH_STATUS = 0
BEGIN
   SELECT @nombre = (SELECT d_nombre FROM cat_usuarios
                     WHERE c_usuario = @cusuario)
   SELECT @apaterno = (SELECT d_apaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SELECT @amaterno = (SELECT d_amaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SET @nomusuario = @nombre + ' ' + @apaterno + ' ' + @amaterno
   IF @cusuariobaja IS NULL
      BEGIN
         SET @nombaja = NULL
         SET @dbaja = NULL
      END
     ELSE
      BEGIN
         SELECT @nombre = (SELECT d_nombre FROM cat_usuarios
                           WHERE c_usuario = @cusuariobaja)
         SELECT @apaterno = (SELECT d_apaterno FROM cat_usuarios
                             WHERE c_usuario = @cusuariobaja)
         SELECT @amaterno = (SELECT d_amaterno FROM cat_usuarios
                             WHERE c_usuario = @cusuariobaja)
         SET @nombaja = @nombre + ' ' + @apaterno + ' ' + @amaterno
         SELECT @dbaja = (SELECT d_baja FROM cat_mot_bajas
                          WHERE c_baja = @cbaja)
```

```
      END
   INSERT INTO temp_proveedores VALUES(@cproveedor, @dproveedor,
      @nomusuario, @falta, @nombaja, @fbaja, @dbaja)
   FETCH NEXT FROM cur_proveedores INTO @cproveedor, @dproveedor,
      @cusuario, @falta, @cusuariobaja, @fbaja, @cbaja
END

CLOSE cur_proveedores
DEALLOCATE cur_proveedores

IF @todos = 1
   SELECT * FROM temp_proveedores
      ORDER BY cproveedor
IF @activos = 1
   SELECT * FROM temp_proveedores
      WHERE fbaja IS NULL
      ORDER BY cproveedor
IF @noactivos = 1
   SELECT * FROM temp_proveedores
      WHERE fbaja IS NOT NULL
      ORDER BY cproveedor

RETURN

GO
```

Consulta de Solicitantes

```
CREATE PROCEDURE sp_consultasolicitantes
   @todos       BIT,
   @autorizados BIT,
   @activos     BIT,
   @noactivos   BIT
AS
DECLARE @csolicitante SMALLINT
DECLARE @dnombre      NVARCHAR(30)
DECLARE @dapaterno    NVARCHAR(40)
```

```
DECLARE @damaterno    NVARCHAR(40)
DECLARE @bautorizado  BIT
DECLARE @cusuario     SMALLINT
DECLARE @falta        DATETIME
DECLARE @dcomentarios NVARCHAR(200)
DECLARE @cusuariobaja SMALLINT
DECLARE @fbaja        DATETIME
DECLARE @cbaja        CHAR(3)
DECLARE @dbaja        NVARCHAR(100)
DECLARE @nombre       NVARCHAR(30)
DECLARE @apaterno     NVARCHAR(40)
DECLARE @amaterno     NVARCHAR(40)
DECLARE @nomsoli      NVARCHAR(110)
DECLARE @nomusuario   NVARCHAR(110)
DECLARE @nombaja      NVARCHAR(110)

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_solicitantes')
   DROP TABLE temp_solicitantes

CREATE TABLE temp_solicitantes
(
   csolicitante SMALLINT,
   nomsoli      NVARCHAR(110),
   bautorizado  BIT,
   nomusuario   NVARCHAR(110),
   falta        DATETIME,
   dcomentarios NVARCHAR(200),
   nombaja      NVARCHAR(110),
   fbaja        DATETIME,
   dbaja        NVARCHAR(100),
)

DECLARE cur_solicitantes CURSOR FOR
   SELECT *
   FROM cat_solicitantes
   ORDER BY c_solicitante
```

```
OPEN cur_solicitantes

FETCH NEXT FROM cur_solicitantes INTO @csolicitante, @dnombre,
   @dapaterno, @damaterno, @bautorizado, @cusuario, @falta,
   @dcomentarios, @cusuariobaja, @fbaja, @cbaja

WHILE @@FETCH_STATUS = 0
BEGIN
   SET @nomsoli = @dnombre + ' ' + @dapaterno + ' ' + @damaterno
   SELECT @nombre = (SELECT d_nombre FROM cat_usuarios
                     WHERE c_usuario = @cusuario)
   SELECT @apaterno = (SELECT d_apaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SELECT @amaterno = (SELECT d_amaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SET @nomusuario = @nombre + ' ' + @apaterno + ' ' + @amaterno
   IF @cusuariobaja IS NULL
      BEGIN
         SET @nombaja = NULL
         SET @dbaja = NULL
      END
     ELSE
      BEGIN
         SELECT @nombre = (SELECT d_nombre FROM cat_usuarios
                           WHERE c_usuario = @cusuariobaja)
         SELECT @apaterno = (SELECT d_apaterno FROM cat_usuarios
                             WHERE c_usuario = @cusuariobaja)
         SELECT @amaterno = (SELECT d_amaterno FROM cat_usuarios
                             WHERE c_usuario = @cusuariobaja)
         SET @nombaja = @nombre + ' ' + @apaterno + ' ' + @amaterno
         SELECT @dbaja = (SELECT d_baja FROM cat_mot_bajas
                          WHERE c_baja = @cbaja)
      END
   INSERT INTO temp_solicitantes VALUES(@csolicitante, @nomsoli,
      @bautorizado, @nomusuario, @falta, @dcomentarios, @nombaja,
      @fbaja, @dbaja)

   FETCH NEXT FROM cur_solicitantes INTO @csolicitante, @dnombre,
```

```
      @dapaterno, @damaterno, @bautorizado, @cusuario, @falta,
      @dcomentarios, @cusuariobaja, @fbaja, @cbaja
END

CLOSE cur_solicitantes
DEALLOCATE cur_solicitantes

IF @todos = 1
   BEGIN
      SELECT * FROM temp_solicitantes
         ORDER BY csolicitante
   END
  ELSE
   BEGIN
      IF @autorizados = 1
         BEGIN
            If (@activos = 0) AND (@noactivos = 0)
               BEGIN
                  SELECT * FROM temp_solicitantes
                     WHERE bautorizado = 1
                     ORDER BY csolicitante
               END
              ELSE
               IF @activos = 1
                  BEGIN
                     SELECT * FROM temp_solicitantes
                        WHERE bautorizado = 1
                          AND fbaja IS NULL
                        ORDER BY csolicitante
                  END
                ELSE
                  BEGIN
                     SELECT * FROM temp_solicitantes
                        WHERE bautorizado = 1
                          AND fbaja IS NOT NULL
                        ORDER BY csolicitante
                  END
         END
```

```
        ELSE
         BEGIN
            If (@activos = 0) AND (@noactivos = 0)
               BEGIN
                  SELECT * FROM temp_solicitantes
                     WHERE bautorizado = 0
                     ORDER BY csolicitante
               END
              Else
               BEGIN
                  If @activos = 1
                     BEGIN
                        SELECT * FROM temp_solicitantes
                           WHERE bautorizado = 0
                             AND fbaja IS NULL
                           ORDER BY csolicitante
                     END
                    Else
                     BEGIN
                        SELECT * FROM temp_solicitantes
                           WHERE bautorizado = 0
                             AND fbaja IS NOT NULL
                           ORDER BY csolicitante
                     END
               END
         END
   END

RETURN

GO
```

Consulta de equipos

```
CREATE PROCEDURE sp_consultaequipos
   @todos     BIT,
   @activos   BIT,
```

```
   @noactivos BIT
AS
DECLARE @cinv         SMALLINT
DECLARE @dmarca       NVARCHAR(20)
DECLARE @dmodelo      NVARCHAR(40)
DECLARE @dparte       NVARCHAR(40)
DECLARE @daa          SMALLINT
DECLARE @cusuario     SMALLINT
DECLARE @falta        DATETIME
DECLARE @cusuariobaja SMALLINT
DECLARE @fbaja        DATETIME
DECLARE @cbaja        NVARCHAR(3)
DECLARE @nombre       NVARCHAR(30)
DECLARE @apaterno     NVARCHAR(40)
DECLARE @amaterno     NVARCHAR(40)
DECLARE @nomusu       NVARCHAR(110)
DECLARE @nombaja      NVARCHAR(110)
DECLARE @dbaja        NVARCHAR(100)

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_equipos')
   DROP TABLE temp_equipos

CREATE TABLE temp_equipos
(
   cinv    SMALLINT,
   dmarca  NVARCHAR(20),
   dmodelo NVARCHAR(40),
   dparte  NVARCHAR(40),
   daa     SMALLINT,
   nomusu  NVARCHAR(110),
   falta   DATETIME,
   nombaja NVARCHAR(110),
   fbaja   DATETIME,
   dbaja   NVARCHAR(100)
)

DECLARE cur_equipos CURSOR FOR
```

```
   SELECT *
   FROM cat_equipos

OPEN cur_equipos

FETCH NEXT FROM cur_equipos INTO @cinv, @dmarca, @dmodelo,
   @dparte, @daa, @cusuario, @falta, @cusuariobaja, @fbaja, @cbaja

WHILE @@FETCH_STATUS = 0
BEGIN
   SELECT @nombre = (SELECT d_nombre FROM cat_usuarios
                     WHERE c_usuario = @cusuario)
   SELECT @apaterno = (SELECT d_apaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SELECT @amaterno = (SELECT d_amaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SET @nomusu = @nombre + ' ' + @apaterno + ' ' + @amaterno
   IF @cusuariobaja IS NULL
      BEGIN
         SET @nombaja = NULL
         SET @dbaja = NULL
      END
     ELSE
      BEGIN
         SELECT @nombre = (SELECT d_nombre FROM cat_usuarios
                           WHERE c_usuario = @cusuariobaja)
         SELECT @apaterno = (SELECT d_apaterno FROM cat_usuarios
                             WHERE c_usuario = @cusuariobaja)
         SELECT @amaterno = (SELECT d_amaterno FROM cat_usuarios
                             WHERE c_usuario = @cusuariobaja)
         SET @nombaja = @nombre + ' ' + @apaterno + ' ' + @amaterno
         SELECT @dbaja = (SELECT d_baja FROM cat_mot_bajas
                          WHERE c_baja = @cbaja)
      END
   INSERT INTO temp_equipos VALUES(@cinv, @dmarca, @dmodelo,
      @dparte, @daa, @nomusu, @falta, @nombaja, @fbaja,
      @dbaja)
   FETCH NEXT FROM cur_equipos INTO @cinv, @dmarca, @dmodelo,
```

```
      @dparte, @daa, @cusuario, @falta, @cusuariobaja, @fbaja, @cbaja
END

CLOSE cur_equipos
DEALLOCATE cur_equipos

IF @todos = 1
   SELECT * FROM temp_equipos
      ORDER BY cinv
IF @activos = 1
   SELECT * FROM temp_equipos
      WHERE fbaja IS NULL
      ORDER BY cinv
IF @noactivos = 1
   SELECT * FROM temp_equipos
      WHERE fbaja IS NOT NULL
      ORDER BY cinv

RETURN

GO
```

Reporte de software no devuelto

```
CREATE PROCEDURE sp_repnodevuelto
AS
DECLARE @cfolioprestamo SMALLINT
DECLARE @cmedio         SMALLINT
DECLARE @csoftware      SMALLINT
DECLARE @csolicitante   SMALLINT
DECLARE @cusuario       SMALLINT
DECLARE @falta          DATETIME
DECLARE @dmedio         NVARCHAR(200)
DECLARE @dsolicitante   NVARCHAR(110)
DECLARE @dusuario       NVARCHAR(110)
DECLARE @nombre         NVARCHAR(30)
DECLARE @apaterno       NVARCHAR(40)
```

```
DECLARE @amaterno       NVARCHAR(40)

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND
TABLE_NAME='temp_software_no_devuelto')
   DROP TABLE temp_software_no_devuelto

CREATE TABLE temp_software_no_devuelto
(
   cfolio       SMALLINT,
   cmedio       SMALLINT,
   dmedio       NVARCHAR(200),
   dsolicitante NVARCHAR(110),
   dusuario     NVARCHAR(110),
   falta        DATETIME
)

DECLARE cur_software_no_devuelto CURSOR FOR
   SELECT c_folio_prestamo, c_medio, c_software, c_solicitante,
      c_usuario, f_alta
   FROM tab_prestamos
   WHERE f_devolucion IS NULL
   ORDER BY c_folio_prestamo

OPEN cur_software_no_devuelto

FETCH NEXT FROM cur_software_no_devuelto INTO @cfolioprestamo, @cmedio,
   @csoftware, @csolicitante, @cusuario, @falta

WHILE @@FETCH_STATUS = 0
BEGIN
   SELECT @dmedio = (SELECT d_medio FROM cat_software
                     WHERE c_medio = @cmedio
                       AND c_software = @csoftware)
   SELECT @nombre = (SELECT d_nombre FROM cat_solicitantes
                     WHERE c_solicitante = @csolicitante)
   SELECT @apaterno = (SELECT d_apaterno FROM cat_solicitantes
                       WHERE c_solicitante = @csolicitante)
```

```
   SELECT @amaterno = (SELECT d_amaterno FROM cat_solicitantes
                       WHERE c_solicitante = @csolicitante)
   SET @dsolicitante = @nombre + ' ' + @apaterno + ' ' + @amaterno
   SELECT @nombre = (SELECT d_nombre FROM cat_usuarios
                     WHERE c_usuario = @cusuario)
   SELECT @apaterno = (SELECT d_apaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SELECT @amaterno = (SELECT d_amaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SET @dusuario = @nombre + ' ' + @apaterno + ' ' + @amaterno
   INSERT INTO temp_software_no_devuelto VALUES(@cfolioprestamo, @cmedio,
      @dmedio, @dsolicitante, @dusuario, @falta)
   FETCH NEXT FROM cur_software_no_devuelto INTO @cfolioprestamo,
@cmedio,
   @csoftware, @csolicitante, @cusuario, @falta
END

CLOSE cur_software_no_devuelto
DEALLOCATE cur_software_no_devuelto

SELECT * FROM temp_software_no_devuelto

RETURN

GO
```

Reporte de registros de software en un rango de tiempo

```
CREATE PROCEDURE sp_repsoftware
   @finicial DATETIME,
   @ffinal   DATETIME
AS
DECLARE @cfolioprestamo SMALLINT
DECLARE @cmedio         SMALLINT
DECLARE @csoftware      SMALLINT
DECLARE @csolicitante   SMALLINT
DECLARE @cusuario       SMALLINT
```

```
DECLARE @falta          DATETIME
DECLARE @destatus       NVARCHAR(15)
DECLARE @fdevolucion    DATETIME
DECLARE @dmedio         NVARCHAR(200)
DECLARE @dsolicitante   NVARCHAR(110)
DECLARE @dusuario       NVARCHAR(110)
DECLARE @nombre         NVARCHAR(30)
DECLARE @apaterno       NVARCHAR(40)
DECLARE @amaterno       NVARCHAR(40)

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_software')
   DROP TABLE temp_software

CREATE TABLE temp_software
(
   cfolio       SMALLINT,
   cmedio       SMALLINT,
   dmedio       NVARCHAR(200),
   dsolicitante NVARCHAR(110),
   dusuario     NVARCHAR(110),
   falta        DATETIME,
   destatus     NVARCHAR(15)
)

DECLARE cur_software CURSOR FOR
   SELECT c_folio_prestamo, c_medio, c_software, c_solicitante,
      c_usuario, f_alta, f_devolucion
   FROM tab_prestamos
   WHERE f_alta BETWEEN @finicial AND (@ffinal + 1)
   ORDER BY c_folio_prestamo

OPEN cur_software

FETCH NEXT FROM cur_software INTO @cfolioprestamo, @cmedio,
   @csoftware, @csolicitante, @cusuario, @falta, @fdevolucion

WHILE @@FETCH_STATUS = 0
```

```
BEGIN
   SELECT @dmedio = (SELECT d_medio FROM cat_software
                     WHERE c_medio = @cmedio
                       AND c_software = @csoftware)
   SELECT @nombre = (SELECT d_nombre FROM cat_solicitantes
                     WHERE c_solicitante = @csolicitante)
   SELECT @apaterno = (SELECT d_apaterno FROM cat_solicitantes
                       WHERE c_solicitante = @csolicitante)
   SELECT @amaterno = (SELECT d_amaterno FROM cat_solicitantes
                       WHERE c_solicitante = @csolicitante)
   SET @dsolicitante = @nombre + ' ' + @apaterno + ' ' + @amaterno
   SELECT @nombre = (SELECT d_nombre FROM cat_usuarios
                     WHERE c_usuario = @cusuario)
   SELECT @apaterno = (SELECT d_apaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SELECT @amaterno = (SELECT d_amaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SET @dusuario = @nombre + ' ' + @apaterno + ' ' + @amaterno
   IF @fdevolucion IS NULL
      BEGIN
         SET @destatus = 'No entregado'
      END
     ELSE
      BEGIN
         SET @destatus = 'Entregado'
      END
   INSERT INTO temp_software VALUES(@cfolioprestamo, @cmedio,
      @dmedio, @dsolicitante, @dusuario, @falta, @destatus)
   FETCH NEXT FROM cur_software INTO @cfolioprestamo, @cmedio,
   @csoftware, @csolicitante, @cusuario, @falta, @fdevolucion
END

CLOSE cur_software
DEALLOCATE cur_software

SELECT * FROM temp_software

RETURN
```

```
GO
```

Reporte de préstamos por solicitante

```
CREATE PROCEDURE sp_repsolicitante
   @csolicitante SMALLINT,
   @finicial     DATETIME,
   @ffinal       DATETIME
AS
DECLARE @cfolioprestamo SMALLINT
DECLARE @cmedio         SMALLINT
DECLARE @csoftware      SMALLINT
DECLARE @cusuario       SMALLINT
DECLARE @falta          DATETIME
DECLARE @destatus       NVARCHAR(15)
DECLARE @fdevolucion    DATETIME
DECLARE @dmedio         NVARCHAR(200)
DECLARE @dusuario       NVARCHAR(110)
DECLARE @nombre         NVARCHAR(30)
DECLARE @apaterno       NVARCHAR(40)
DECLARE @amaterno       NVARCHAR(40)

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_rep_solicitantes')
   DROP TABLE temp_rep_solicitantes

CREATE TABLE temp_rep_solicitantes
(
   cfolio       SMALLINT,
   cmedio       SMALLINT,
   dmedio       NVARCHAR(200),
   dusuario     NVARCHAR(110),
   falta        DATETIME,
   destatus     NVARCHAR(15)
)
```

```
DECLARE cur_rep_solicitantes CURSOR FOR
   SELECT c_folio_prestamo, c_medio, c_software,
      c_usuario, f_alta, f_devolucion
   FROM tab_prestamos
   WHERE c_solicitante = @csolicitante
     AND f_alta BETWEEN @finicial AND (@ffinal + 1)
   ORDER BY c_folio_prestamo

OPEN cur_rep_solicitantes

FETCH NEXT FROM cur_rep_solicitantes INTO @cfolioprestamo, @cmedio,
   @csoftware, @cusuario, @falta, @fdevolucion

WHILE @@FETCH_STATUS = 0
BEGIN
   SELECT @dmedio = (SELECT d_medio FROM cat_software
                     WHERE c_medio = @cmedio
                       AND c_software = @csoftware)
   SELECT @nombre = (SELECT d_nombre FROM cat_usuarios
                     WHERE c_usuario = @cusuario)
   SELECT @apaterno = (SELECT d_apaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SELECT @amaterno = (SELECT d_amaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SET @dusuario = @nombre + ' ' + @apaterno + ' ' + @amaterno
   IF @fdevolucion IS NULL
      BEGIN
         SET @destatus = 'No entregado'
      END
     ELSE
      BEGIN
         SET @destatus = 'Entregado'
      END
   INSERT INTO temp_rep_solicitantes VALUES(@cfolioprestamo, @cmedio,
      @dmedio, @dusuario, @falta, @destatus)
   FETCH NEXT FROM cur_rep_solicitantes INTO @cfolioprestamo, @cmedio,
   @csoftware, @cusuario, @falta, @fdevolucion
END
```

```
CLOSE cur_rep_solicitantes
DEALLOCATE cur_rep_solicitantes

SELECT * FROM temp_rep_solicitantes

RETURN

GO
```

Reporte de préstamos por usuario

```
CREATE PROCEDURE sp_repusuario
   @cusuario SMALLINT,
   @finicial DATETIME,
   @ffinal   DATETIME
AS
DECLARE @cfolioprestamo SMALLINT
DECLARE @cmedio         SMALLINT
DECLARE @csoftware      SMALLINT
DECLARE @csolicitante   SMALLINT
DECLARE @falta          DATETIME
DECLARE @destatus       NVARCHAR(15)
DECLARE @fdevolucion    DATETIME
DECLARE @dmedio         NVARCHAR(200)
DECLARE @dsolicitante   NVARCHAR(110)
DECLARE @nombre         NVARCHAR(30)
DECLARE @apaterno       NVARCHAR(40)
DECLARE @amaterno       NVARCHAR(40)

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_rep_usuarios')
   DROP TABLE temp_rep_usuarios

CREATE TABLE temp_rep_usuarios
(
   cfolio       SMALLINT,
```

```
   cmedio       SMALLINT,
   dmedio       NVARCHAR(200),
   dsolicitante NVARCHAR(110),
   falta        DATETIME,
   destatus     NVARCHAR(15)
)

DECLARE cur_rep_usuarios CURSOR FOR
   SELECT c_folio_prestamo, c_medio, c_software,
      c_solicitante, f_alta, f_devolucion
   FROM tab_prestamos
   WHERE c_usuario = @cusuario
     AND f_alta BETWEEN @finicial AND (@ffinal + 1)
   ORDER BY c_folio_prestamo

OPEN cur_rep_usuarios

FETCH NEXT FROM cur_rep_usuarios INTO @cfolioprestamo, @cmedio,
   @csoftware, @csolicitante, @falta, @fdevolucion

WHILE @@FETCH_STATUS = 0
BEGIN
   SELECT @dmedio = (SELECT d_medio FROM cat_software
                     WHERE c_medio = @cmedio
                       AND c_software = @csoftware)
   SELECT @nombre = (SELECT d_nombre FROM cat_solicitantes
                     WHERE c_solicitante = @csolicitante)
   SELECT @apaterno = (SELECT d_apaterno FROM cat_solicitantes
                       WHERE c_solicitante = @csolicitante)
   SELECT @amaterno = (SELECT d_amaterno FROM cat_solicitantes
                       WHERE c_solicitante = @csolicitante)
   SET @dsolicitante = @nombre + ' ' + @apaterno + ' ' + @amaterno
   IF @fdevolucion IS NULL
      BEGIN
         SET @destatus = 'No entregado'
      END
     ELSE
      BEGIN
```

```
         SET @destatus = 'Entregado'
      END
   INSERT INTO temp_rep_usuarios VALUES(@cfolioprestamo, @cmedio,
      @dmedio, @dsolicitante, @falta, @destatus)
   FETCH NEXT FROM cur_rep_usuarios INTO @cfolioprestamo, @cmedio,
   @csoftware, @csolicitante, @falta, @fdevolucion
END

CLOSE cur_rep_usuarios
DEALLOCATE cur_rep_usuarios

SELECT * FROM temp_rep_usuarios

RETURN

GO
```

Reporte de Mantenimiento Preventivo

```
CREATE PROCEDURE sp_repmantprev
   @finicial DATETIME,
   @ffinal   DATETIME
AS
DECLARE @cfolio     SMALLINT
DECLARE @cinv       SMALLINT
DECLARE @dmarca     NVARCHAR(20)
DECLARE @dmodelo    NVARCHAR(40)
DECLARE @dparte     NVARCHAR(40)
DECLARE @dproveedor NVARCHAR(50)
DECLARE @dusuario   NVARCHAR(110)
DECLARE @falta      DATETIME
DECLARE @cproveedor SMALLINT
DECLARE @cusuario   SMALLINT
DECLARE @nombre     NVARCHAR(30)
DECLARE @apaterno   NVARCHAR(40)
DECLARE @amaterno   NVARCHAR(40)
```

```
IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_rep_mant_prev')
   DROP TABLE temp_rep_mant_prev

CREATE TABLE temp_rep_mant_prev
(
   cfolio       SMALLINT,
   cinv         SMALLINT,
   dmarca       NVARCHAR(20),
   dmodelo      NVARCHAR(40),
   dparte       NVARCHAR(40),
   dproveedor   NVARCHAR(50),
   dusuario     NVARCHAR(110),
   falta        DATETIME
)

DECLARE cur_rep_mant_prev CURSOR FOR
   SELECT c_folio, c_inv, c_proveedor, c_usuario, f_alta
   FROM tab_mant_prev
   WHERE f_alta BETWEEN @finicial AND (@ffinal + 1)
   ORDER BY f_alta

OPEN cur_rep_mant_prev

FETCH NEXT FROM cur_rep_mant_prev INTO @cfolio, @cinv,
   @cproveedor, @cusuario, @falta

WHILE @@FETCH_STATUS = 0
BEGIN
   SELECT @dmarca = (SELECT d_marca FROM cat_equipos
                     WHERE c_inv = @cinv)
   SELECT @dmodelo = (SELECT d_modelo FROM cat_equipos
                      WHERE c_inv = @cinv)
   SELECT @dparte = (SELECT d_parte FROM cat_equipos
                     WHERE c_inv = @cinv)
   SELECT @dproveedor = (SELECT d_proveedor FROM cat_proveedores
                         WHERE c_proveedor = @cproveedor)
   SELECT @nombre = (SELECT d_nombre FROM cat_usuarios
```

```
                    WHERE c_usuario = @cusuario)
   SELECT @apaterno = (SELECT d_apaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SELECT @amaterno = (SELECT d_amaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SET @dusuario = @nombre + ' ' + @apaterno + ' ' + @amaterno
   INSERT INTO temp_rep_mant_prev VALUES(@cfolio, @cinv,
      @dmarca, @dmodelo, @dparte, @dproveedor, @dusuario, @falta)
   FETCH NEXT FROM cur_rep_mant_prev INTO @cfolio, @cinv,
   @cproveedor, @cusuario, @falta
END

CLOSE cur_rep_mant_prev
DEALLOCATE cur_rep_mant_prev

SELECT * FROM temp_rep_mant_prev

RETURN

GO
```

Reporte de Mantenimiento Correctivo

```
CREATE PROCEDURE sp_repmantcorr
   @finicial DATETIME,
   @ffinal   DATETIME
AS
DECLARE @cfolio     SMALLINT
DECLARE @cinv       SMALLINT
DECLARE @dmarca     NVARCHAR(20)
DECLARE @dmodelo    NVARCHAR(40)
DECLARE @dparte     NVARCHAR(40)
DECLARE @dfalla     NVARCHAR(100)
DECLARE @dproveedor NVARCHAR(50)
DECLARE @dusuario   NVARCHAR(110)
DECLARE @falta      DATETIME
DECLARE @destatus   NVARCHAR(10)
```

```
DECLARE @dobscierre NVARCHAR(100)
DECLARE @fcierre    DATETIME
DECLARE @cproveedor SMALLINT
DECLARE @cusuario   SMALLINT
DECLARE @nombre     NVARCHAR(30)
DECLARE @apaterno   NVARCHAR(40)
DECLARE @amaterno   NVARCHAR(40)

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_rep_mant_corr')
   DROP TABLE temp_rep_mant_corr

CREATE TABLE temp_rep_mant_corr
(
   cfolio       SMALLINT,
   cinv         SMALLINT,
   dmarca       NVARCHAR(20),
   dmodelo      NVARCHAR(40),
   dparte       NVARCHAR(40),
   dfalla       NVARCHAR(100),
   dproveedor   NVARCHAR(50),
   dusuario     NVARCHAR(110),
   falta        DATETIME,
   destatus     NVARCHAR(10),
   dobscierre   NVARCHAR(100)
)

DECLARE cur_rep_mant_corr CURSOR FOR
   SELECT c_folio, c_inv, c_proveedor, d_falla, c_usuario, f_alta,
   f_cierre, d_obs_cierre
   FROM tab_mant_corr
   WHERE f_alta BETWEEN @finicial AND (@ffinal + 1)
   ORDER BY f_alta

OPEN cur_rep_mant_corr

FETCH NEXT FROM cur_rep_mant_corr INTO @cfolio, @cinv, @cproveedor,
   @dfalla, @cusuario, @falta, @fcierre, @dobscierre
```

```
WHILE @@FETCH_STATUS = 0
BEGIN
   SELECT @dmarca = (SELECT d_marca FROM cat_equipos
                     WHERE c_inv = @cinv)
   SELECT @dmodelo = (SELECT d_modelo FROM cat_equipos
                      WHERE c_inv = @cinv)
   SELECT @dparte = (SELECT d_parte FROM cat_equipos
                     WHERE c_inv = @cinv)
   SELECT @dproveedor = (SELECT d_proveedor FROM cat_proveedores
                         WHERE c_proveedor = @cproveedor)
   SELECT @nombre = (SELECT d_nombre FROM cat_usuarios
                     WHERE c_usuario = @cusuario)
   SELECT @apaterno = (SELECT d_apaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SELECT @amaterno = (SELECT d_amaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SET @dusuario = @nombre + ' ' + @apaterno + ' ' + @amaterno
   IF @fcierre IS NULL
      BEGIN
         SET @destatus = 'Abierto'
      END
     ELSE
      BEGIN
         SET @destatus = 'Cerrado'
      END
   INSERT INTO temp_rep_mant_corr VALUES(@cfolio, @cinv, @dmarca,
      @dmodelo, @dparte, @dfalla, @dproveedor, @dusuario, @falta,
      @destatus, @dobscierre)
   FETCH NEXT FROM cur_rep_mant_corr INTO @cfolio, @cinv, @cproveedor,
   @dfalla, @cusuario, @falta, @fcierre, @dobscierre
END

CLOSE cur_rep_mant_corr
DEALLOCATE cur_rep_mant_corr

SELECT * FROM temp_rep_mant_corr
```

```
RETURN

GO
```

Reporte de Cambios de Ubicación

```
CREATE PROCEDURE sp_repubicaciones
   @finicial DATETIME,
   @ffinal   DATETIME
AS
DECLARE @cinv       SMALLINT
DECLARE @dmarca     NVARCHAR(20)
DECLARE @dmodelo    NVARCHAR(40)
DECLARE @dparte     NVARCHAR(40)
DECLARE @dubicacion NVARCHAR(50)
DECLARE @dusuario   NVARCHAR(110)
DECLARE @falta      DATETIME
DECLARE @cusuario   SMALLINT
DECLARE @nombre     NVARCHAR(30)
DECLARE @apaterno   NVARCHAR(40)
DECLARE @amaterno   NVARCHAR(40)

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_rep_ubicaciones')
   DROP TABLE temp_rep_ubicaciones

CREATE TABLE temp_rep_ubicaciones
(
   cinv         SMALLINT,
   dmarca       NVARCHAR(20),
   dmodelo      NVARCHAR(40),
   dparte       NVARCHAR(40),
   dubicacion   NVARCHAR(50),
   dusuario     NVARCHAR(110),
   falta        DATETIME
)
```

```
DECLARE cur_rep_ubicaciones CURSOR FOR
   SELECT c_inv, d_ubicacion, c_usuario, f_alta
   FROM his_ubicaciones
   WHERE f_alta BETWEEN @finicial AND (@ffinal + 1)
   UNION ALL
   SELECT c_inv, d_ubicacion, c_usuario, f_alta
   FROM tab_ubicaciones
   WHERE f_alta BETWEEN @finicial AND (@ffinal + 1)
   ORDER BY f_alta

OPEN cur_rep_ubicaciones

FETCH NEXT FROM cur_rep_ubicaciones INTO @cinv,
   @dubicacion, @cusuario, @falta

WHILE @@FETCH_STATUS = 0
BEGIN
   SELECT @dmarca = (SELECT d_marca FROM cat_equipos
                     WHERE c_inv = @cinv)
   SELECT @dmodelo = (SELECT d_modelo FROM cat_equipos
                      WHERE c_inv = @cinv)
   SELECT @dparte = (SELECT d_parte FROM cat_equipos
                     WHERE c_inv = @cinv)
   SELECT @nombre = (SELECT d_nombre FROM cat_usuarios
                     WHERE c_usuario = @cusuario)
   SELECT @apaterno = (SELECT d_apaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SELECT @amaterno = (SELECT d_amaterno FROM cat_usuarios
                       WHERE c_usuario = @cusuario)
   SET @dusuario = @nombre + ' ' + @apaterno + ' ' + @amaterno
   INSERT INTO temp_rep_ubicaciones VALUES(@cinv,
      @dmarca, @dmodelo, @dparte, @dubicacion, @dusuario, @falta)
   FETCH NEXT FROM cur_rep_ubicaciones INTO @cinv,
   @dubicacion, @cusuario, @falta
END

CLOSE cur_rep_ubicaciones
DEALLOCATE cur_rep_ubicaciones
```

```
SELECT * FROM temp_rep_ubicaciones

RETURN

GO
```

Eliminación de tablas temporales

```
CREATE PROCEDURE sp_eliminatablastemporales
AS

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_usuarios')
   DROP TABLE temp_usuarios

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_solicitantes')
   DROP TABLE temp_solicitantes

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_proveedores')
   DROP TABLE temp_proveedores

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_equipos')
   DROP TABLE temp_equipos

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND
TABLE_NAME='temp_software_no_devuelto')
   DROP TABLE temp_software_no_devuelto

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_software')
   DROP TABLE temp_software
```

```
IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_rep_solictantes')
   DROP TABLE temp_rep_solicitantes

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_rep_usuarios')
   DROP TABLE temp_rep_usuarios

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_rep_mant_prev')
   DROP TABLE temp_rep_mant_prev

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_rep_mant_corr')
   DROP TABLE temp_rep_mant_corr

IF EXISTS(SELECT 1 FROM INFORMATION_SCHEMA.TABLES
   WHERE TABLE_TYPE='BASE TABLE' AND TABLE_NAME='temp_rep_ubicaciones')
   DROP TABLE temp_rep_ubicaciones

RETURN

GO
```

Anexo 6. Código fuente para cifrar/descifrar cadenas

```
Imports Microsoft.VisualBasic
Imports System
Imports System.IO
Imports System.Text
Imports System.Security.Cryptography

Public Class cls_tripledes
    Private key() As Byte = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, _
    16, 17, 18, 19, 20, 21, 22, 23, 24}
    Private iv() As Byte = {65, 110, 68, 26, 69, 178, 200, 219}

    Public Function Encripta(ByVal Texto As String) As Byte()
        REM Declara un objeto UTF8Encoding para que se utilice
        REM el método GetByte para transformar el texto a un
        REM arreglo Byte
        Dim utf8encoder As UTF8Encoding = New UTF8Encoding()
        Dim introduceBytes() As Byte = utf8encoder.GetBytes(Texto)

        REM Crea un nuevo proveedor de servicio TripleDES
        Dim proveedorTDES As TripleDESCryptoServiceProvider = New
TripleDESCryptoServiceProvider()

        REM La interface ICrypTransform utiliza la encriptación TripleDES
        REM junto con la llave de encriptación y el vector inicial de
información
        Dim transformaCryptografia As ICryptoTransform =
proveedorTDES.CreateEncryptor(Me.key, Me.iv)

        REM Toda función criptográfica necesita un flujo de salida para
la información
        REM encriptada.
        Dim flujoEncriptado As MemoryStream = New MemoryStream()
        Dim flujoCrypt As CryptoStream = New
CryptoStream(flujoEncriptado, _
        transformaCryptografia, CryptoStreamMode.Write)
```

```
        REM Escribe la información encriptada en el flujo. Vacía la
información
        REM cuando se asegura que todo ha salido del buffer
        flujoCrypt.Write(introduceBytes, 0, introduceBytes.Length)
        flujoCrypt.FlushFinalBlock()
        flujoEncriptado.Position = 0

        REM Lee el flujo de vuelta al arreglo Byte y lo devuelve al
        REM método que lo invoca
        Dim resultado(flujoEncriptado.Length - 1) As Byte
        flujoEncriptado.Read(resultado, 0, flujoEncriptado.Length)
        flujoCrypt.Close()
        Return resultado
    End Function

    Public Function Desencripta(ByVal introduceBytes() As Byte) As String
        REM UTFEncoding es utilizado para transformar la información
        REM del arreglo Byte en una cadena
        Dim utf8encoder As UTF8Encoding = New UTF8Encoding
        Dim proveedorTDES As TripleDESCryptoServiceProvider = New
TripleDESCryptoServiceProvider()

        REM Como antes, dbemos proveer la llave de
encriptación/desencriptación
        REM junto con el vector inicial
        Dim transformaCrypto As ICryptoTransform =
proveedorTDES.CreateDecryptor(Me.key, Me.iv)

        REM Se provee de un flujo de memoria para desencriptar la
información
        Dim flujoDesencriptado As MemoryStream = New MemoryStream()
        Dim flujoCrypt As CryptoStream = New
CryptoStream(flujoDesencriptado, _
        transformaCrypto, CryptoStreamMode.Write)
        flujoCrypt.Write(introduceBytes, 0, introduceBytes.Length)
        flujoCrypt.FlushFinalBlock()
        flujoDesencriptado.Position = 0
```

```
        REM Lee el flujo de memoria y lo covierte en una cadena
        Dim resultado(flujoDesencriptado.Length - 1) As Byte
        flujoDesencriptado.Read(resultado, 0, flujoDesencriptado.Length)
        flujoCrypt.Close()
        Dim miUtf As UTF8Encoding = New UTF8Encoding()
        Return miUtf.GetString(resultado)
    End Function
End Class
```

Bibliografía y Referencias

Bibliografía:

[1] *Enciclopedia de Microsoft Visual Basic*, Fco. Javier Ceballos, Editorial Alfaomega Ra-Ma, 2006

[2] Material de apoyo Diplomado *"Diseño de Sistemas de Información orientado a Negocios con SQL Server y Oracle", tercera generación*, Coordinación Académica: Ing. Pedro Gabriel Ramírez Hernández, FES Aragón, 2006

[3] *Microsoft Visual Basic .NET Lenguaje y aplicaciones*, Fco. Javier Ceballos, Editorial Alfaomega Ra-Ma, 2006

Referencias:

[4] *Ingeniería de Software*, http://www.monografias.com/trabajos5/inso/inso2.shtml, 2008

[5] *Modelo de Red*, http://www.angelfire.com/my/jimena/bdat1/guia8.htm, 2008

[6] *SQL Server 2000: Data Transformation Services (DTS)*, http://www.microsoft.com/technet/prodtechnol/sql/2000/deploy/dtssql2k.mspx, 2008

[7] *String Encryption With Visual Basic .Net*, http://www.devarticles.com/c/a/VB.Net/String-Encryption-With-Visual-Basic-.NET/, 2008

[8] *Técnicas Aplicadas para el Desarrollo de Sistemas, capítulo 8.6.2*,
http://www.ongei.gob.pe/publica/metodologias/Lib5081/0300.htm,
2008

[9] *Wikipedia, la enciclopedia libre*,
http://es.wikipedia.org,
2008

[10] *x2PDF – Transalet to PDF*,
http://garvander.com/acecomputing/default.htm,
2008