# Universidad Nacional Autónoma de México

## Facultad de Estudios Superiores Aragón


# “CHECKPOINT: DESCARGADOR DE REGISTROS DE ASISTENCIA DE DISPOSITIVOS BIOMÉTRICOS DE HUELLA DIGITAL PARA EL PROCEDIMIENTO DE ASISTENCIA DE LA COFEPRIS”

## TRABAJO ESCRITO

**EN LA MODALIDA DE DESARROLLO DE UN CASO PRÁCTICO PARA OBTENER EL TÍTULO DE INGENIERO EN COMPUTACIÓN**

**PRESENTA:**

**FRANCISCO SALVADOR HERNÁNDEZ PÉREZ**

**ASESOR: MTRO. JUAN GASTALDI PÉREZ**

**MÉXICO 2015**

## ÍNDICE

## INTRODUCCIÓN

En este documento se presenta el desarrollo de un Sistema de información que tiene la capacidad de descargar registros de asistencia de dispositivos biométricos de huella digital, este sistema forma parte del Proceso de Control de Asistencia de la Comisión Federal para la Protección Contra Riesgos Sanitarios (COFEPRIS).

Se comienza con la descripción organizacional de la COFEPRIS, en el Capítulo I se menciona el procedimiento de control de Asistencia y su sustento documental, se analiza la operación técnica del procedimiento en general y las problemáticas que presenta.

Posteriormente en el Capítulo II se plantea el desarrollo del Sistema CHECKPOINT, se analizan las capas que estructuran el sistema y las herramientas de desarrollo que lo conforman. En el Capítulo III se pone en marcha el sistema y se muestran los resultados obtenidos. Finalmente se concluye los temas con el análisis de los resultados con respecto a los objetivos planteados en el caso del desarrollo, para dar inicio con el desarrollo del caso práctico se plantean los antecedentes de la COFEPRIS.

La COFEPRIS es un órgano desconcentrado de la Secretaria de Salud Pública que tiene la misión proteger a la población mexicana de riesgos a la salud provocados por el uso y consumo de bienes y servicios, insumos para la salud, así como por su exposición a factores ambientales y laborales, la ocurrencia de emergencias sanitarias y prestación de servicios de salud, mediante la regulación, el control y la prevención de riegos de salud[1].

La COFEPRIS fue creada el 5 de Julio 2001, como órgano desconcentrado con facultades de autonomía técnica, administrativa y operativa[2]. El 2 de Julio de 2012

[1] Misión y Visión de la COFEPRIS, Sitio Web COFEORIS http://www.cofepris.gob.mx/cofepris/Paginas/VisionYMision.aspx

[2] Historia de la COFEPRIS, Sitio Web COFEPRIS http://www.cofepris.gob.mx/cofepris/Paginas/Historia.aspx

la COFEPRIS fue reconocida como Autoridad Reguladora Nacional de Referencia Regional de Medicamentos y Productos Biológicos por la Organización Panamericana de la Salud (OPS)[3]. En 2013 COFEPRIS inicio el proceso de Certificación en materia de buenas prácticas en medicamentos y vacunas ante la Organización Mundial de la Salud (OMS)[4].

COFEPRIS está conformada por una plantilla aproximada de 1650 empleados divididos en cuatro órganos y ocho unidades administrativas que trabajan en conjunto para cumplir la misión de esta Comisión Federal.

> *"La Comisión Federal contará con los siguientes órganos y unidades administrativas, para su debida organización y funcionamiento"[5].*

**I. Órganos:**

a. El Consejo Interno.
b. El Consejo Científico.
c. El Consejo Consultivo Mixto.
d. El Consejo Consultivo de la Publicidad.

**II. Unidades administrativas:**

a. Comisión de Evidencia y Manejo de Riesgos.
b. Comisión de Fomento Sanitario.
c. Comisión de Autorización Sanitaria.
d. Comisión de Operación Sanitaria.
e. Comisión de Control Analítico y Ampliación de Cobertura.
f. Coordinación General del Sistema Federal Sanitario.
g. Coordinación General Jurídica y Consultiva.

[3] OPS agencia de salud pública internacional más antigua del mundo. Brinda cooperación técnica y moviliza asociaciones para mejorar la salud y la calidad de vida en los países de las Américas

[4] OMS autoridad directiva y coordinadora de la acción sanitaria en el sistema de las Naciones Unidas.

[5] Reglamento de la Comisión Federal para la Protección de Riesgos Sanitarios, Capítulo II. Integración de la Comisión Federal Art. 4 p. 15

h. Secretaría General.

> *"La Secretaria General Establece, con la aprobación del Comisionado Federal, las políticas, normas, sistemas y procedimientos para la programación, presupuestación y administración integral de los recursos humanos, materiales y financieros de que disponga la Comisión Federal".*[6]

La Secretaría General es la Unidad Administrativa de apoyo a todo el personal de del resto de las áreas operativas de COFEPRIS, en el ámbito de la administración de recursos humanos se encuentra el control de asistencia para el caso de estudio de este caso práctico que en el siguiente capítulo se detalla.

[6] Reglamento de la Comisión Federal para la Protección de Riesgos Sanitarios, Capítulo IV De las Unidades Administrativas de la Comisión Federal, Art. 19, p.50

# CAPITULO I

# "DESCRIPCIÓN Y PROBLEMÁTICA DEL PROCEDIMIENTO DE CONTROL DE ASISTENCIA DE LA COFEPRIS"

## 1.1 Procedimiento para el Registro y Control de Asistencia de la COFEPRIS.

> *"La Dirección Ejecutiva de Recursos Humanos, de la Secretaría General, a través de la Gerencia Ejecutiva de Desarrollo Humano y Servicios al Personal, será la responsable de llevar a cabo el control de asistencias del personal estableciendo los límites de tolerancia de conformidad con la normatividad establecida en las Condiciones Generales de Trabajo vigentes en la Secretaria de Salud"*[7]*.*

Una de las principales tareas que lleva acabo la Secretaría General a través de la de la Dirección Ejecutiva de Recursos Humanos y la Gerencia de Desarrollo Humano y Servicios al Personal es el *"Procedimiento para el Registro y Control de Asistencia del Personal"* establecido en el Manual de Procedimientos de la misma unidad administrativa, el cual tiene alcance a las áreas antes mencionadas y toda la COFEPRIS, este procedimiento tiene como propósito específico controlar la puntualidad y las incidencias del personal adscrito a la COFEPRIS mediante un registro de asistencia adecuado para llevar a cabo las acciones que procedan de acuerdo a la normatividad establecida en las condiciones generales de trabajo vigente en la Secretaría de Salud[8].

[7] Manual de Procedimientos Secretaría General, Procedimiento para Registro y Control de Asistencia de Personal SG-DERH-P-08, Políticas de Operación, normas y lineamientos 3.0

[8] Ibídem, Propósito 1.0

## 1.2 Descripción general del procedimiento de control de asistencia.

El procedimiento de Asistencia de la Dirección Ejecutiva de Recursos Humanos es fundamental para la COFEPRIS, ya que tiene un impacto al personal de todas las unidades administrativas de esta Comisión Federal y es complementario de otra serie de procesos, tales como son, el Procedimiento de Pagos de Nomina, Procedimientos de Estímulos y Recompensas, Procedimiento de Pago de Aguinaldos, por mencionar los más importantes, la estructura del procedimiento se muestra en la imagen 1.1.

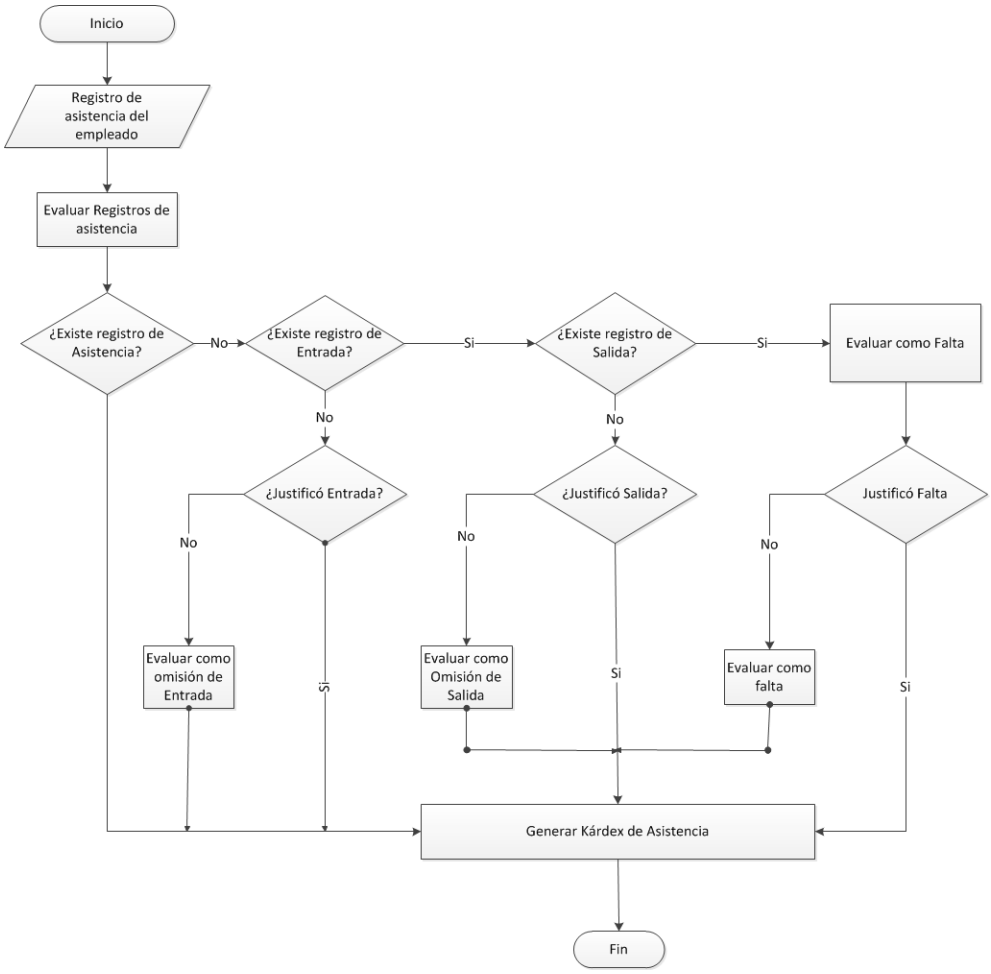



Imagen 1.1 Diagrama de flujo general del procedimiento de Registro y Control de Asistencia

El procedimiento consiste inicialmente en crear un Registro de Asistencia del empleado, se produce una evidencia de la entrada y/o la salida del personal mediante un record de la hora y la fecha del evento, esto es por cada día hábil, es decir, de lunes a viernes excepto días festivos de acuerdo a la jornada laboral de cada empleado, enseguida se determina si está dentro de los parámetros establecidos por el horario del empleado, finalmente se genera el Kárdex de asistencia del empleado, en caso contrario, si no existe algún registro de entrada o de salida se evalúa de la siguiente manera:

a) En el caso de no contar con el registro de entrada se califica como omisión de entrada.
b) Si no se encuentra el registro de salida se evalúa como omisión de salida.

c) Si no se cuenta con ambos se evalúa como falta.

Las tres incidencias mencionas se pueden justificar con algún documento del catálogo de justificación de incidencias de la Dirección Ejecutiva de Recursos Humanos, el empleado tiene determinado tiempo para cubrir la justificación de la incidencia, de lo contrario la evaluación se aplica conforme se mencionó al kárdex de asistencia de cada empleado y se procede con el descuento económico, para finalmente cumplir con el objetivo este proceso.

La Dirección Ejecutiva de Recursos Humanos de la COFEPRIS con el objetivo de cumplir puntualmente con el Procedimiento para el Registro y Control de Asistencia del Personal de manera precisa, ha automatizado este procedimiento, a lo largo de trece años se ha apoyado de Sistemas de Información y dispositivos especializados para el control de asistencia, obteniendo como resultados la reducción de costos económicos, reducción de recursos humanos y la reducción de tiempo de procesamiento, esto se ve reflejado en la precisión de la información generada con la implementación de estas tecnologías.

## 1.3 Estructura y operación del procedimiento de control de asistencia

El procedimiento de control de asistencia se basa en la ejecución de un Sistema de control de asistencia interconectado a dispositivos biométricos, la biometría es la capacidad de reconocer características propios de cada persona tales como los rasgos faciales, las huellas digitales, la voz, la iris entre otros. Es por ello se define lo siguiente.

> *"La biometría es el estudio de métodos automáticos para el reconocimiento único de rasgos humanos basados en uno o más rasgos conductuales o físicos intrínsecos"*[9].

Los dispositivos biométricos se utilizan para el procedimiento de control de asistencia tienen la capacidad de reconocer al personal mediante la huella digital, este dispositivo genera el registro de asistencia. En la Imagen 1.2 se observa el inicio del procedimiento con la captura de Huella Digital del empleado, se realiza la validación de la misma, si es válido se genera un registro con los datos de la persona tales como el número del empleado, la fecha y la hora en la que se realizó la operación, este conjunto de tareas se produce en el dispositivo biométrico, a continuación se descargan los registros recolectados por el dispositivo mediante una aplicación y se almacenan en una base de datos, a continuación la aplicación envía los registros a la base de datos principal del sistema de control de asistencia, posteriormente se genera el kárdex de cada empleado y finalmente se publica a los empleados para su consulta y en su caso realizar las justificaciones necesarias.

[9] Fundamentos de Biometría, UNAM Facultad de Ingeniería Biometría informática http://redyseguridad.fip.unam.mx/proyectos/biometria/

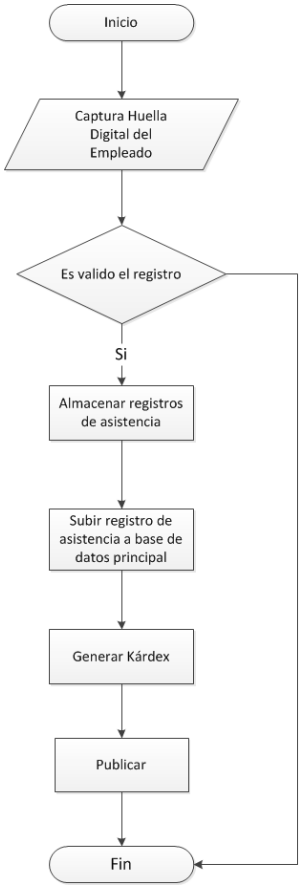



Imagen 1.2 Diagrama de flujo de la operación del procedimiento de Registro y Control de Asistencia

El sistema de control de asistencia se estructura por una seria de operaciones relacionadas entre sí, en algunos caso estas operaciones depende de otra para poder ejecutarse. La imagen 1.3 muestra las operaciones del Sistema, la interacción entre los diferentes casos y la dependencia que existe en cada uno de ellos con otras operaciones para lograr satisfacer los requerimientos del procedimiento de control de asistencia. El método de entrada del sistema es la captura de huella digital, es esencial que el caso de almacenamiento del registro de asistencia exista, ya que de no ser incluido se perderían datos, posteriormente se extiende la operación para subir el registro a la base de datos principal que está incluido para el caso del procesamiento del registro de asistencia, después se incluye para la generación de kárdex y finalmente es incluido para la publicación de la asistencia en el sitio web.

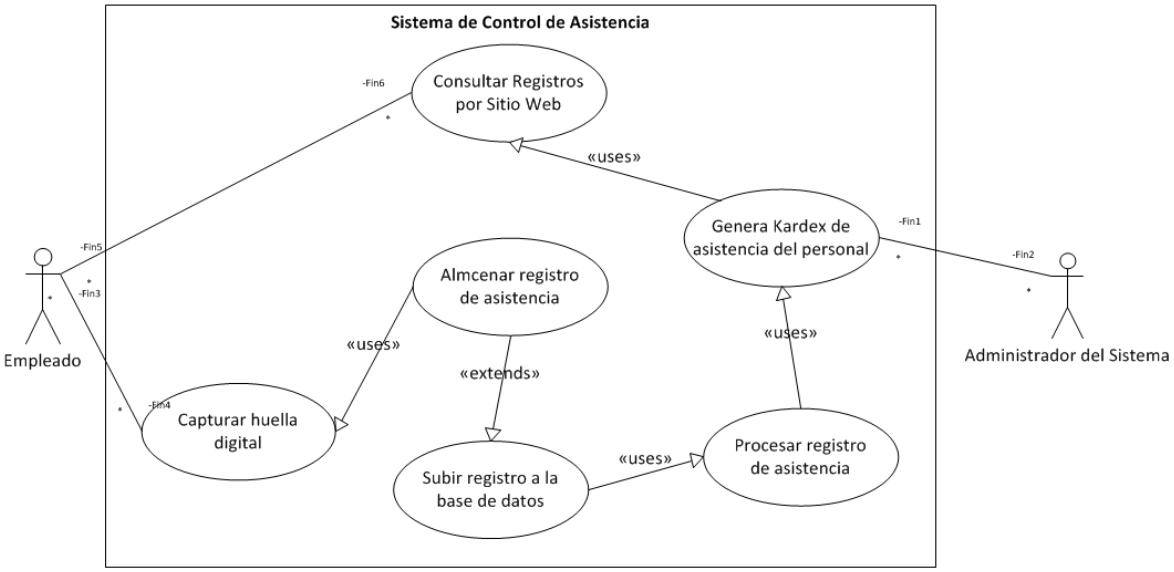



Imagen 1.3 Diagrama de casos de uso del procedimiento de Registro y Control de Asistencia

## 1.4 Software y hardware para el procedimiento de control de asistencia.

Una vez mencionadas las operaciones mediante las cuales se ejecuta el Procedimiento del Control del Registro y la Asistencia de manera general, se describe a continuación las herramientas de hardware y software requeridas por el procedimiento, analizando la conectividad entre cada una de ellas y clasificando respecto al tipo de tecnología de la que se trata como se aprecia en el siguiente diagrama.

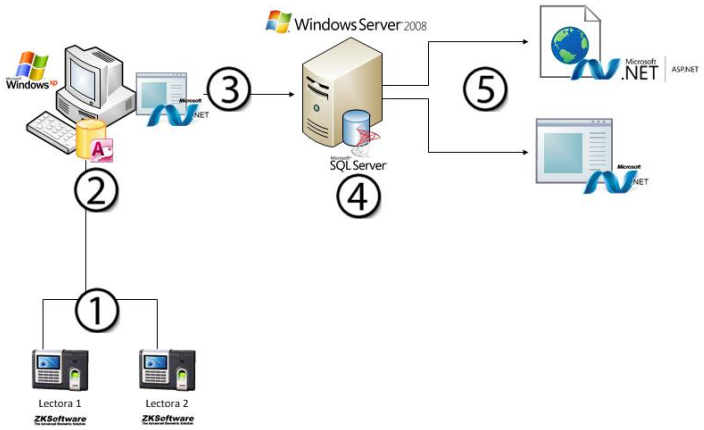



Imagen 1.4 Diagrama de conectividad del Sistema de Control de Asistencia

### 1.4.1 Bloque 1: captura de huellas digitales en dispositivos biométricos y almacenamiento de registros de asistencia.

En primera instancia el proceso requiere de dispositivos biométricos para capturar la huella digital del empleado a la hora de entrada y la hora de salida, el responsable de esta tarea es el Lector Biométrico de Huella Digital x628–C, desarrollado por la Compañía ZK-Software[10] siendo líder en soluciones tecnológicas biométricas entre las que se encuentran dispositivos con la capacidad de reconocimiento facial, reconocimiento de huella digital y lectores RFID[11] utilizados principalmente en Sistemas de Control de Accesos y Control de Asistencia, esta compañía está situada en Nueva Jersey, Estados Unidos, es reconocida por su algoritmo de autentificación mediante dispositivos biométricos.

Imagen 1.5 Dispositivo Biométrico ZK Software x628-c

*"El dispositivo x628-C es un innovador lector biométrico de huellas digitales para aplicaciones de Asistencia. El x628-C ofrece un rendimiento sin precedentes utilizando un algoritmo de verificación avanzada confiable, exacto y con la velocidad más rápida de comparación de huellas dactilares"[12].*

El Lector Biométrico de huella digital x628-C diseñado específicamente para el control de asistencia, sus principales características son la conectividad mediante el protocolo TCP/IP[13], ofrece la capacidad de almacenamiento de 3000 huellas digitales y 100000 registros de asistencia en su memoria interna. Este dispositivo es el encargado de ejecutar la primera etapa del procedimiento de asistencia, realiza la validación de la huella digital del personal.

[10] Company Overview, ZK-Software http://www.zktechnology.com/CompanyOverView.aspx

[11] RFID (siglas de Radio Frequency Identification, en español identificación por radiofrecuencia), tecnología que transmite la identidad de un objeto mediante etiquetas, tarjetas, o tags RFID

[12] ZK Software, Data sheet, x628 – C

[13] TCP/IP es un conjunto de protocolos de red basados en Internet que permitan la transmisión de datos entre equipos de computo

### 1.4.2 Bloque 2 y bloque 3: Sistema AD10 de descarga de registros de asistencia de dispositivos biométricos, operación y estructura.

El sistema AD10 es una aplicación que descarga los registros de asistencia de los dispositivos biométricos ZK Software x625-C, se encarga de ejecutar los bloques II y III mostrados en la imagen 1.4. El sistema AD10 se conecta a través de la red al dispositivo biométrico para descargar los registros almacenados en su memoria, después el sistema los inserta en su propia base de datos y posteriormente los inserta en la base de datos principal del Sistema de Control de Asistencia.

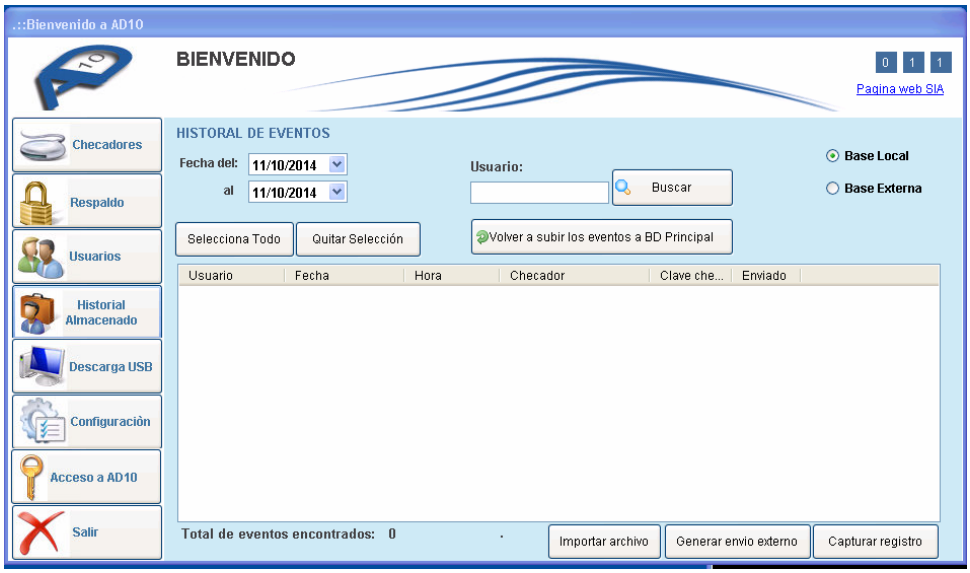



Imagen 1.6 Panel de control del Sistema AD10

El panel de control del Sistema AD10 se observa en la imagen 1.6, este sistema fue desarrollado con tecnología Microsoft, específicamente con en el Framework 3.5 de .Net, Windows Forms y lenguaje de programación C#. Las aplicaciones Windows Forms son un conjunto de herramientas de desarrollo de sistemas de información orientados a la programación de aplicaciones con interfaces de usuario sencillas y ofrecen alto rendimiento al ejecutarse, la desventaja que presenta este tipo de aplicaciones es que se ejecutan de manera local en un equipo de cómputo únicamente con Sistemas Operativos Windows.

*"Windows Forms es una de las dos tecnologías que se utiliza en Visual C# para crear aplicaciones cliente inteligentes basadas en Windows que se ejecutan en .NET Framework. Windows Forms es especialmente adecuado para escenarios de desarrollo rápido de aplicaciones donde la prioridad principal no es una interfaz gráfica de usuario compleja. El Diseñador de Windows Forms se utiliza para crear la interfaz de usuario, y permite obtener acceso a otras características de diseño y ejecución"*[14].

*".NET Framework es una tecnología que admite la compilación y ejecución de la siguiente generación de aplicaciones y servicios Web XML. El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes:*

- *Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.*
- *Proporcionar un entorno de ejecución de código que minimiza los conflictos en el despliegue y versionado de software.*
- *Ofrecer un entorno de ejecución de código que promueva la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.*
- *Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan scripts o intérpretes de comandos.*
- *Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.*

[14] Windows Forms, Microsoft Developer Network http://msdn.microsoft.com/es-mx/library/hk4ts42s(v=vs.90).aspx

- *Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código"*[15].

  *"C# es un lenguaje de programación que se ha diseñado para compilar diversas aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos. Las numerosas innovaciones de C# permiten desarrollar aplicaciones rápidamente y mantener la expresividad y elegancia de los lenguajes de estilo de C"*[16].

  *"Visual Basic está diseñado para la creación de aplicaciones de manera productiva con seguridad de tipos y orientado a objetos. Visual Basic permite a los desarrolladores centrar el diseño en Windows, la web y dispositivos móviles. Como ocurre con todos los lenguajes destinados a Microsoft .NET Framework, los programas escritos en Visual Basic se benefician de la seguridad y la interoperabilidad de los lenguajes"*[17].

Visual Basic y Visual C# Son lenguajes de Programación Orientada a Objetos, con sintaxis simple basados en el Lenguaje de Programación C, ambos se ejecutan con el Framework .NET de Microsoft, el Framework tiene origen en el año 2001 en su versión 1.0, hasta la más actual 4.0 lanzada en 2013.

La base de datos del AD10 está desarrollada con Microsoft Office[18] Access 2003, Access es un sistema gestor de base de datos relacionales que tuvo origen el año

[15] Información general acerca de .NET Framework, Microsoft Developer Network http://msdn.microsoft.com/es-es/library/zw4w595w(v=vs.110).aspx

[16] Visual C#, Microsoft Developer Network http://msdn.microsoft.com/es-es/library/kx37x362.aspx

[17] Visual Basic, Microsoft Developer Network http://msdn.microsoft.com/es-es/library/2x7h1hfk.aspx

[18] Microsoft Office es un conjunto integrado de aplicaciones de software de negocios para las computadoras Windows y Macintosh. El Office incluye un procesador de texto, una hoja de cálculo, presentación de gráficos y un programa de comunicación de correo electrónico que le da funcionalidad y que es usado generalmente para la conducción de los negocios de una oficina.

de 1993, diez años después mejoró y presentó una interfaz de usuario sencilla que permite crear bases de datos sin tener experiencia especifica en el tema, es muy limitado por su poca capacidad de almacenamiento de datos no recomendada para aplicaciones empresariales con gran escalabilidad.

El descargador de Registros de Asistencia AD10 se instaló en un equipo de cómputo de escritorio COMPAQ EVO D310, este equipo fue fabricado por la compañía Hewlett Packard (HP) entre los años 2002 y 2003, la pc contiene un procesador Intel Pentium 4 y una memoria RAM de 1.2 GB, el equipo funciona con el Sistema Operativo Windows XP Service Pack[19] 3 lanzado por Microsoft en 2001, durante el auge del Sistema Operativo fue el más estable, fue utilizado por las más importantes organizaciones, ya que ofreció alto rendimiento y estabilidad además de nuevas y mejoradas opciones de conectividad para los equipos de cómputo.

### 1.4.3 Bloque 4: Procesamiento de registros, generación del kárdex de asistencia.

> *"SQL es el lenguaje estándar para interactuar con bases de datos relacionales y es soportado prácticamente por todos los Sistemas Administradores de Bases de datos actuales"*[20].

Una de las tareas con mayor relevancia para el procedimiento del Control de Asistencia de la COFEPRIS es la generación del kárdex de cada empleado, esta operación se lleva acabo con la recolección de los Registros enviados por el Sistema AD10, que se recaban en los dispositivos biométricos, el método realiza la recepción de los registros en la base de datos principal del sistema implementada en el Sistema Gestor de Base de Datos SQL Server 2008 R2.

---

[19] Service Pack es un conjunto de Actualizaciones que Microsoft Ofrece a sus Sistemas Operativos para la mejora de proceso y corrección de errores

[20] Ceballos, Francisco Javier, C# Lenguaje y Aplicaciones p. 342

> *"Microsoft SQL Server 2008 R2 es una plataforma de base de datos que se basa en Microsoft SQL Server. SQL Server R2 facilita el desarrollo de aplicaciones controladas por datos con gran variedad de funciones, que mejoran la seguridad del almacenamiento y se implementan con rapidez".*[21]

SQL Server 2008 R2 es un sistema gestor basado en el modelo entidad relación de base de datos. Entre sus principales características se encuentran el soporte de consultas o transacciones del estándar SQL y el soporte para ejecutar procedimientos almacenados, permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los clientes se conectan a través de la red y sólo acceden a la información. La estructura fundamental de este gestor es el lenguaje Transact-SQL (T–SQL) que es el principal medio de interacción con el Servidor, permite realizar las principales operaciones en SQL Server, incluyendo la creación y modificación de esquemas de la base de datos, la introducción y edición de los datos en la base de datos, así como la administración del servidor. Esto se realiza mediante el envío de sentencias de T-SQL y declaraciones que son procesadas por el servidor y los resultados que retornan a la aplicación cliente.

> *"Un procedimiento almacenado (Stored Procedure) es un conjunto de comandos SQL que pueden almacenarse en el servidor de Base de Datos. Una vez que se hace, los clientes no necesitan relanzar los comandos individuales pero pueden en su lugar referirse al procedimiento almacenado"*[22]*.*

Los procedimientos almacenados son un conjunto de consultas SQL del tipo Insert, Select , Update y Delete, así como estructuras de control de flujo IF, WHILE, CASE y operaciones lógicas y aritméticas que se ejecutan de manera estructurada. La ventaja de los procedimientos almacenados es la ejecución

[21] Información general de SQL Server, Microsoft TechNet http://technet.microsoft.com/es-es/dd367804.aspx

[22] Procedimientos almacenados y funciones, MySQL 5.0 Reference Manual http://dev.mysql.com/doc/refman/5.0/es/stored-procedures.html

dentro del servidor de base de datos, esto evita que se delegue las operaciones al código fuente de la aplicación que lo invoca, esto conlleva a brindar seguridad dado que no se expone el nombre de las tablas de la base de datos en ninguna línea de código de la aplicación, otra ventaja sobresaliente es la ejecución de “n” consultas y “n” operaciones la base de datos invocando al Stored Procedure, pueden tener parámetros de entrada y salida o no, esto significa que se pueden retornar “n” numero de variables después de ejecutar el procedimiento almacenado.

La Base de Datos del Sistema de Control de Asistencia cuenta con veintitrés procedimientos almacenados que se encargan de ejecutar todas las funciones del sistema de control de asistencia, los más importantes son:

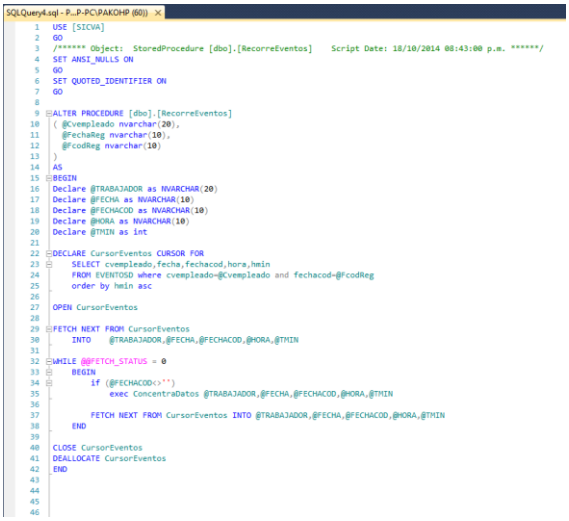
“Recalificar”, mostrado en la imagen 1.7, Es el procedimiento que desencadena a otros procedimientos almacenados de acuerdo la existencia del registro de asistencia, al día laboral o alguna justificación de asistencia.

SQLQuery3.sql - MJ...Administrador (60))* | SQLQuery1.sql - MJ...Administrador (57))*

```
USE [SICVA]
GO
/****** Object:  StoredProcedure [dbo].[Recalificar]    Script Date: 13/10/2014 01:01:01 p.m. ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[Recalificar]
( @Cvempleado nvarchar(20),
  @FechaReg nvarchar(10),
  @FcodReg nvarchar(10)
)
AS
BEGIN

Declare @TRABAJADOR as NVARCHAR(20)
Declare @FECHA as NVARCHAR(10)
Declare @FECHACOD as NVARCHAR(10)
Declare @HORA as NVARCHAR(10)
Declare @TMIN as int
Declare @Esdialaboral as int
Declare @CadenaFecha as nvarchar(10)
Declare @NumDia as nvarchar(4)
DECLARE @sql nvarchar(4000)
Declare @ClaveHorarioTrab as nvarchar(20)
Declare @Lenguje as nvarchar(20)

Set @CadenaFecha=@FcodReg
Set @Numdia=(SELECT DATEPART(DW,@CadenaFecha))
set @Lenguje=(select @@language)
if (@Lenguje='us_english')
    Set @Numdia=@Numdia-1

Set @ClaveHorarioTrab=(Select dbo.ClaveHorarioEmpleado(@Cvempleado))
Set @Esdialaboral=(Select dbo.EsDiaLaboral(@Numdia,@ClaveHorarioTrab)) ---verifica por el dia de entrada

PRINT @CadenaFecha
PRINT @ClaveHorarioTrab
PRINT @Numdia
--BORRAR PRIMERO EL DIA DEL MOVIMIENTO
Set @sql='Delete from MovimientoD where '
SET @sql=@sql + 'Cvempleado=''' + @Cvempleado  + ''''
SET @sql=@sql + 'and fechacod=''' + @FcodReg  + ''''
EXEC sp_executesql @sql
-- MANDAR EL BLANCO DEL MOVIMIENTO
print 'ES DIA LABORAL'
print @Esdialaboral
if (@Esdialaboral=1)
   EXEC MandaDiaFalta @Cvempleado,@FechaReg,@FcodReg
else
   EXEC MandaDiaNoLaboral @Cvempleado,@FechaReg,@FcodReg

--MANDAR EL RECORRIDO DE EVENTOS ENCONTRADOS
PRINT 'MANDA A RECORRER EVENTOS'
EXEC RecorreEventos @Cvempleado,@FechaReg,@FcodReg
-- MANDAR A VERIFICAR SI ENCUENTRA INCIDENCIAS
PRINT 'MANDA A VERIFICAR INCIDENCIAS'
EXEC VerificadorIncidencia @Cvempleado,@FechaReg,@FcodReg
--MANDAR A AJUSTAR LAS OMISIONES DE ENTRADA Y SALIDA
EXEC AjustarOmisiones @Cvempleado,@FechaReg,@FcodReg
END
```

Imagen 1.7 Código del procedimiento almacenado “Recalificar”

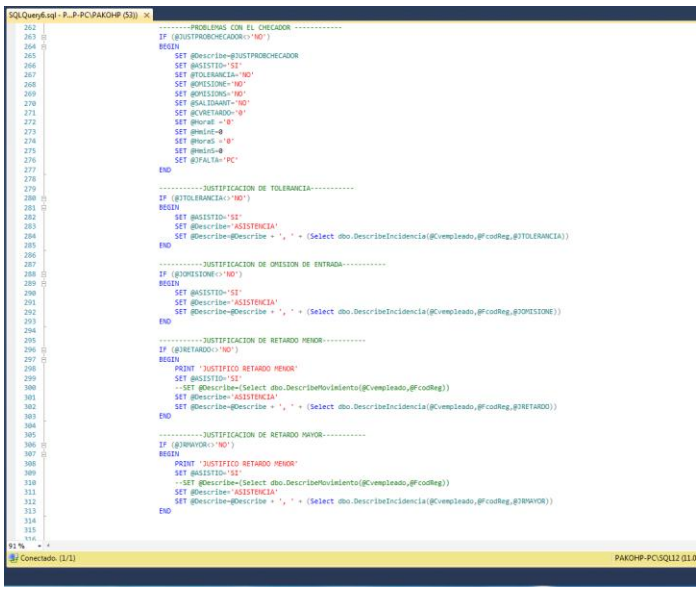
El procedimiento “recorreEventos” se desencadena solo si es día laboral para consultar la existencia de los registros de asistencia del empleado, en caso de existir los evalúa conforme el horario de labores del empleado, el código del procedimiento se observa en la imagen 1.8



```
USE [SICVA]
GO
/****** Object:  StoredProcedure [dbo].[RecorreEventos]    Script Date: 18/10/2014 08:43:00 p.m. ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER PROCEDURE [dbo].[RecorreEventos]
( @Cvempleado nvarchar(20),
  @FechaReg nvarchar(10),
  @FcodReg nvarchar(10)
)
AS
BEGIN
Declare @TRABAJADOR as NVARCHAR(20)
Declare @FECHA as NVARCHAR(10)
Declare @FECHACOD as NVARCHAR(10)
Declare @HORA as NVARCHAR(10)
Declare @TMIN as int

DECLARE CursorEventos CURSOR FOR
    SELECT cvempleado,fecha,fechacod,hora,hmin
    FROM EVENTOSD where cvempleado=@Cvempleado and fechacod=@FcodReg
    order by hmin asc

OPEN CursorEventos

FETCH NEXT FROM CursorEventos
    INTO    @TRABAJADOR,@FECHA,@FECHACOD,@HORA,@TMIN

WHILE @@FETCH_STATUS = 0
    BEGIN
        if (@FECHACOD<>'')
            exec ConcentraDatos @TRABAJADOR,@FECHA,@FECHACOD,@HORA,@TMIN

        FETCH NEXT FROM CursorEventos INTO @TRABAJADOR,@FECHA,@FECHACOD,@HORA,@TMIN
    END

CLOSE CursorEventos
DEALLOCATE CursorEventos
END
```

Imagen 1.9 Código Procedimiento almacenado “recorreEventos”

El procedimiento almacenado “verificadorIncidencia” verifica si existe alguna justificación y valida que corresponda con el tipo de incidencia, se ejecuta solo si no existen registros de asistencia,  parte del código del procedimiento se observa en la imagen 1.10.



```
--------PROBLEMAS CON EL CHECADOR ----------
IF (@JUSTPROBCHECADOR<>'NO')
BEGIN
    SET @Describe=@JUSTPROBCHECADOR
    SET @ASISTID='SI'
    SET @TOLERANCIA='NO'
    SET @OMISIONE='NO'
    SET @OMISIONS='NO'
    SET @SALIDAANT='NO'
    SET @CVRETARDO='0'
    SET @HoraE ='0'
    SET @HminE=0
    SET @HoraS ='0'
    SET @HminS=0
    SET @JFALTA='PC'
END

----------JUSTIFICACION DE TOLERANCIA----------
IF (@JTOLERANCIA<>'NO')
BEGIN
    SET @ASISTID='SI'
    SET @Describe='ASISTENCIA'
    SET @Describe=@Describe + ', ' + (Select dbo.DescribeIncidencia(@Cvempleado,@FcodReg,@JTOLERANCIA))
END

----------JUSTIFICACION DE OMISION DE ENTRADA----------
IF (@JOMISIONE<>'NO')
BEGIN
    SET @ASISTID='SI'
    SET @Describe='ASISTENCIA'
    SET @Describe=@Describe + ', ' + (Select dbo.DescribeIncidencia(@Cvempleado,@FcodReg,@JOMISIONE))
END

----------JUSTIFICACION DE RETARDO MENOR----------
IF (@JRETARDO<>'NO')
BEGIN
    PRINT 'JUSTIFICO RETARDO MENOR'
    SET @ASISTID='SI'
    --SET @Describe=(Select dbo.DescribeMovimiento(@Cvempleado,@FcodReg))
    SET @Describe='ASISTENCIA'
    SET @Describe=@Describe + ', ' + (Select dbo.DescribeIncidencia(@Cvempleado,@FcodReg,@JRETARDO))
END

----------JUSTIFICACION DE RETARDO MAYOR----------
IF (@JRMAYOR<>'NO')
BEGIN
    PRINT 'JUSTIFICO RETARDO MENOR'
    SET @ASISTID='SI'
    --SET @Describe=(Select dbo.DescribeMovimiento(@Cvempleado,@FcodReg))
    SET @Describe='ASISTENCIA'
    SET @Describe=@Describe + ', ' + (Select dbo.DescribeIncidencia(@Cvempleado,@FcodReg,@JRMAYOR))
END
```

Imagen 1.10 Código Procedimiento almacenado “verificadorIncidencia”

Finalmente el procedimiento “ingresaMovimiento” se invoca cuando se forma la cadena de datos válida para el kárdex de asistencia de empleado cualquiera que haya sido sea la evaluación del registro de asistencia del empleado y se inserta en la tabla de la base de datos donde se alojan el kárdex del empleado, parte del código que lo conforma se muestra en la imagen 1.11.

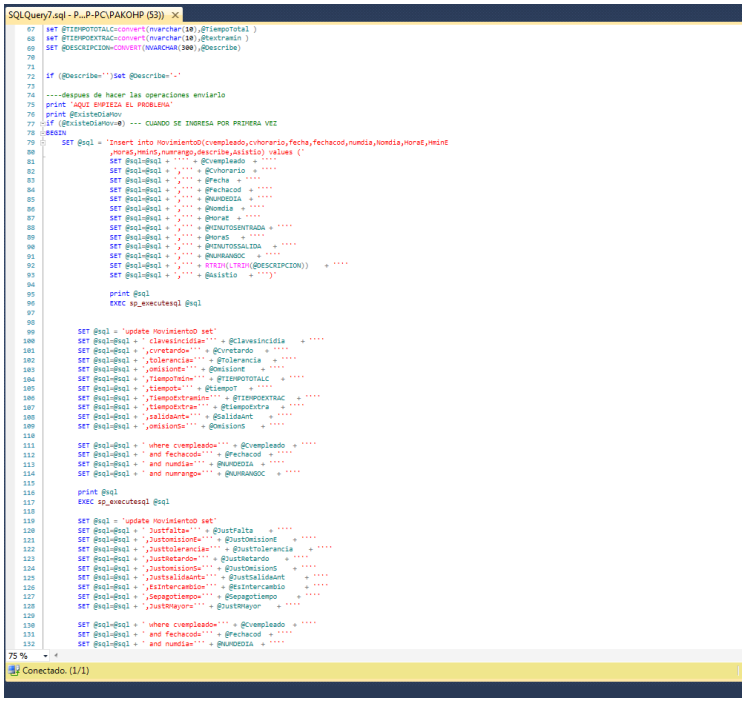



Imagen 1.11 Código Procedimiento almacenado “ingresaMovimiento”

Es así como concluye la descripción del cuarto bloque, se puede observar que es el bloque con mayor procesamiento de datos que depende totalmente de la precisión de la descarga de registros de asistencia para poder brindar los resultados esperados para el procedimiento de control de asistencia.

### 1.4.4 Bloque 5: consulta y publicación de resultados

En este último bloque se describe el procedimiento para la publicación de los resultados obtenidos en los anteriores bloques, para realizarlo únicamente se

consultan a la base de datos principal con el Sistema de Control de Asistencia Alife5 desarrollado con la tecnología Windows Forms mencionada.

En la imagen 1.12 se observa el kárdex de evaluación de asistencia del empleado en un rango de quince días, se aprecia la fecha y el nombre del día de la semana en la que se efectuó el evento, el rango de la asistencia, la hora de entrada, la hora de salida, la descripción de la evaluación y las horas que transcurrieron desde la hora de entrada hasta la salida. Esta información es solo accesible para la Dirección Ejecutiva de Recursos Humanos y la Gerencia Ejecutiva de Desarrollo Humano y Servicios al Personal para llevar a cabo el procedimiento de Registro y Control de Asistencia.

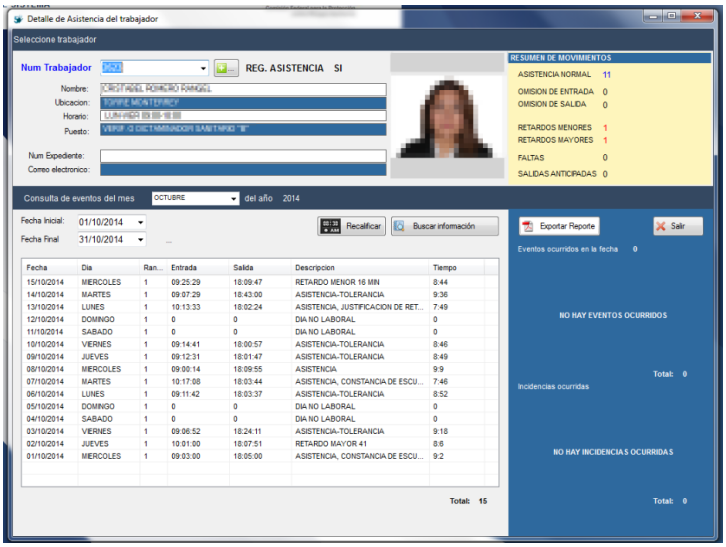



Imagen 1.12 Kárdex de asistencia del empleado

Finalmente el procedimiento de asistencia es publicado a través de una página web en la intranet institucional COFEPRIS, esta página es accesible para todos los empleados, en ella se consulta el kárdex de asistencia desde cualquier equipo de cómputo, insertando el número de empleado y el rango de fechas, desde la fecha inicial y hasta la final, en esta publicación se observan la fecha y nombre del día en que se evalúa la asistencia, la hora de entrada, la hora de salida y la evaluación recibida por cada día de asistencia, el cuerpo de la página web se muestra en la imagen 1.13.


Imagen 1.13 Kárdex de asistencia del empleado en la página web institucional

La principal ventaja que brinda la página web al Procedimiento de Asistencia es la accesibilidad que ofrece a un número ilimitado de empleados, quienes tienen la opción de consultar sus registros de asistencia en el momento que lo deseen, contar con información precisa, en tiempo y forma ya que los registros se actualizan a diario, en caso que exista alguna incidencia el empleado cuenta con mayor tiempo para poder justificarla y evitar algún descuento económico.

La página web se ejecuta en el Servidor Web IIS 7 (Internet Information Services), desarrollada con ASP (Active Server Pages) y el Lenguaje C# definido anteriormente trabajo con el Framework .NET

> *"IIS es un servidor web y un conjunto de servicios para sistemas Operativos Microsoft Windows que permiten compartir información con usuarios en Internet, en una intranet o en una extranet"*[23].

[23] Servidor Web IIS, Microsoft TechNet http://technet.microsoft.com/es-mx/library/cc753433(v=ws.10).aspx

Los servidores web son programas que se publican a través de las redes (internet, intranet, extranet) que tienen la capacidad de compartir información mediante páginas web, archivos multimedia tales como video, música y fotografías, además de ser el medio para conectarse a bases de datos, consultar información.

> *"ASP.NET es un modelo de desarrollo Web unificado que incluye los servicios necesarios para crear aplicaciones Web empresariales con el código mínimo. ASP.NET forma parte de .NET Framework y al codificar las aplicaciones ASP.NET tiene acceso a las clases en .NET Framework. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el Common Language Runtime (CLR), entre ellos Microsoft Visual Basic y C#. Estos lenguajes permiten desarrollar aplicaciones ASP.NET que se benefician del Common Language Runtime, seguridad de tipos, herencia, etc"*[24].

ASP.NET es una tecnología de Microsoft Systems para el desarrollo e implementación de Páginas Web Dinámicas en la red directamente asociado con código escrito en los lenguajes de programación C# y Visual Basic, se ejecutan con el Framework .NET instalado en el servidor IIS. ASP tiene origen en 1996 con la versión IIS 3.0, Active Server Pages, como se le conoce en el idioma inglés, se complementa con otras tecnologías, tales como el lenguaje de marcado de hipertexto (HTML) que es un estándar para la elaboración de páginas web, el cual define una estructura básica para el contenido de la misma, y las hojas de estilo en cascada (CSS) es un lenguaje para definir la presentación de un documento HTML, en la imagen 1.14 se describe la vinculación entre las diferentes tecnologías antes mencionadas.

[24] Información General sobre ASP.NET, Microsoft TechNet http://msdn.microsoft.com/eses/library/4w3ex9c2(v=vs.100).aspx

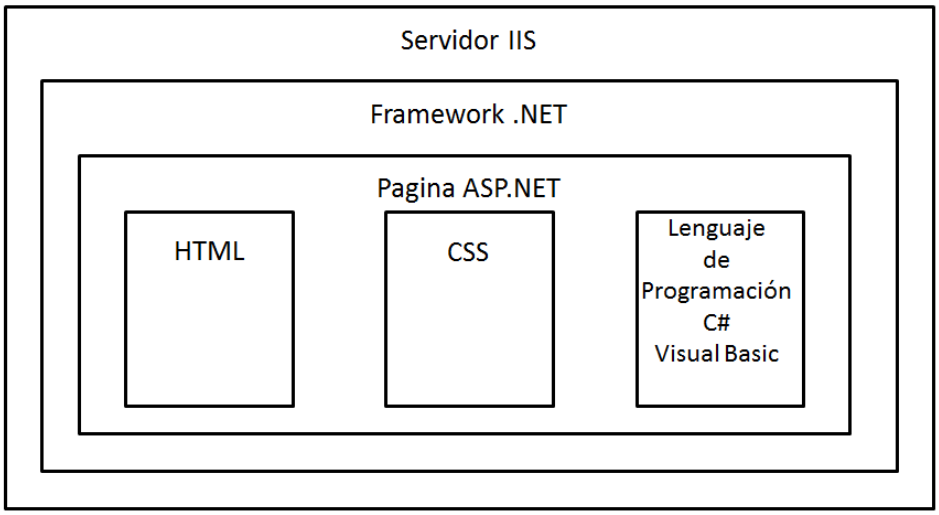



Imagen 1.14 Estructura del Servidor IIS

De manera general, se ha mencionado la operación del Procedimiento de Registro y Control de Asistencia de la COFEPRIS, como se observa tiene un grado de complejidad alto, puesto que cada operación que se realiza con las diferentes herramientas que estructuran este procedimiento tienen total dependencia incluida y extendida de las demás operaciones como fue mostrado en la imagen 1.4 del diagrama de casos de uso.

## 1.5 Definición del problema

El procedimiento de Registro y Control de Asistencia de COFEPRIS en los últimos meses ha presentado inestabilidad en su funcionamiento, no se han obtenido los resultados esperados, esto se refleja con el incremento de casos de empleados a los que se hace descuentos económicos en periodos quincenales, debido a la falta de registros de asistencia de algunos días de asistencia, se ha analizo todo el procedimiento en general, la complejidad de todo el sistema no permite identificar la falla que ocasiona este inconveniente de manera sencilla, es por ello que se trazó el siguiente plan de acción para la búsqueda del problema y la corrección del mismo.

El plan de acción se organizó de manera estructurada de acuerdo al flujo del procedimiento, se revisó puntualmente cada una de las operaciones, sistema y equipos que intervienen en este procedimiento, en el orden que a continuación se muestra:

I) Se realizó mantenimiento a los dispositivos biométricos con el proveedor de ZK Software en México, se analizó de manera general los dispositivos, se revisaron los puertos de conexión de red, el mecanismo lector de huella digital, la fuente de alimentación, el display de los dispositivos y se determinó finalmente que el funcionamiento de los dispositivos es correcto, los dispositivos trabajan de manera adecuada. Solo se dio limpieza general a los dispositivo para descartar un funcionamiento no adecuado por lo que se descarta que los dispositivos biométricos generen los problemas en el procedimiento.

II) Se revisó el equipo COMPAQ EVO310 en el que está instalado el Sistema AD10 Descargador de Registros de Asistencia, se encontró que la respuesta del equipo al abrir un programa es lenta, además se observó bastante polvo en el interior del CPU del equipo, específicamente en los componentes de la tarjeta madre, memoria RAM y en las terminales de los conexión del disco duro. Otro inconveniente que se detecto fue el Sistema Operativo Windows XP, Microsoft Systems dejo brindar Soporte Técnico el 8 de Abril de 2014, este cambio afecto a las actualizaciones de software y opciones de seguridad, por lo que ya no es recomendable el uso de este sistema, ya que en caso de colapsar ya no será sencillo dar mantenimiento a la PC, por lo que se realizó servicio de limpieza al equipo pero no se observaron mejoras, por lo que se concluye que

el equipo de cómputo ya no tiene las características ideales para continuar con la descarga de registros de asistencia con el Sistema AD10, es recomendable reemplazar la PC por un equipo con características más actual.

III) Se revisó al sistema AD10, se encontró que debido al bajo rendimiento de la PC en el que se encuentra instalado no es posible almacenar los registros en su Base de Datos de Access y la consecuencia es la perdida de registros de asistencia. Se determinó como medida de acción monitorear el desempeño del sistema y se observaron los siguientes errores producidos por el AD10 mostrados en las imágenes 1.15 a la 1.18.

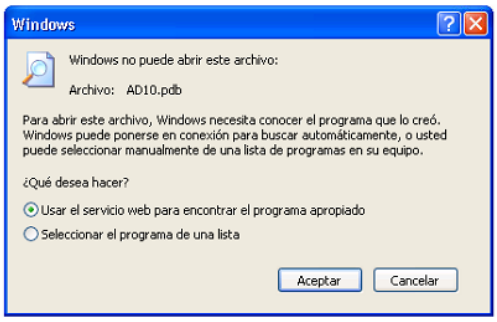



Imagen 1.15 Error 1 Sistema AD10

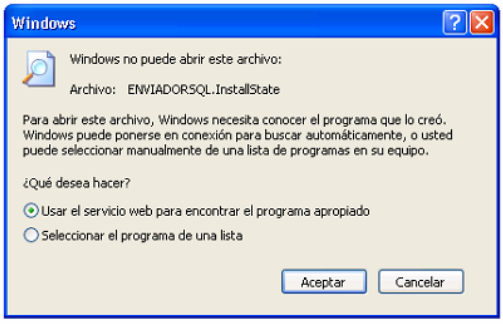



Imagen 1.16 Error 2 Sistema AD10


Imagen 1.17 Error 3 Sistema AD10

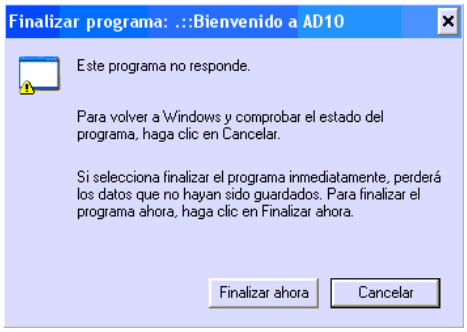



Imagen 1.18 Error 4 Sistema AD10

Para dar solución a esta situación se recomendó migrar el Sistema AD10 a un equipo con mejores recursos con un Sistema Operativo más actual y estable, específicamente Windows 7, pero no fue la opción más viable ya que el AD10 requiere de licencias de liberación de software para operar con los dispositivos biométricos como se muestra a continuación.

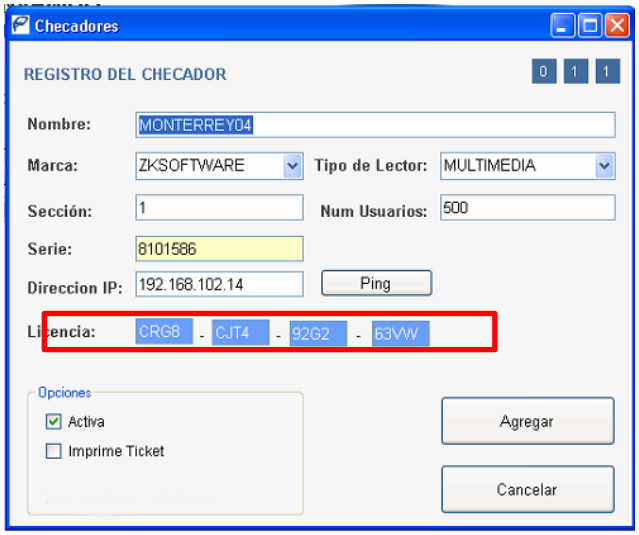



Imagen 1.19 Clave de Liberación Dispositivos Biométricos Sistema AD10

En la ventana de la imagen 1.19 cuatro cajas de texto con una serie de caracteres, dichas líneas forman una licencia de liberación, esta licencia es indispensable para lograr la comunicación con los dispositivos biométricos, en caso de no existir el código de liberación no hay posibilidades de conectarse con los equipo biométricos y por ende no se descargan los registros de asistencia. La medida tomada fue contactar al desarrollador del sistema, lo que se fue que el sistema AD10 no cuenta con se actualizaciones y se dejó de dar soporte técnico por parte del fabricante, sin ofrecer otra alternativa no se consiguió la información necesaria para poder brindar una solución inmediata.

IV) En seguimiento al análisis de la problemática del procedimiento de asistencia y descartar posibles errores la Base de Datos principal se respaldó completamente como soporte informativo y se depuro los registros con año y medio de antigüedad para evitar errores de lógica con los procedimientos almacenados que operan el sistema, este análisis se realizó sin inconvenientes, sin embargo no hubo cambios significativos al respecto, por lo que se descarta que la Base de Datos principal ocasione los resultados no esperados para el Procedimiento de Control de Asistencia.

V) Se descartó totalmente al sistema Alife5 y al Sitio Web de Consulta de Registros de Asistencia, debido a que ambos Sistemas no intervienen en el procesamiento de registros de asistencia, solo son sistemas de consulta de resultados procesados por lo que no tiene mayor injerencia en esta problemática.

Finalmente se concluye en base a las acciones tomadas para la localización del problema y a efecto de brindar una solución, se determinó que la falla es provocada por el equipo de cómputo COMPAQ EVO310, ya no cuenta con las características ideales para ejecutar el sistema AD10 que se ejecuta de manera inestable, no se cuenta con la información necesaria para migrar al sistema a otro equipo de cómputo, en caso de mantener al AD10 se seguirá la perdida de Registros de Asistencia del Personal.

# CAPITULO II

# "DESARROLLO DEL CHECKPOINT"

## 2.1 PROPUESTA

De acuerdo a la problemática antes mencionada se propuso el desarrollo de un nuevo Sistema de Descarga de Registros de Asistencia para el Procedimiento de Asistencia nombrado CHECKPOINT Descargador de Registros de Asistencia, el nombre se elige por la traducción al español "Punto de Control", para reemplazar al Sistema AD10. El CHECKPOINT debe cumplir las condiciones operativas del AD10 y además debe brindar precisión en la descarga de registros para satisfacer las necesidades del Procedimiento de Registro y Control de Asistencia de la COFEPRIS.

## 2.2 Objetivo

Desarrollar un Sistema de Descarga de Registros de Asistencia, denominado CHECKPOINT, con la capacidad de adaptarse adecuadamente y satisfacer los requerimientos del Procedimiento Registro y Control de Asistencia de La Comisión Federal para la Protección Contra Riesgos Sanitarios.

### 2.2.1 Objetivos Específicos

I) Programar un Sistema de Descarga de Registros de Asistencia que sea capaz de conectarse con los dispositivos Biométricos ZK Software x628-C y ofrezca la funciones del sistema AD10.

II) Crear una base de datos sobre un Sistema Gestor de Base de Datos sólido para el sistema de Descarga de Registros de Asistencia.

III) Programar un sistema de Descarga de Registros de Asistencia con tecnologías Microsoft compatible con el resto del procedimiento de control de asistencia.

### 2.2.2 Justificación

La implementación del CHECKPOINT mejorará la descarga de registros de Asistencia de los dispositivos biométricos, brindará precisión en la captura de registros de asistencia en la base de datos y reducirá los casos de empleados que se reportan con descuentos económicos quincenales en un 20% desde el momento que sea implementado.

## 2.3 Requerimientos del sistema

Para lograr cubrir los objetivos del sistema se deben cumplir las siguientes operaciones:

a) La descarga de los registros de asistencia de los Dispositivos Biométricos ZKSOFTWARE x628-C.

b) La descarga de los registros de asistencia de los Dispositivos Biométricos ZKSOFTWARE x628-C que se almacenan en la memoria del dispositivo.

c) Búsqueda de registros de asistencia en la Base de Datos del Sistema CHECKPOINT por número de empleado y fecha.

d) Gestión de dispositivos biométricos, registro de nuevos dispositivos, habilitar y deshabilitar dispositivos, modificar la fecha y la hora de los dispositivos, modificación el nombre de los dispositivos.

e) Control de acceso a los usuarios del Sistema CHECKPOINT con usuario y contraseña.

## 2.4 Estructura del CHECKPOINT

Dados los requerimientos de operación del Sistema CHECKPOINT, se propuso dividir en dos bloques de trabajo al sistema, con el objetivo reducir posibles fallos en la descarga de registros de asistencia de los Dispositivos Biométricos y realizar de manera independiente la consulta de registros de asistencia y la gestión de los dispositivos biométricos como se observa en la siguiente imagen 2.1.

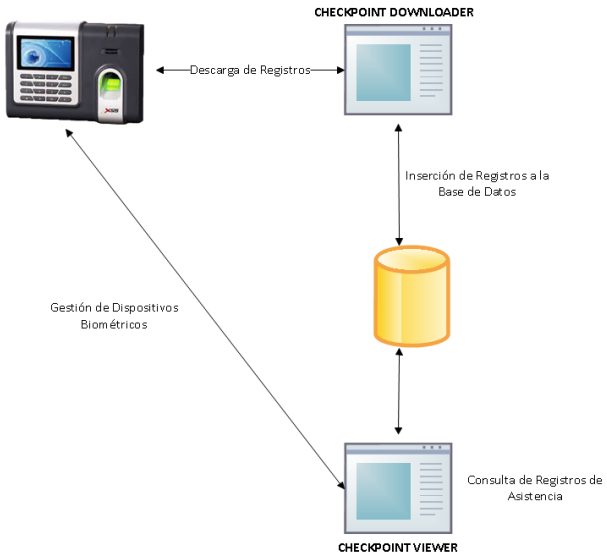



Imagen 2.1 Estructura del Sistema CHECKPOINT

### 2.4.1 CHECKPOINT DOWNLOADER

El CHECKPOINT DOWNLOADER descargará los registros de asistencia de los dispositivos biométricos ZK Software, esta aplicación debe permanecer en ejecución desde el momento en que se enciende el equipo de cómputo e inmediatamente intentar conectarse a los dispositivos, en caso de conectarse se descargarán los registros almacenados en la memoria interna de los dispositivos y se insertarán en la base de datos del CHECKPOINT, posteriormente cuando la aplicación detecte un registro en el dispositivo lo insertará en la base de datos SQL Server en tiempo real, al transcurrir sesenta minutos la aplicación monitorea el estado de la conexión con el dispositivo para garantizar la comunicación entre el dispositivos y el equipo de cómputo; en caso de no lograr la conexión la aplicación insistirá hasta conectarse nuevamente. El proceso se ejecutará indefinidamente hasta que sea interrumpida manualmente por los usuarios de sistema, el procedimiento de esta aplicación se aprecia en la imagen 2.2.

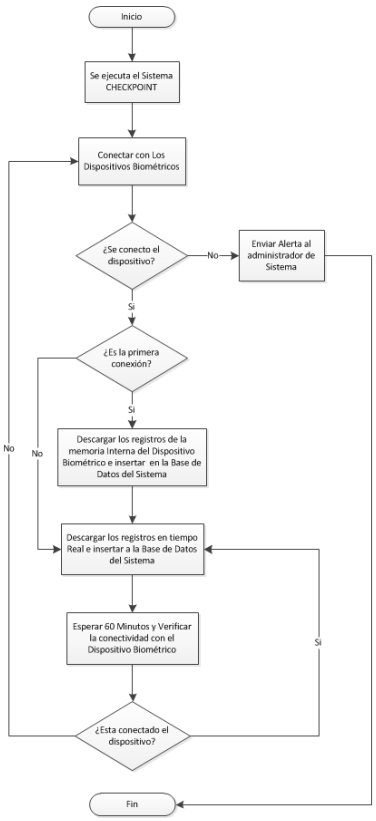



Imagen 2.2 Diagrama de Flujo del CHECKPOINT DOWNLOADER

### 2.4.2 CHECKPOINT VIEWER

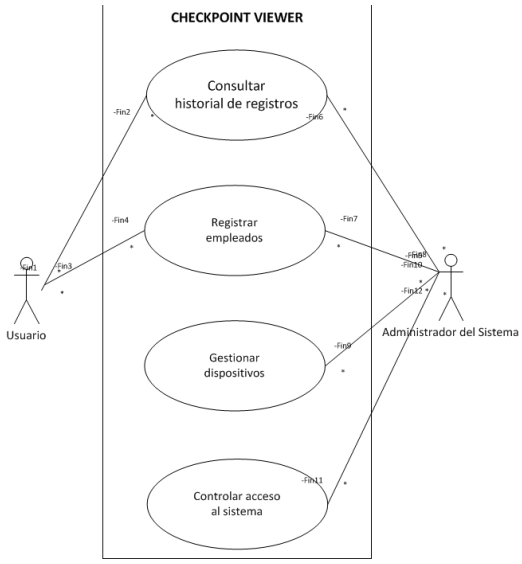



Imagen 2.3 Diagrama de Casos de uso CHECKPOINT VIEWER

El CHECKPOINT VIEWER es el medio para consultar el historial de los registros de asistencia recolectados por el CHECKPOINT DOWNLOADER, así mismo registrará a los empleados de nuevo ingreso a la dependencia incluidos en el procedimiento de control de asistencia, además se integrará un control de acceso para ejecutar las operaciones mencionadas solo accesibles para el perfil de usuarios generales y también para el perfil de administradores del sistema. El control de acceso del sistema delimitará ciertas operaciones, específicamente en la gestión de los dispositivos y otorgará el acceso a nuevos usuarios con el perfil de administrador y/o usuario general, estas operaciones se muestran en la imagen 2.3. El CHECKPOINT contiene los valores por omisión para el control de acceso usuario “admin” y contraseña “root”.

Los bloques del CHECKPOINT se conectarán a una base de datos en común, la base de datos se deber instalar en un Servidor de Base de Datos SQL 2012 dentro de la red donde se conecta el equipo de cómputo donde se ejecuta el sistema para ser accesible por el CHECKPOINT DOWNLOADER que insertará registros de asistencia y por CHECKPOINT VIEWER que consultará los registros de asistencia, gestionará y configurará los dispositivos biométricos.

## 2.5 Metodología de programación del CHECKPOINT

El método de programación para los bloques del Sistema CHECKPOINT debe diseñarse y desarrollarse de manera ordenada para garantizar un avance continuo del proyecto y lograr como resultado un producto de calidad que alcance los objetivos planteados, para ello el método utilizado para este desarrollo es la programación en capas, que consiste en dividir el código fuente según su funcionalidad principal, la definición de esta metodología es la siguiente.

*"La programación por capas es una técnica de ingeniería de software propia de la programación orientada a objetos, éstos se organizan principalmente en 3 capas: la capa de presentación, la capa de lógica de negocio y la capa de datos"*[25].

La principal ventaja que ofrece este modelo de programación es la independencia que hay entre las capas, el resultado que se obtiene al implementarlo es el manejo de errores y actualizaciones solo se modifican las capas deseadas sin tener que modificar el código de todo el sistema. El esquema de esta metodología se observa en la imagen 2.4.

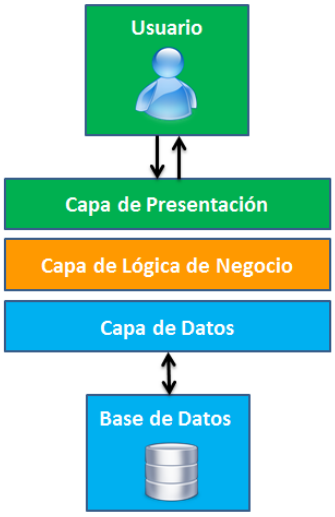



Imagen 2.5 Esquema del método de programación en tres capas

[25] Ceballos Francisco Javier, C# Lenguaje y aplicación p. 332

*"La Capa de Presentación del programa ante el usuario, debe manejar interfaces que cumplan con el objetivo principal de este componente, el cual es facilitar al usuario la interacción con la aplicación"*[26].

En la capa de presentación contiene los objetos encargados de comunicar al usuario con el sistema mediante el intercambio de información, capturando y desplegando los datos necesarios para realizar alguna tarea. En esta capa los datos se procesan de manera superficial por ejemplo, para determinar la validez de su formato o para darles algún orden específico.

*"La Capa de la Lógica de Negocio es la que define todas las reglas que se deben cumplir para una correcta ejecución del programa, es donde se encuentra toda la lógica del programa, las estructuras de datos y objetos encargados para la manipulación de los datos existentes, así como el procesamiento de la información ingresada o solicitada por el usuario en la capa de presentación".*[27]

La Lógica de Negocio obtiene la información ingresada por el usuario mediante la Capa de Presentación, si la aplicación se comunica con otros sistemas que actúan en conjunto, lo hace mediante esta capa. También se comunica con la capa de datos para obtener información existente o ingresar nuevos datos.

*"La Capa de Datos realiza las transacciones con bases de datos y con otros sistemas para obtener o ingresar información al sistema. El manejo de los datos debe hacerse de tal forma que los datos que se ingresan así como los que se extraen de las bases de datos sean consistentes y precisos".*[28]

La capa de datos se definen las consultas a realizar en la base de datos, tanto las consultas simples como las consultas complejas parla generación de reportes más específicos, esta capa envía la información directamente a la capa de reglas de negocio para que sea procesada e ingresada en objetos según se necesite.

[26] Ceballos Francisco Javier, C# Lenguaje y aplicación p. 333

[27] Ibídem, p. 334

[28] Ibídem, p. 336

Al realizar el análisis del Sistema CHECKPOINT y teniendo el modelo de programación que estructura al sistema se procede a describir las herramientas de desarrollo para este proyecto.

## 2.6 Herramientas de desarrollo del CHECKPOINT

### - Visual Studio 2010

> *"Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C# y Visual C++ utilizan todos el mismo entorno de desarrollo integrado (IDE), que habilita el uso compartido de herramientas y hace más sencilla la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes utilizan las funciones de .NET Framework, las cuales ofrecen acceso a tecnologías clave para simplificar el desarrollo de aplicaciones web ASP y Servicios Web XML"*[29].

Visual Studio es un entorno de desarrollo que integra todas las herramientas de programación del Microsoft Framework .NET, para el caso de estudio se utiliza el lenguaje de programación C# mencionado, principalmente para el desarrollo de la Capa de Negocio y la conexión con el dispositivo biométrico, así mismos de Windows Forms para la implementación de la Capa de Presentación del sistema CHECKPOINT a los usuario.

[29] Introducción a Visual Studio, Microsoft MSDN http://msdn.microsoft.com/es-mx/library/fx6bk1f4(v=vs.90).aspx

**- SDK: SOFTWARE DEVELOPMENT KIT ZKSOFTWARE (KIT DE DESARROLLO INTEGRADO)**

> *"El SDK de comunicación es una interfaz para la comunicación de datos con los dispositivos de huella digital, dispositivos de control de acceso y dispositivos de tarjeta RFID. Se puede utilizar para administrar convenientemente la información del usuario y las huellas digitales, descarga de registros de asistencia, registros de operación, información del usuario y de huellas digitales, ajuste de dispositivos, y configuración el control de acceso"*[30].

El SDK de ZK Software es el medio de comunicación entre el Sistema CHECKPOINT y los dispositivos biométricos, la versión requerida para el desarrollo es la 6.2.4.1, esta versión es compatible con el lenguaje C# de Microsoft Framework .NET y forma parte de la Capa de Negocio, los métodos que contiene el SDK son los siguientes.

- Descarga los registros de asistencia.
- Carga y descarga la información del usuario, información de la tarjeta, las huellas digitales, y consulta de información.
- Establece reglas de control de acceso a los dispositivos de control de acceso.
- Ajusta la hora del dispositivo biométrico.
- Disparador de eventos de los dispositivos en tiempo real, por ejemplo, la verificación de huellas dactilares.
- Registrar a los usuarios en línea.

Para el caso del desarrollo del Sistema CHECKPOINT y en base a sus requerimientos operativos se utilizaron las operaciones "descarga de Registros de

[30] SDK Description, TFT Series Communication Protocol SDK Development Handbook P. 9

Asistencia y Ajuste de la hora del dispositivo" que están contenidos en SDK ZK Software.

- **SQL SERVER 2012**

> *"SQL Server 2012 permite a los clientes crear aplicaciones de vital importancia y soluciones de gran cantidad de datos mediante el uso de tecnología en memoria de alto rendimiento de almacenamiento de datos, inteligencia de negocios y cargas de trabajo de análisis sin tener dispositivos avanzados. SQL Server 2012 usa un conjunto común de herramientas para implementar y administrar bases de datos tanto de forma local como en la nube, lo que facilita que los clientes aprovechen la nube con sus conocimientos existentes"*[31].

SQL Server 2012 es un Sistema Gestor de Base de Datos, fue creado por Microsoft Systems, actualmente es la versión más estable de ediciones SQL Server, ofrece buen rendimiento de operación, tiene la capacidad de almacenar gran cantidad de datos, brinda una amplia gama de utilidades de seguridad, es por ello que se eligió SQL 2012 como gestor de base de datos del CHECKPOINT, que a su vez se incorpora a la Capa de Datos del Sistema.

- **SQL MAGNAMENT STUDIO 2012**

> *"SQL Server Management Studio es un entorno integrado para obtener acceso, configurar, administrar y desarrollar todos los componentes de SQL Server. SQL Server Management Studio combina un amplio grupo de herramientas gráficas con una serie de editores de script enriquecidos que permiten a desarrolladores y*

[31] SQL Server, Sever and cloud Platform http://www.microsoft.com/es-xl/server-cloud/products/sql-server/

*administradores de todos los niveles obtener acceso SQL Server"*[32].

SQL Magnament Studio se utiliza para interactuar con el sistema gestor de base de datos de manera gráfica, se podrá crear la Base de Datos del CHECKPOINT y las tablas que la conforman, así como también los procedimientos almacenados que operarán las consultas de datos que requiera el CHECKPOINT.

## 2.7 Programación de la Capa de Datos

La base de datos del CHECKPOINT aloja los registros de asistencia recolectados por el CHECKPOINT DOWNLOADER, la lógica de la base de datos fue implementada con referencia al modelo de base datos relacional, el autor Cesar Pérez menciona lo siguiente de las modelos de datos relacionales.

*"Lo primero que hay que tener presente a la hora de diseñar una base de datos relacional es el propio concepto de modelo relacional, que organiza los datos en una base de datos como una colección de tablas teniendo presente inicialmente los siguiente:*

- *Cada tabla tiene un nombre que la identifica únicamente.*
- *Cada tabla tiene cero o más columnas nominadas, que están dispuestas en un orden específico de izquierda a derecha.*
- *Cada tabla tiene cero o más filas, conteniendo cada una un único valor en cada columna. Las filas están desordenadas.*
- *Todos los valores de una columna determinada tienen el mismo tipo de datos y estos están extraídos de un de un conjunto de valores legales llamado el dominio de la columna"*[33].

Las tablas de la base de datos se relacionan con otras por los datos que contienen. El modelo de bases de datos relacional utiliza Claves Principales y Claves externas para representar estas relaciones entre tablas. A continuación se

[32] Usar SQL Server Magnament, Microsoft MSDN Studio http://msdn.microsoft.com/es-MX/library/ms174173.aspx

[33] César Pérez, MySQL para Windows y Linux 2° Edición, Diseño de una Base de Datos Relacional p.45

presenta el modelo de Base de Datos Relacional para el Sistema CHECKPOINT en la imagen 2.6.

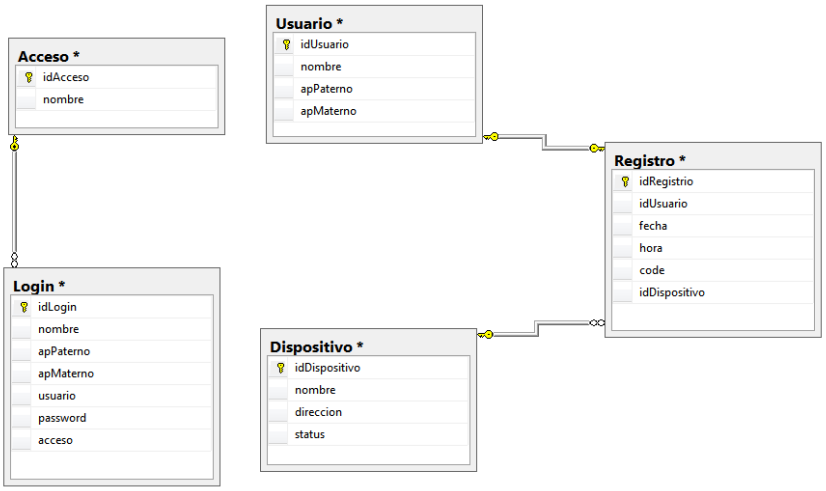



Imagen 2.6 Esquema de la base de datos del CHECKPOINT

### 2.7.1 Diccionario de datos

**Nombre de la Tabla:** Login

**Descripción:** Esta tabla contiene datos del control de acceso al Sistema CHECKPOINT

| Campo | Tipo | Tamaño | Descripción |
|---|---|---|---|
| idLogin | int | | Es el número consecutivo de identificación del usuario del sistema |
| nombre | nvarchar | 50 | Es el nombre del Usuario del sistema |
| apPaterno | nvarchar | 50 | Es el Apellido paterno del usuario del sistema |
| apMaterno | nvarchar | 50 | Es el Apellido Materno del usuario del sistema |
| usuario | nvarchar | 10 | Es la palabra con la que el usuario inicia sesión |
| password | nvarchar | 10 | Es la contraseña del usuario del sistema para iniciar sesión |
| acceso | int | | Es el nivel de acceso al sistema del |

| | | | |
|---|---|---|---|
| | | | usuario del sistema |

| | |
|---|---|
| **Relaciones:** acceso con Acceso | **Campos Clave:** idLogin |

**Nombre de la Tabla:** Usuario

**Descripción:** Esta tabla contiene datos requeridos por el CHECKPOINT para el control de acceso al sistema del usuario

| Campo | Tipo | Tamaño | Descripción |
|---|---|---|---|
| idUsuario | int | | Es el número consecutivo de identificación del usuario del sistema |
| nombre | nvarchar | 50 | Es el nombre del usuario del sistema |
| apPaterno | nvarchar | 50 | Es el Apellido paterno del usuario del sistema |
| apMaterno | nvarchar | 50 | Es el Apellido Materno del usuario del sistema |

| | |
|---|---|
| **Relaciones:** | **Campos Clave:** idUsuario |

**Nombre de la Tabla:** Dispositivo

**Descripción:** Esta tabla contiene datos del empleado que registra asistencia en dispositivos biométricos

| Campo | Tipo | Tamaño | Descripción |
|---|---|---|---|
| idDispositivo | int | | Es el número de identificación del dispositivo biométrico |
| nombre | nvarhchar | 50 | Es el nombre del dispositivo biométrico |
| direccion | nvarhchar | 50 | Es la dirección IP dispositivo biométrico |
| status | int | | Es el estatus del dispositivo |

| | |
|---|---|
| **Relaciones:** | **Campos Clave:** idDispositivo |

**Nombre de la Tabla:** Usuario

**Descripción:** Esta tabla contiene datos requeridos por el CHECKPOINT para el control de acceso al sistema del usuario

| Campo | Tipo | Tamaño | Descripción |
|---|---|---|---|
| idUsuario | int | | Es el nombre del empleado, usuario del dispositivo biométrico |
| nombre | nvarchar | 50 | Es el nombre del usuario del empleado, usuario del dispositivo biométrico |
| apPaterno | nvarchar | 50 | Es el Apellido paterno del usuario del empleado, usuario del dispositivo biométrico |
| apMaterno | nvarchar | 50 | Es el Apellido Materno del usuario del empleado, usuario del dispositivo biométrico |

| **Relaciones:** | **Campos Clave:** idUsuario |
|---|---|

**Nombre de la Tabla:** Registro

**Descripción:** Esta tabla contiene datos de la descarga de registros de los dispositivos biométricos

| Campo | Tipo | Tamaño | Descripción |
|---|---|---|---|
| idRegistro | Int | | Es el número consecutivo de identificación del registro de asistencia. |
| idUsuario | nvarchar | 50 | Es el número de identificación del empleado, usuario del dispositivo biométrico |
| Fecha | nvarchar | 50 | Es fecha del registro de asistencia |
| hora | nvarchar | 50 | Es la hora del registro de asistencia |
| code | nvarchar | 10 | Es el código de verificación del registro del sistema |
| idDispositivo | nvarchar | 10 | Es el número de identificación del |

| | | | |
|---|---|---|---|
| | | | dispositivo |

| | |
|---|---|
| **Relaciones:** idUsuario a Usuario<br>idDispositivo a Dispositivo | **Campos Clave:** idRegistro |

A continuación se crea la base de datos del CHECKPOINT dada la definición de su estructura y el tipo de datos que conforman las tablas del esquema de datos, para ello se requieren los datos de acceso (dirección IP del servidor de base de datos y la contraseña de acceso) del servidor donde se alojará la base de datos de SQL Server, para realizar este procedimiento se utilizó SQL Magnament Studio 2012 en modo gráfico como se observa en la imagen 2.7.

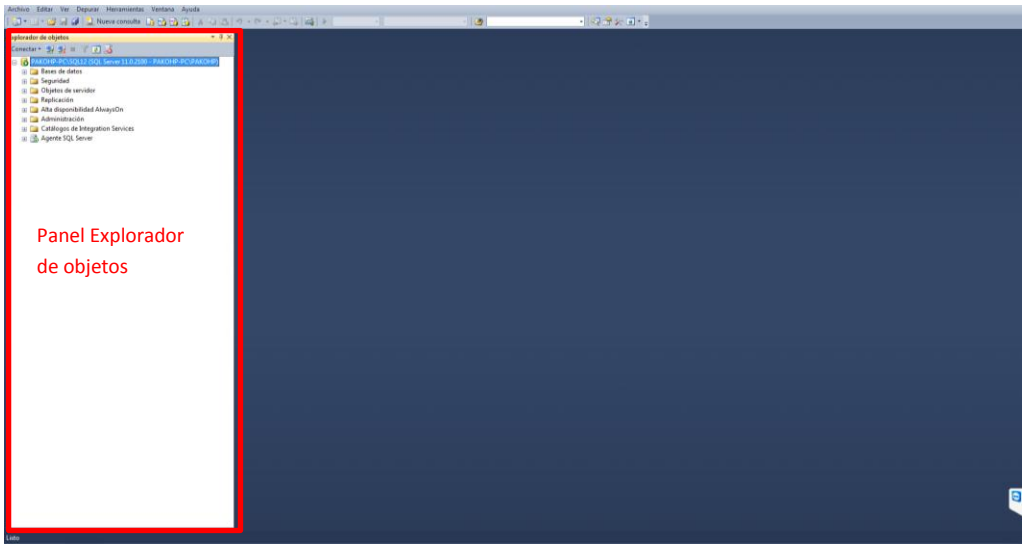



Imagen 2.7 Explorador de objetos SQL Magnament Studio

Para Comenzar con el procedimiento de creación de la base de datos se inicia SQL Magnament Studio, en la opción “Bases de datos” se da clic derecho y abre el menú de opciones, en ese menú se elige la opción “Nueva Base de datos” tal como se observa en la imagen 2.8.


Imagen 2.8 Creación de la base de dato

Al realizar el procedimiento anterior se muestra una ventana para crear la base de datos, para ello solo se escribe el nombre de la base en el cuadro de texto que se indica, se mantiene la configuración por defecto que se muestra en la ventana y se da clic en el botón aceptar plasmado en la imagen 2.9.

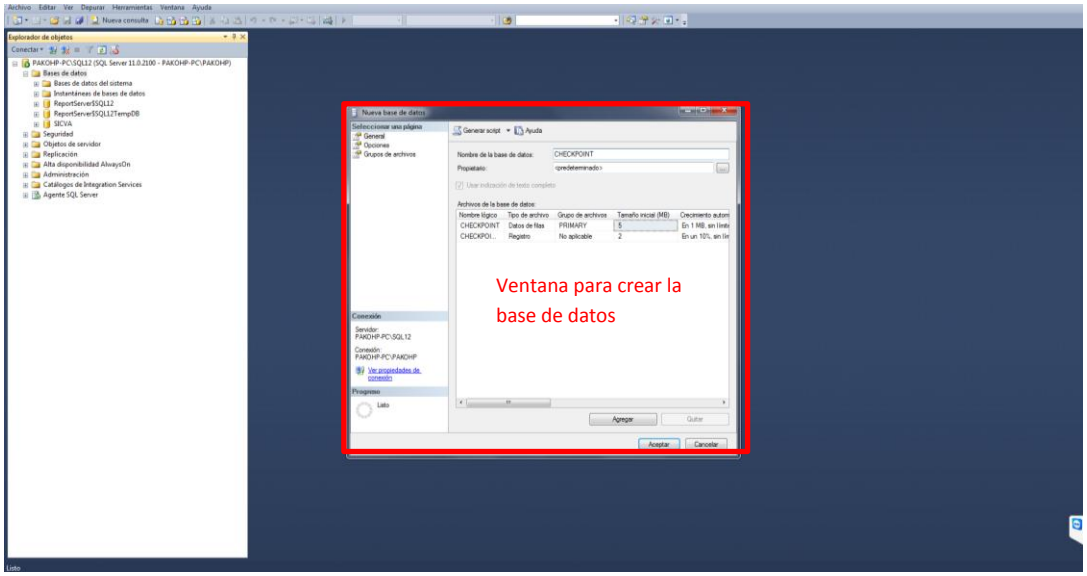



Imagen 2.9 Definición de la base de datos

Despues de concluir con los pasos antes mencionados se observa en el explorador de objetos de Magnament Studio que la Base de Datos CHECKPOINT fue creada exitosamente en la imagen 2.10.

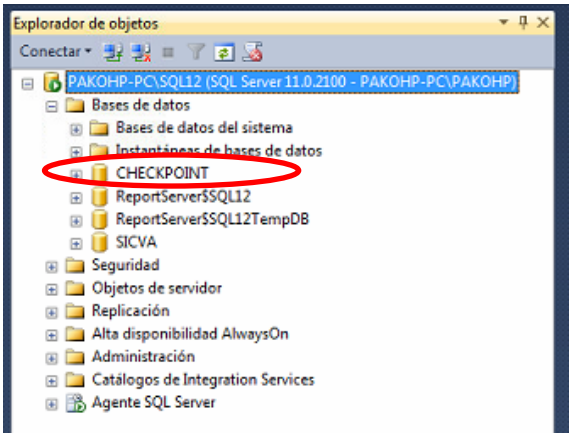



Imagen 2.10 Comprobación de la base de datos

Continuando con el desarrollo del CHECKPOINT se procede con la creación de las tablas, para ello se expande la base de datos y se despliega en un menú con una seria de opciones propias del esquema de datos, se da clic derecho sobre la opción tabla para abrir otro menú de opciones para finalmente se selecciona la opción de “Nueva tabla” que se aprecia en la imagen 2.11.

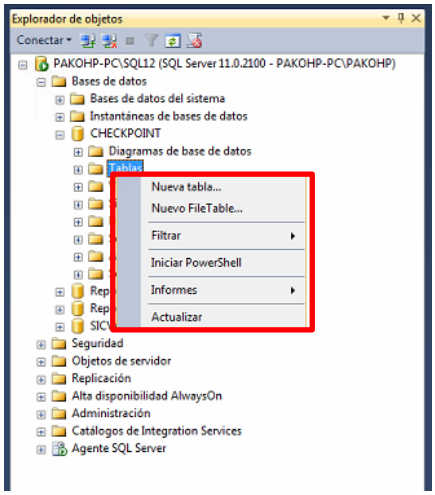



Imagen 2.11 Comprobación de la base de datos

A continuación se abre en el área de trabajo las propiedades para la creación de la tabla, se construye la tabla Login en base al análisis antes mencionado, se definen los nombres de los campos de la tabla y el tipo de dato de los mismos. En el caso del campo "idLogin" en la opción de "Especificación Identidad" del panel de "Propiedades de columna" el parámetro se marca en "Si" y el "Incremento de la Identidad" en uno, de acuerdo al procedimiento anterior esta configuración permitirá que el campo idLogin se incremente cada vez que se ingrese un registro a la tabla conforme al análisis del diccionario de datos, la configuración se aprecia en la imagen 2.12.

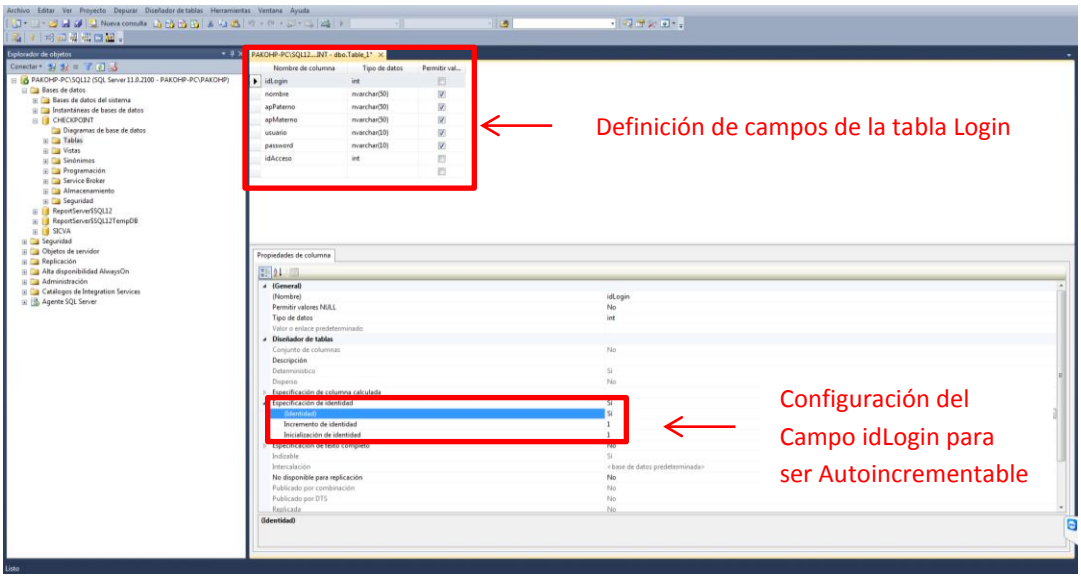



Imagen 2.12 Definición de la tabla "Login"

El campo idLogin además de ser autoincrementable también tiene la propiedad de ser Clave Principal de la tabla, se define de la siguiente manera

> *"Una Clave Principal es una columna o combinación de columnas dentro de una tabla, cuyos valores identifican unívocamente a cada fila de la tabla. Cada tabla tiene una única clave Principal"*[34].

[34] César Pérez, MySQL para Windows y Linux 2° Edición, Diseño de una Base de Datos Relacional p.45

Las claves principales sirven para representar las relaciones entre cada tabla, en este caso se requiere configurar el esquema de la base de datos del CHECKPOINT en el campo “idLogin” como Clave principal de la tabla “Login”, para realizarlo se abre el “Diseño de la tabla” y se da clic sobre el campo en cuestión, se despliega un menú con la propiedades del campo y se elige la opción “Establecer como clave principal”, se observa que al realizar el procedimiento en el campo se colocó una signo de llave, lo que significa que el campo ya es reconocido como  Clave Principal de la tabla ilustrado en la imagen 2.13.

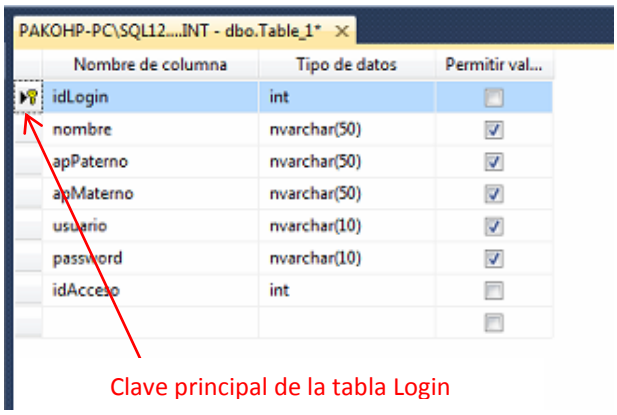



Imagen 2.13 Clave principal de la tabla “Login”

Para concluir la creación de la tabla Login en la pestaña superior se da clic derecho, se despliega el menú de opciones de tabla y se elige la opción de “Guardar Tabla”, el Gestor de base de datos solicita escribir el nombre de la tabla tal cual se aprecia en la imagen 2.14, que en esta caso se nombró Login.

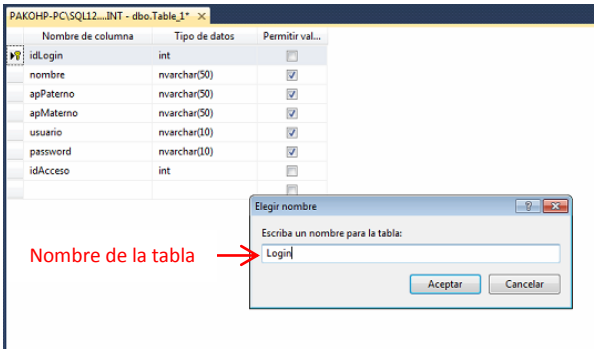



Imagen 2.14 nombre de la tabla “Login”

Se da clic en el boton aceptar y se cierra el espacio de trabaja de la definición de la estructura de la tabla Login. Finalmente el “Explorador de Objetos” en el árbol de la Base de datos CHECKPOINT se observa la tabla que se creo exitosamente que se comprueba en la imagen 2.15.

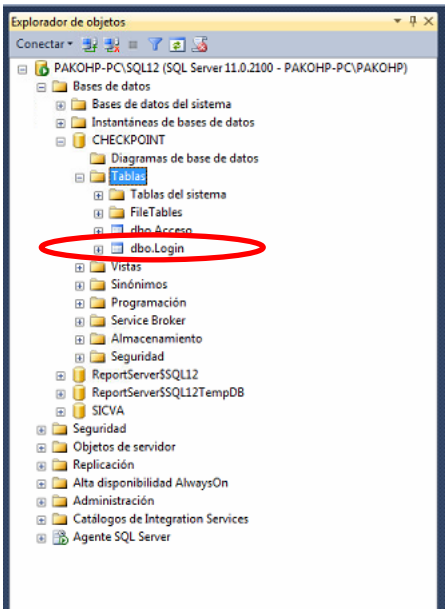



Imagen 2.15 Comprobación de la tabla “Login”

Una vez creada la tabla Login se relaciona con la tabla Acceso con el campo en “idAcceso” para cumplir con el esquema planteado de la base de datos, cabe mencionar que este campo determina el perfil del usuario del sistema. La tabla Acceso contiene el Campo con Clave Principal y la Tabla Login contiene el campo con Clave Externa, por ello se toma como base la tabla Login.

En el Explorador de Objetos de Magnament Studio se da clic derecho sobre la tabla Login y se selecciona la opción “Diseño”, se abrirá nuevamente el área de trabajo de la definición de datos utilizado en la creación de la tabla “Login”, se da clic derecho sobre el campo idAcceso para abrir el menú de propiedades del campo, y se elige la opción de “Relaciones”, se abrirá la ventana que se muestra en la imagen 2.16.

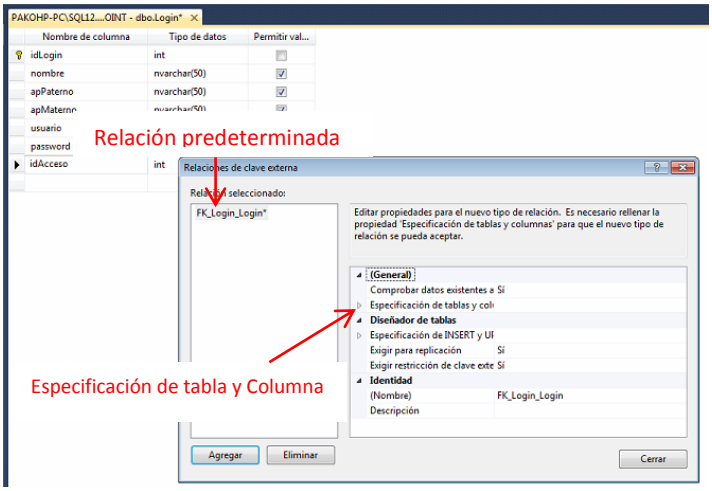



Imagen 2.16 Relación de la tabla “Login” con la tabla Acceso

Posteriormente se da clic en el botón agregar y se genera una relación predeterminada con el nombre FK_Login_Login*, en el menú de propiedades se selecciona la opción de “Especificación de tabla y columna”, se despliega la siguiente ventana de la imagen 2.17.

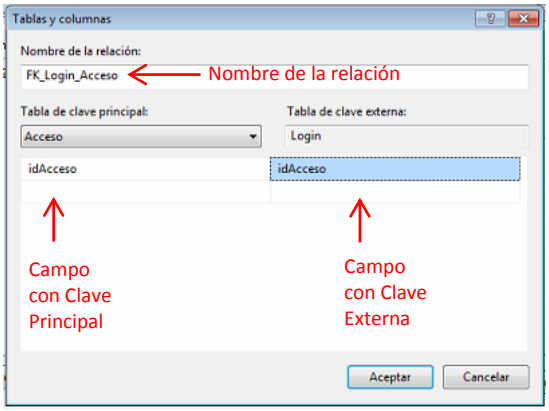



Imagen 2.17 Especificación de la relación de las tablas “Login” respecto “Acceso”

En la imagen 2.17 se observa la configuración requerida para relacionar las tablas de la base de datos, el procedimiento consta de seleccionar la tabla con el Campo Clave Principal, en este caso la tabla es Acceso y el Campo Clave es idAcceso, después se selecciona el Campo Clave Externo, es decir, se selecciona el campo idAcceso de la Tabla Login, para concluir con este procedimiento se da clic en el botón aceptar, internamente el Sistema Gestor de Base de datos interpreta la

relación de ambas tablas. El procedimiento para crear el resto de las tablas y las relaciones con otras tablas de la base de datos del Sistema se omitirá, puesto que los pasos realizados para la creación de la tabla “Login” y la relación con la tabla “Acceso” es el mismo, con ello se concluye el análisis y desarrollo de la base de datos del Sistema CHECKPOINT.

### 2.7.2 Procedimientos Almacenados del CHECKPOINT

Los procedimientos almacenados son un conjunto de sentencias SQL que se ejecutan en el Servidor de Base de Datos, en el caso de SQL Server este tipo de programación se basa en el lenguaje T-SQL (Transact SQL). Para la creación de un Procedimiento en SQL Server se requiere de SQL Magnament Studio, en el Explorador de Objetos se expanden los apartados de la Base de Datos, en este caso CHECKPOINT y se da clic sobre “Programación”, así mismo se expande un submenú de opciones, se da clic derecho a la opción “Procedimientos Almacenados” que se muestra a continuación, finalmente se da clic en la opción “Nuevo procedimiento almacenado”.

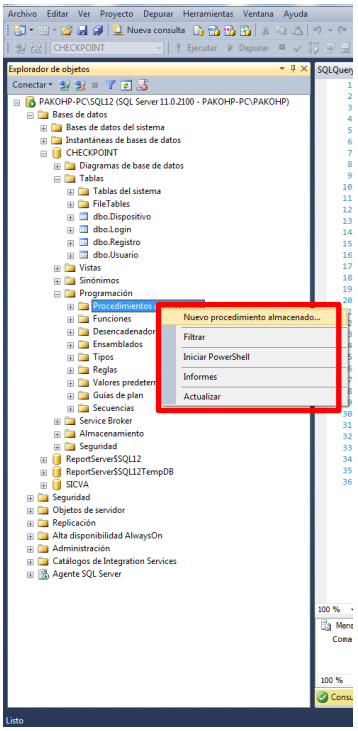



Imagen 2.18 Creación de Procedimientos Almacenados SQL Magnamente Studio

En el área de trabajo se abre una plantilla de un procedimiento almacenado por omisión de Magnament Studio, esta plantilla de modifica de acuerdo a las necesidades de programación requeridas. La estructura del procedimiento almacenado es simple, primero se declara el nombre del procedimiento almacenado, enseguida se declaran los parámetros o resultados de entrada y/o salida de acuerdo al tipo de dato declarado, los parámetros de salida se declaran con la palabra reservada “OUTPUT”, estos parámetros son utilizados por el conjunto de comandos SQL de consulta, modificación y borrado y operaciones lógicas y aritméticas necesarias para cumplir las funciones del procedimiento almacenado. La estructura general del Procedimiento almacenado se observa en la imagen 2.19.

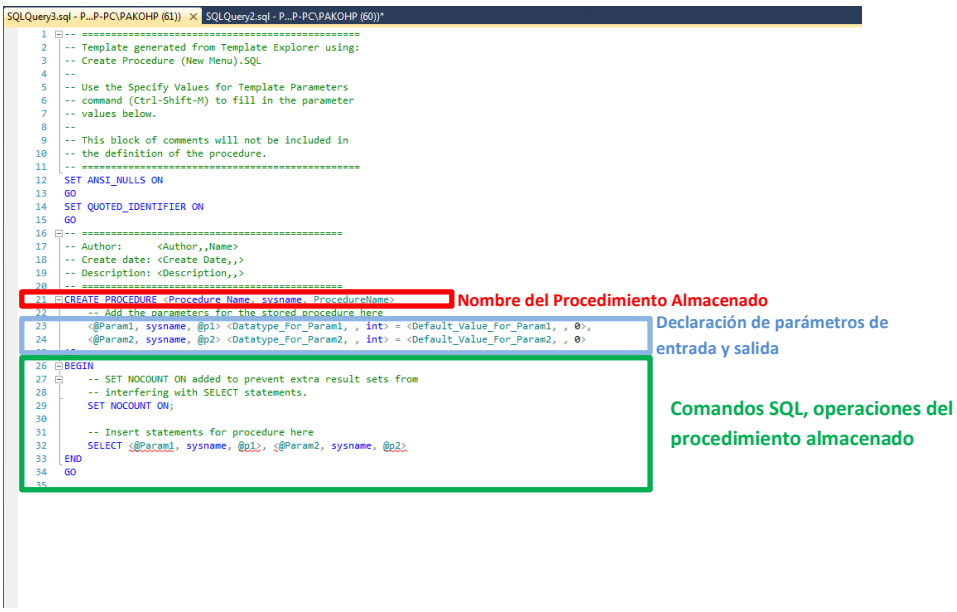



Imagen 2.19 Estructura general de un procedimiento almacenado en SQL Server

Para el caso del CHECKPOINT se creó el procedimiento almacenado “insertaRegistro” para la inserción de registros de asistencia de los dispositivos biométricos, se declararon parámetros de entrada que corresponden a los datos del registro de asistencia y un parámetro de salida que indica el resultado de la operación en caso de ser exitosa retorna un valor de uno, se utilizó la estructura de control de manejo de excepciones TRY-CATCH para intentar insertar un registro de asistencia con el comando de inserción, en caso de no realizarlo

genera una excepción y retorna un valor de cero que indica que no se insertó el registro, la estructura de este procedimiento almacenado se aprecia en la imagen 2.20.

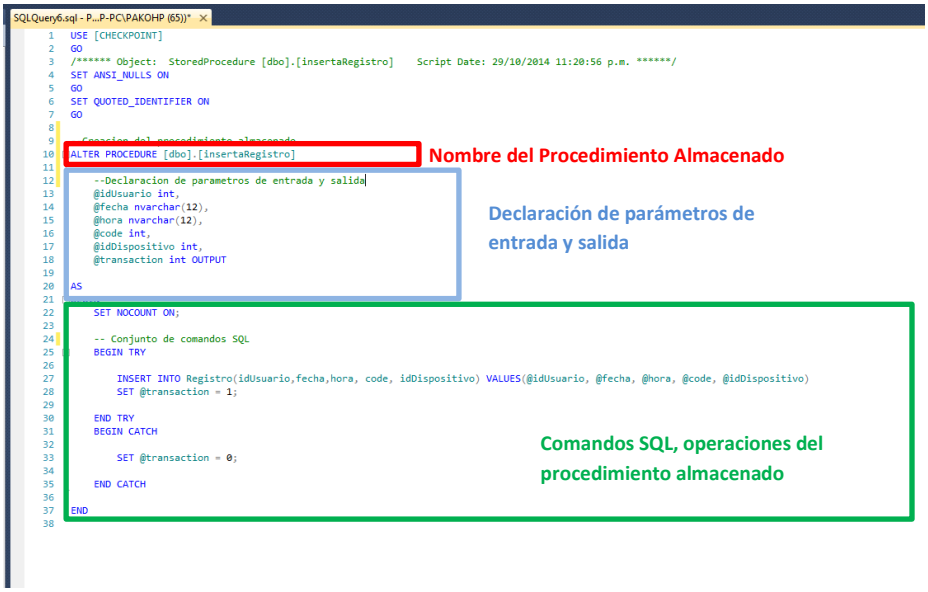



Imagen 2.20 Código del Procedimiento almacenado “insertaRegitros”

El Sistema CHECKPOINT requirió 19 procedimientos almacenados, de manera general se muestra el desarrollo en cada uno de ellos, se observa que la estructura del código es similar a la del procedimiento “insertaRegistro”.

- **Procedimiento Almacenado “access”**

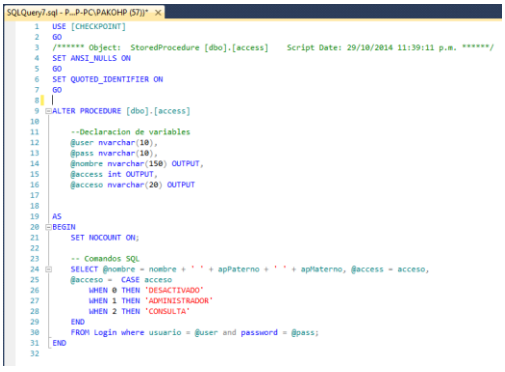


Imagen 2.21 Código del Procedimiento almacenado “access”

Este procedimiento almacenado es utilizado para el Control de Acceso al Sistema CHECKPOINT, retorna un valor numérico en caso de que se autentifique al usuario del sistema.

- **Procedimiento Almacenado "actualizaDispositivo"**

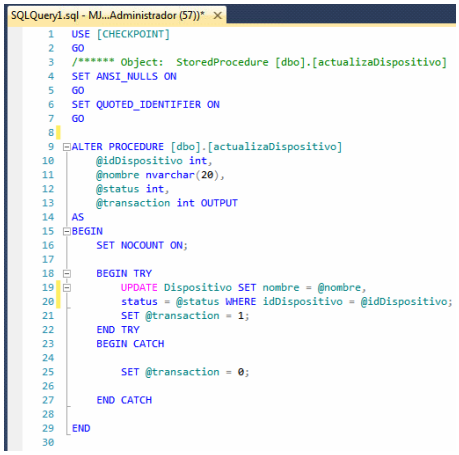



Imagen 2.22 Código del Procedimiento almacenado "actualizaDispositivo"

Este procedimiento almacenado se encarga de actualizar el nombre y el estado del dispositivo biométrico desde el panel de control CHECKPOINT VIEWER.

- **Procedimiento Almacenado "actualizaLogin"**

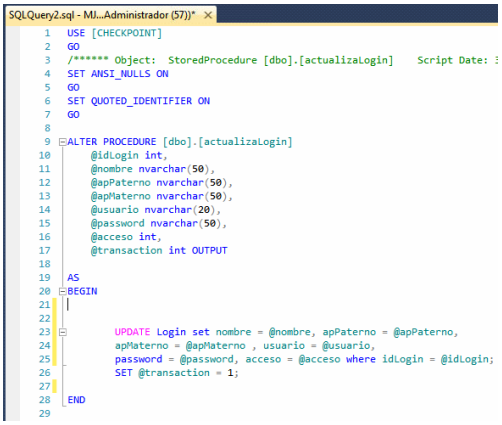



Imagen 2.23 Código del Procedimiento almacenado "actualizaLogin"

Este procedimiento almacenado se encarga de actualizar los datos de acceso de los Usuario al sistema con el CHEKPOINT VIEWER.

- **Procedimiento Almacenado “actualizaUsuario”**

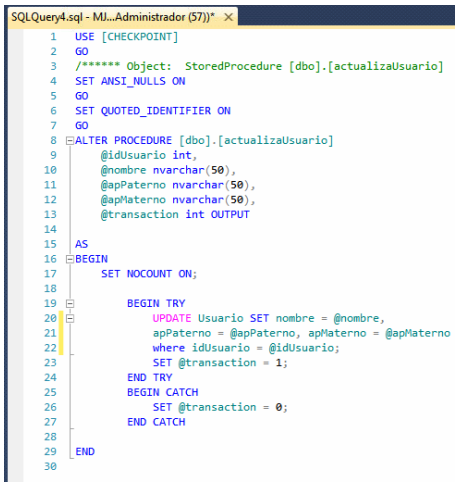



Imagen 2.24 Segmento de código del Procedimiento almacenado “actualizaUsuario”

Este procedimiento almacenado se encarga de actualizar los datos del control de acceso de las cuentas de usuario, utilizado en CHECKPOINT VIEWER.

- **Procedimiento Almacenado “buscaUsuario”**

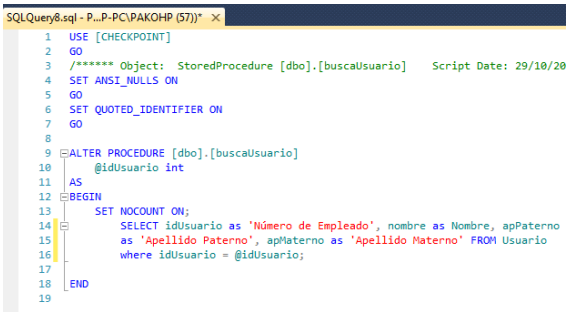



Imagen 2.25 Código del Procedimiento almacenado “buscaUsuario”

Este procedimiento almacenado tiene la función de retornar el nombre, el apellido paterno, el apellido materno de un empleado de acuerdo al número de empleado.

- **Procedimiento Almacenado "consultaAllDev"**

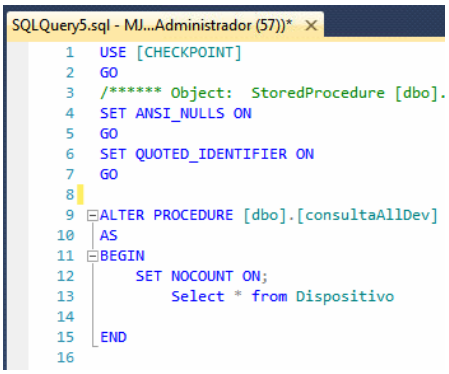



Imagen 2.26 Código del Procedimiento almacenado "consultaAllDev"

El procedimiento almacenado se encarga de consultar la tabla donde se registró a los dispositivos biométricos registrados sin importar si están o no habilitados, este procedimiento se utiliza en el CHECKPOINT DOWNLOADER, también es requerido por el CHECKPOINT VIEWER.

- **Procedimiento Almacenado "consultaDispositivo"**

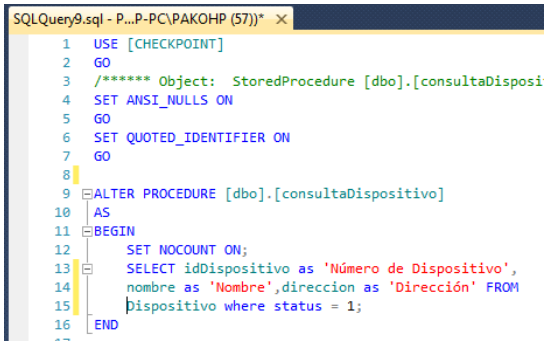



Imagen 2.27 Código del Procedimiento almacenado "consultaDispostivo"

Esl

para realizar la conexión con los dispositivos biométricos habilitados, es utilizado por el CHECKPOIN DOWNLOADER.

- **Procedimiento Almacenado “consultaLogin”**

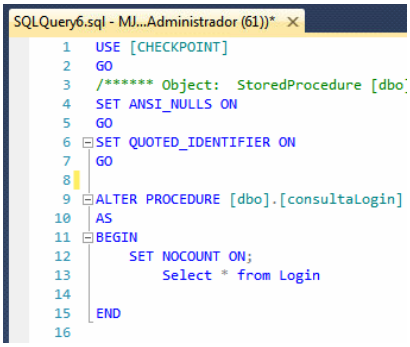



Imagen 2.28 Código del Procedimiento almacenado “consultaLogin”

Este procedimiento almacenado se encarga de retornar los datos de todas las cuentas de usuarios del sistema en el CHECKPOINT VIEWER.

- **Procedimiento Almacenado contador**

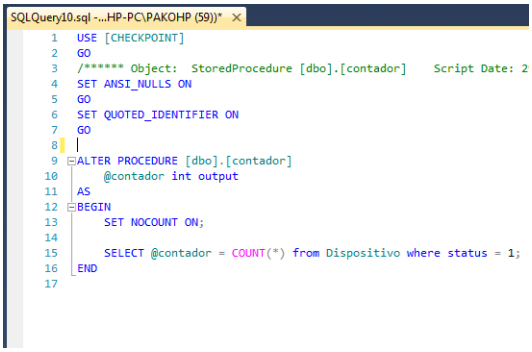



Imagen 2.29 Código del Procedimiento almacenado “contador”

Este procedimiento almacenado realiza el conteo de los dispositivos biométricos que se encuentran habilitados, esto lo lleva a cabo el CHECKPOIN DOWNLOADER.

- **Procedimiento Almacenado “cuentaFecha”**

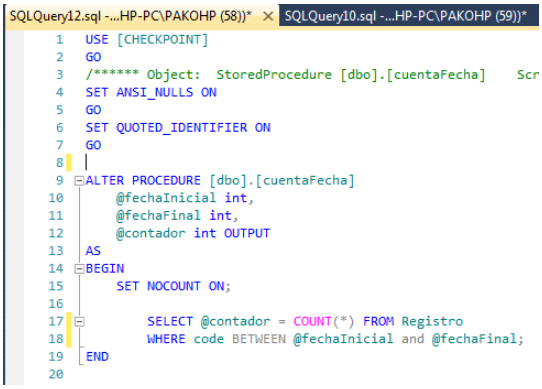



Imagen 2.30 Código del Procedimiento almacenado “cuentaFecha”

Este procedimiento almacenado es responsable realizar el conteo de registros de asistencia en un periodo de fechas determinados, se utiliza en el CHECKPOINT VIEWER.

- **Procedimiento Almacenado “cuentaRegUser”**

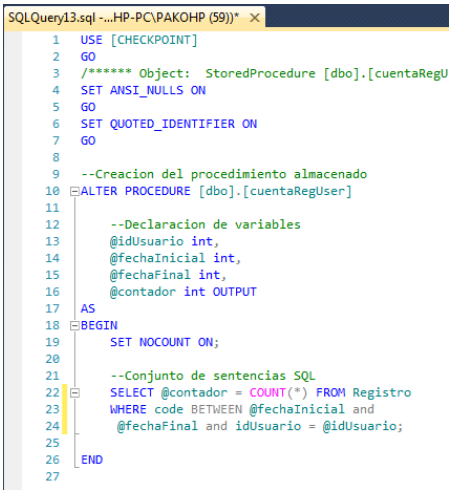



Imagen 2.31 Código del Procedimiento almacenado “cuentaRegUser”

Este procedimiento almacenado tiene la función de realizar el conteo de registros de asistencia por rango de fechas y número de empleado determinados por el usuario del sistema.

- **Procedimiento Almacenado “insertaBack”**

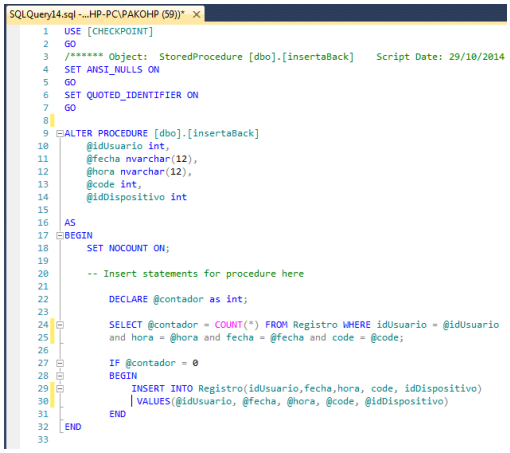



Imagen 2.32 Código del Procedimiento almacenado “insertaBack”

Este procedimiento almacenado inserta los registros de asistencia que son almacenados en la memoria interna del dispositivo, en caso de que ya existan no los inserta, es utilizado por el CHECKPOINT DOWNLOADER.

- **Procedimiento Almacenado “insertaDispositivo”**

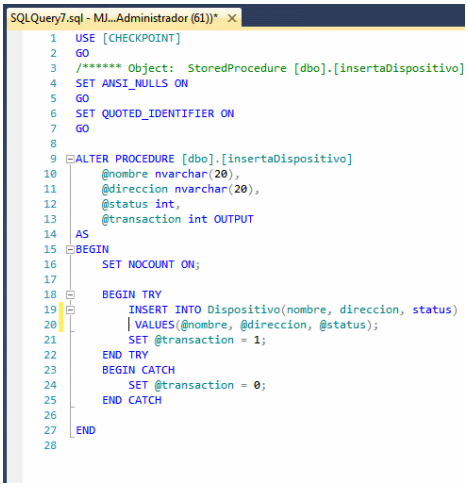



Imagen 2.33 Código del Procedimiento almacenado “insertaBack”

Este procedimiento almacenado registra nuevos dispositivos biométricos, inserta los datos de configuración de cada dispositivo, se utiliza en el CHECKPOINT VIEWER.

- **Procedimiento Almacenado “insertaLogin”**

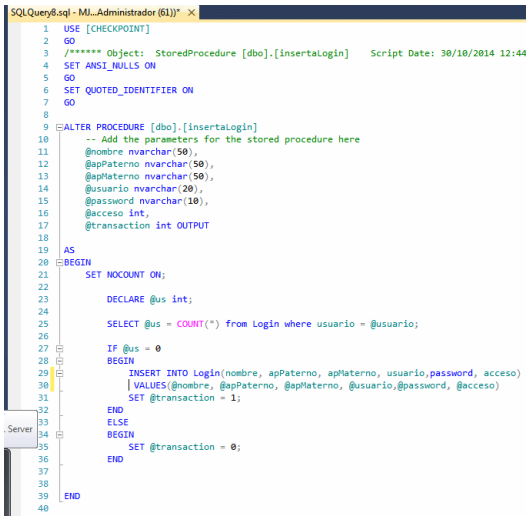



Imagen 2.34 Código del Procedimiento almacenado “insertaLogin”

Este procedimiento almacenado inserta usuarios nuevos a la tabla Login para control de acceso del sistema CHECKPOINT VIEWER.

- **Procedimiento Almacenado “listUsuario”**

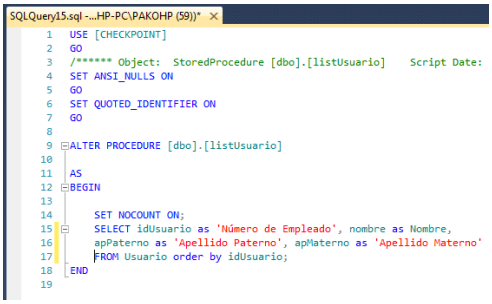



Imagen 2.35 Código del Procedimiento almacenado “listUsuario”

Este procedimiento almacenado consulta los usuarios del sistema, los ordena por número de registro en el CHECKPOINT VIEWER.

- **Procedimiento Almacenado “registrarUsuario”**

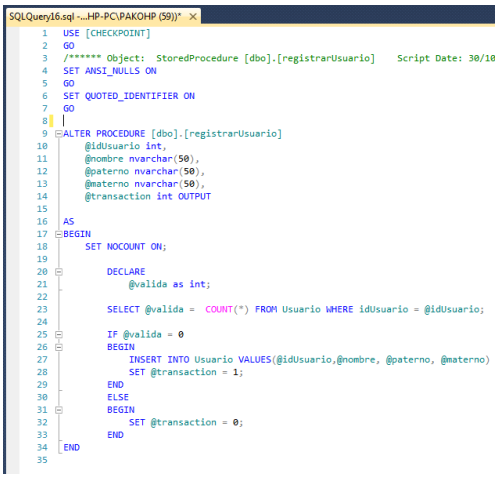



Imagen 2.36 Código del Procedimiento almacenado “listUsuario”

Este procedimiento almacenado se encarga de registrar a un nuevo empleado que se incluya en el procedimiento de asistencia, se utiliza en el CHECKPOINT VIEWER.

- **Procedimiento Almacenado “registroFecha”**

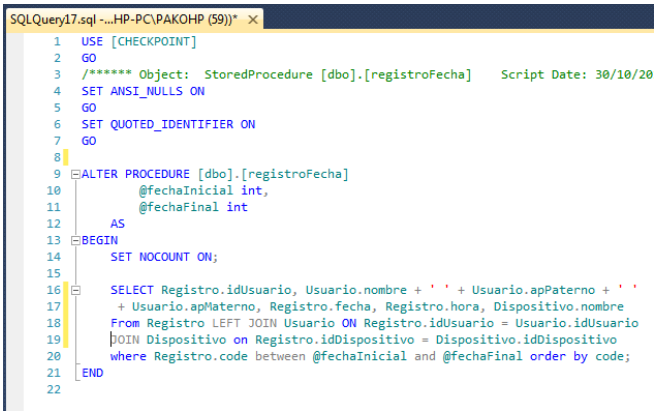
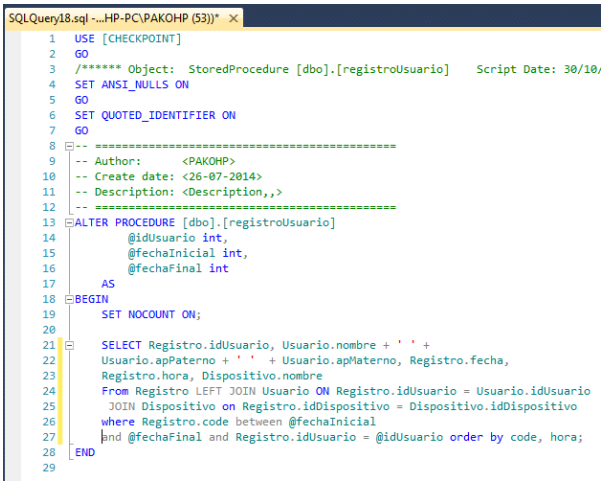



Imagen 2.37 Código del Procedimiento almacenado “registroFecha”

Este procedimiento almacenado consulta los registros de asistencia de todos los empleados en general por rangos de

fecha establecidos por el usuario del sistema, es requerido por el CHECKPOINT VIEWER.

- **Procedimiento Almacenado "registroUsuario"**



```
SQLQuery18.sql -...HP-PC\PAKOHP (53))*
1  USE [CHECKPOINT]
2  GO
3  /****** Object:  StoredProcedure [dbo].[registroUsuario]    Script Date: 30/10/
4  SET ANSI_NULLS ON
5  GO
6  SET QUOTED_IDENTIFIER ON
7  GO
8  -- =============================================
9  -- Author:      <PAKOHP>
10 -- Create date: <26-07-2014>
11 -- Description: <Description,,>
12 -- =============================================
13 ALTER PROCEDURE [dbo].[registroUsuario]
14         @idUsuario int,
15         @fechaInicial int,
16         @fechaFinal int
17     AS
18 BEGIN
19     SET NOCOUNT ON;
20
21     SELECT Registro.idUsuario, Usuario.nombre + ' ' +
22     Usuario.apPaterno + ' '  + Usuario.apMaterno, Registro.fecha,
23     Registro.hora, Dispositivo.nombre
24     From Registro LEFT JOIN Usuario ON Registro.idUsuario = Usuario.idUsuario
25      JOIN Dispositivo on Registro.idDispositivo = Dispositivo.idDispositivo
26     where Registro.code between @fechaInicial
27     and @fechaFinal and Registro.idUsuario = @idUsuario order by code, hora;
28 END
29
```

Imagen 2.38 Código del Procedimiento almacenado "registroUsuario"

El procedimiento almacenado "registroUsuario" consulta los registros de asistencia de un empleado por rangos de fecha y el número de empleado, este procedimiento almacenado forma parte del historial de registros de asistencia del CHECKPOINT VIEWER.

Al concluir el desarrollo de la base de datos y de los procedimientos almacenados se finaliza la programación de la capa de datos, para dar continuidad con el flujo del desarrollo del CHECKPOINT con la Capa de Negocio.

## 2.8 Programación de la Capa de Negocio

La Capa de Negocio, definida anteriormente, es la interfaz de conexión entre la Capa de Datos y la Capa de Presentación, está diseñada para realizar las operaciones aritméticas y lógicas del Sistema, es aplicable para el CHECKPOINT DOWNLOADER y para el CHECKPOINT VIEWER, así mismo, es la responsable de conectarse con los dispositivos biométricos ZK SOFTWARE y realizar todo lo

relacionado con la descarga de los registros, la consulta de registros de asistencia y de la gestión de los dispositivos.

La Capa de Lógica de Negocio fue desarrollada con el lenguaje C# de Microsoft .NET con el Entorno de Desarrollo Visual Studio 2010, a continuación se describe el procedimiento realizado.

La estructura del Sistema CHECKPOINT está basada en la programación orientada objetos, este tipo de programación se sustenta en la representación de objetos y métodos de un mismo tipo como se hace en el mundo real, el paradigma de programación objetos lo define el autor Francisco Javier Ceballos de la siguiente manera.

> *"La programación orientada a objetos es un modelo de programación que utiliza objetos ligados mediante mensajes, para la solución de problemas. La idea central es simple: es organizar los programas a imagen y semejanza de los objetos de mundo real"*[35].

El modelo de programación orientada a objetos se utilizó para desarrollar las clases que forman la lógica de negocio de la aplicación, el desarrollo se llevó a cabo con Visual Studio 2010, se crearon dos proyectos de Windows Forms llamadas CHECKPOINT VIEWER y CHECKPOINT DOWNLOADER, para el caso práctico se describe el procedimiento del desarrollo de la clase "Device" del CHEKCPOINT DOWNLOADER, para ello se definen algunos conceptos.

> *"Una clase equivale a la generalización de un tipo específico de objetos, pero cada objeto que se construya de cada clase tiene sus propios datos"*[36].

> *"un objeto consta de un estado y de un comportamiento, que a su vez constan respectivamente de datos almacenados y*

[35] Ceballos, Francisco Javier, C# Lenguaje y Aplicaciones p. 27

[36] Ceballos, Francisco Javier, C# Lenguaje y Aplicaciones p. 28

*de tareas realizables durante el tiempo de ejecución. Un objeto puede ser creado instanciando de una clase, como ocurre en la programación orientada a objetos, o mediante escritura directa de código y la replicación otros objetos"*[37].

*"Un programa orientado a objetos se compone solamente de objetos. Cada uno de ellos es una entidad que tiene ciertas propiedades particulares, atributos, y unas formas de operar sobre ellos, los métodos"*[38].

Una clase es un plantilla donde se crean objetos, al crear un objeto se invoca a la clase, estos objetos tienen ciertas características llamadas atributos y pueden ser modificados por métodos declarados en la misma clase.

En Visual Studio se crea un proyecto del tipo Windows Forms, en este caso se utiliza la versión 3.5 del Framework de .NET, se nombra a la solución CHECKPOINT como se demuestra en la imagen 2.39.

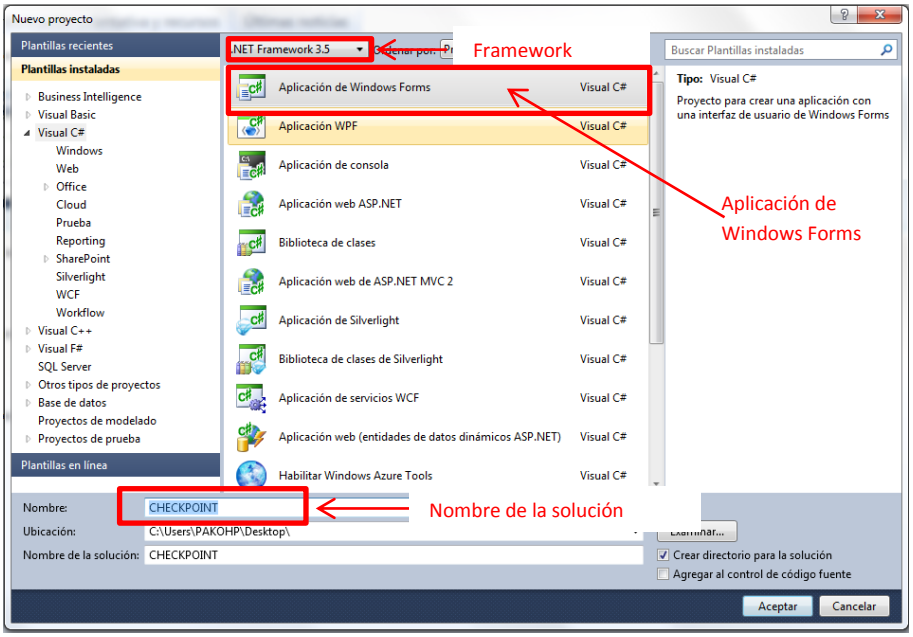



Imagen 2.39 Creación del Proyecto CHECKPOIN en Visual Studio

[37] Ibídem, p.28

[38] Ibídem, p.28

La solución Windows Forms se crea predeterminadamente con una ventana y ciertas propiedades de configuración, más adelante se retoma este tema dado que es parte de la capa de presentación, por ahora en el "Explorador de Soluciones", ubicado en la parte superior derecha, específicamente en el nombre del proyecto se da clic derecho y se despliega un menú, se debe seleccionar la opción de "Agregar Clase", la cual posteriormente se nombra "Device", esta clase es la base para creación de objetos del tipo Device, en ella se encuentran métodos como la conexión con el dispositivo biométrico, la descarga de registros del dispositivo biométrico, la configuración del dispositivo biométrico por mencionar los más importantes, en el área de trabajo se abre una plantilla de una clase predeterminada con referencia a ciertas librerías de configuración de la aplicación Windows Forms.

La imagen 2.40 muestra la plantilla mencionada y la definición de los Atributos y de los Métodos de la Clase que ejecutan las operaciones relacionadas con la conexión, funcionamiento y gestión del dispositivo biométrico ZK Software.

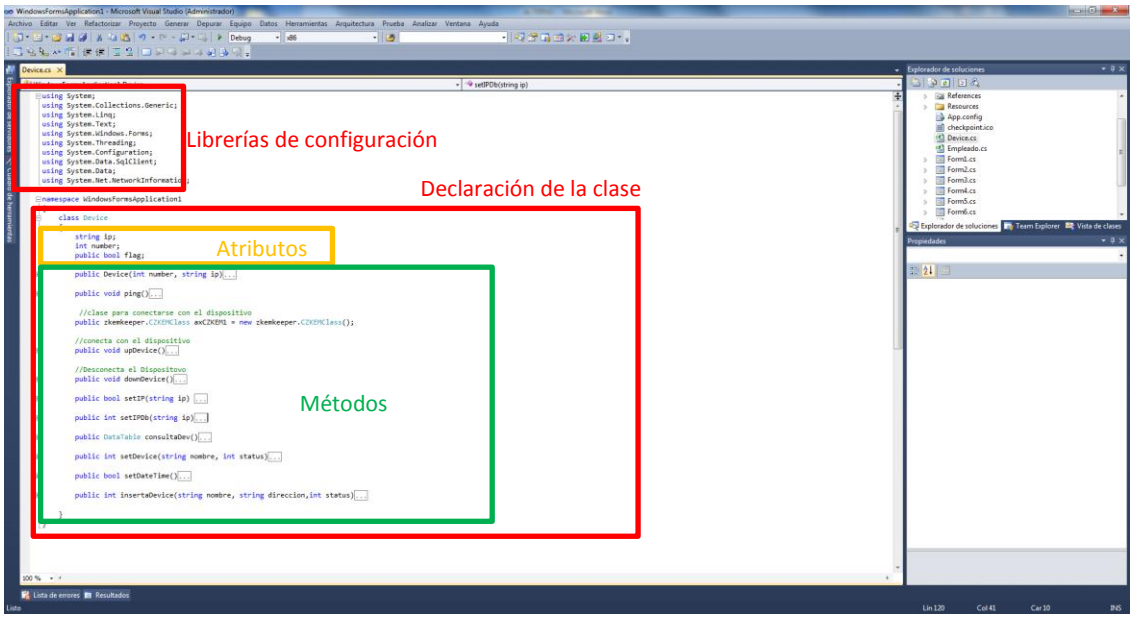



Imagen 2.40 Declaración de la clase "Device" en Visual Studio

Las operaciones del Sistema respecto a los dispositivos biométricos requieren del SDK de ZK Software, la versión que se utilizó para el desarrollo de esta es la 6.2.4.1, disponible en la página oficial de ZK Software, el SDK se trata de una librería "dll" que contiene las funciones necesarias para la comunicación con los dispositivos biométricos.

> *"Un archivo dll es una biblioteca que contiene el código y los datos que se pueden utilizar por más de un programa al mismo tiempo. Por lo tanto, cada programa puede usar la funcionalidad contenida en este archivo dll para implementar un cuadro de diálogo Abrir. Esto ayuda a promover la reutilización de código y el uso eficaz de la memoria"*[39].

La librería del SDK se incluye en el proyecto Visual Studio del CHECKPOINT, basta con ubicar en equipo de cómputo el archivo dll mediante el menú de "Referencias" del "Explorador de Soluciones", para comprobar que el SDK se incluyó basta con expandir el árbol de opciones del CHECKPOINT en el "Explorador de Soluciones" como se demuestra en la imagen 2.41.

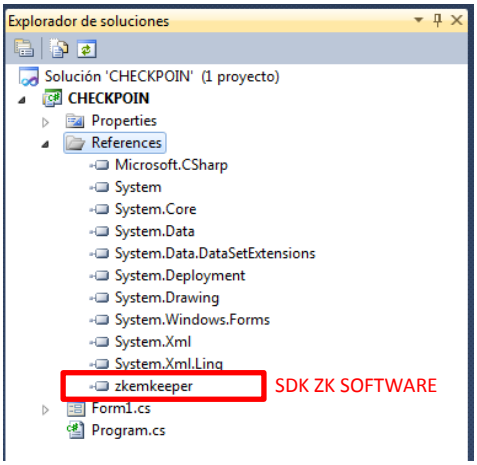



Imagen 2.41 Inserción del SKD ZK Software

[39] ¿Qué es un archivo dll?, Microsoft Soporte http://support2.microsoft.com/kb/815065/es

Al incluir el SDK en el proyecto basta con insertar la siguiente línea de código para comenzar a utilizar las funciones del mismo, se observa en la imagen 2.42.

```
//metodo para conectarse con el dispositivo
private zkemkeeper.CZKEMClass axCZKEM1 = new zkemkeeper.CZKEMClass();
```

Imagen 2.42 Línea de código para invocar el SDK ZK Software

Una vez realizado el procedimiento anterior se pueden programar los métodos de la clase para dar la funcionalidad al sistema CHECKPOINT, a partir de ello se crea el método “upDevice”, este método tiene la función de conectarse con el dispositivo biométrico a través de la red, el procedimiento consta de declarar el método “upDevice”, a continuación se declaran variables para la validación de la conexión, enseguida se declara la línea de código que intenta conectarse con el dispositivo biométrico, si se realiza la conexión por primera vez invoca el método de descarga de registros de asistencia almacenados en la memoria interna del dispositivo, posteriormente modifica el estado del dispositivo en conectado, finalmente se invoca el método de descarga de registros de asistencia en tiempo real, el código del método “UpDevice” se observa en la imagen 2.43.

```
//conecta con el dispositivo
private void upDevice() {
    //variable de validacion de conexion
    bool statusConexion = false;
    //variable del mensaje de conexion
    string message = string.Empty;

    //conexion con el dispositivo biometrico
    statusConexion = axCZKEM1.Connect_Net(this.ip, 4370);

    //validacion de la conexion
    if (statusConexion)
    {
        //verifica si es la primera vez que se conecta
        if (this.contador == 0)
        {
            try
            {
                //intenta descargar los registros de la memoria de dispositivo biometrico
                download();
            }
            catch
            {
                //habilita el dispositivo bimetrico
                axCZKEM1.EnableDevice(this.number, true);//enable the device
            }

        }

        //invoca al metodo de descarga de registros en tiempo real
        if (axCZKEM1.RegEvent(this.number, 65535))
        {
            this.axCZKEM1.OnAttTransactionEx += new zkemkeeper._IZKEMEvents_OnAttTransactionExEventHandler(axCZKEM1_OnAttTransactionEx);
            this.mensaje = "Conectado";

        }

    }
    else
    {
        //envia el mensaje en caso de no conectarse con el dispositivo bimetrico
        this.mensaje = "No conectado";
    }

}
```

Imagen 2.43 Código del método “UpDevice”

El método con mayor trascendencia de la clase “Device” es el método de descarga de registros de asistencia de los dispositivos biométricos en tiempo real, este método se auxilia de una cadena conexión a la base de datos, la cadena contiene el nombre con el que se invoca en este caso es “CPoint”, seguido de la dirección IP del servidor de la base de datos para el CHECKPOINT es “192.168.198.67”, el nombre de la base de datos, el usuario “sa” y la contraseña de acceso al servidor de base de datos y la librería que abre la comunicación entre el cliente y el servidor, esta línea de código se incrusta en un archivo de configuración (app.config) del proyecto de Visual Studio, esta cadena se observa en la imagen 2.44 .

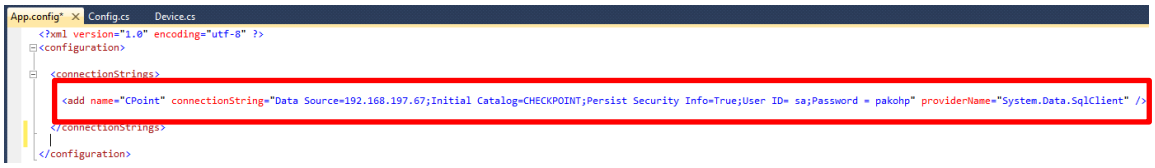



Imagen 2.44 Cadena de conexión a la base de datos del archivo app.config”

El archivo de configuración (app.config) es una archivo del tipo XML, se utiliza en el caso del CHECKPOINT para para realizar la conexión a la base de datos, el archivo tiene la capacidad de reconocer cadenas de conexión a base datos unicamente escribiendo los parametros de conexión, el mayor beneficio que ofrece este tipo archivos es la seguridad con la que maneja la cadena de conexión, esta cadena no se espone dado que se encapsula, este archivo se invoca en cada Clase del proyecto donde se consulte, inserte o modifique la base de datos.

El método de descarga de registros de asistencia en tiempo real (imagen 2.45) para realizar sus operaciones además de la conexión a la base de datos requiere los parametros de número de empleado, la fecha y la hora de registro de asistencia, invoca al procedimiento almacenado “insertaRegistro”, si todo esta configurado de acuerdo a los parámetros establecidos se envian los datos al procedimiento almacenado y se inserta en la tabla “Registro” el registro de asistencia del empleado, si la transacción se realiza exitosamente se retorna un

valor numerico mayor a cero, en caso contrario no se inserta en la base de datos, queda almacenado en el dispositivo biométrico y se insertará hasta que la aplicación se vuela a conectar con el dispositivo.

```
//Registra los eventos en tiempo real
private void axCZKEM1_OnAttTransactionEx(string sEnrollNumber, int iIsInValid, int iAttState, int iVerifyMethod, int anio, int mes, int dia, int Hora, int minuto, int segundo, int codigo)
{
    //declaracion de numero de empleado
    int idUsuario = Convert.ToInt32(sEnrollNumber);

    //declaracion de la hora registrada del evento
    DateTime hour = Convert.ToDateTime(Hora + ":" + minuto + ":00");

    string hora = hour.ToString("HH:mm:ss");

    DateTime date = Convert.ToDateTime(dia + "/" + mes + "/" + anio);

    string fecha = date.ToString("dd/MM/yyyy");

    string cod = date.ToString("yyyyMMdd");

    //intenta la conexion a la base de datos del CHECKPOINT
    string conexionString = ConfigurationManager.ConnectionStrings["CPoint"].ConnectionString;
    SqlConnection sql = new SqlConnection(conexionString);
    try
    {
        sql.Open();
        // se invoca al procedimiento almacenado
        SqlCommand cmd = new SqlCommand("insertaRegistro", sql);

        cmd.CommandType = CommandType.StoredProcedure;
        // se envian los datos al procedimiento almacenado
        cmd.Parameters.Add("@idUsuario", System.Data.SqlDbType.Int).Value = idUsuario;
        cmd.Parameters.Add("@fecha", System.Data.SqlDbType.NVarChar,12).Value = fecha;
        cmd.Parameters.Add("@hora", System.Data.SqlDbType.NVarChar, 12).Value = hora;
        cmd.Parameters.Add("@code", System.Data.SqlDbType.Int).Value = cod;
        cmd.Parameters.Add("@idDispositivo", System.Data.SqlDbType.Int).Value = this.number;

        cmd.Parameters.Add("@transaction", SqlDbType.Int).Direction = ParameterDirection.Output;

        cmd.ExecuteNonQuery();

        sql.Close();

    }
    catch(Exception ex)
    {
        //en caso de no poder realizar la conexion manda una alerta
        MessageBox.Show(ex.ToString());
        sql.Close();
    }
    finally
    {
        sql.Dispose();
    }
}
```

Imagen 2.45 Código del Método de descarga de registros de asistencia en tiempo real

Para comprobar la operación correcta del método de descarga del Sistema, se monitorea la conexión con el dispositivo biométrico, es por ello que se creó el método hilo, el código se observa en la imagen 2.46, este método, como su nombre lo dice, es un hilo que se ejecuta cada cierto tiempo, un hilo es un subproceso que se crea paralelamente con otros procesos o subprocesos, este hilo se creó para verificar la conexión entre los dispositivos biométricos y el Sistema CHECKPOINT, esta tarea se realiza en rangos de 60 minutos mientras el sistema se ejecute, en C# un hilo está definido por la palabra reservada “THREAD” y su definición es la siguiente.

> *“Un hilo es una ruta independiente de ejecución, capaz de ejecutarse simultáneamente con otros hilos. Un programa C# arranca en un hilo principal creado automáticamente por el CLR y el sistema operativo y puede estar compuesto de*

*múltiples subprocesos mediante la creación de hilos adicionales"*[40].

Antes de ejecutar el hilo se verifica que exista en la red un dispositivo con la dirección IP con la que se registró el dispositivo, esta verificación se realiza mediante el comando PING, que permite que una aplicación determine si un equipo remoto está accesible o conectado en red, este comando se define de la siguiente manera.

*"Ping es la abreviatura de Packet Internet Groper, es una herramientas de diagnóstico utilizada en la administración de redes, permite verificar el estado de conexión de los equipos que forman parte de una red de computadoras. Para lograr su objetivo, el Ping envía paquetes de un equipo a otro, esperando una respuesta, para así poder diagnosticar el estado completo de la conexión y su velocidad"*[41].

```
public void hilo()
{
    // se realiza un ping
    Ping ping = new Ping();
    int timeout = 10;

    //si se alcanza el dispositivo intenta realizar la conexion con el dispositivo invocando el metodo de conexion
    if (ping.Send(ip, timeout).Status == IPStatus.Success)
    {
        //se crea un hilo para gestionar la conexiomn
        Thread t = new Thread(upDevice);
        t.Start();
        t.Join();
        //se crea un timer
        System.Windows.Forms.Timer timer1 = new System.Windows.Forms.Timer();
        //se cuenta una hora
        timer1.Interval = 3600000;
        timer1.Start();
        timer1.Tick += (s, e) =>
        {
            timer1.Dispose();
            t.Abort();
            contador++;
            hilo();

        };
    }
    else
    {
        this.mensaje = "No contectado";
    }

}
```

Imagen 2.46 Código del Método "hilo"

[40] THREAD (Clase), Microsoft MSDN http://msdn.microsoft.com/es-es/library/system.threading.thread(v=vs.110).aspx

[41] Ping (Clase), Microsoft http://msdn.microsoft.com/es-es/library/system.net.networkinformation.ping(v=vs.110).aspx

La clase “Device” contiene otros métodos relacionados con la gestión, el control y funcionamiento de los dispositivos biométricos, dichos métodos están programados con líneas de código muy similares a los métodos de “hilo” y el de descarga de registros de asistencia.

Otro método que forma parte de la clase “Config” diseñada para la descarga de datos de comunicación y configuración de los dispositivos biométricos, es el método “consultaDevice”, este método requiere de una conexión a la base de datos para invocar al procedimiento almacenado “consultaDispositivo”, los valores que le son retornados los almacena en un DataTable (tabla de datos) , que son necesarios para la conexión con los dispositivos biometricos ya que esta tabla contiene la dirección IP de los dispositivos y el número de identificación del dispositivo biométrico.

> *“Un objeto DataTable representa una tabla de datos relacionales en la memoria, se puede crear y usar de manera independiente o lo pueden usar otros objetos de .NET Framework, normalmente como miembro de un objeto DataSet”*[42]*.*

La estructura del método consultaDevice se observa en la imagen 2.47.

[42] DataTables, Microsoft MSDN http://msdn.microsoft.com/es-es/library/t31h6yhs(v=vs.110).aspx

```
        public DataTable consultaDevice()
        {
            DataTable dt = new DataTable();
            string conexionString = ConfigurationManager.ConnectionStrings["CPoint"].ConnectionString;
            SqlConnection sql = new SqlConnection(conexionString);

            try
            {
                sql.Open();
                SqlDataAdapter da = new SqlDataAdapter("consultaDispositivo", conexionString);
                da.SelectCommand.CommandType = CommandType.StoredProcedure;
                da.Fill(dt);
                sql.Close();
            }
            catch
            {

                sql.Close();
            }
            finally
            {
                sql.Dispose();
            }

            return dt;
        }

    }
```

Imagen 2.47 Código del Método “consultDevice”

La Capa de la Lógica de negocio está conformada por tres clases más, la clase “Empleado” que se encargar de buscar registrar y buscar empleado, la clase Login se encarga del control de acceso al Sistema, registrar nuevos usuarios y actualizar el perfil de cada usuario, y la clase “Registro” que se encarga de la inserción y consulta de registros de asistencia en la base de datos, el código fuente de esas clases es muy similar al de la clase “Device”, por lo que se omite su análisis y se continúa con el desarrollo de la capa de presentación del sistema.

## 2.9 Programación de la Capa de Presentación

> *“Una de las grandes ventajas de trabajar con ventanas es que todas se comportan de la misma forma independiente del sistema operativo con el que se trabaje y, en muchas ocasiones, utilizan los mismos componentes básicos para introducir órdenes”*[43].

[43] Ceballos, Francisco Javier, Java Interfaces Gráficas y Aplicaciones para Internet p.1

La capa de presentación del sistema tiene el propósito de ser amigable y lo más precisa posible para lograr que los usuarios interactúen con la aplicación y obtener los resultados esperados, es por ello que con apoyo de0 las herramientas que Windows Forms ofrece se diseñó una interfaz de usuario simple, que cumple con las necesidades requeridas por el sistema, el resultado fue el siguiente.

La interfaz gráfica del CHECKPOINT DOWNLOADER, es básica, solo requiere de un botón para reiniciar el sistema en el momento que se desee, es decir intentar conectarse con los dispositivos biométricos o manejar cualquier fallo, en la ventana se enlistan los dispositivos biométricos registrados y el estado en el que se encuentran, en caso de no estar conectados la aplicación lo indica con una cruz en rojo, el resultado obtenido se observa en la imagen 2.48.

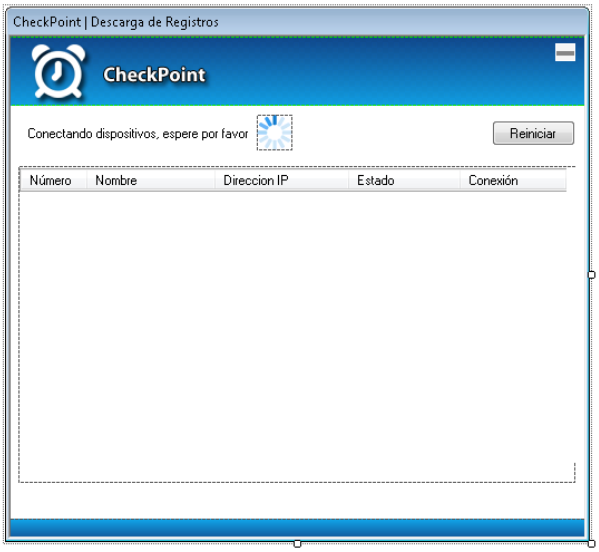



Imagen 2.48 Interfaz Gráfica del CHECKPOINT DOWNLOADER

La interfaz gráfica del CHECKPOINT VIEWER es un poco más compleja, se utilizan botones que generan eventos para la creación de objetos y uso de métodos de las clases previamente programadas para da la funcionalidad del sistema. Como se observó en el análisis del Sistema, CHEKPOINT VIEWER ofrece varias operaciones, por tal motivo se agruparon ordenadamente en pestañas, el resultado de la interfaz gráfica del CHECKPOINT VIEWER se aprecia en la imagen 2.49.

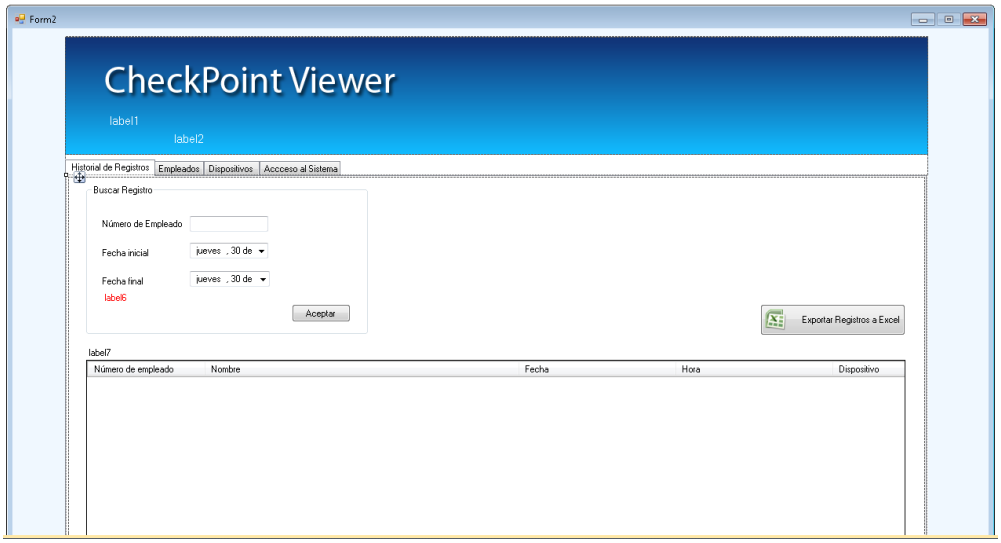



Imagen 2.49 Interfaz Gráfica del CHECKPOINT VIEWER

Para dar funcionalidad a los componentes de la interfaz gráfica se programa eventos en cada botón, principalmente se valida cada dato que el usuario insertó para evitar problemas con las operaciones de la base de datos, a continuación se muestra un ejemplo del procedimiento de la creación de eventos para cada botón.

```
private void button10_Click(object sender, EventArgs e)
{
    Cursor = Cursors.WaitCursor;
    dataGridView4.Rows.Clear();
    label12.Visible = true;
    label12.Text = "Buscando Usuarios, espere por favor";


    Login log = new Login();

    DataTable data = new DataTable();

    data = log.consultaLogin();

    int index = data.Rows.Count;
    int i = 0;

    while (i < index)
    {
        string estado = string.Empty;

        if (Convert.ToInt32(data.Rows[i][6]) == 1)
        {
            estado = "ADMINISTRADOR";
        }
        else if (Convert.ToInt32(data.Rows[i][6]) == 2)
        {
            estado = "CONSULTA";
        }
        else
        {
            estado = "INACTIVO";
        }


        dataGridView4.Rows.Add(data.Rows[i][0].ToString(), data.Rows[i][1].ToString(), data.Rows[i][2].ToString(), data.Rows[i][3].ToString(), data.Rows[i][4].ToString(),
        i++;

        label12.Text = "Usuarios encontrados " + i;
    }
```

Imagen 2.50 Segmento de Código para invocar a las Clase del CHECKPOINT

El segmento de código de la imagen 2.50 demuestra el procedimiento de asignación de eventos a los botones del CHECKPOINT VIEWER, en este caso al “botón 10” se le asigna la invocación a la clase “Login”, se crean los objetos de las mismas y se manejan los métodos que los operan, en esencia se validan ciertos parámetros de entrada, dado el caso de que sean válidos los datos se crean los objetos de la clase y se desencadena todo el procedimiento hasta concluir con la inserción en la base de datos. La asignación de funciones a los botones del sistema en general es semejante al ejemplo antes descrito.

Es así como se concluye con el desarrollo del Sistema CHECKPOINT, para generar la versión final del sistema con intercambiar compilar el proyecto en modo “Release”, en la barra de herramientas de Visual Studio en el selector de configuración de solución se elige la opción “Release” y se ejecuta el proyecto, el resultado de la compilación se observa en la raíz de la carpeta “bin” de la solución, se generó la carpeta llamada “Release”, en ella está contenido el archivo ejecutable del sistema CHECKPOINT.

# CAPITULO III

# "IMPLEMETANCIÓN DEL CHECKPOINT"

## 3.1 Implementación

Una vez concluido el desarrollo del Sistema CHECKPOINT se continúa con la instalación e implementación, para ello se recomienda una serie de requisitos mínimos para el correcto funcionamiento del Sistema:

- Equipo de cómputo con 2 Gb de memoria RAM.
- Capacidad mínima de 20 Gb de disco duro.
- Procesador Intel o AMD 2.0 GHz
- Sistema Operativo Windows 7 o superior de 32 y 64 bits.
- Microsoft Framework .NET 3.5 o superior instalado.
- Conexión en red respecto al servidor de base de datos.

La instalación del CHECKPOINT se realizó en un equipo de la marca Lenovo modelo THINKCENTRE Serie M, cuenta con 2.99 Gb de memoria utilizable, procesador AMD ATHLON a 3.2 GHz y 180 Gb de capacidad en disco duro. El equipo cumple con las características ideales para ejecutar al sistema.

## 3.2 Instalación del CHECKPOINT DOWNLOADER

Para la instalación del CHECKPOINT DOWNLOADER se requiere la carpeta "Release" ubicada en la carpeta "Bin" del proyecto desarrollado en Visual Studio.

> *"La configuración "Release" del programa es totalmente optimizada y no contiene información de depuración simbólica. La información de depuración se puede generar en archivos base del programa"*[44].

[44] Configuración Debug y Release, Microsoft MSDN http://msdn.microsoft.com/es-mx/library/aa292277(v=vs.71).aspx

La carpeta “Release” se copia en la carpeta “Archivos de Programa” del Sistema Operativo, generalmente se localiza en la ruta de acceso “C:\Program Files”, además se debe reemplazar el nombre “Release” por el nombre de CHECKPOINT DOWNLOADER, en la imagen se muestra el resultado del procedimiento descrito.

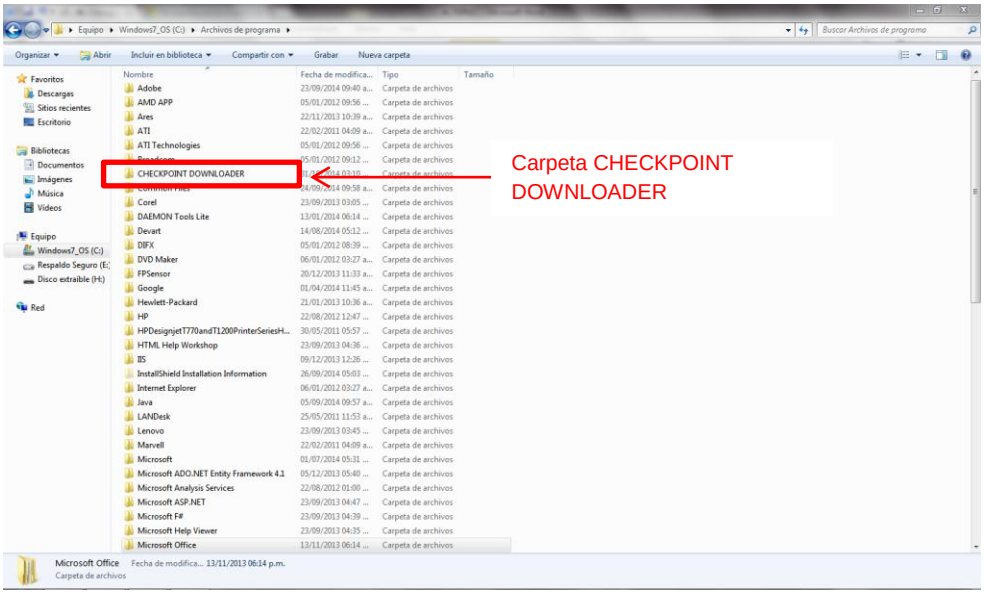



Imagen 3.1 Carpeta de instalación del CHECKPOINT DOWNLOADER

El CHECKPOINT DOWNLOADER debe ejecutar mientras el equipo se encuentre encendió, en necesario crear una tarea programada que tenga la capacidad de ejecutar la aplicación cada vez que se encienda el equipo, o bien, cada que se inicia el Sistema Operativo del equipo.

Las tareas programadas son procesos que ejecutan programas o configuran archivos de un equipo de cómputo de manera automática de acuerdo a cierto tiempo o fecha establecida definida por el usuario del equipo, para el caso en cuestión se creó una tarea programada para ejecutar el CHECKPOINT DOWNLOADER cada vez que se encienda el equipo, a continuación se describe el procedimiento.

En el menú se inició de Windows 7 se elige las opciones Accesorios – Herramientas del Sistema – Programador de Tareas, posteriormente se abre la ventana de la imagen 3.2.

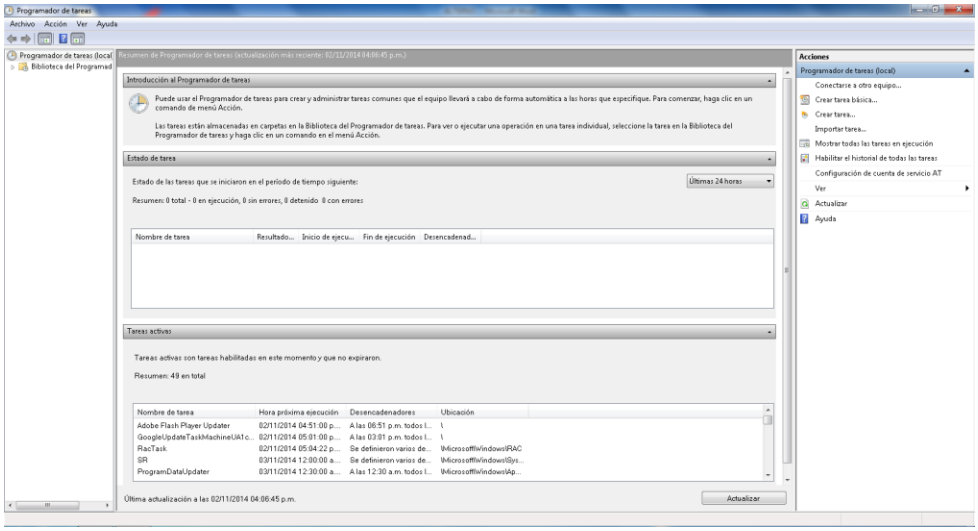



Imagen 3.2 Creación de tareas programadas

En el cuadro de “Acciones” se selecciona la opción de “Crear tarea básica” y se muestra el siguiente formulario de la imagen 3.3.

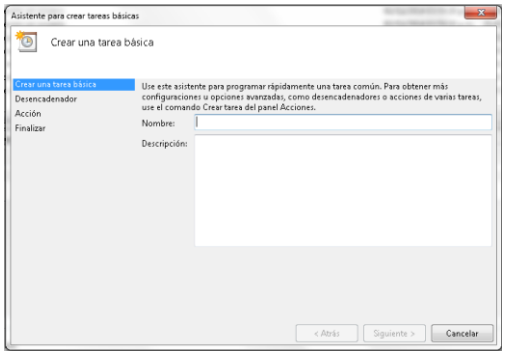



Imagen 3.2 Formulario para nombrar a la tarea programada

En el formulario se escribe el nombre con el que se identifica la tarea programada, para este caso se nombró “CHECKPOINT DOWNLOADER”, a continuación se habilita el botón “Siguiente” y se da clic. Posteriormente se abre la ventana de la imagen 3.3.

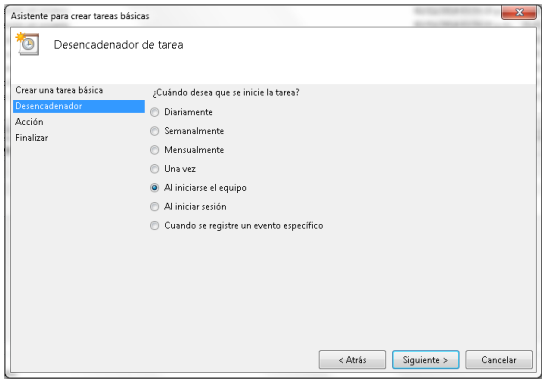



Imagen 3.3 Ventana para desencadenar la tarea

La ventana muestra una seria de opciones que determinan el momento en el que se desencadena la tarea, para este caso particular se eligió la opción de “Al iniciarse el equipo” con el objetivo de ejecutar al CHECKPOINT DOWLOADER cada vez que se encienda el equipo de cómputo. A continuación se abre el formulario para elegir la acción que se desencadena cada vez que se inicie la computadora, en este caso se seleccionó la opción “Iniciar un programa” como se aprecia en la imagen 3.4.

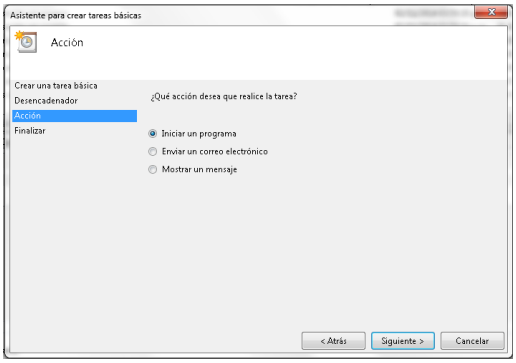



Imagen 3.4 Ventana para desencadenar la tarea

La tarea programada requiere que se le indique la ruta de acceso del programa que debe ejecutar, por ello se ingresa la ruta del CHECKPOINT DOWNLOADER “C:\ProgramFiles\CHECKPOINTDOWNLOADER\CHECKPOINTDOWNLOADER.exe”, para continuar con el procedimiento se da clic en el botón “Siguiente”.

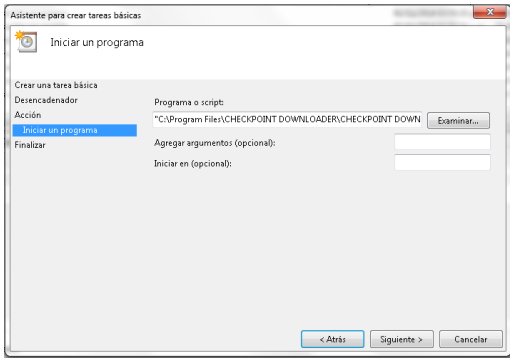



Imagen 3.5 Ubicación de la ruta del CHECKPOINT DOWNLOADER

Una vez realizados los pasos anteriores se muestra la ventana con la que se finaliza la creación de la tarea programada, en ella se muestran los parámetros de configuración requeridos para el funcionamiento del CHECKPOINT DOWNLOADER

Al concluir la creación de la tarea programada se requiere comprobar que el funcionamiento sea el ideal para ejecutar el CHECKPOINT DOWNLOADER cada vez que se encienda la computadora, la manera de hacerlo es reiniciar el equipo de cómputo y observar los resultados. En la prueba se observó que se abre la aplicación después de reiniciar la computadora como se muestra a en la imagen 3.6.

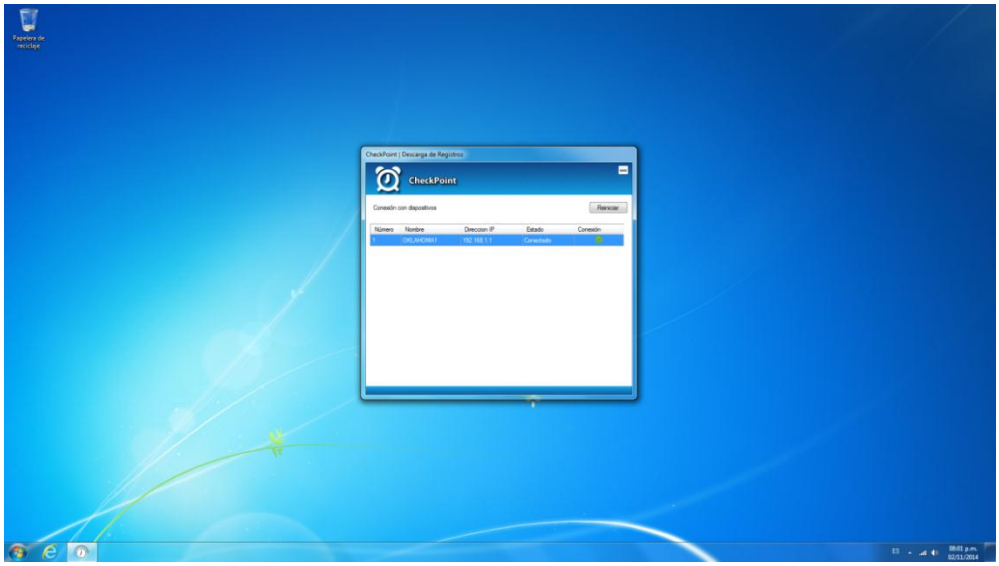



Imagen 3.6 Comprobación del funcionamiento del CHECKPOINT DOWNLOADER

Se observa en la imagen 3.7 que el CHECKPONT DOWLOADER se ejecuta sin problemas con la Tarea Programada que se creó, se logra la conexión con los equipos biométricos ZK Software x628-C, también se logra la descarga de los Registros de Asistencia del dispositivo que se comprueba en el CHECKPOINT VIEWER, por lo que la implementación del Sistema CHECKPOINT en el bloque de conexión de descarga de registros ha sido exitosa.

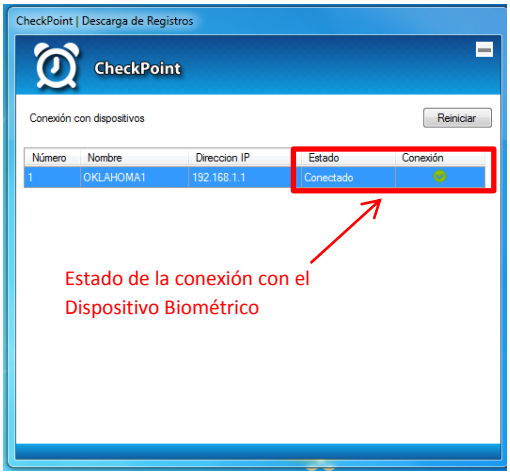



Imagen 3.7 Conexión con los Dispositivos Biométricos

## 3.3 INSTALACIÓN CHECKPOINT VIEWER

La instalación del CHECKPOINT VIEWER es similar a la del CHECKPOINT DOWNLOADER, el procedimiento requiere de la carpeta “Release” del proyecto de Visual Studio, copiarla en el directorio de “Archivos de Programa” del equipo y modificarle el nombre por “CHECKPOINT VIEWER”, tal cual se observa en la imagen 3.8.

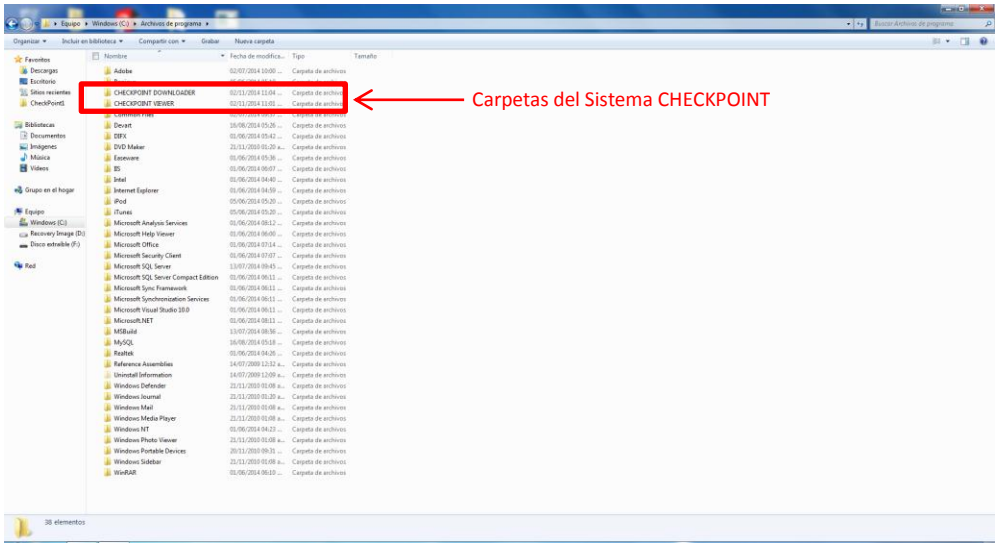



Imagen 3.8 Carpeta de instalación del CHECKPOINT VIEWER

Después de copiar la carpeta al directorio indicado se debe de crear un “Acceso Directo” para facilitar la ejecución del Sistema a los usuarios, es decir, crear un icono de referencia en el escritorio del equipo de cómputo para ejecutar la aplicación. El procedimiento es sencillo, se debe ubicar el archivo ejecutable CHECKPOINT VIEWER.exe dentro de la carpeta del mismo proyecto, una vez ubicado el archivo se da clic derecho sobre el ejecutable y se abre el menú de opciones, se debe seleccionar la opción de “Crear Acceso Directo” como se observa en la imagen 3.9.

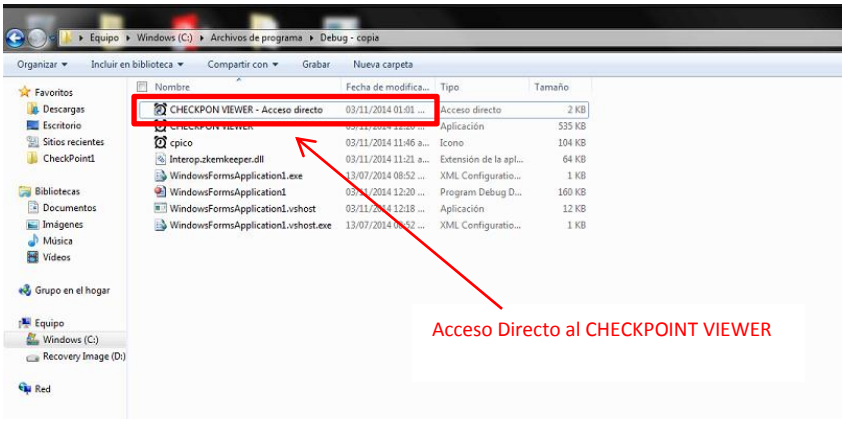



Imagen 3.9 Creación del acceso directo del CHECKPOINT VIEWER

Después de crear el acceso directo se copió en el escritorio del equipo de cómputo donde se instaló la aplicación para que el usuario tenga acceso al sistema, para comprobar el funcionamiento del acceso únicamente se da clic sobre el icono del acceso directo en la imagen 3.10.

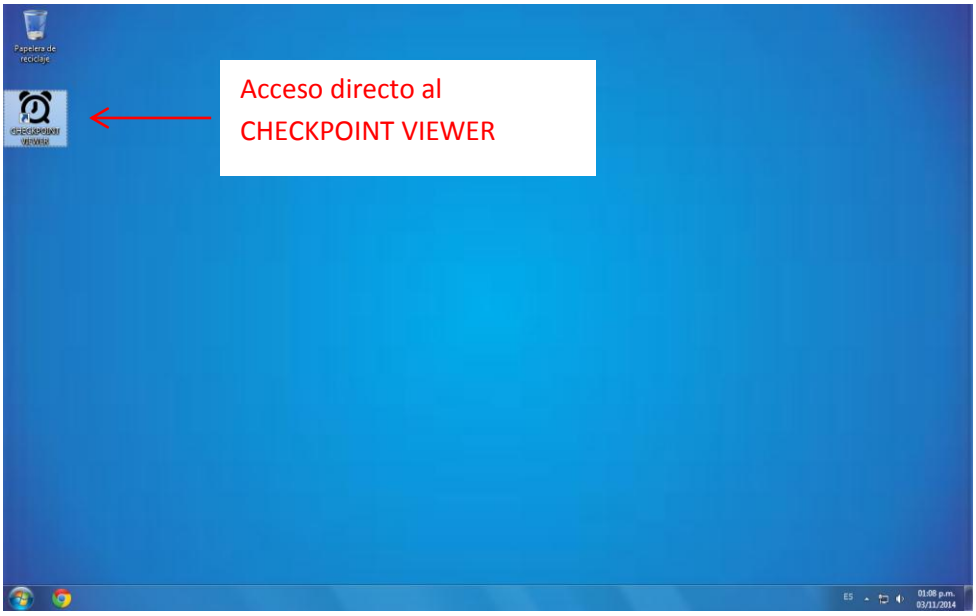



Imagen 3.10 Acceso directo del CHECKPOINT VIEWER

Al comprobar el funcionamiento del acceso directo del sistema se abrió la ventana del control de accesos del Sistema de la imagen 3.11, la cual requiere de usuario y contraseña para poder ingresar al sistema, para este caso predeterminadamente se ingresó directamente a la base de datos el usuario “admin” con la contraseña “root”.

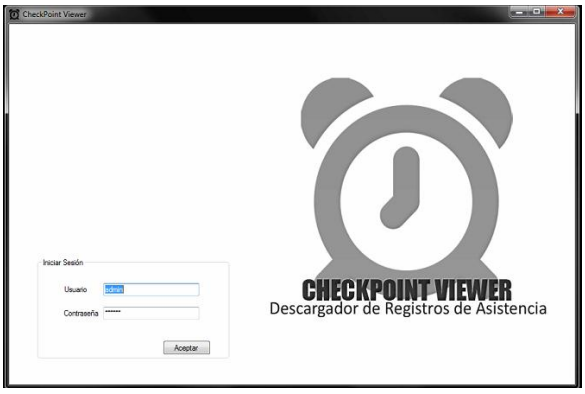



Imagen 3.11 Control de Acceso del CHECKPOINT VIEWER

Al ingresar al sistema se muestra la ventana de la imagen 3.12, en ella se aprecia la estructura general de la aplicación de escritorio, esta se conforma por un encabezado con el nombre de la aplicación y los datos del usuario que ingreso al sistema, debajo del encabezado se aprecia un menú con pestañas, las cuales contienen las funciones requeridas del sistema.

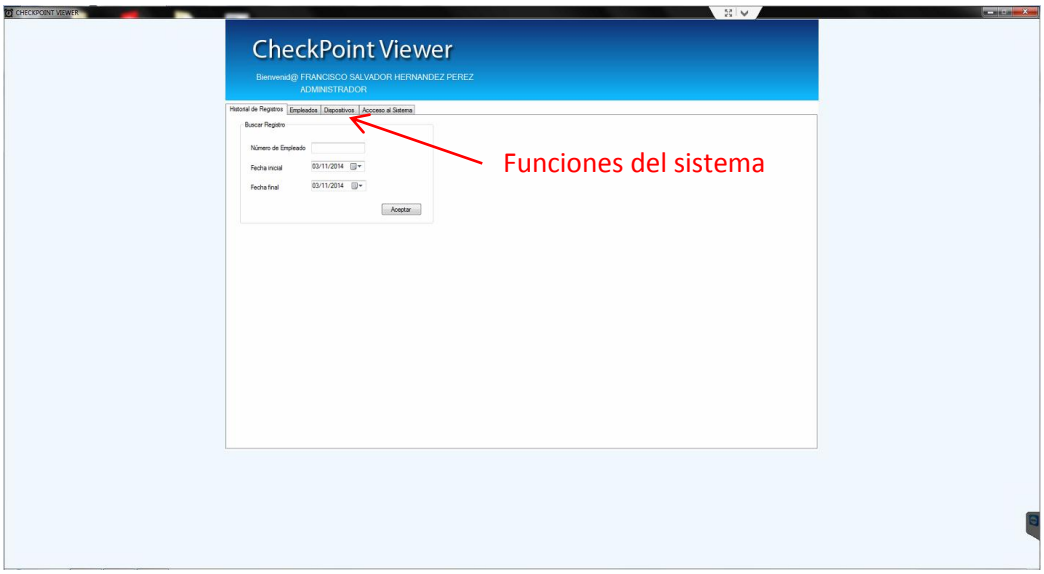



Imagen 3.12 Inicio del Sistema CHECKPOINT VIEWER

La opción “Historial de Registros” contiene la función para consultar los registros de asistencia de asistencia descargados de los dispositivos biométricos a través del CHECKPOINT DOWNLOADER, la búsqueda es por criterios de número de empleado y de fecha inicial y fecha final, en caso de existir muestra todos los registros de acuerdo al criterio de busqueda con el número de empleado, nombre del empleado, fecha del registro, hora del registro y el dispositivo que capturó el registro como se aprecia en la imagen 3.13.

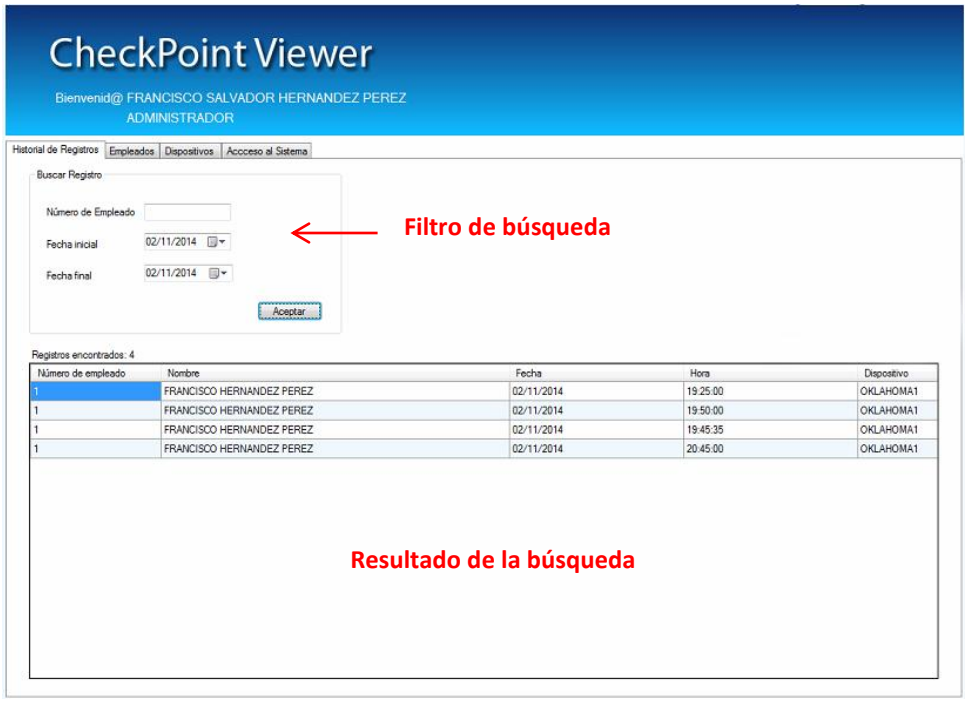



Imagen 3.13 Historial de Registros del Sistema CHECKPOINT VIEWER

La opción “Empleados” registra nuevos empleados, así mismo consulta la plantilla de los empleados que registran asistencia.

La opción “Dispositivos” se encarga de consultar los dispositivos biométricos que se registraron previamente para la captura del registro de asistencia del personal, además ofrece la opción de Agregar “n” número de dispositivos sin registrar licencia para liberar la conexión con el dispositivo biométrico.

Otra de las funciones de la opción “Dispositivos” es gestionar las propiedades del dispositivo, modificar nombre, fecha y hora, dirección IP y el estado del dispositivo, estas operaciones se realizan mediante el formulario de la imagen 3.14.

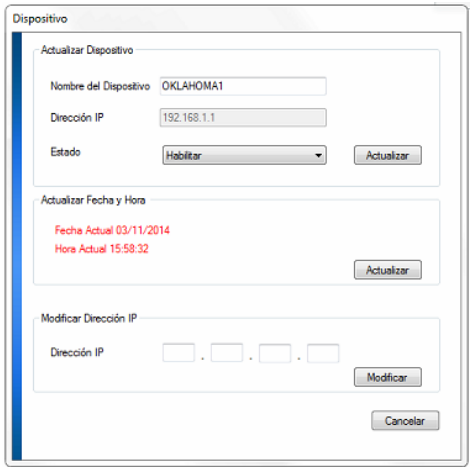



Imagen 3.14 Gestión de Dispositivos Biométricos Sistema CHECKPOINT VIEWER

Finalmente la opción “Acceso al sistema”, de la imagen 3.15, controla el acceso al sistema, en ella se pueden registrar nuevos usuarios, restringir o habilitar privilegios a las operaciones del sistema y habilitar o deshabilitar cuentas de usuario, con esta opción se concluye la implementación del CHECKPOINT VIEWER.

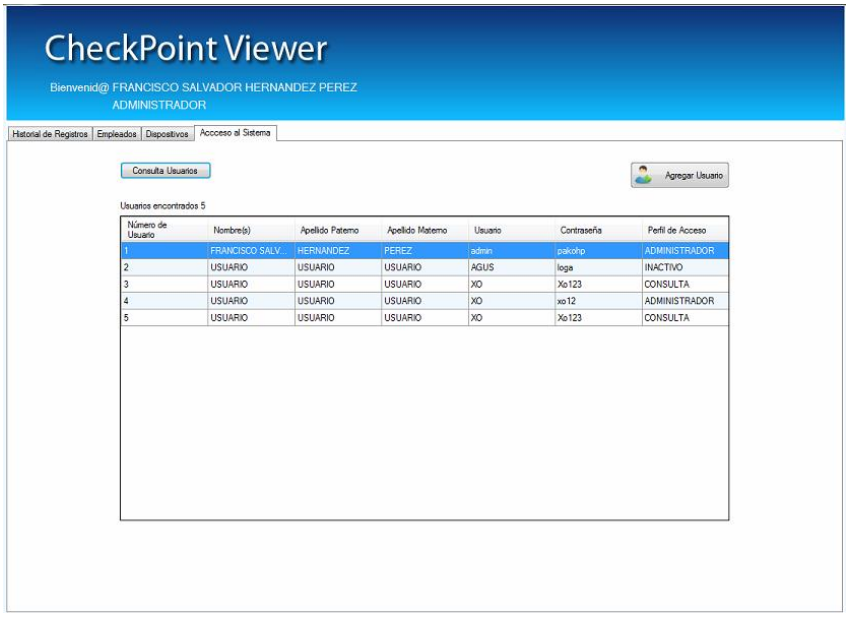



Imagen 3.14 Gestión del Control de Acceso al Sistema CHECKPOINT VIEWER

La implementación de los bloques funcionales del Sistema CHECKPOINT se logró sin inconvenientes, los objetivos planteados en el análisis del sistema se han alcanzado, cumpliendo con las operaciones necesarias para adaptarse al Procedimiento de Registro y Control de Asistencia de COFEPRIS.

### 3.4 Resultados obtenidos

Al concluir la implementación del Sistema CHECKPOINT se determinó ejecutarlo simultáneamente con el Sistema AD10 para monitorear el desempeño de cada Sistema en base a los resultados ofrecidos por cada aplicación, entre los meses de Abril y Junio de 2014 se realizó el conteo los registros capturados e insertados en la base de datos de cada sistema, al concluir la pruebas se observó lo siguiente.

En el mes de Abril el Sistema CHECKPOINT capturó 19270 registros contra 14966 del Sistema AD10, la diferencia fue de 4309.

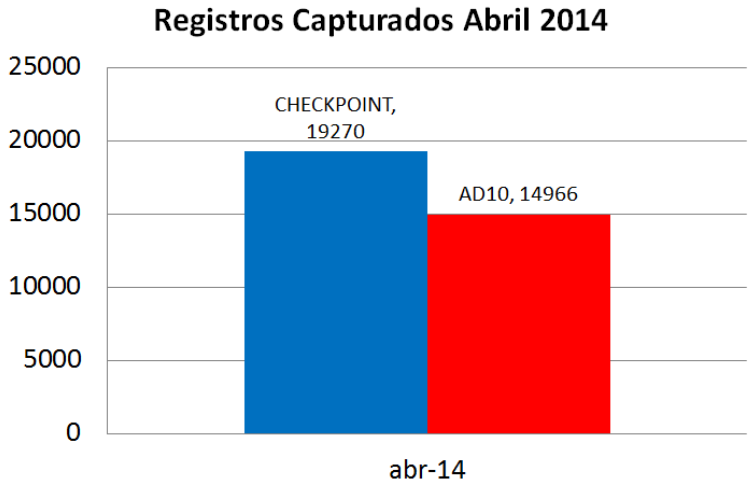



Imagen 3.15 Gráfica de descarga de registros de asistencia del mes de Abril de 2014 por el Sistema CHECKPOINT

Con respecto al mes de Mayo se obtuvieron 18124 CHECKPOINT con respecto a los 14230 del AD10, la diferencia fue de 3824.

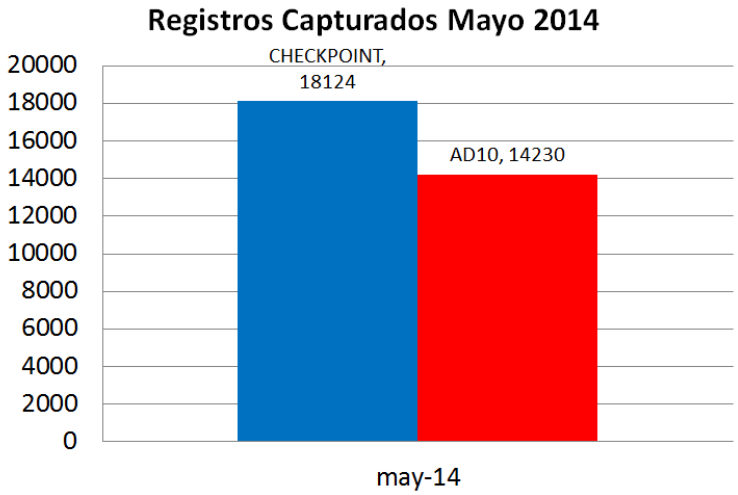



Imagen 3.16 Gráfica de descarga de registros de asistencia del mes de Mayo de 2014 por el Sistema CHECKPOINT

Finalmente en el mes de Junio el CHEKPOINT capturó 19309 contra 15830 del AD10, la diferencia 3479.

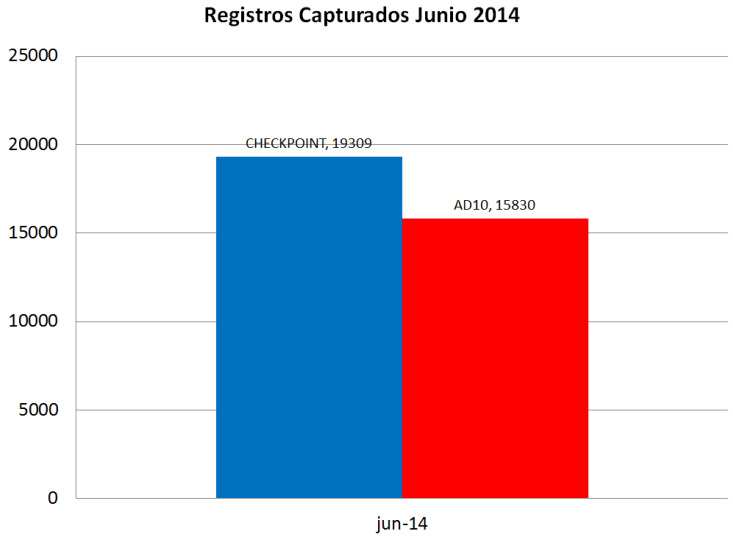



Imagen 3.17 Gráfica de descarga de registros de asistencia del mes de Junio de 2014 por el Sistema CHECKPOINT

Al concentrar los resultados de los tres meses de monitoreo en una gráfica se observa que los parámetros de descarga de cada sistema no varían mucho entre

cada mes, únicamente se aprecia que el sistema AD10 tiene un déficit de registros perdidos en promedio mensual de 3800.

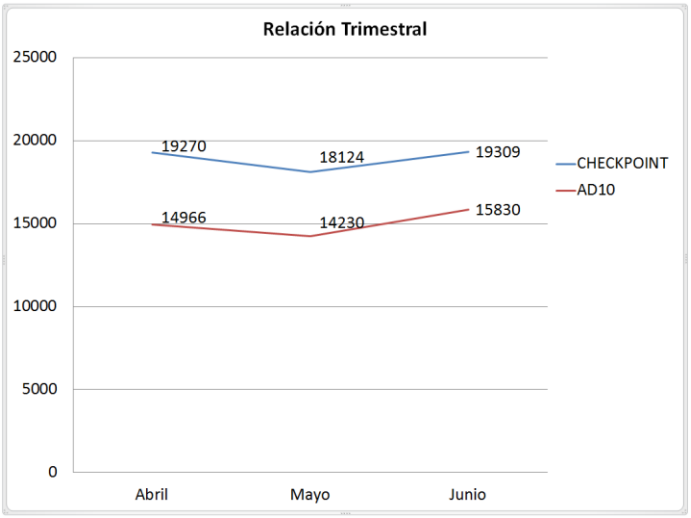



Imagen 3.18 Gráfica del concentrado trimestral de la descarga de registros de asistencia por el Sistema CHECKPOINT

Con los resultados obtenidos de la implementación del sistema se concluye este capítulo del caso práctico, el desarrollo del sistema CHECKPOINT favorece a la mejora del procedimiento de control de asistencia, la implementación del Sistema CHECKPOINT ha sido exitosa, se ha logrado incrementar en un 28% la descarga de registros de asistencia de los dispositivos biométricos ZK Software X628 – C.

# CONCLUSIÓN

Después de tres meses de monitoreo se determinó sustituir al Sistema AD10 por el Sistema CHECKPOINT en base a los resultados obtenidos durante las pruebas realizadas, además se consideró el rendimiento y funcionamiento del sistema. Durante seis meses el CHECKPOINT no ha presentado fallas técnicas, con respecto a la configuración del sistema ni a la conectividad con los dispositivos biométricos. En cuanto a la adaptación con el Procedimiento de Registro y Control de Asistencia no se han presentado inconvenientes.

En el análisis del sistema sobre sale el método de programación en capas, con él se logró dividir los procesos del sistema en diferentes bloques de ejecución que conlleva a un buen manejo de posibles errores. Las herramientas de desarrollo utilizadas se adaptan a las versiones del Sistema Windows 7 y superiores. La estructura de la base facilita la inserción de registros y la gestión de datos requerido por el sistema. Con ello se cumplen los objetivos específicos planteados en el desarrollo del sistema satisfaciendo las necesidades del Procedimiento de Asistencia con respecto a la problemática presentada.

La implementación del Sistema CHECKPOINT de Descarga de Registros de Asistencia ha sido exitosa, se ha reportado en los últimos seis meses la disminución de casos de personal con descuentos económicos con respecto al procedimiento de Registro y Control de Asistencia en un 35%, con ello cumple con el objetivo planteado para este caso práctico, se desarrolló un Sistema de Descarga de Registros de Asistencia de Dispositivos Biométricos que favorece a la mejora del Procedimiento de Asistencia de la Comisión para la Protección Contra Riesgos Sanitarios.

## Bibliografía y Cibergrafía

¿Qué es un archivo dll?, Microsoft Soporte http://support2.microsoft.com/kb/815065/es

Ceballos, Francisco Javier,C# Lenguaje y Aplicaciones

Ceballos, Francisco Javier, Java Interfaces Gráficas y Aplicaciones para Internet

César Pérez, MySQL para Windows y Linux 2° Edición, Diseño de una Base de Datos Relacional

Company Overview, ZK-Software http://www.zktechnology.com/CompanyOverView.aspx

Configuración Debug y Release, Microsoft MSDN http://msdn.microsoft.com/es-mx/library/aa292277(v=vs.71).aspx

DataTables, Microsoft MSDN http://msdn.microsoft.com/es-es/library/t31h6yhs(v=vs.110).aspx

Fundamentos de Biometría, UNAM Facultad de Ingeniería Biometría informática http://redyseguridad.fip.unam.mx/proyectos/biometria/

Historia de la COFEPRIS, Sitio Web COFEPRIS http://www.cofepris.gob.mx/cofepris/Paginas/Historia.aspx

Información general acerca de .NET Framework, Microsoft Developer Network http://msdn.microsoft.com/es-es/library/zw4w595w(v=vs.110).aspx

Información general de SQL Server, Microsoft TechNet http://technet.microsoft.com/es-es/dd367804.aspx

Información General sobre ASP.NET, Microsoft TechNet http://msdn.microsoft.com/eses/library/4w3ex9c2(v=vs.100).aspx

Introducción a Visual Studio, Microsoft MSDN http://msdn.microsoft.com/es-mx/library/fx6bk1f4(v=vs.90).aspx

Manual de Procedimientos Secretaría General, Procedimiento para Registro y Control de Asistencia de Personal SG-DERH-P-08, Políticas de Operación, normas y lineamientos 3.0

Misión y Visión de la COFEPRIS, Sitio Web COFEPRIS http://www.cofepris.gob.mx/cofepris/Paginas/VisionYMision.aspx

Página Web de la OMS http://www.who.int/es/

Página Web OPS http://www.paho.org/hq/?lang=es

Ping (Clase), Microsoft http://msdn.microsoft.com/es-es/library/system.net.networkinformation.ping(v=vs.110).aspx

Procedimientos almacenados y funciones, MySQL 5.0 Reference Manual http://dev.mysql.com/doc/refman/5.0/es/stored-procedures.html

Reglamento de la Comisión Federal para la Protección de Riesgos Sanitarios, Capítulo II. Integración de la Comisión Federal Art. 4 p. 15

Reglamento de la Comisión Federal para la Protección de Riesgos Sanitarios, Capítulo II. Integración de la Comisión Federal Art. 4 p. 15

SDK Description, TFT Series Communication Protocol SDK Development Handbook P. 9

Servidor Web IIS, Microsoft TechNet http://technet.microsoft.com/es-mx/library/cc753433(v=ws.10).aspx

SQL Server, Sever and cloud Platform http://www.microsoft.com/es-xl/server-cloud/products/sql-server/

THREAD (Clase), Microsoft MSDN http://msdn.microsoft.com/es-es/library/system.threading.thread(v=vs.110).aspx

Usar SQL Server Magnament, Microsoft MSDN Studio http://msdn.microsoft.com/es-MX/library/ms174173.aspx

Visual Basic, Microsoft Developer Network http://msdn.microsoft.com/es-es/library/2x7h1hfk.aspx

Visual C#, Microsoft Developer Network http://msdn.microsoft.com/es-es/library/kx37x362.aspx

ZK Software, Data sheet, x628 – C