SISTEMA DE INFORMACIÓN PARA EL MANEJO DEL TALENTO HUMANO EN C.I. TÉCNICAS BALTIME DE COLOMBIA (SETHUM)

JAIME RAFAEL ABELLA ALVAREZ

WILMAN JAVIER RINCÓN BAUTISTA

UNIVERSIDAD DEL MAGDALENA

FACULTAD DE INGENIERÍA

PROGRAMA INGENIERÍA DE SISTEMAS

SANTA MARTA, D.T.C.H.

2005

SISTEMA DE INFORMACIÓN PARA EL MANEJO DEL TALENTO HUMANO EN C.I. TÉCNICAS BALTIME DE COLOMBIA (SETHUM)

JAIME RAFAEL ABELLA ALVAREZ
WILMAN JAVIER RINCÓN BAUTISTA

Trabajo de Memoria de Grado presentado para optar al título de Ingeniero de Sistemas

Director
LUIS GARRIDO BARRIOS
Ingeniero de Sistemas

UNIVERSIDAD DEL MAGDALENA
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA DE SISTEMAS
SANTA MARTA, D.T.C.H.
2005

Nota de aceptación

_____

_____

_____

_____
Presidente del Jurado

_____
Jurado

_____
Jurado

Santa Marta, Agosto de 2005

DEDICATORIAS

A mis padres Ana y Jaime por su apoyo incondicional

Jaime.

A mi familia y compañeros de trabajo por su colaboración.

Wilman.

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a:

La Universidad del Magdalena por ofrecernos la oportunidad de enriquecer nuestros conocimientos.

Ing. Luís Garrido, Director de Sistemas de Técnicas Baltime de Colombia, por su gran ayuda profesional, humano y apoyo en la elaboración de este proyecto.

A todos los docentes de la Universidad del Magdalena, Universidad de Antioquia, Universidad Nacional de Colombia, Universidad Industrial de Santander y Universidad de los Andes que nos brindaron sus conocimientos para la culminación de nuestros estudios.

CONTENIDO

1 INTRODUCCION

Hoy en día, el desarrollo humano en la empresa juega un papel predominante, para que la organización logre ser competitiva. Para lograr esta competitividad, un ingrediente fundamental, es el poder tener una fuerza laboral donde se logre una interacción adecuada entre persona, trabajador y funciones. Esto significa que las destrezas y las aspiraciones del trabajador estén alineadas con la visión de las organizaciones, y que las organizaciones soporten la parte humana del empleado.

Por esto la administración de este personal es de vital importancia dentro de las organizaciones, hasta llegar a convertirse el área de Recursos Humanos en el eje central de cualquier empresa.

Un aspecto importante del talento humano es poder comprender no solo la parte académica y las oportunidades de mejorar dentro de la organización, si no también tener como referencia el aspecto social donde el individuo se desenvuelve, permitiendo a las organizaciones brindar un apoyo integral a sus trabajadores desde esta área.

En C.I. TECNICAS BALTIME DE COLOMBIA S.A. actualmente no cuentan con la información necesaria que permita a los gerentes de RRHH la toma de decisiones en el aspecto laboral, de capacitación y personal de sus empleados.

Manejando parte de la información de los empleados en forma manual y otra procesada por sistemas aislados, generando procesos independientes y con altos costos de procesamiento y de personal.

2 PLANTEAMIENTO DEL PROBLEMA

El procesamiento de la información de los empleados de una manera eficiente y segura, generar estadísticas para la toma de decisiones en el departamento de RRHH de la compañía, La interacción con el usuario final, el acceso desde cualquier lugar del mundo y demás aspectos, requieren de una robusta aplicación que aproveche las características de Internet, pero para aprovechar estas características el desarrollo de la misma se hace necesariamente muy complejo, provocando la presencia de un gran número de puntos críticos que deben considerarse.

La seguridad de las aplicaciones debe ser una premisa que debe estar presente en toda tarea de desarrollo, teniendo siempre en cuenta que cualquier cliente que haga uso del servicio puede ser un potencial atacante para el mismo.

Es fundamental disponer de una metodología de desarrollo estructurada y con una serie de objetivos claros, como son facilitar las tareas de revisión y modificación de código, actualizaciones y ampliaciones de los servicios proporcionados, permitiendo que en todo momento se tenga el control por parte del servidor de lo acontecido en la ejecución de las aplicaciones evitando así que puedan ejecutarse acciones imprevistas que den pie a la vulneración del servicio.

Algunos de los problemas comunes en el desarrollo de este tipo de aplicaciones son:

- *Implementaciones poco robustas de mecanismos de autenticación o de mantenimiento de sesiones.*
- *Errores en el tratamiento de entrada de datos: el incorrecto procesamiento de éstos puede llevar a una malversación por parte del atacante.*
- *Ejecución de aplicaciones con máximos privilegios sobre el sistema: como consecuencia de esto un fallo en la aplicación puede comprometer completamente a la plataforma de ejecución.*
- *Tratamiento incorrecto de los posibles errores y los mensajes generados al producirse éstos.*

3 ANTECEDENTES

No sólo las empresas colombianas se encuentran enfrentando la incertidumbre del cambio. Cualquier organización en el mundo esta sintiendo la apelación que produce uno solo de los elementos que caracterizan el entorno económico de hoy: la competitividad. Para enfrentar el desafío de la competitividad, los administradores deben estar abiertos a intentar nuevas prácticas de gestión, nuevas ideas para construir organizaciones más productivas, que proyecten su éxito a largo plazo, como también el de los empleados que trabajan para ella.

La primacía de la información fuente de desarrollo en la era moderna, y con ella el proceso de globalización y los acelerados y permanentes cambios tecnológicos, han generado ya sus efectos en la transformación de las estructuras empresariales, en la forma de hacer las cosas.

En consecuencia, una nueva organización del trabajo está marcando la pauta en el mundo actual, obligando a elevar las condiciones de competitividad. Estos cambios han influenciado directamente sobre la composición de los empleos, los cuales han pasado de una concepción de puestos de trabajo a una de ocupaciones. El trabajo, así visto, cambia de la orientación al esfuerzo, por la orientación al cerebro.

De ahí que debamos generar nuevas ideas de gestión organizacional que respondan más a las condiciones actuales y proyectadas al mañana. En este nuevo entorno, la gestión del talento humano tendrá que ser diferente. Los perfiles de los trabajadores serán diferentes, el trabajo manual se reducirá, el intelectual se incrementará. La evaluación del desempeño tendrá otro sentido y la experiencia otra valoración. Igualmente serán diferentes los conceptos de motivación y compromiso empresarial. Será más importante ser capaz de aprender que saber.

Todo esto exige un diseño diferente de los programas de educación y capacitación. Generando nuevas maneras de concebir los conceptos de motivación y compromiso empresarial.

De acuerdo con esto, el sistema de información que se pretende implantar en C.I. Técnicas Baltime de Colombia S.A. tiene como fin gestionar el proceso de administración integral del Recurso Humano. Fundamentada en la utilización de algunas concepciones, como la gestión y competencia, herramientas administrativas que, organizadas en un proceso lógico de dirección, ayuden a mejorar la productividad organizacional y humana.

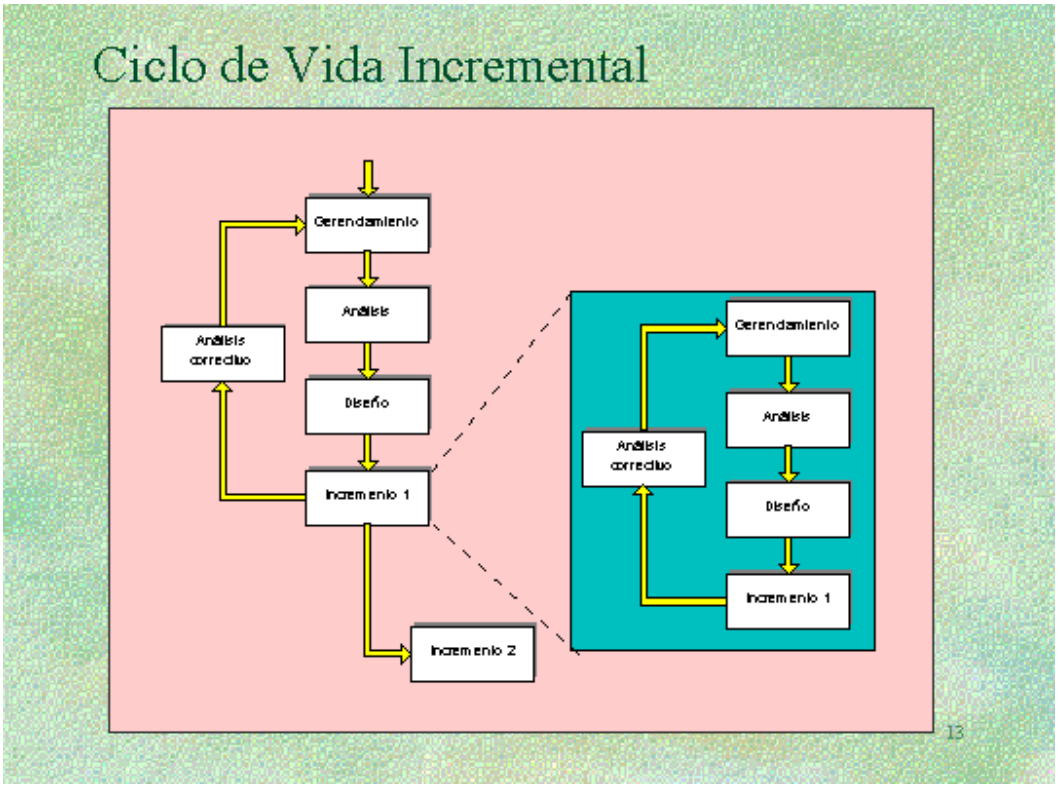
Así el Sistema Electrónico de Talento Humano, les permitirá identificar las necesidades al nivel de capacitación y educación, programas de salud ocupacional y

programas de trabajo social, como también un registro de la información laboral del empleado el cual es el verdadero activo de la compañía representado en su capital humano.

4 MARCO TEORICO

4.1 CICLO DE VIDA DE LOS SISTEMAS

El ciclo de vida de un sistema de información esta ligado al ciclo de vida del sistema de base de datos sobre el que se apoya. Al ciclo de vida de los sistemas de información también se le denomina ciclo de vida de desarrollo del software.



Las etapas típicas del ciclo de vida de desarrollo de software son:

- *Planificación*
- *Recolección y Análisis de los requisitos*
- *Diseño de base de datos y aplicación*
- *Creación de prototipos*
- *Prueba*
- *Implementación*

Este ciclo de vida hace énfasis en la identificación de las funciones que realiza el área de RRHH y en el desarrollo de las aplicaciones que lleven a cabo estas funciones. Se dice que el ciclo de vida de desarrollo del software sigue un enfoque orientado a funciones, ya que los sistemas se ven desde el punto de vista de las funciones que llevan a cabo. Por esta razón, el análisis estructurado hace énfasis en los diagrama de flujo de datos, siguiendo el movimiento de los datos a través de una secuencia de transformaciones, y refinando estas a través de una serie de niveles. Lo mismo ocurre en el diseño estructurado que va a un sistema como una función que se descompone sucesivamente en niveles o subfunciones.

4.2 MODELO DE BASE DE DATOS

4.2.1 Modelo Entidad- Relación

El modelo entidad-relación se basa en una percepción del mundo real que consiste en un conjunto de objetos básicos llamaos entidades y de relaciones entre estos objetos. Fue creado como una notación orientada al diseño del esquema conceptual de una base de datos, pues permite la descripción del esquema conceptual sin preocuparse por problema de diseño físico o de eficiencia. Dentro del modelo entidad-relación se tienen los siguientes elementos:

Entidades: *Es una "persona, lugar, cosa, concepto o suceso real o abstracto". Es aquel objeto sobre el cual queremos almacenar información en la base de datos. La entidad esta representada por un conjunto de atributos. Para cada atributo existe un rango de valores permitidos, llamado dominio del atributo.*

Atributos: *En cada una de las propiedades que tiene una entidad. El conjunto de posibles valores que puede tomar el atributo recibe el nombre de dominio. Entre todos los atributos de una entidad debe existir uno o varios que identifiquen unívoca y ínimamente cada una de las ocurrencias de una entidad, este atributo se conoce como identificador principal.*

Relaciones: *Es una asociación entre (varias) entidades. También se puede definir como la interrelación entre entidades. Una relación se caracteriza por tener un nombre*

por medio del cual la identificamos, un grado el cual es el número de entidades que participan en la relación y un tipo de relación el cual es el numero máximo de ocurrencias de una entidad que puede intervenir por cada ocurrencia de otra entidad asociada en la relación.

La estructura lógica general de una base de datos puede expresarse en forma grafica por medio de un Modelo Entidad-Relación. El modelo Entidad-Relación se representa de dos formas, una para representar las entidades y sus relaciones llama Diagrama Entidad Relación, y la otra, para representar sus atributos llamada Modelo Relacional.

*4.3 **PROGRAMACIÓN ORIENTADA A OBJETOS***

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación. Pensar en términos de objetos es muy parecido a cómo lo haríamos en la vida real.

4.3.1 Clases en POO

Las clases son declaraciones de objetos, también se podrían definir como abstracciones de objetos. Esto quiere decir que la definición de un objeto es la clase. Cuando programamos un objeto y definimos sus características y funcionalidades en realidad lo que estamos haciendo es programar una clase.

4.3.2 Propiedades en clases

Las propiedades o atributos son las características de los objetos. Cuando definimos una propiedad normalmente especificamos su nombre y su tipo. Nos podemos hacer a la idea de que las propiedades son algo así como variables donde almacenamos datos relacionados con los objetos.

4.3.3 Métodos en las clases

Son las funcionalidades asociadas a los objetos. Cuando estamos programando las clases las llamamos métodos. Los métodos son como funciones que están asociadas a un objeto.

4.3.4 Objetos en POO

Los objetos son ejemplares de una clase cualquiera. Cuando creamos un ejemplar tenemos que especificar la clase a partir de la cual se creará. Esta acción de crear un objeto a partir de una clase se llama instanciar (que viene de una mala traducción de la palabra instace que en inglés significa ejemplar).

4.3.5 Estados en objetos

Cuando tenemos un objeto sus propiedades toman valores. El valor concreto de una propiedad de un objeto se llama estado.

4.4 XML (Extensible Markup Language)

XML es un metalenguaje, esto es, un lenguaje para definir lenguajes. Está basado en el anterior estándar SGML (Standard Generalized Markup Language, ISO 8879), que data de 1986, pero que empezó a gestarse desde principios de los años 70. Éste no es más que un modelo de objetos (en forma de API) que permite acceder a las diferentes partes que pueden componer un documento XML o HTML.

Los alcances de XML van más allá de su aplicación en Internet, más que eso se propone como lenguaje de bajo nivel (a nivel de aplicación, no de programación) para intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo, y casi cualquier cosa que podamos pensar.

4.5 XSL (Extensible Stylesheet Language)

El XSL es un lenguaje que nos permite definir una presentación o formato para un documento XML. Un mismo documento XML puede tener varias hojas de estilo XSL que lo muestren en diferentes formatos (HTML, PDF, RTF, VRML, PostScript, sonido, etc.).

Una hoja de estilo XSL es una serie de reglas que determina cómo va a ocurrir la transformación. Cada regla se compone de un patrón y una acción o plantilla.

De este modo, cada regla afecta a uno o varios elementos del documento XML. El efecto de las reglas es recursivo, para que un elemento situado dentro de otro elemento pueda ser también transformado. Las hojas de estilo tienen una regla raíz que, además de ser procesada, llama a las reglas adecuadas para los elementos hijos.

4.6 PATRONES DE DISEÑO

Algunas de las características de los patrones de diseño es dar soluciones a problemas específicos que se repiten en muchas aplicaciones diferentes. Estas soluciones puede que requieran algo más de esfuerzo que una solución "ad hoc" del problema que tengamos que afrontar, pero este esfuerzo se verá recompensado al permitirnos construir aplicaciones más flexibles, escalables y de fácil mantenimiento.

Además, la existencia de catálogos de patrones nos permite aprovechar la experiencia de otros diseñadores y mejorar nuestras habilidades en el diseño de software. En estos catálogos, los patrones se representan en un formato estándar (problema-solución-consecuencias) que codifica de una forma digerible las soluciones a problemas recurrentes que diseñadores expertos han descubierto en un contexto dado.

Los patrones de diseño nos permiten reutilizar buenos diseños y hacer explícito el conocimiento del diseño de software, mediante abstracciones de las soluciones encontradas. Amplían nuestro vocabulario, usualmente, a cada patrón de diseño se le asocia un nombre con el que hacerle referencia, en otras palabras, los patrones de diseño nos permiten no sólo mejorar nuestros diseños sino ampliar nuestras expectativas de aprendizaje y nuestra capacidad de comunicación con otros diseñadores.

4.7 PLATAFORMA .NET

Plataforma.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años con el objetivo de obtener una plataforma sencilla y potente para distribuir el software en forma de servicios que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados.

4.7.1 Common Language Runtime (CLR)

El "Common Language Runtime" es el núcleo de la plataforma .NET. Es el motor encargado de gestionar la ejecución de las aplicaciones para ella desarrolladas y a las que ofrece numerosos servicios que simplifican su desarrollo y favorecen su fiabilidad y seguridad.

4.7.2 Microsoft Intermediate Language (MSIL)

Ninguno de los compiladores que generan código para la plataforma .NET produce código máquina para CPUs x86 ni para ningún otro tipo de CPU concreta, sino que generan código escrito en el lenguaje intermedio conocido como "Microsoft Intermediate Lenguage (MSIL)". El CLR da a las aplicaciones la sensación de que se están ejecutando sobre una máquina virtual, y precisamente MSIL es el código máquina de esa máquina virtual. Es decir, MSIL es el único código que es capaz de interpretar el CLR, y por tanto cuando se

dice que un compilador genera código para la plataforma .NET lo que se está diciendo es que genera MSIL.

4.7.3 Common Type System (CTS)

El "Common Type System" (CTS) o Sistema de Tipo Común es el conjunto de reglas que han de seguir las definiciones de tipos de datos para que el CLR las acepte. Es decir, aunque cada lenguaje gestionado disponga de su propia sintaxis para definir tipos de datos, en el MSIL resultante de la compilación de sus códigos fuente se han de cumplir las reglas del CTS.

4.7.4 Ensamblados

Un ensamblado es una agrupación lógica de uno o más módulos o ficheros de recursos (ficheros .GIF, .HTML, etc.) que se engloban bajo un nombre común. Un programa puede acceder a información o código almacenados en un ensamblado sin tener que conocer cuál es el fichero en concreto donde se encuentran, por lo que los ensamblados nos permiten abstraernos de la ubicación física del código que ejecutemos o de los recursos que usemos. Por ejemplo, podemos incluir todos los tipos de una aplicación en un mismo ensamblado pero colocando los más frecuentemente usados en un cierto módulo y los menos usados en otro, de modo que sólo se descarguen de Internet los últimos si es que se van a usar.

4.7.5 Librería de clase base (BCL)

La Librería de Clase Base (BCL) es una librería incluida en el ".NET Framework" formada por cientos de tipos de datos que permiten acceder a los servicios ofrecidos por el CLR y a las funcionalidades más frecuentemente usadas a la hora de escribir programas. Además, a partir de estas clases prefabricadas el programador puede crear nuevas clases que mediante herencia extiendan su funcionalidad y se integren a la perfección con el resto de clases de la BCL. Por ejemplo, implementando ciertos interfaces podemos crear nuevos tipos de colecciones que serán tratadas exactamente igual que cualquiera de las colecciones incluidas en la BCL.

4.7.6 Características de C#

Alguna de las características aquí señaladas no son exactamente propias del lenguaje sino de la plataforma .NET en general, y si aquí se comentan es porque tienen una repercusión directa en el lenguaje:

- ***Sencillez:** C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:*

  - *El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile (no como en C++), lo que facilita la portabilidad del código.*

- *No se incluyen elementos poco útiles de lenguajes como C++ tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (.) acceder a miembros de espacios de nombres (::)*

- ***Modernidad:*** *C# incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, como un tipo básico "decimal" que permita realizar operaciones de alta precisión con reales de 128 bits (muy útil en el mundo financiero), la inclusión de una instrucción "foreach" que permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico "string" para representar cadenas o la distinción de un tipo "bool" específico para representar valores lógicos.*

- ***Orientación a objetos:*** *Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos, aunque eso es más bien una característica del CTS que de C#. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código. C# soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo.*

En lo referente a la encapsulación es importante señalar que aparte de los típicos modificadores "public", "private" y "protected", C# añade un cuarto modificador llamado "internal", que puede combinarse con "protected" e indica que al elemento a cuya definición precede sólo puede accederse desde su mismo ensamblado.

Respecto a la herencia C# sólo admite herencia simple de clases ya que la múltiple provoca más quebraderos de cabeza que facilidades y en la mayoría de los casos su utilidad puede ser simulada con facilidad mediante herencia múltiple de interfaces. De todos modos, esto vuelve a ser más bien una característica propia del CTS que de C#.

Por otro lado y a diferencia de Java, en C# se ha optado por hacer que todos los métodos sean por defecto sellados y que los redefinibles hayan de marcarse con el modificador "virtual" (como en C++), lo que permite evitar errores derivados de redefiniciones accidentales. Además, un efecto secundario de esto es que las llamadas a los métodos serán más eficientes por defecto al no tenerse que buscar en la tabla de funciones virtuales la implementación de los mismos a la que se ha de llamar. Otro efecto secundario es que permite que las llamadas a los métodos virtuales se puedan hacer más eficientemente al contribuir a que el tamaño de dicha tabla se reduzca.

- ***Orientación a componentes:*** *La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros).*

- ***Gestión automática de memoria:*** *Como ya se comentó, todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando éste se active –ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente-, C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción "using".*

- ***Seguridad de tipos:*** *C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evita que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura. Para ello se toman medidas del tipo:*

- *Sólo se admiten conversiones entre tipos compatibles. Esto es, entre un tipo y antecesores suyos, entre tipos para los que explícitamente se haya definido un operador de conversión, y entre un tipo y un tipo hijo suyo del que un objeto del primero almacenase una referencia del segundo (downcasting) Obviamente, lo último sólo puede comprobarlo en tiempo de ejecución el CLR y no el compilador, por lo que en realidad el CLR y el compilador colaboran para asegurar la corrección de las conversiones.*
- *No se pueden usar "variables no inicializadas". El compilador da a los campos un valor por defecto consistente en ponerlos a cero y controla mediante análisis del flujo de control de la fuente que no se lea ninguna variable local sin que se le haya asignado previamente algún valor.*
- *Se comprueba que todo acceso a los elementos de una tabla se realice con índices que se encuentren dentro del rango de la misma.*
- *Se puede controlar la producción de desbordamientos en operaciones aritméticas, informándose de ello con una excepción cuando ocurra. Sin embargo, para conseguirse un mayor rendimiento en la aritmética estas comprobaciones no se hacen por defecto al operar con variables sino sólo con constantes (se pueden detectar en tiempo de compilación)*
- *C# incluye "delegados", que son similares a los punteros a funciones de C++ pero siguen un enfoque orientado a objetos, pueden almacenar referencias a varios métodos simultáneamente, y se comprueba que los*

métodos a los que apunten tengan parámetros y valor de retorno del tipo indicado al definirlos.

- ***Instrucciones seguras:*** *Para evitar errores muy comunes, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, la guarda de toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (==) con el de asignación (=); y todo caso de un "switch" ha de terminar en un "break" o "goto" que indique cuál es la siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación.*

- ***Sistema de tipos unificado:*** *A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada "System.Object", por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán "objetos")*

  En C# esto también es aplicable a los tipos de datos básicos Además, para conseguir que ello no tenga una repercusión negativa en su nivel de rendimiento, se ha incluido un mecanismo transparente de "boxing" y "unboxing" con el que se consigue que sólo sean tratados como objetos cuando la situación lo requiera, y mientras tanto puede aplicárseles optimizaciones específicas.

El hecho de que todos los tipos del lenguaje deriven de una clase común facilita enormemente el diseño de colecciones genéricas que puedan almacenar objetos de cualquier tipo.

- ***Extensibilidad de tipos básicos:*** *C# permite definir, a través de "estructuras", tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos. Es decir, que se puedan almacenar directamente en pila (luego su creación, destrucción y acceso serán más rápidos) y se asignen por valor y no por referencia. Para conseguir que lo último no tenga efectos negativos al pasar estructuras como parámetros de métodos, se da la posibilidad de pasar referencias a pila a través del modificador de parámetro "ref".*

- ***Extensibilidad de operadores:*** *Para facilitar la legibilidad del código y conseguir que los nuevos tipos de datos básicos que se definan a través de las estructuras estén al mismo nivel que los básicos predefinidos en el lenguaje, al igual que C++ y a diferencia de Java, C# permite redefinir el significado de la mayoría de los operadores -incluidos los de conversión, tanto para conversiones implícitas como explícitas- cuando se apliquen a diferentes tipos de objetos.*

  Las redefiniciones de operadores se hacen de manera inteligente, de modo que a partir de una única definición de los operadores **++** *y* **--** *el compilador puede*

*deducir automáticamente como ejecutarlos de manera prefijas y postfija; y definiendo operadores simples (como **+**), el compilador deduce cómo aplicar su versión de asignación compuesta (**+=**) Además, para asegurar la consistencia, el compilador vigila que los operadores con opuesto siempre se redefinan por parejas (por ejemplo, si se redefine **==**, también hay que redefinir **!=**)*

*También se da la posibilidad, a través del concepto de "indizador", de redefinir el significado del operador **[]** para los tipos de dato definidos por el usuario, con lo que se consigue que se pueda acceder al mismo como si fuese una tabla. Esto es muy útil para trabajar con tipos que actúen como colecciones de objetos.*

- ***Extensibilidad de modificadores:** C# ofrece, a través del concepto de **atributos**, la posibilidad de añadir a los metadatos del módulo resultante de la compilación de cualquier fuente información adicional a la generada por el compilador que luego podrá ser consultada en tiempo ejecución a través de la librería de reflexión de .NET . Esto, que más bien es una característica propia de la plataforma .NET y no de C#, puede usarse como un mecanismo para definir nuevos modificadores.*

- ***Versionable:** C# incluye una "política de versionado" que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen*

errores difíciles de detectar en tipos hijos previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos.

Si una clase introduce un nuevo método cuyas redefiniciones deban seguir la regla de llamar a la versión de su padre en algún punto de su código, difícilmente seguirían esta regla miembros de su misma signatura definidos en clases hijas previamente a la definición del mismo en la clase padre; o si introduce un nuevo campo con el mismo nombre que algún método de una clase hija, la clase hija dejará de funcionar. Para evitar que esto ocurra, en C# se toman dos medidas:

- *Se obliga a que toda redefinición deba incluir el modificador "override", con lo que la versión de la clase hija nunca sería considerada como una redefinición de la versión de miembro en la clase padre ya que no incluiría "override". Para evitar que por accidente un programador incluya este modificador, sólo se permite incluirlo en miembros que tengan la misma signatura que miembros marcados como redefinibles mediante el modificador "virtual". Así además se evita el error tan frecuente en Java de creerse haber redefinido un miembro, pues si el miembro con "override" no existe en la clase padre se producirá un error de compilación.*
- *Si no se considera redefinición, entonces se considera que lo que se desea es ocultar el método de la clase padre, de modo que para la clase hija sea*

como si nunca hubiese existido. El compilador avisará de esta decisión a través de un mensaje de aviso que puede suprimirse incluyendo el modificador "new" en la definición del miembro en la clase hija para así indicarle explícitamente la intención de ocultación.

- ***Eficiente:*** *En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador "unsafe") y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad procesamiento muy grandes.*

- ***Compatible:*** *Para facilitar la migración de programadores, C# mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes.*

5 JUSTIFICACIÓN

Alrededor de la información pertinente a la administración del personal de C.I. Técnicas Baltime de Colombia S.A. se ha notado la necesidad de implementar un sistema el cuál permita obtener y manejar de forma rápida y eficaz información relacionada al empleado, como, información familiar, información ocupacional, capacitaciones, información socioeconómica.

Por tal motivo la organización se ha visto incapacitada para tomar decisiones o formar proyectos sociales que involucren a los empleados.

Otro factor que influye es que la información de los empleados de las fincas es manejada por sistemas locales, y cualquier modificación de esta obliga al empleado a transportarse hasta la ciudad, generando con esto inconsistencia y duplicidad de la información.

En la compañía los empleados antiguos, tanto como los nuevos deben ser capacitados y reforzados anualmente en procedimientos ISO. Este control de capacitaciones se lleva manualmente, imposibilitando al departamento de RRHH el control de empleados que han sido capacitados.

A todo esto se suma que esta área maneja un gran volumen de información por lo tanto se encuentra urgida de una solución informática a la medida.

En la actualidad no se lleva registro de los datos contenidos en las hojas de vida y por lo tanto es muy difícil la elaboración de informes.

Se hace complicado controlar toda la información que el personal presenta a la oficina.

Todos los informes se realizan en herramientas de Office por lo tanto se presenta la posibilidad que estos archivos se extravíen y que los datos puedan variar de un informe a otro.

Dentro de estos factores, también se puede destacar la dificultad de la obtención de informes, proyecciones o estadísticas de la información de acuerdo a los requerimientos de los usuarios.

Con todos estos factores que influyen en los procesos que se llevan en el área de Recursos Humanos, generando un vació en esta y un atraso en algunas labores.

El sistema de información SETHUM el cual se implementara en C.I. Técnicas Baltime de Colombia S.A. es un sistema orientado a Web que manejará una base de datos

centralizada, lo cuál permitirá la actualización inmediata desde diferentes sedes, tales como Serteba ubicada en la Troncal del Caribe, Asociación Agrícola Eufemia ubicada en la Zona Bananera, Agropecuaria San Gabriel ubicada en la Zona de la Aguja.

Con esto se eliminara la redundancia e inconsistencia de la información vital del empleado, tales como socioeconómica, sociofamiliar, ocupacional y personal.

Estas características del sistema, apoyada en un diseño de interfaz modular harán posible el procesamiento de información única, actualizada, con mayor rapidez y veracidad total en su presentación.

6 OBJETIVOS

6.1 OBJETIVO GENERAL

Analizar, desarrollar e implantar un sistema de información de recursos humanos que gestione la información social, económica, familiar, de capacitación, ocupacional de los empleados en C.I. Técnicas Baltime de Colombia S.A.

6.2 OBJETIVOS ESPECÍFICOS

- *Facilitar la realización y procesamiento de la información de los empleados de una manera eficiente, rápida y segura desde su punto de trabajo.*

- *Generar estadísticas para la toma de decisiones en el departamento de RRHH de la compañía.*

- *Diseñar un sistema modular e integral que permita eliminar la duplicidad de información de los empleados.*

- *Reducir los costos en transporte, control y procesamiento de información de la compañía.*

- *Realizar un control de la seguridad de la información de RRHH mediante mecanismos de control de acceso y registros de auditoria.*

7 FORMULACIÓN

El presente proyecto busca desarrollar una robusta aplicación para solucionar el problema con el manejo de la información del talento humano en C.I. Técnicas Baltime de Colombia S.A.

Este proyecto propone un desarrollo aplicando estándares bien definidos en el desarrollo de software y usando las ultimas tecnologías empleadas en el desarrollo de aplicaciones Web, para presentarle al usuario una sencilla y agradable interfaz grafica, evitando con esto la resistencia al cambio impuestas muchas veces por el personal involucrado en los procesos debido a la complejidad en el manejo de las soluciones informáticas.

8 DISEÑO METODOLOGICO

El Sistema Electrónico de Talento Humano se elaboro bajo esta metodología la cual se llevo a cabo con el cronograma planteado. En el anexo A se presenta el cronograma de actividades.

8.1 DELIMITACION DEL ESPACIO TEMPORAL Y GEOGRÁFICO

El espacio temporal en el cual se realizará el trabajo de memoria de grado esta comprendido entre el momento que se aprobó el proyecto y 5 meses después, tiempo en el cual se puso en funcionamiento el sistema, luego de haber sido sometido a las pruebas pertinentes.

Este trabajo se desarrollo de manera colaborativa en las instalaciones de la Compañía C.I. Técnicas Baltime de Colombia S.A. Ubicada en la Troncal de Caribe Km. 2 vía a Gaira Ubicada en la ciudad de Santa Marta DTCH, capital del departamento del Magdalena.

8.2 RECOPILACION Y ANALISIS DE LA INFORMACION

8.2.1 Recopilación de la información

Para tener una mejor compilación de las necesidades del usuario, se recolecto información concerniente al departamento de Sistemas y de Recursos Humanos, se utilizaron las siguientes técnicas:

- *Entrevistas directas al gerente de Recursos Humanos, trabajadora social y usuarios finales.*
- *Observación directa de las condiciones actuales y los procedimientos empleados por el personal encargado de manejar la información.*
- *Entrevistas con el jefe de sistemas, para conocer la infraestructura de comunicaciones y los software de desarrollo con que cuenta la empresa.*

Además de las consultas bibliografías que se constituyeron el soporte teórico del diseño, desarrollo e implementación del Sistema Electrónico de Talento Humano para C.I. Técnicas Baltime de Colombia S.A. En este proceso se tuvieron en cuenta los siguientes puntos:

- *Procesos y procedimientos en C.I. Técnicas Baltime de Colombia S.A.*
- *Política de seguridad de la información en C.I. Técnicas Baltime de Colombia S.A.*
- *Requerimientos y necesidades del Departamento de Recursos Humanos de C.I. Técnicas Baltime de Colombia S.A.*
- *Manejo actual de los reportes.*
- *Infraestructura (Redes) y equipos.*
- *Recursos materiales y humanos que hacen parte del desarrollo e implementación del Software*
- *Herramientas necesarias para la realización del proyecto.*

8.2.2 Análisis de la información

El ciclo de vida utilizado en el Sistema Electrónico de Talento Humano es el ciclo de vida incremental. Se opto por este modelo debido a que la aplicación se entrego a producción por módulos.

También se definió los módulos que integrarían la aplicación, entre los cuales encontramos:

- *Modulo información personal del empleado*
- *Modulo información referente al empleado.*
- *Modulo de la información familiar del empleado.*
- *Modulo de vivienda del empleado.*
- *Modulo de capacitación.*
- *Modulo de dotación.*
- *Modulo de EPP.*
- *Modulo de ausentismos.*

Además se definió la plataforma y los lenguajes de programación que se trabajaría el sistema, en este proceso se opto por Microsoft Visual Estudio C#.NET.

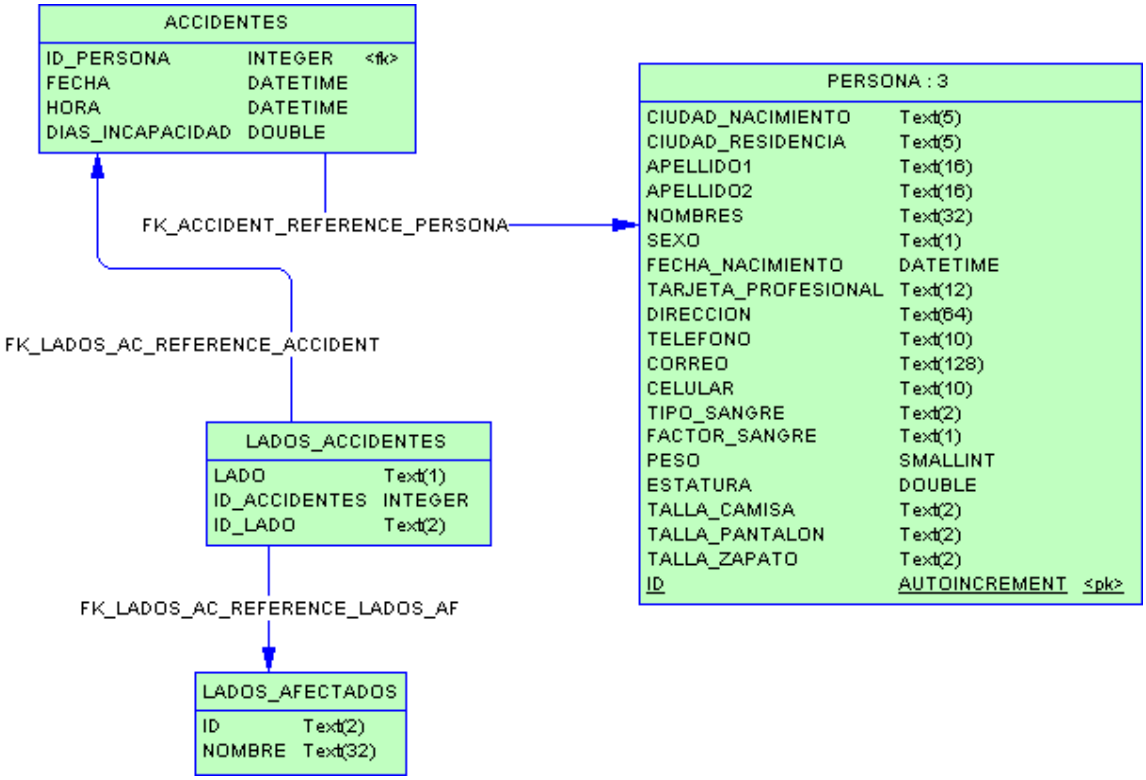
8.3 DISEÑO DEL PROYECTO

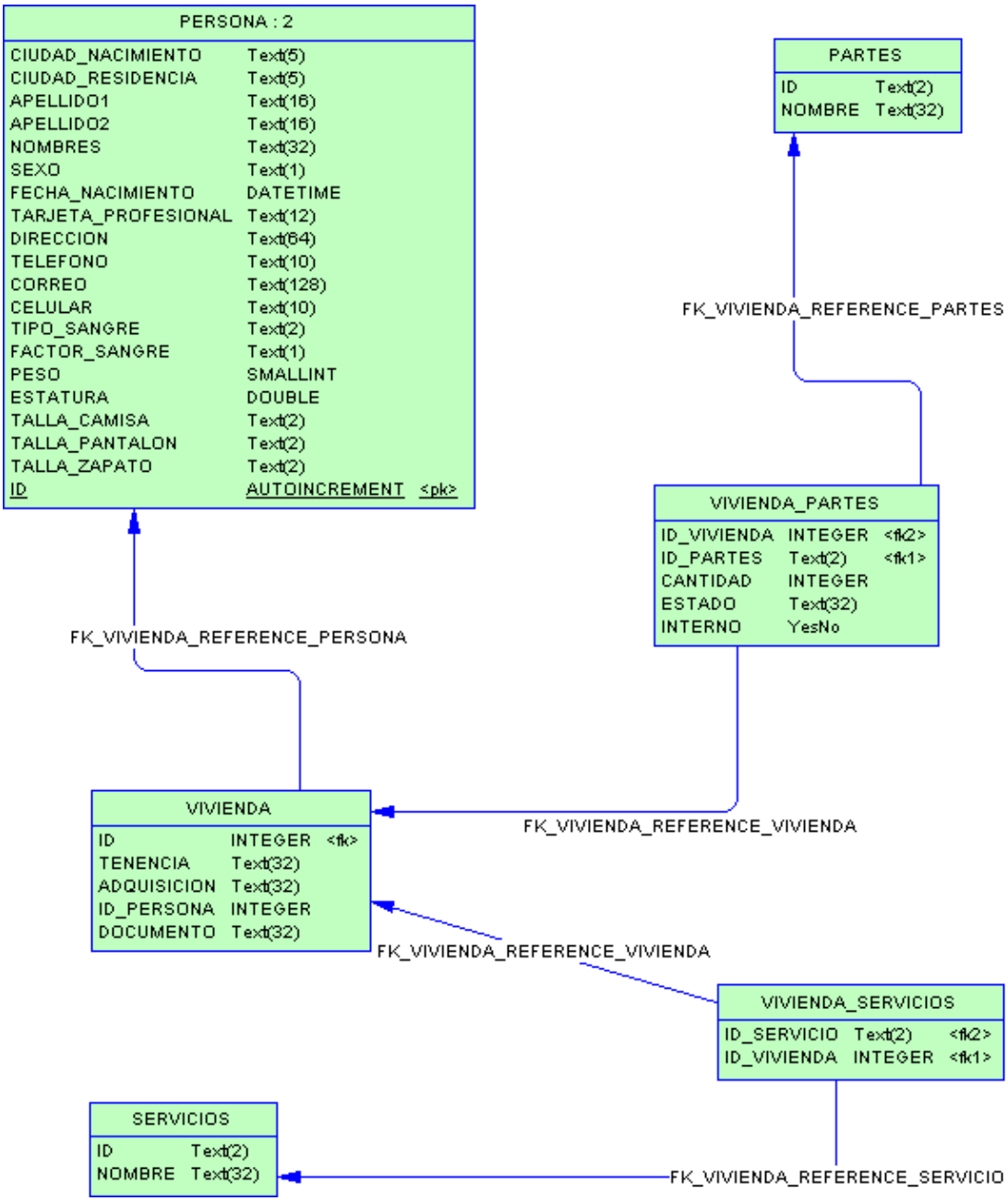
8.3.1 Diseño de la Base de Datos

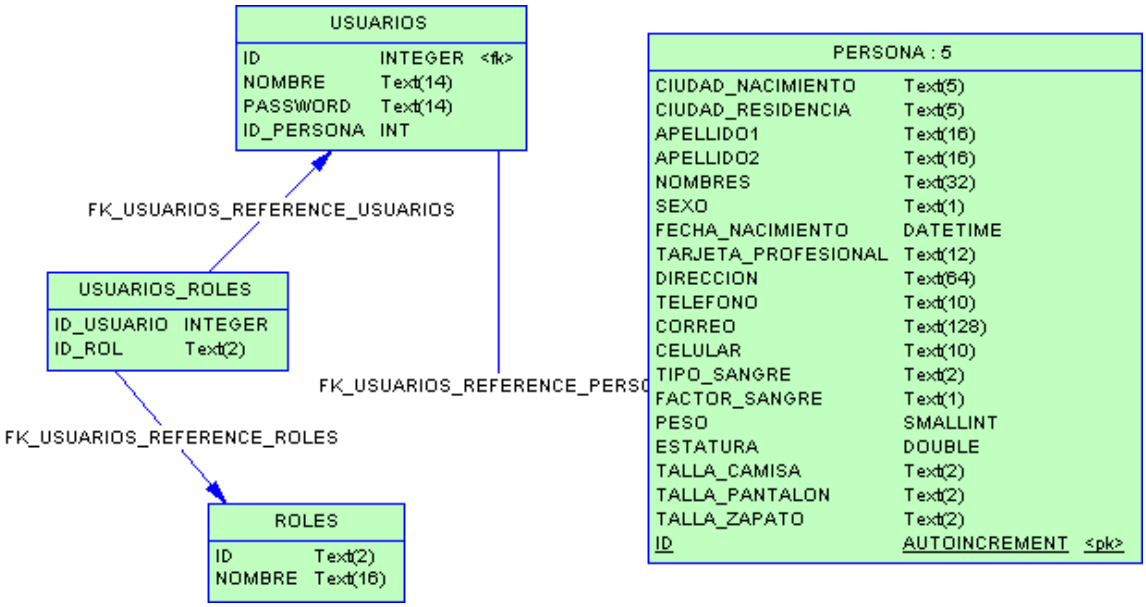
En esta etapa se elaboraron los modelos lógicos y funcionales que conformarían el nuevo sistema de información, iniciando por el modelo entidad/relación, seguido del modelo relacional. Esta etapa implico lo siguiente:

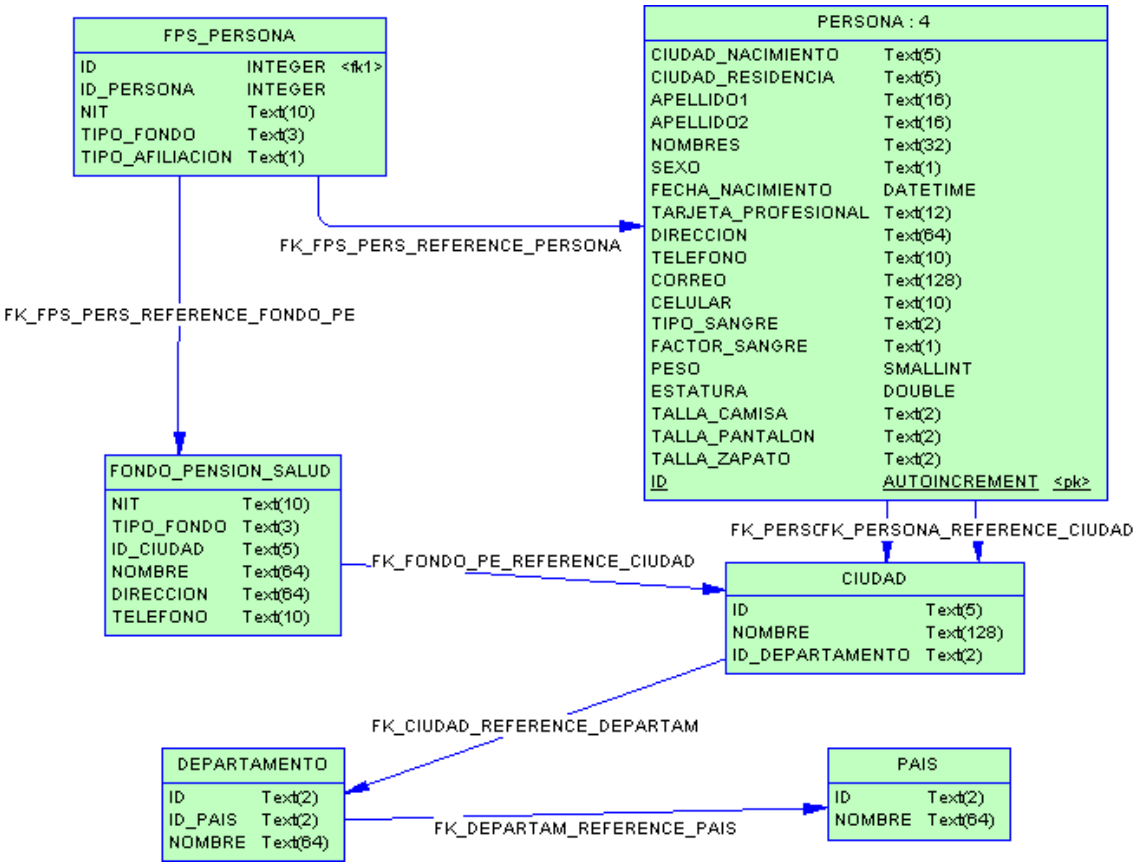
- *Definición de modelos de diseño, es decir se elaboro la modelación lógica de los datos recolectados en la etapa de análisis.*
- *Confrontación de los modelos realizados con la realidad.*

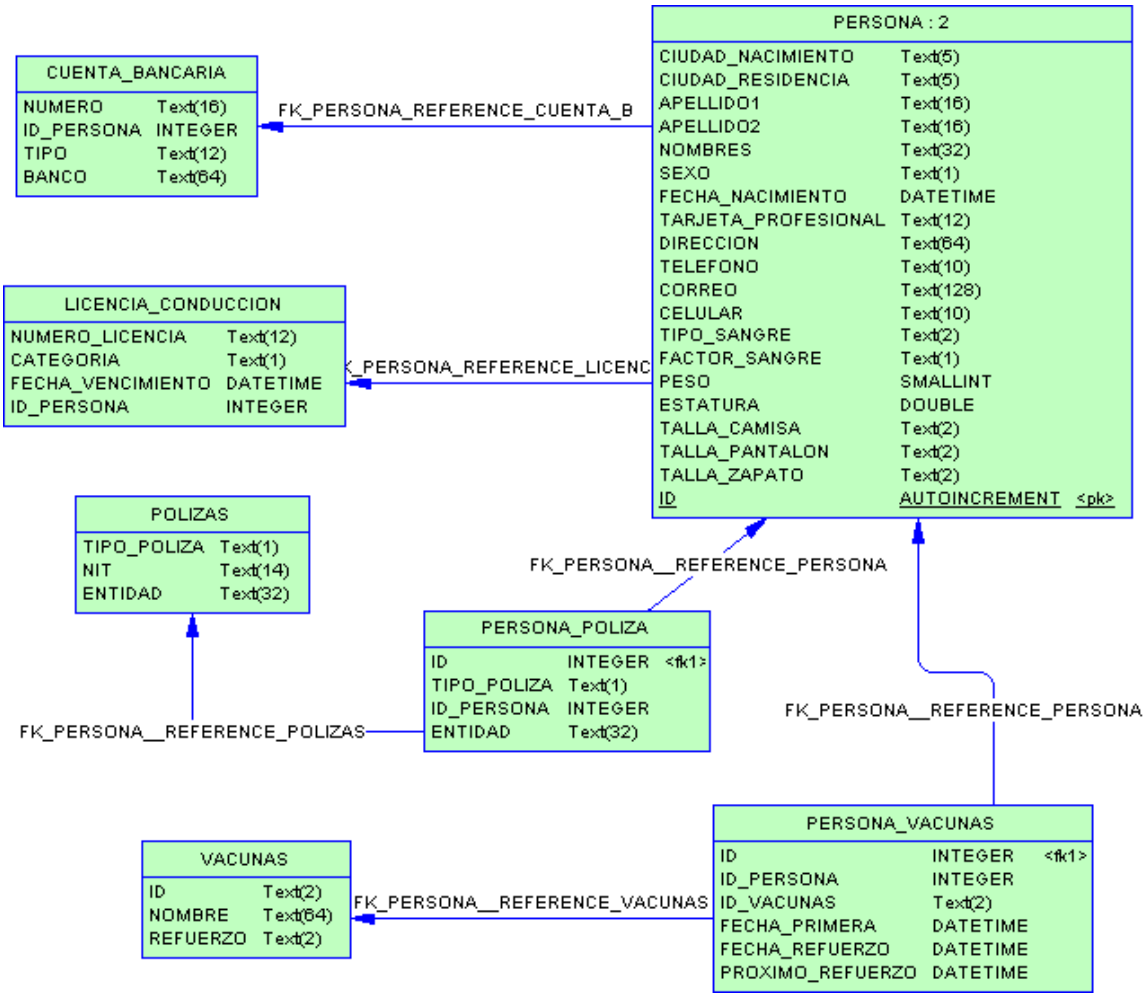
8.3.2 Grafica del modelo relacional

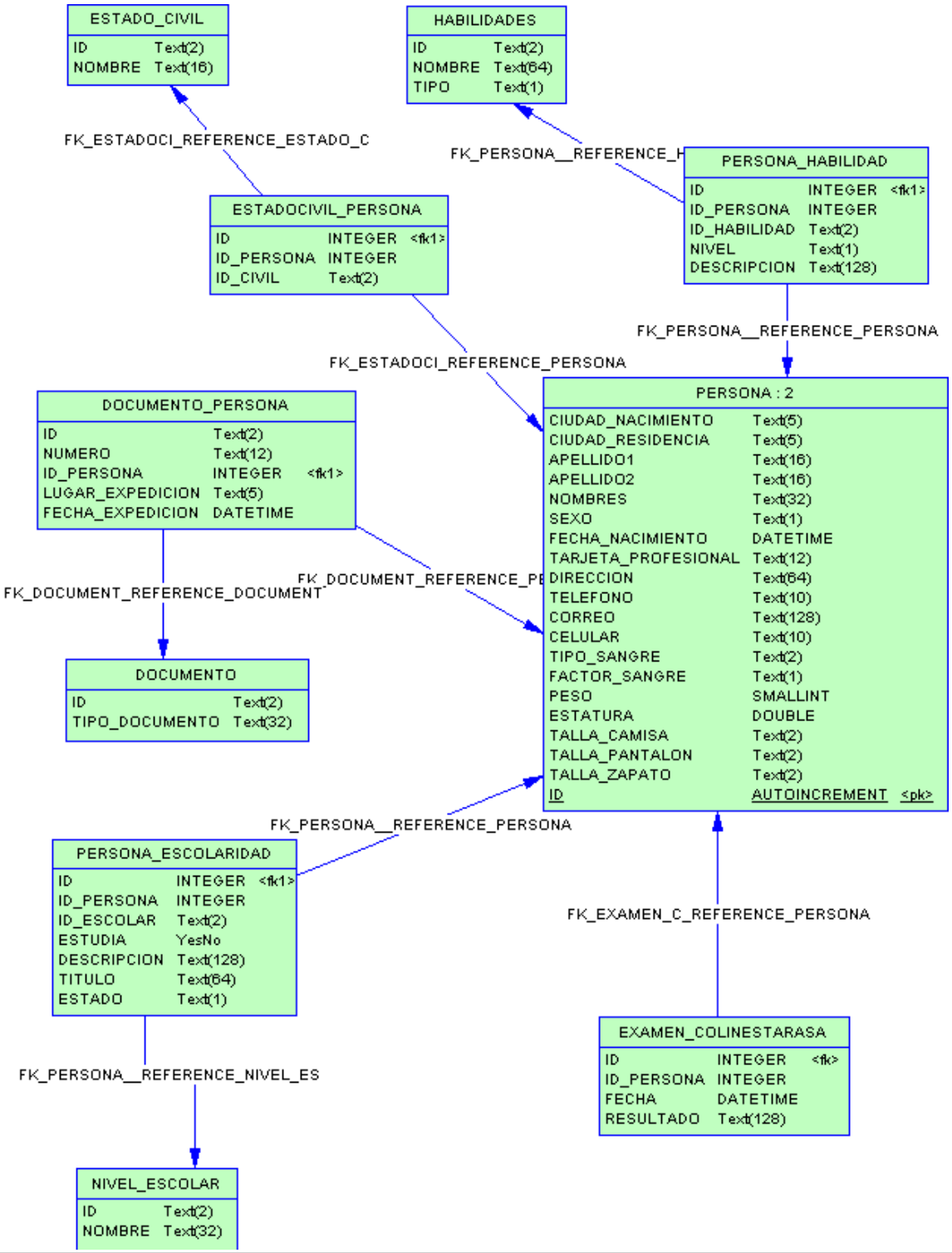


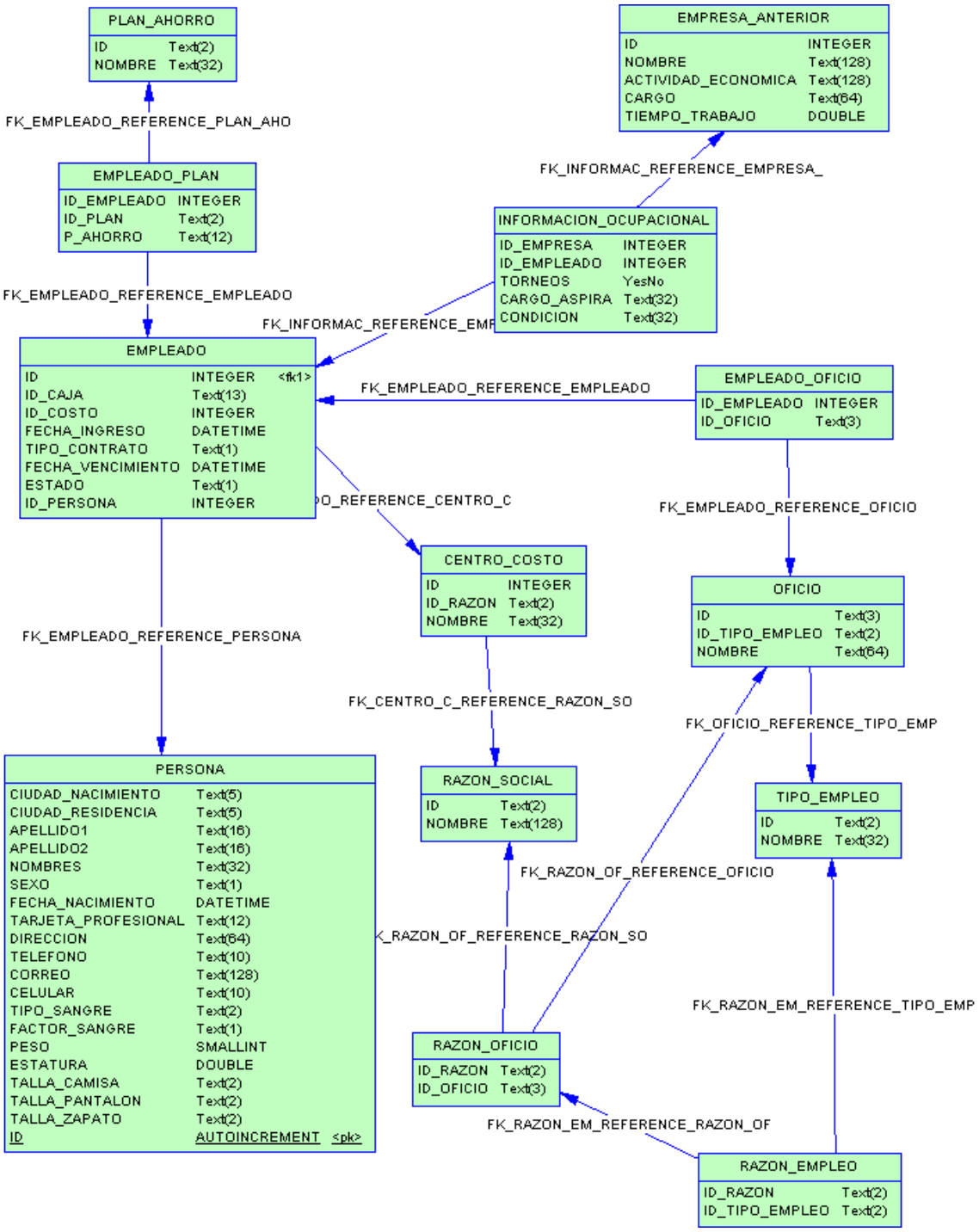

PERSONA : 2
CIUDAD_NACIMIENTO Text(5)
CIUDAD_RESIDENCIA Text(5)
APELLIDO1 Text(16)
APELLIDO2 Text(16)
NOMBRES Text(32)
SEXO Text(1)
FECHA_NACIMIENTO DATETIME
TARJETA_PROFESIONAL Text(12)
DIRECCION Text(64)
TELEFONO Text(10)
CORREO Text(128)
CELULAR Text(10)
TIPO_SANGRE Text(2)
FACTOR_SANGRE Text(1)
PESO SMALLINT
ESTATURA DOUBLE
TALLA_CAMISA Text(2)
TALLA_PANTALON Text(2)
TALLA_ZAPATO Text(2)
ID AUTOINCREMENT <pk>
PARTES
ID Text(2)
NOMBRE Text(32)
FK_VIVIENDA_REFERENCE_PARTES
VIVIENDA_PARTES
ID_VIVIENDA INTEGER <fk2>
ID_PARTES Text(2) <fk1>
CANTIDAD INTEGER
ESTADO Text(32)
INTERNO YesNo
FK_VIVIENDA_REFERENCE_PERSONA
VIVIENDA
ID INTEGER <fk>
TENENCIA Text(32)
ADQUISICION Text(32)
ID_PERSONA INTEGER
DOCUMENTO Text(32)
FK_VIVIENDA_REFERENCE_VIVIENDA
FK_VIVIENDA_REFERENCE_VIVIENDA
VIVIENDA_SERVICIOS
ID_SERVICIO Text(2) <fk2>
ID_VIVIENDA INTEGER <fk1>
SERVICIOS
ID Text(2)
NOMBRE Text(32)
FK_VIVIENDA_REFERENCE_SERVICIO

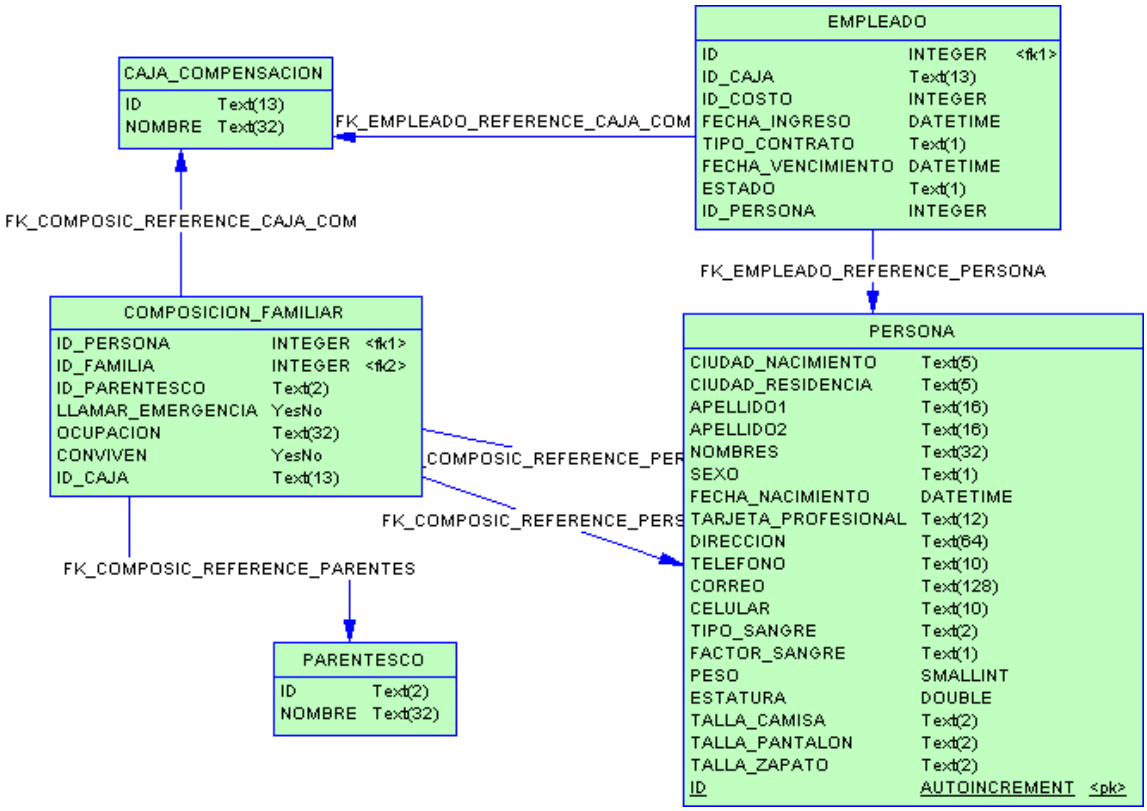

USUARIOS
ID INTEGER <fk>
NOMBRE Text(14)
PASSWORD Text(14)
ID_PERSONA INT
FK_USUARIOS_REFERENCE_USUARIOS
USUARIOS_ROLES
ID_USUARIO INTEGER
ID_ROL Text(2)
FK_USUARIOS_REFERENCE_PERSO
FK_USUARIOS_REFERENCE_ROLES
ROLES
ID Text(2)
NOMBRE Text(16)
PERSONA : 5
CIUDAD_NACIMIENTO Text(5)
CIUDAD_RESIDENCIA Text(5)
APELLIDO1 Text(16)
APELLIDO2 Text(16)
NOMBRES Text(32)
SEXO Text(1)
FECHA_NACIMIENTO DATETIME
TARJETA_PROFESIONAL Text(12)
DIRECCION Text(64)
TELEFONO Text(10)
CORREO Text(128)
CELULAR Text(10)
TIPO_SANGRE Text(2)
FACTOR_SANGRE Text(1)
PESO SMALLINT
ESTATURA DOUBLE
TALLA_CAMISA Text(2)
TALLA_PANTALON Text(2)
TALLA_ZAPATO Text(2)
ID AUTOINCREMENT <pk>


FPS_PERSONA
ID INTEGER <fk1>
ID_PERSONA INTEGER
NIT Text(10)
TIPO_FONDO Text(3)
TIPO_AFILIACION Text(1)
PERSONA : 4
CIUDAD_NACIMIENTO Text(5)
CIUDAD_RESIDENCIA Text(5)
APELLIDO1 Text(16)
APELLIDO2 Text(16)
NOMBRES Text(32)
SEXO Text(1)
FECHA_NACIMIENTO DATETIME
TARJETA_PROFESIONAL Text(12)
DIRECCION Text(64)
TELEFONO Text(10)
CORREO Text(128)
CELULAR Text(10)
TIPO_SANGRE Text(2)
FACTOR_SANGRE Text(1)
PESO SMALLINT
ESTATURA DOUBLE
TALLA_CAMISA Text(2)
TALLA_PANTALON Text(2)
TALLA_ZAPATO Text(2)
ID AUTOINCREMENT <pk>
FK_FPS_PERS_REFERENCE_PERSONA
FK_FPS_PERS_REFERENCE_FONDO_PE
FONDO_PENSION_SALUD
NIT Text(10)
TIPO_FONDO Text(3)
ID_CIUDAD Text(5)
NOMBRE Text(64)
DIRECCION Text(64)
TELEFONO Text(10)
FK_PERSCFK_PERSONA_REFERENCE_CIUDAD
FK_FONDO_PE_REFERENCE_CIUDAD
CIUDAD
ID Text(5)
NOMBRE Text(128)
ID_DEPARTAMENTO Text(2)
FK_CIUDAD_REFERENCE_DEPARTAM
DEPARTAMENTO
ID Text(2)
ID_PAIS Text(2)
NOMBRE Text(64)
PAIS
ID Text(2)
NOMBRE Text(64)
FK_DEPARTAM_REFERENCE_PAIS


PERSONA : 2
CIUDAD_NACIMIENTO Text(5)
CIUDAD_RESIDENCIA Text(5)
APELLIDO1 Text(16)
APELLIDO2 Text(16)
NOMBRES Text(32)
SEXO Text(1)
FECHA_NACIMIENTO DATETIME
TARJETA_PROFESIONAL Text(12)
DIRECCION Text(64)
TELEFONO Text(10)
CORREO Text(128)
CELULAR Text(10)
TIPO_SANGRE Text(2)
FACTOR_SANGRE Text(1)
PESO SMALLINT
ESTATURA DOUBLE
TALLA_CAMISA Text(2)
TALLA_PANTALON Text(2)
TALLA_ZAPATO Text(2)
ID AUTOINCREMENT <pk>
CUENTA_BANCARIA
NUMERO Text(16)
ID_PERSONA INTEGER
TIPO Text(12)
BANCO Text(64)
FK_PERSONA_REFERENCE_CUENTA_B
LICENCIA_CONDUCCION
NUMERO_LICENCIA Text(12)
CATEGORIA Text(1)
FECHA_VENCIMIENTO DATETIME
ID_PERSONA INTEGER
K_PERSONA_REFERENCE_LICENC
POLIZAS
TIPO_POLIZA Text(1)
NIT Text(14)
ENTIDAD Text(32)
FK_PERSONA__REFERENCE_PERSONA
PERSONA_POLIZA
ID INTEGER <fk1>
TIPO_POLIZA Text(1)
ID_PERSONA INTEGER
ENTIDAD Text(32)
FK_PERSONA__REFERENCE_POLIZAS
FK_PERSONA__REFERENCE_PERSONA
PERSONA_VACUNAS
ID INTEGER <fk1>
ID_PERSONA INTEGER
ID_VACUNAS Text(2)
FECHA_PRIMERA DATETIME
FECHA_REFUERZO DATETIME
PROXIMO_REFUERZO DATETIME
VACUNAS
ID Text(2)
NOMBRE Text(64)
REFUERZO Text(2)
FK_PERSONA__REFERENCE_VACUNAS


ESTADO_CIVIL
ID Text(2)
NOMBRE Text(16)
HABILIDADES
ID Text(2)
NOMBRE Text(64)
TIPO Text(1)
FK_ESTADOCI_REFERENCE_ESTADO_C
FK_PERSONA__REFERENCE_H
PERSONA_HABILIDAD
ID INTEGER <fk1>
ID_PERSONA INTEGER
ID_HABILIDAD Text(2)
NIVEL Text(1)
DESCRIPCION Text(128)
ESTADOCIVIL_PERSONA
ID INTEGER <fk1>
ID_PERSONA INTEGER
ID_CIVIL Text(2)
FK_PERSONA__REFERENCE_PERSONA
FK_ESTADOCI_REFERENCE_PERSONA
PERSONA : 2
CIUDAD_NACIMIENTO Text(5)
CIUDAD_RESIDENCIA Text(5)
APELLIDO1 Text(16)
APELLIDO2 Text(16)
NOMBRES Text(32)
SEXO Text(1)
FECHA_NACIMIENTO DATETIME
TARJETA_PROFESIONAL Text(12)
DIRECCION Text(64)
TELEFONO Text(10)
CORREO Text(128)
CELULAR Text(10)
TIPO_SANGRE Text(2)
FACTOR_SANGRE Text(1)
PESO SMALLINT
ESTATURA DOUBLE
TALLA_CAMISA Text(2)
TALLA_PANTALON Text(2)
TALLA_ZAPATO Text(2)
ID AUTOINCREMENT <pk>
DOCUMENTO_PERSONA
ID Text(2)
NUMERO Text(12)
ID_PERSONA INTEGER <fk1>
LUGAR_EXPEDICION Text(5)
FECHA_EXPEDICION DATETIME
FK_DOCUMENT_REFERENCE_PE
FK_DOCUMENT_REFERENCE_DOCUMENT
DOCUMENTO
ID Text(2)
TIPO_DOCUMENTO Text(32)
FK_PERSONA__REFERENCE_PERSONA
PERSONA_ESCOLARIDAD
ID INTEGER <fk1>
ID_PERSONA INTEGER
ID_ESCOLAR Text(2)
ESTUDIA YesNo
DESCRIPCION Text(128)
TITULO Text(64)
ESTADO Text(1)
FK_EXAMEN_C_REFERENCE_PERSONA
EXAMEN_COLINESTARASA
ID INTEGER <fk>
ID_PERSONA INTEGER
FECHA DATETIME
RESULTADO Text(128)
FK_PERSONA__REFERENCE_NIVEL_ES
NIVEL_ESCOLAR
ID Text(2)
NOMBRE Text(32)


PLAN_AHORRO
ID Text(2)
NOMBRE Text(32)
FK_EMPLEADO_REFERENCE_PLAN_AHO
EMPLEADO_PLAN
ID_EMPLEADO INTEGER
ID_PLAN Text(2)
P_AHORRO Text(12)
FK_EMPLEADO_REFERENCE_EMPLEADO
EMPLEADO
ID INTEGER <fk1>
ID_CAJA Text(13)
ID_COSTO INTEGER
FECHA_INGRESO DATETIME
TIPO_CONTRATO Text(1)
FECHA_VENCIMIENTO DATETIME
ESTADO Text(1)
ID_PERSONA INTEGER
EMPRESA_ANTERIOR
ID INTEGER
NOMBRE Text(128)
ACTIVIDAD_ECONOMICA Text(128)
CARGO Text(64)
TIEMPO_TRABAJO DOUBLE
FK_INFORMAC_REFERENCE_EMPRESA_
INFORMACION_OCUPACIONAL
ID_EMPRESA INTEGER
ID_EMPLEADO INTEGER
TORNEOS YesNo
CARGO_ASPIRA Text(32)
CONDICION Text(32)
FK_INFORMAC_REFERENCE_EMP
FK_EMPLEADO_REFERENCE_EMPLEADO
EMPLEADO_OFICIO
ID_EMPLEADO INTEGER
ID_OFICIO Text(3)
O_REFERENCE_CENTRO_C
FK_EMPLEADO_REFERENCE_OFICIO
CENTRO_COSTO
ID INTEGER
ID_RAZON Text(2)
NOMBRE Text(32)
OFICIO
ID Text(3)
ID_TIPO_EMPLEO Text(2)
NOMBRE Text(64)
FK_EMPLEADO_REFERENCE_PERSONA
FK_CENTRO_C_REFERENCE_RAZON_SO
FK_OFICIO_REFERENCE_TIPO_EMP
PERSONA
CIUDAD_NACIMIENTO Text(5)
CIUDAD_RESIDENCIA Text(5)
APELLIDO1 Text(16)
APELLIDO2 Text(16)
NOMBRES Text(32)
SEXO Text(1)
FECHA_NACIMIENTO DATETIME
TARJETA_PROFESIONAL Text(12)
DIRECCION Text(64)
TELEFONO Text(10)
CORREO Text(128)
CELULAR Text(10)
TIPO_SANGRE Text(2)
FACTOR_SANGRE Text(1)
PESO SMALLINT
ESTATURA DOUBLE
TALLA_CAMISA Text(2)
TALLA_PANTALON Text(2)
TALLA_ZAPATO Text(2)
ID AUTOINCREMENT <pk>
RAZON_SOCIAL
ID Text(2)
NOMBRE Text(128)
TIPO_EMPLEO
ID Text(2)
NOMBRE Text(32)
FK_RAZON_OF_REFERENCE_OFICIO
K_RAZON_OF_REFERENCE_RAZON_SO
FK_RAZON_EM_REFERENCE_TIPO_EMP
RAZON_OFICIO
ID_RAZON Text(2)
ID_OFICIO Text(3)
FK_RAZON_EM_REFERENCE_RAZON_OF
RAZON_EMPLEO
ID_RAZON Text(2)
ID_TIPO_EMPLEO Text(2)



8.3.3 Diseño de la Aplicación

La división de un sistema en distintas capas o niveles de abstracción es una de las técnicas más comunes empleadas por los ingenieros para construir sistemas complejos.

Desde el punto de vista de la Ingeniería del Software, la división de un sistema en capas facilita el diseño modular y permite la construcción de sistemas débilmente acoplados

Una de las partes mas difíciles en este desarrollo fue decidir cuántas capas utilizar y qué responsabilidades asociarle a cada capa.

La aplicación esta diseñada con una arquitectura de 4 capas. A continuación explicaremos cada capa en detalle.

8.3.3.1 Capa de presentación

La capa de presentación es la encargada de interactuar con el usuario de la aplicación mediante una interfaz de usuario. Nuestro aplicación usa una interfaz Web, en la que con ayuda de el Lenguaje de programación JavaScript desarrollamos un cliente enriquecido.

En esta capa utilizamos una técnica llamada "Remote Scripting" que consiste en que solo recargamos los datos estrictamente necesarios, conservando así, a través de la actualización de datos, los elementos estáticos a nuestro propósito, tal como imágenes, estructura html, scripts embebidos, etcétera. Logrando así agilizar de forma significativa el tiempo de carga, mostrando con menor demora los datos que realmente interesan.

8.3.3.2 Comunicación entre cliente y servidor WEB

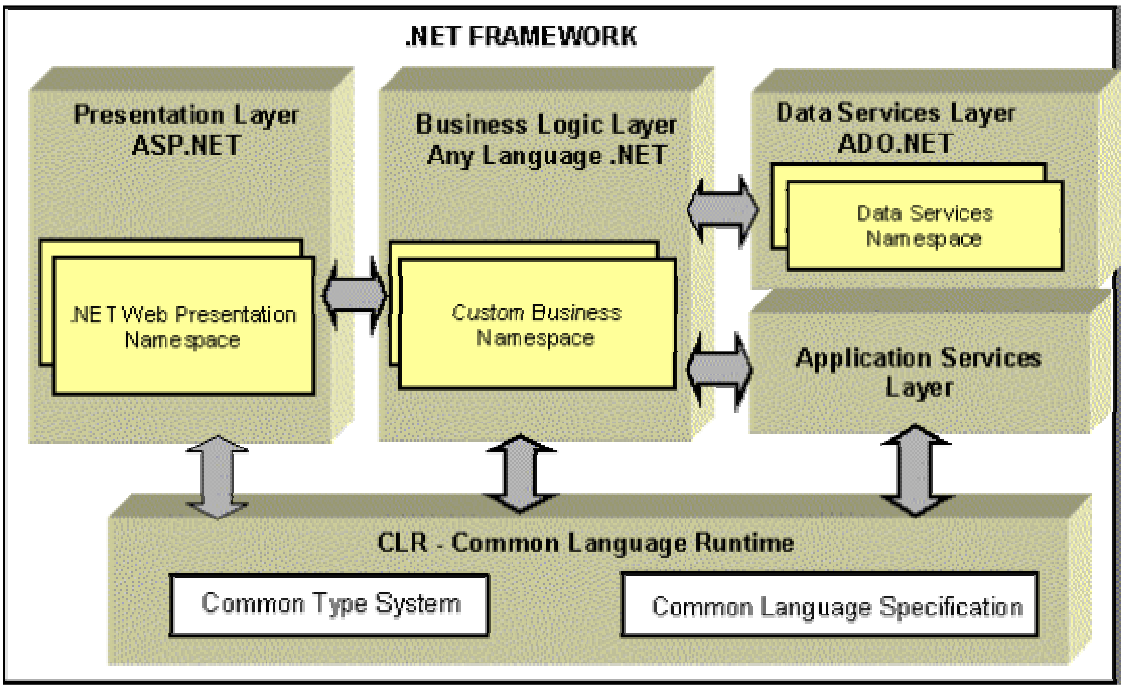
Hicimos uso del objeto "XMLHttpRequest", que permite hacer una conexión HTTP sobre una URL dada sin recargar la página, desde dentro de un JavaScript. Compatible bajo IE

y "Mozilla". Es especialmente útil a la hora de invocar una URL y obtener XML que por ejemplo, sirve para generar combos dinámicos sin recargar toda la página.

Usamos además "Sarissa", que es una librería JavaScript que provee una interfaz común para el manejo del DOM XML tanto en Mozilla como en Explorer. Algunas de las tares cubiertas es la carga de objetos DOM desde fuentes remotas o cadenas, control de transformaciones XSLT, optimizaciones de consultas "Xpath" , etc.

8.3.3.3 Capa de Servicio

Aunque usamos un modelo orientado a objetos del dominio de la aplicación, fue necesario crear una capa intermedia entre la capa de presentación y la lógica de la aplicación a la que se suele denominar capa de servicio o de control. Es la que responde a eventos y solicitudes del usuario y delega inmediatamente a los objetos responsables de cada tarea (Objetos ubicados en la lógica del negocio)



.NET Framework 0-1

Una vez que tuvimos separados el núcleo de la aplicación de sus distintos interfaces, solo nos quedaba por decidir cómo organizar la implementación de la lógica asociada a la aplicación. El diseño tenia que ser sencillo, pero deseábamos que el modelo estuviese bien preparado para soportar las modificaciones que hubiesen de realizarse a futuro.

El diseño de la lógica de la aplicación se ajusta al patrón de diseño "Modelo del Dominio", que consiste en crear un modelo orientado a objetos del dominio de la aplicación, donde cada objeto es responsable de realizar las tareas que lo atañen directamente.

Cabe señalar que la aplicación no esta atada a la Plataforma .NET de Microsoft, se realizaron pruebas en Mono, que es un compilador C# y un "Runtime" para .Net que cumple la especificación de ECMA (Asociación de industrias dedicada a la estandarización de información y sistemas de comunicación).

8.3.3.4 Lógica del Negocio

En la lógica de la aplicación usamos un modelo orientado a objetos del dominio de la aplicación.

8.3.3.5 Acceso a Datos

La capa de acceso a los datos, encargada de gestionar el almacenamiento de los datos y en general de la comunicación del sistema con cualquier otro sistema que realice tareas auxiliares. Usamos el motor de base de datos SQL Server.

Para el acceso a los datos utilizamos ADO.NET que es la nueva arquitectura de acceso de datos propuesta en la plataforma .NET.

8.4 REFACTORIZACIÓN

Un "refactoring" es un cambio realizado a la estructura interna del software sin modificar su comportamiento observable desde el exterior. Los cambios de este tipo pueden ser útiles si hacen que el software sea más fácil de comprender y de modificar cuando hemos de adaptarlo a nuevas necesidades. Refactorizar es reestructurar el

software aplicando una serie de modificaciones sin cambiar su comportamiento. Cuando se realiza esta tarea de forma adecuada, lo que estamos haciendo es mejorar la calidad de nuestros diseños "para lo cual muchas veces echaremos mano de patrones de diseño". En otras palabras, aunque no estemos añadiéndole funciones al programa, estamos facilitando su desarrollo futuro.

El "refactoring" no es una actividad más a la que se le reserva un espacio en la planificación de un proyecto, sino que es algo que se va haciendo sobre la marcha conforme hace falta.

En el proceso de la optimización de la aplicación la refactorización cumplió un papel importante al permitirnos eliminar lógica duplicada como resultado del desarrollo colaborativo, también fue muy útil cuando se añadió funcionalidad a la aplicación, pues sirvió como paso inicial para que el código resultara más fácil de entender facilitando con esto el añadir la funcionalidad deseada.

8.5 PRUEBAS DE UNIDAD

Cuando implementamos el software, comprobamos que el código que escribimos funciona correctamente. Para ello implementamos "tests" que verifican que nuestro programa genera los resultados esperados. Conforme íbamos añadiéndole nuevas funcionalidades, creábamos nuevos "tests" con los que pudimos medir nuestros progresos y comprobar que lo que antes funcionaba sigue funcionando "test de regresión".

Las pruebas de unidad también son muy importantes en el "refactoring". Aunque en ese proceso no se añade nueva funcionalidad, estamos modificando la estructura interna de nuestro programa y debemos comprobar que no hemos introducido errores.

Señalamos en este punto que las pruebas de Unidad debido al esfuerzo y tiempo extra que requieren solo elaboramos " test" a los puntos críticos del sistema.

Para la automatización de las pruebas usamos una herramienta especialmente diseñada para realizar este tipo de pruebas de unidad llamada NUnit.

Los Tests NUnit son test de "calidad de las clases", es decir, con un test NUnit lo que hacemos es comprobar si una clase funciona exactamente como debería.

La importancia de estos tests radica en que es la forma más completa y ordenada de comprobar que dichas clases están libres de fallos.

8.6 PRUEBAS

En esta etapa llevamos al sistema al límite de situaciones en las cuales se creía que el software podría fallar. El aplicativo fue probado en dos entornos de ejecución.

El primero en un computador con el sistema operativo Windows 2000 Server, "Internet Information Server IIS" , el compilador y el "runtime" provisto por VS.NET.

El segundo entorno fue suministrado por un computador con el sistema operativo Linux Debían "Woody" y el compilador y el "runtime" provisto por el Proyecto Mono.

Entre las pruebas realizadas se encuentra:

- *Intentos de logeo de la aplicación de personal no autorizado o no registrados.*
- *El acceso recurrente a la base de datos.*
- *Tiempos de inactividad de un usuario dentro de la aplicación.*
- *Prueba de veracidad de los reportes.*
- *Prueba de consistencia de la información almacenada en la base de datos.*

En el anexo C se encuentra el formato utilizado para la documentación de las pruebas realizadas.

8.7 IMPLEMENTACION Y ENTREGA FINAL DE SOFTWARE

En esta etapa se realizo el entrenamiento requerido al personal que manejaría la aplicación, además se prosiguió con la entrega y presentación formal del sistema a la dependencia de Recursos Humanos para que verificaran la utilidad y la efectividad del mismo.

El desarrollo del Sistema de Información para el Manejo de talento Humano requirió de recursos tecnológicos, humanos y físicos (Papeleria, Tinta, fotocopias etc.), Los cuales acarearon un costo. En el anexo B se encuentra el presupuesto utilizado en el proyecto.

9 CONCLUSIONES

Por medio del diseño e implementación del Sistema Electrónico de Talento Humano en C.I. Técnicas Baltime de Colombia S.A. se hizo un aporte importante a la compañía, en cuanto a la automatización de los procesos y procedimientos llevados a cabo en la dependencia de Recursos Humanos. Con esto la gerente de Recursos Humanos ha podido gestionar proyectos de vivienda a los empleados, coordinar cronogramas de capacitaciones, controlar ausentismos etc.

El diseño e implementación del Sistema Electrónico de Talento Humano a C.I. Técnicas Baltime de Colombia S.A., el cual por su características de fácil acceso, confiabilidad, seguridad y de consulta personalizada, garantizo la obtención de resultado favorables para la compañía puesto que ellos significo el ahorro de recursos humanos y materiales, como también la sastifacción de los usuarios.

10 ANALISIS DE RIESGO

10.1 RIESGOS INTERNOS

RIESGO	PROBABILIDAD	IMPACTO	MEDIDA
Conflictos en la asignación de responsabilidades entre el grupo de trabajo.	Alta	Alto	Comprometer al grupo de trabajo en el desarrollo del proyecto.
Problemas de comunicación en el equipo, debido a que este labora en distintas sedes.	Alta	Medio	Aprovechar las tecnologías disponibles (correo, chat, etc.) para establecer canales de comunicación eficientes.

10.2 RIESGOS INTRINSECOS

RIESGO	PROBABILIDAD	IMPACTO	MEDIDA
Falta de precisión o veracidad en la información obtenida en el proceso de recolección de información.	Media	Alto	Confrontar las entrevistas con la observación de los procesos que se llevan a cabo.

11 GLOSARIO

Cliente-Servidor. *Modelo de interacción en un sistema distribuido en el cual un programa en un sitio envía una petición a un programa en otro sitio y espera una respuesta. Al programa solicitante se le llama Cliente y al programa que satisface la petición se le llama Servidor.*

Internet. *Es una unión enorme de computadoras conectadas a través de redes por todo el mundo. Desarrollado en los 70's por el ARPANET (Deparment of Defense's Advanced Research Projects Agency)*

Interoperabilidad. *La capacidad de componentes o sistemas electrónicos producidos por diferentes fabricantes, para comunicarse directamente y satisfactoriamente entre ellos y/o sus usuarios. La tendencia hacia adoptar estándares, ha impulsado en gran medida el proceso de interoperabilidad.*

JavaScript. *Es un lenguaje de programación utilizado para crear pequeños programitas encargados de realizar acciones dentro del ámbito de una página Web.*

SOAP *(Simple Access Protocol o SOAP), Protocolo de Acceso a Objetos Simple. (Simple Access Protocol o SOAP), que se basa en la utilización del protocolo HTTP para el transporte de mensajes y el lenguaje XML para la escritura del cuerpo de estos mensajes, y que ha sido enviado al W3C por Microsoft en mayo de 2000 para su estandarización.*

Xpath: *XPath es un lenguaje para direccionar partes de un documento XML, diseñado para ser utilizado tanto por "XSL" como por "XPointer".*

12 BIBLIOGRAFIA

Kendall, Kenneth E. y Kendall, Julie E. Análisis y diseño de sistemas (Tercera Edición). Pearson Education 1997.

Jose Manuel, Huidoro Moya. Redes y servicios de telecomunicaciones (Tercera Edición). Paraninfo 2001

Abraham Siberschatz, Henry F. Korth. Fundamentos de base de datos (Tercera Edición) McGrawHill 1998

Long, Larry. Introducción a la informática y al procesamiento de información. Prentice-Hall Hispanoamericana S.A. 1986 p. 4, 310

POWELL Tomas, SCHNEIDER Fritz, JavaScript: Manual de Referencia, Editorial McGraw-Hill, Primera Edición, p. 603.

COMUNICACIONES WORLD, La revista de los profesionales de redes y telecomunicaciones, < URL: www.idg.es >

DEVELOPER CONNECTION, <URL: http://developer.apple.com/internet/webcontent/>

SITEPOINT, XMLHttpRequest, <URL:http://www.sitepoint.com/blog-post-view.php?id=185942 >

Proyecto MONO, http://www.mono-project.com/about/index.htmlMicrosoft .NET

http://www.programacion.com/tutorial/csharp/

http://msdn.microsoft.com/net

http://www.csharphelp.com/

www.microsoft.com/mspress/spain/books/book17507.htm

http://www.openresources.com/es/magazine/xml-tutorial/

http://msdn.microsoft.com/library/spa/

http://www.monohispano.org/tutoriales/csharp/

http://www.gotdotnet.com/community/resources/home.aspx

http://www.ecma-international.org/

http://www.3devnet.com/download/punit.ppt

http://www-106.ibm.com/developerworks/library/wa-resc/?dwzone=web

http://www.programacion.com/articulo/remote_scripting/

http://elvex.ugr.es/decsai/csharp/design/layers.xml

http://sarissa.sourceforge.net/

http://www.nunit.org/getStarted.html

http://elvex.ugr.es/decsai/csharp/

ANEXOS

Anexo A

CRONOGRAMA DE ACTIVIDADES

Anexo B

PRESUPUESTO

Anexo C

FORMATO DE PRUEBAS DE LOS SISTEMAS

1. MATERIALES					
Ítem	*Descripción*	*Cantidad*	*Unidad*	*Vr Unitario*	*Valor Total*
1.1	*PAPEL*	*2*	*RESMAS*	*$7,800*	*$15,600*
1.2	*TRANSPORTE OTROS*	*8*	*Unidad*	*$80,000*	*$640,000*
1.3	*FOTOCOPIAS*	*500*	*Unidad*	*$80*	*$40,000*
1.4	*TINTA IMPRESORA*	*1*	*Unidad*	*$77,000*	*$77,000*
1.5	*MATERIALES VARIOS**		*Unidad*		*$20,000*
Subtotal					*$792,600*
2. HERRAMIENTAS					
Ítem	*Descripción*	*Cantidad*	*Unidad*	*Vr Unitario*	*Valor Total*
2.1	*COMPUTADORES*	*2*	*Unidad*	*1,000,000*	*$2,000,000*
2.2	*INTERNET*	*500*	*HORAS*	*$2,500*	*$1,250,000*
Subtotal					*$3.250.000*
3. MANO DE OBRA					
Ítem	*Descripción*	*Cantidad*	*Unidad*	*Vr Unitario*	*Valor Total*
3.1	*DESARROLLADORES*	*180*	*DIAS*	*$66,600*	*$12,000,000*
Subtotal					*$12,000,000*

4. LICENCIAS					
Ítem	*Descripción*	*Cantidad*	*Unidad*	*Vr Unitario*	*Valor Total*
4.1	*SQL SERVER*	*1*	*Unidad*	*$1,500,000*	*$1,500,000*
4.2	*VISUAL STUDIO .NET*	*1*	*Unidad*	*$3,500,000*	*$3,500,000*
Subtotal					*$5,000,000*
Total valor del Desarrollo: $21.042.600					

Tiempo / Actividad	MES 1				MES 2				MES 3				MES 4				MES 5			
	SEM 1	SEM 2	SEM 3	SEM 4	SEM 1	SEM 2	SEM 3	SEM 4	SEM 1	SEM 2	SEM 3	SEM 4	SEM 1	SEM 2	SEM 3	SEM 4	SEM 1	SEM 2	SEM 3	SEM 4
Recolección de Información																				
Análisis de la Información																				
Diseño del Entorno																				
Pruebas del Diseño																				
Corrección de Errores (Mejoramiento Incremental)																				
Entrega Formal																				

Numero de Control <Numero de control>	Referencia del Proceso de Pruebas: < ID Proceso de Pruebas>		Tipo de Variación: <Describa la variación de la prueba>
Responsable: <Responsable>	Fecha de Ejecución: <fecha de Ejecucuion de la prueba>	Coordinado de la prueba: <Nombre del Coordinador de la Prueba>	Grupo de Prueba: <grupo de Pruebas>

Pasos	Accion	Menu	Uso especifico de los datos	Resultados esperados	Resultado Actual	Ciclo de tiempo esperado	Ciclo de Tiempo Actual

MANUAL TÉCNICO

SISTEMA DE INFORMACIÓN PARA EL MANEJO DEL TALENTO HUMANO EN C.I. TÉCNICAS BALTIME DE COLOMBIA (SETHUM)

Autores:	Jaime Abella Wilman Rincón
Fecha Creación:	19/06//2005
Ultima Actualización:	18/07/2005
Versión:	1.00

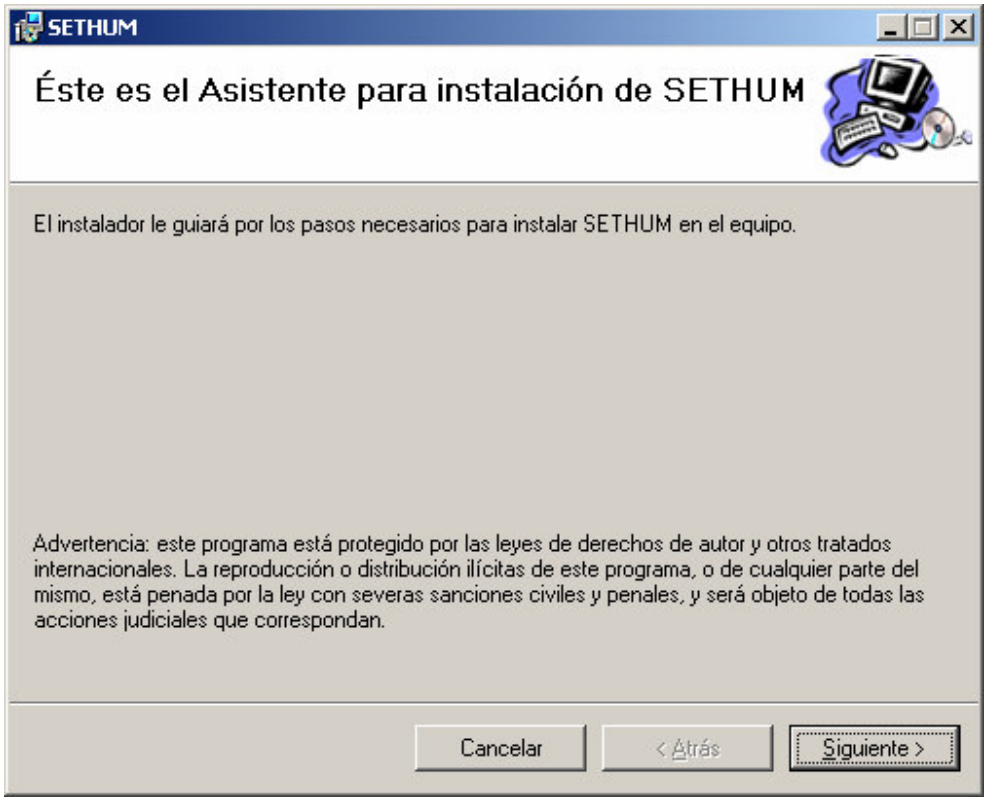
TABLA DE CONTENIDO

1. INSTALACION DE SETHUM

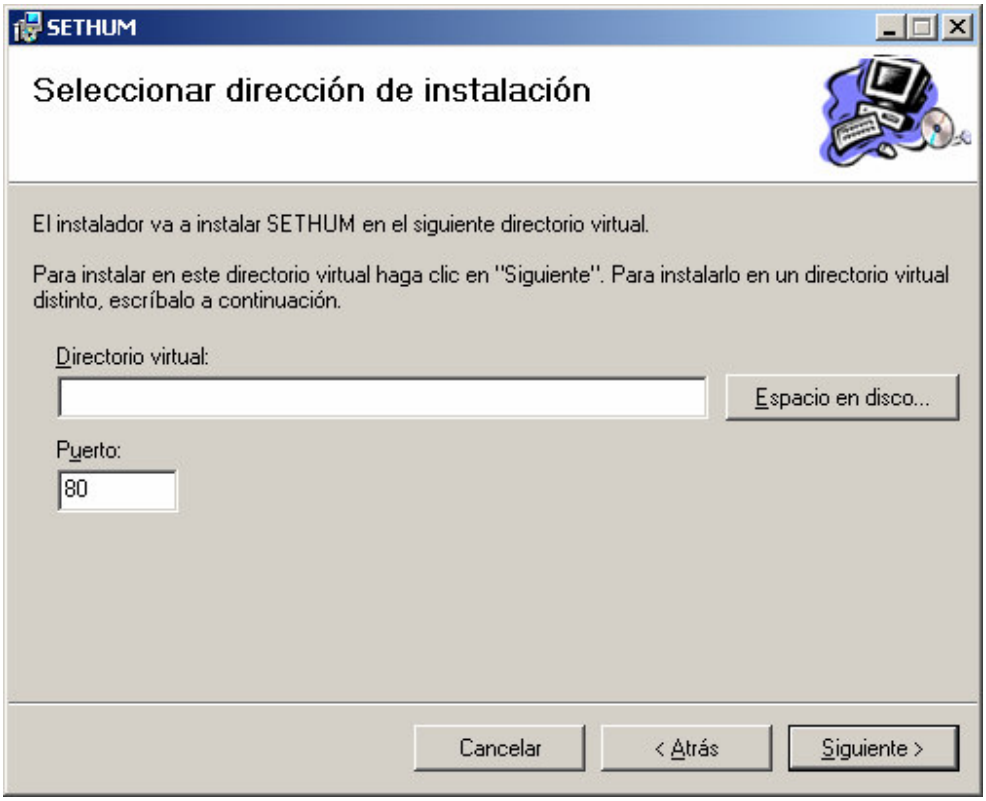
Los lenguajes que utilizamos comúnmente para desarrollar páginas Web (Asp, Php, Jsp) requieren para la instalación de sus sitios Web copiar los archivos en la carpeta raíz o en un directorio virtual del servidor.

.NET utiliza para la instalación de sus sitios Web, un paquete instalador como si se tratara de una aplicación de escritorio. Este paquete consta de un archivo (setup) que se copia en el servidor y se ejecuta ahí mismo, con lo cual aparece el asistente. Los pasos a seguir son los siguientes:

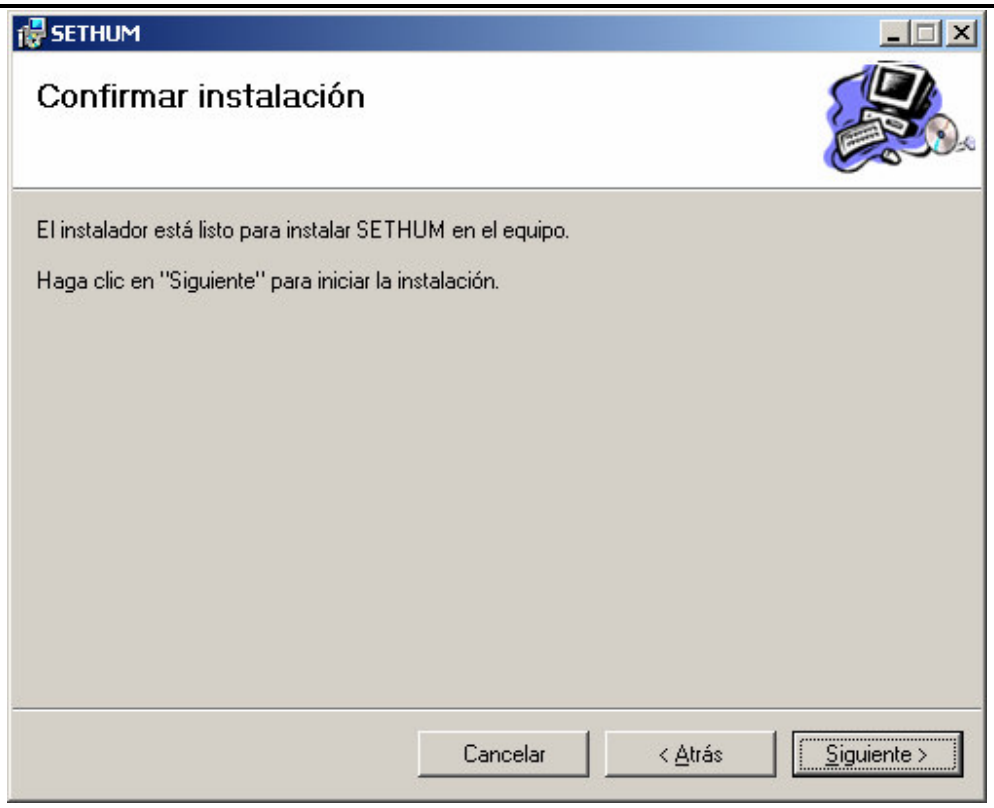
1. Este es el asistente que aparece luego de haber ejecuta el instalador, esta primera pantalla es informativa es una presentación del software que se va a instalar. Para avanzar a la siguiente pantalla damos click en el botón siguiente.



2. En esta pantalla debemos especificar un directoria virtual para la aplicación y el puerto de conexión con que trabajara la aplicación. Por defecto si se deja el campo directorio virtual en blanco la aplicación se instala en el directorio raíz y se comunica con el puerto 80.



3. En esta pantalla se confirma la instalación de la aplicación, si se desea instalar la aplicación se da click en el botón siguiente, en caso contrario que se desee cancelar la instalación de la aplicación se da clic en el botón cancelar, con lo cual provocara que el asistente se cierre.



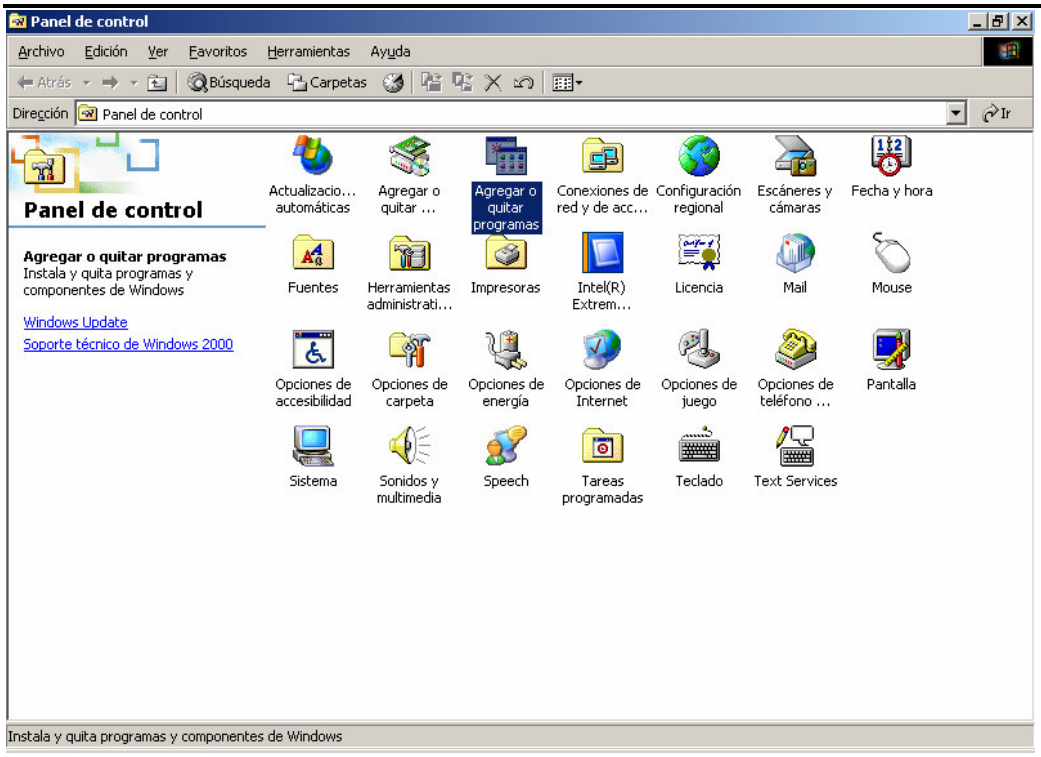
2. CONFIGURACION DE LA APLICACIÓN

La aplicación SETHUM funciona en un modelo cliente-servidor, el cual su objetivo es el intercambio de información entre los clientes Web y los servidores HTTP.

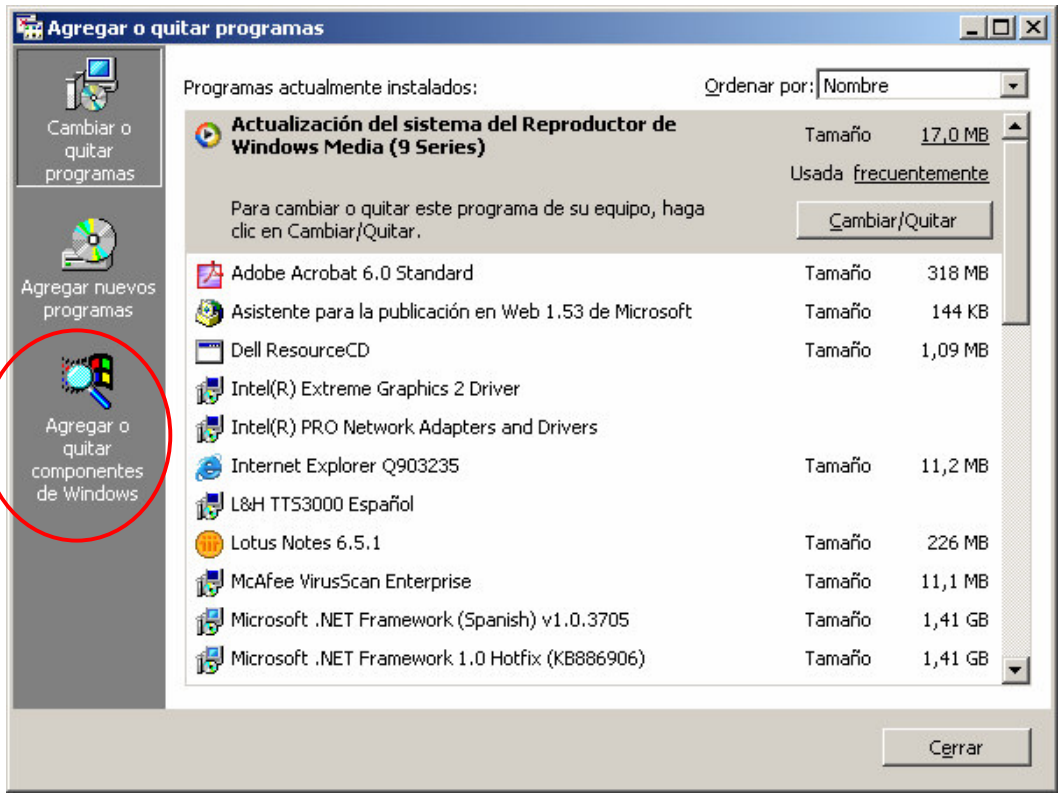
En nuestro caso necesitamos que el servidor tenga configurado el servicio Internet Information Server (IIS). Si hemos instalado un sistema operativo como el Windows 2000 Server o el Windows NT no es necesario configurar este servicio ya que estos lo configuran por defecto.

Si este no es el caso necesitamos configurar en el servidor el servicio de IIS. Para esto seguimos los siguientes pasos:

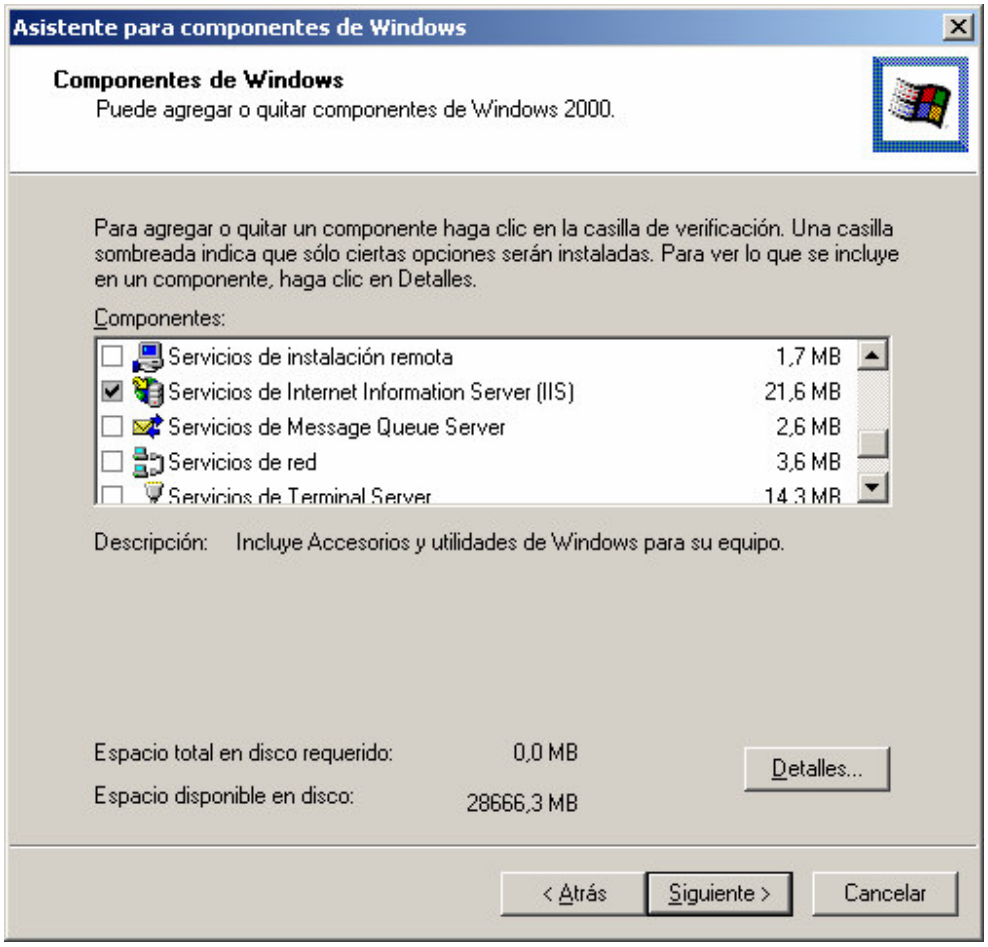
1. Abrimos el Panel de Control y seleccionamos la opción Agregar o Quitar Programas.



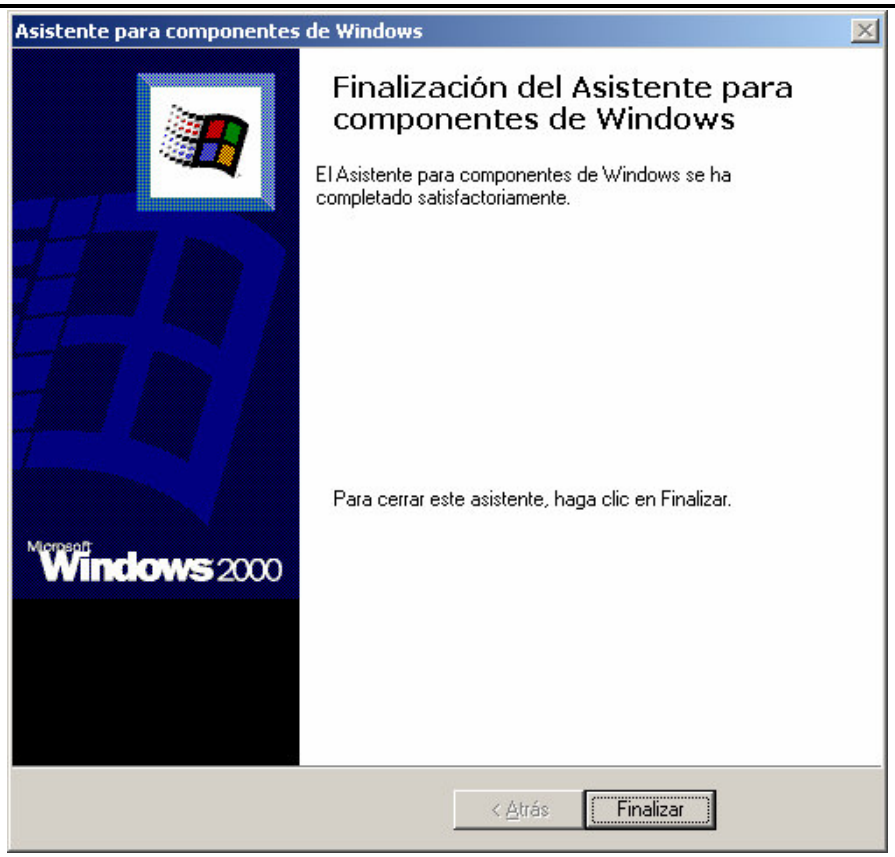
2. En la pantalla que nos aparece escojamos la opción que dice Agregar o Quitar Componentes de Windows



3. Con esto nos aparece una pantalla en la cual podemos verificar los servicios que tiene instalado el servidor, en nuestro caso señalamos el que dice Servicios de Internet Information Server (IIS) y damos click en el botón siguiente.



4. Con esto hemos finalizado la instalación del servicio IIS, con lo cual ahora contamos con un servidor Web o un servidor HTTP.



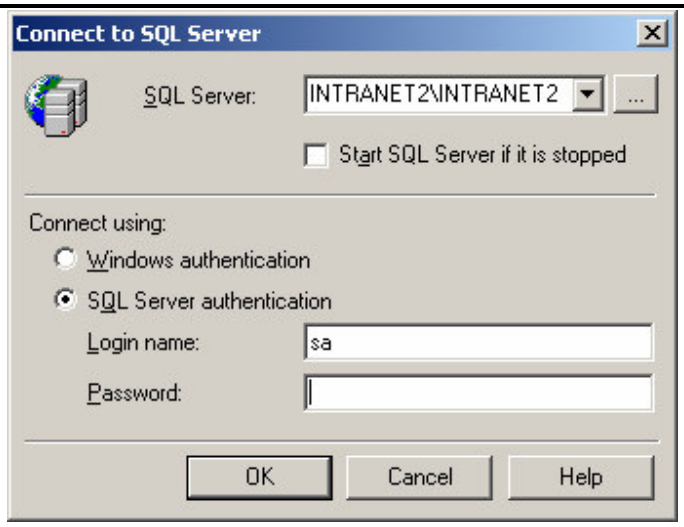
3. SEGURIDAD

3.1. NIVELES DE SEGURIDAD

La aplicación SETHUM cuenta con diferentes niveles de seguridad, en el servidor, en la base de datos y en la aplicación como tal.

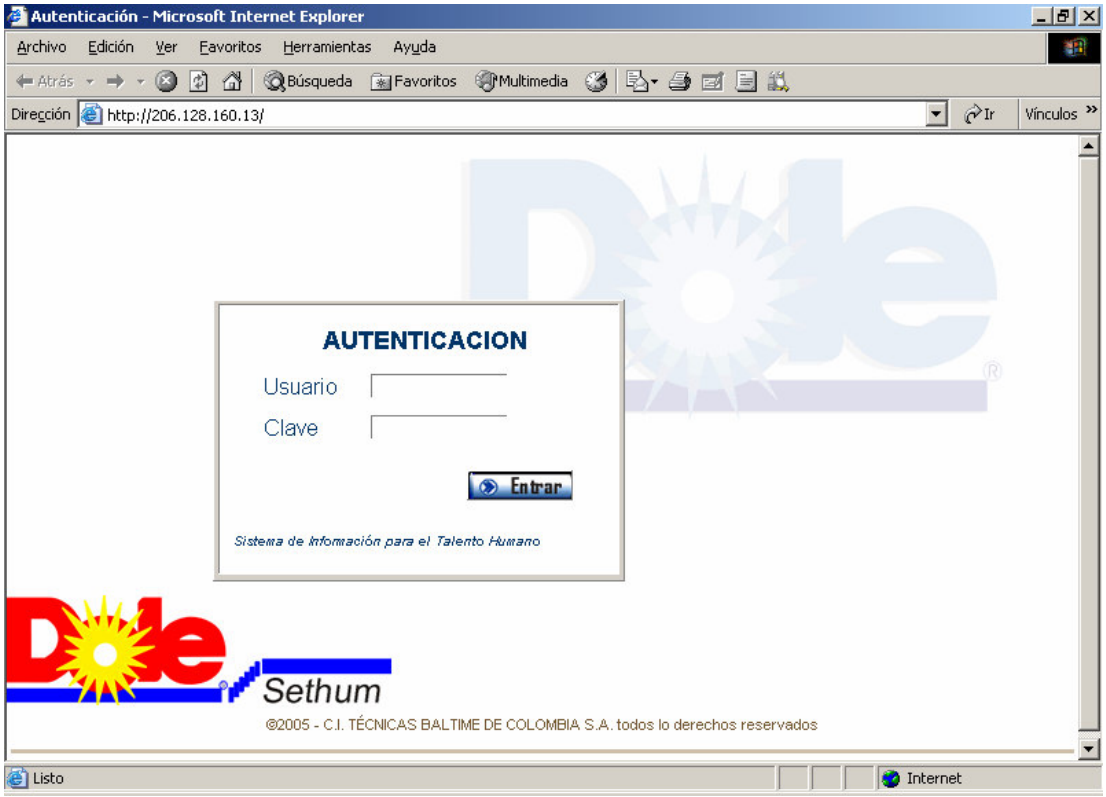
Servidor: La seguridad en el servidor esta a cargo del Sistema Operativo, este trabaja con Windows 2000 Server. En el se encuentra registrados los usuarios que tienen acceso al sistema, así a la hora de entrar al servidor es necesario realizar una autenticación (usuario y password), la cual nos podrá dar acceso al sistema si estamos registrados o en el casos contrario truncarnos el acceso.

Base de Datos: En la base de datos se encuentra registrados (nombre de usuario y password) los usuarios que tienen acceso a la base de datos. Esta autenticación es realizada por el motor de la base de datos, en nuestro caso SQL Server. Cuando tratamos de ingresar a la base de datos a realizar mantenimiento, backup etc.. es necesario autenticarnos, como lo muestra la siguiente figura.

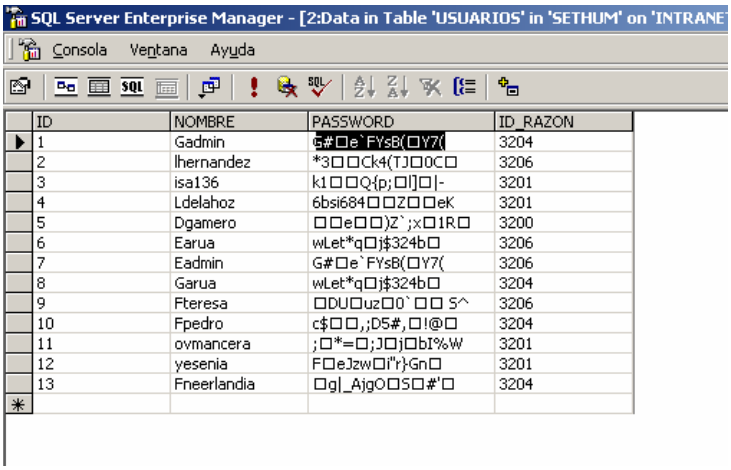


Aplicación: La aplicación cuenta también con niveles de seguridad, los cuales se implementan para garantizar la consistencia de la información. Estos son:

- Autenticación: La aplicación implementa una autenticación mediante un formulario, que consta del registro de un nombre de usuario y una contraseña. Estos usuarios deben estar registrados en la base de datos.



- Encriptación de Claves: La aplicación encripta las claves de los usuarios en la base de datos. Así se protege el password de los usuarios a la vista de cualquier persona que tenga acceso a la base de datos.



ID	NOMBRE	PASSWORD	ID_RAZON
1	Gadmin	G#□e`FYsB(□Y7(	3204
2	lhernandez	*3□□Ck4(TJ□0C□	3206
3	isa136	k1□□Q{p;□l]□|-	3201
4	Ldelahoz	6bsi684□□Z□□eK	3201
5	Dgamero	□□e□□)Z`;x□1R□	3200
6	Earua	wLet*q□j$324b□	3206
7	Eadmin	G#□e`FYsB(□Y7(	3206
8	Garua	wLet*q□j$324b□	3204
9	Fteresa	□DU□uz□0`□□ S^	3206
10	Fpedro	c$□□,;D5#,□!@□	3204
11	ovmancera	;□*=□;J□j□bI%W	3201
12	yesenia	F□eJzw□i"r}Gn□	3201
13	Fneerlandia	□g|_AjgO□S□#'□	3204

- Roles: La aplicación maneja 4 tipos de roles los cuales le dan permiso a los usuarios. Cabe indicar que dependiendo del rol del usuario se carga el menú principal de la aplicación. Los roles que maneja la aplicación SETHUM son Ejecutivo, Secretaria, Supervisor, Administrador.

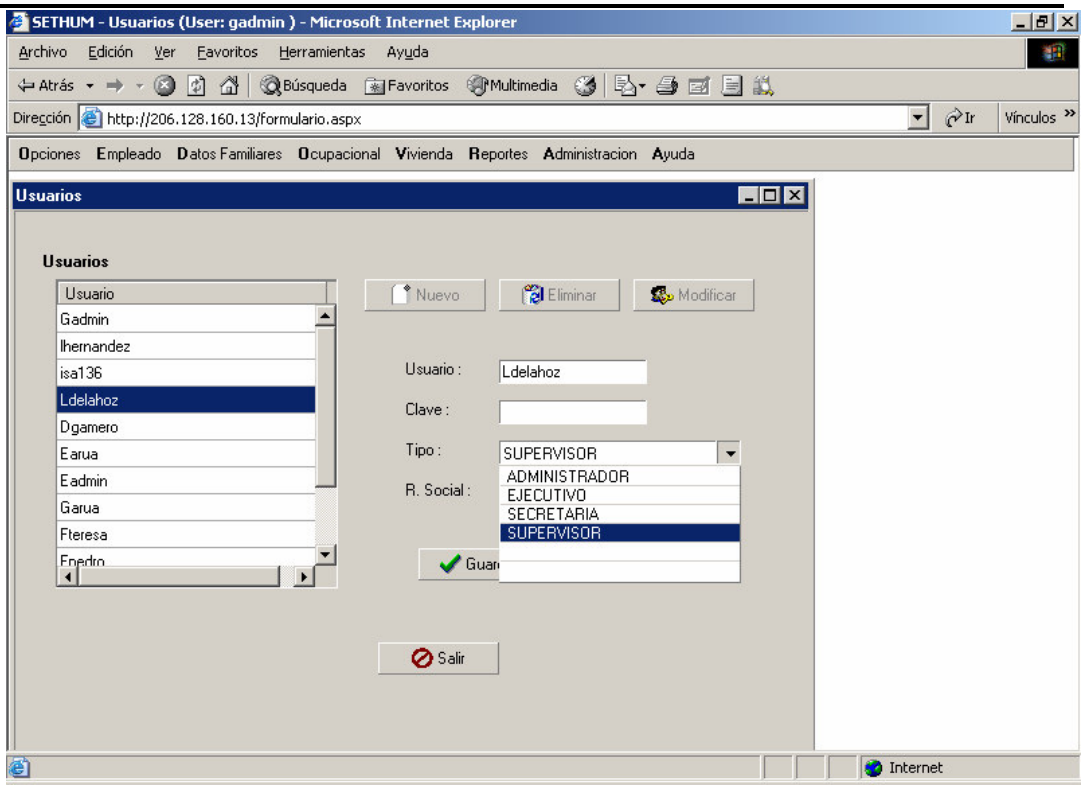
  Los usuarios tipo ejecutivo solo pueden leer información de los formularios, no pueden guardar nuevos registros, ni modificar los que ya se encuentran en la base de datos.

  Los usuarios tipo secretaria pueden ingresar información y modificar la que ya se encuentra, pero no tienen acceso al menú de Reportes.

  Los usuarios tipo Supervisor pueden ingresar y modificar la información, también tienen acceso al menú de Reportes pero no tienen acceso al menú de Administración.

  Los usuarios tipo Administrador tienen todos los permisos, hasta pueden eliminar información almacenada en la base de datos.

  La creación de los usuarios solo es accesible por el administrador de la aplicación, cuando se crea el usuario se le asigna el permiso que va a tener en la aplicación. En la siguiente figura se muestra como es la creación de los usuarios.

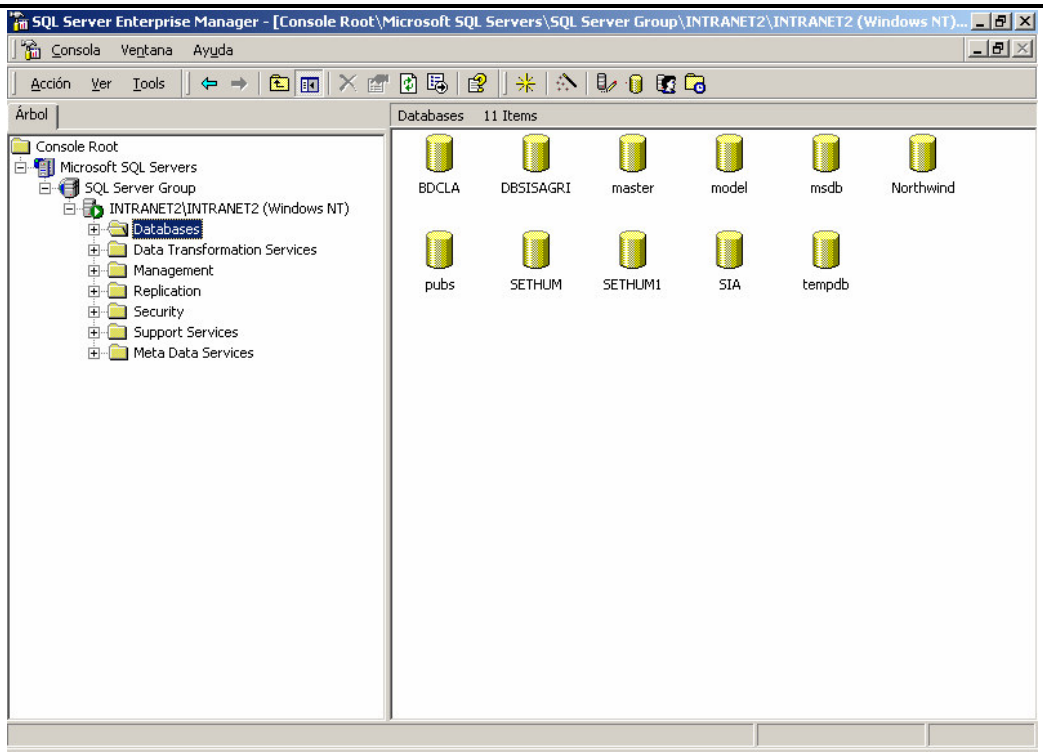


3.2. SEGURIDAD DE LOS DATOS

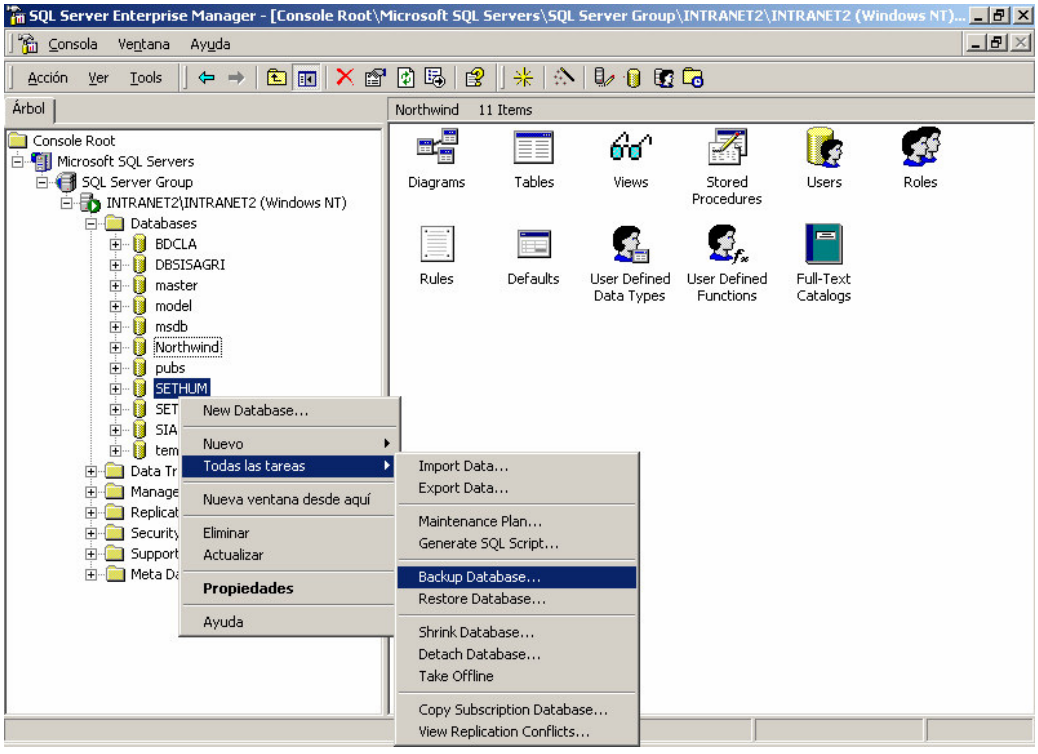
3.2.1. Backup de la Base de Datos

Para resguardar la información que se encuentra en la base de datos y evitar la perdida de esta ante cualquier percance, se le realiza cada 2 días una copia de seguridad. Esta copia es guardada en un servidor con la fecha de creación, ya que semanalmente se guarda en una cinta de backup todas las copias que se encuentren. Para realizar la copia de seguridad se realiza los siguientes pasos:

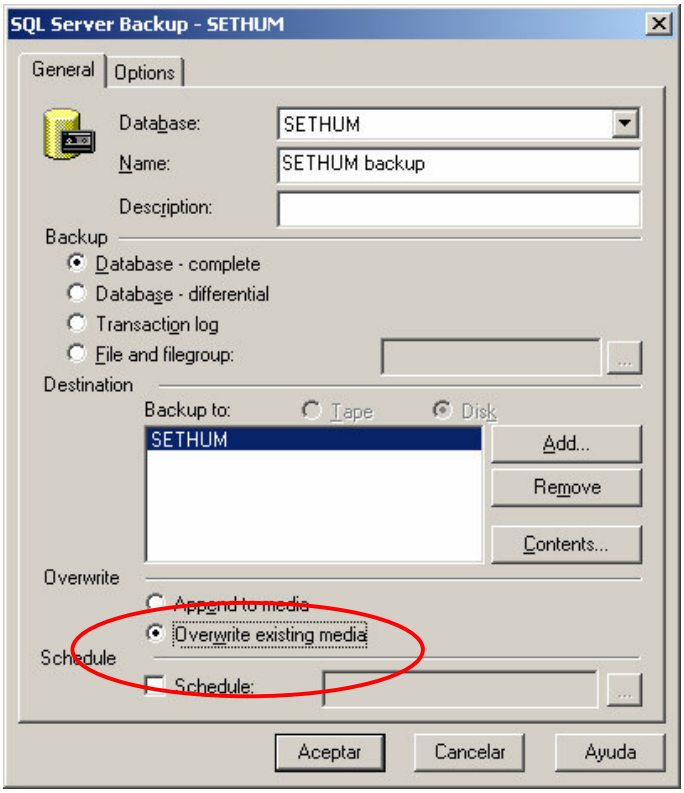
1. Se debe ejecutar el SQL Server Enterprise Manager, la cual es una consola de administración provista por el motor de datos, como vemos en la siguiente figura.



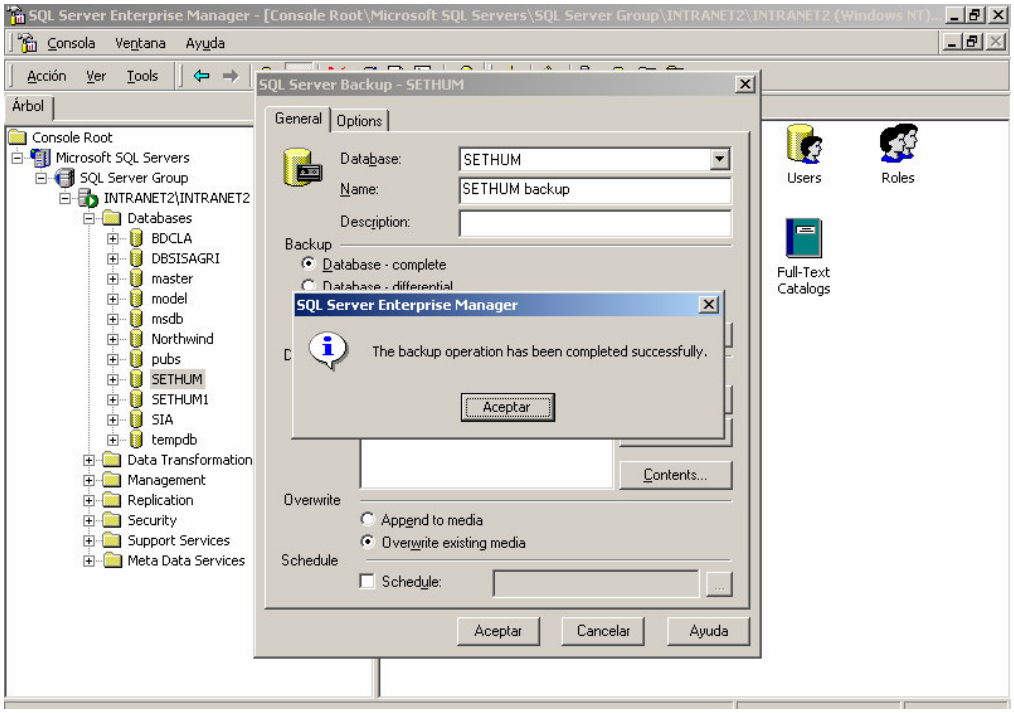
2. Se da un click derecho sobre la base de datos que se desea hacer la copia de seguridad. Del menú contextual que aparece se escoge la opción Todas las tareas y luego la opción Backup Database.



3. Con esto aparece la siguiente pantalla, aquí escogemos la opción Overwrite existing media y damos click en el botón Aceptar.



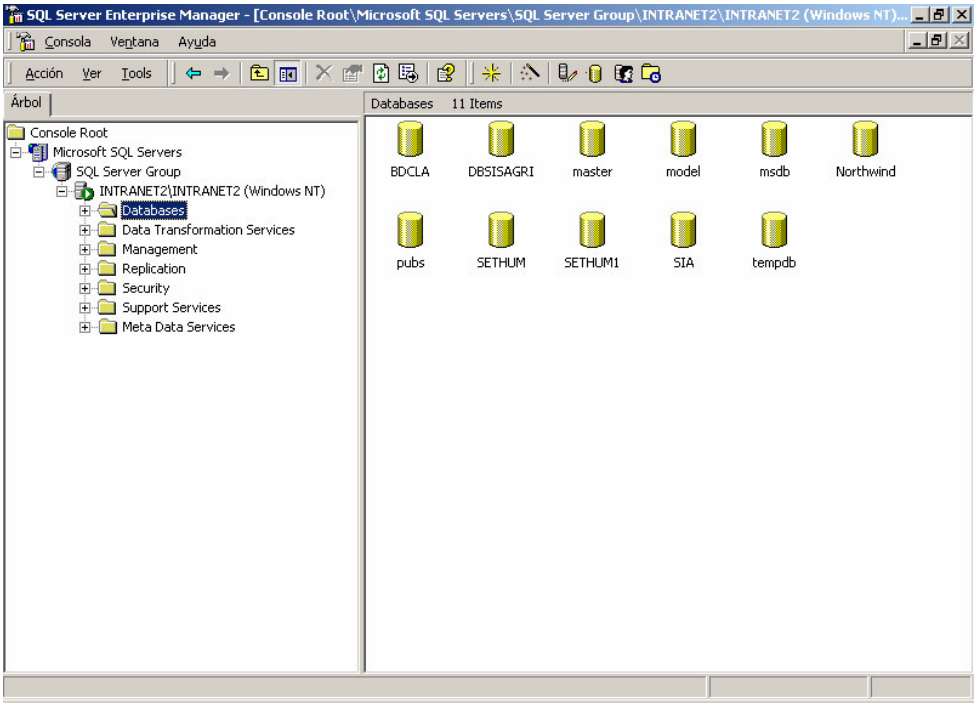
4. La siguiente pantalla nos muestra que la copia de seguridad se ha realizado con éxito.



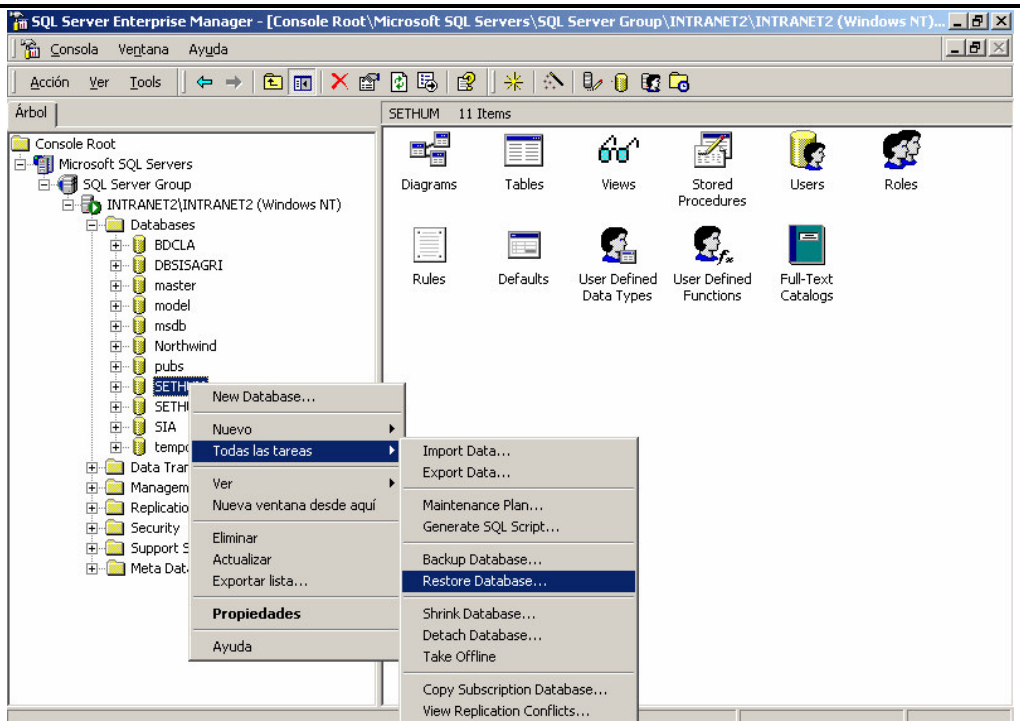
3.2.2. Restore de la Base de Datos

Cuando necesitemos restaurar la base de datos con una copia de seguridad, debemos seguir los siguientes pasos:

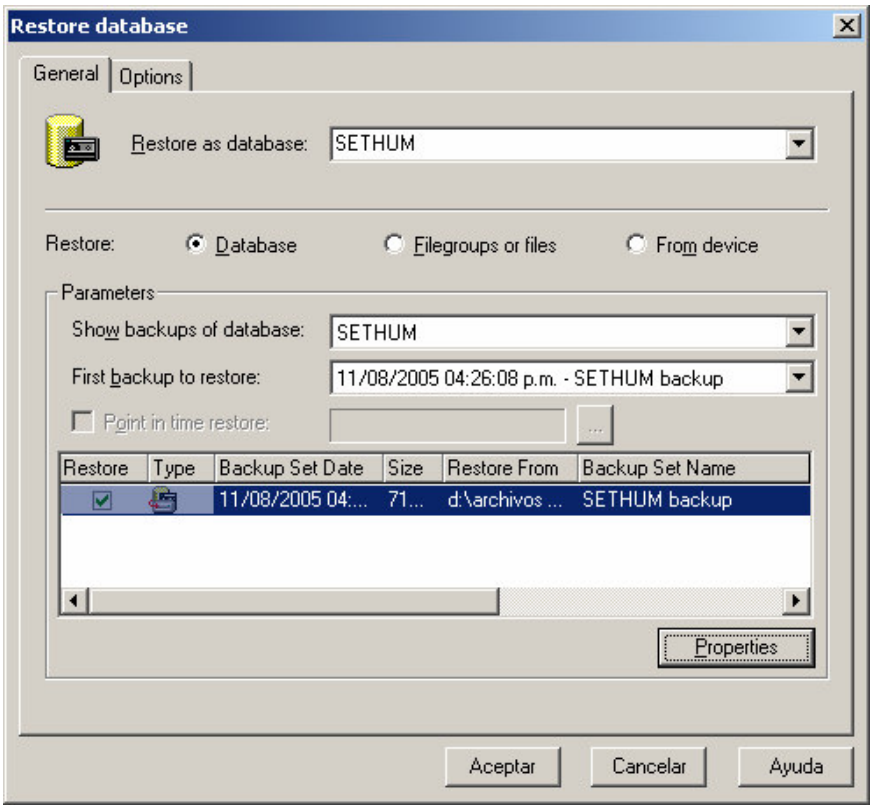
1. Se debe ejecutar el SQL Server Enterprise Manager, la cual es una consola de administración provista por el motor de datos, como vemos en la siguiente figura



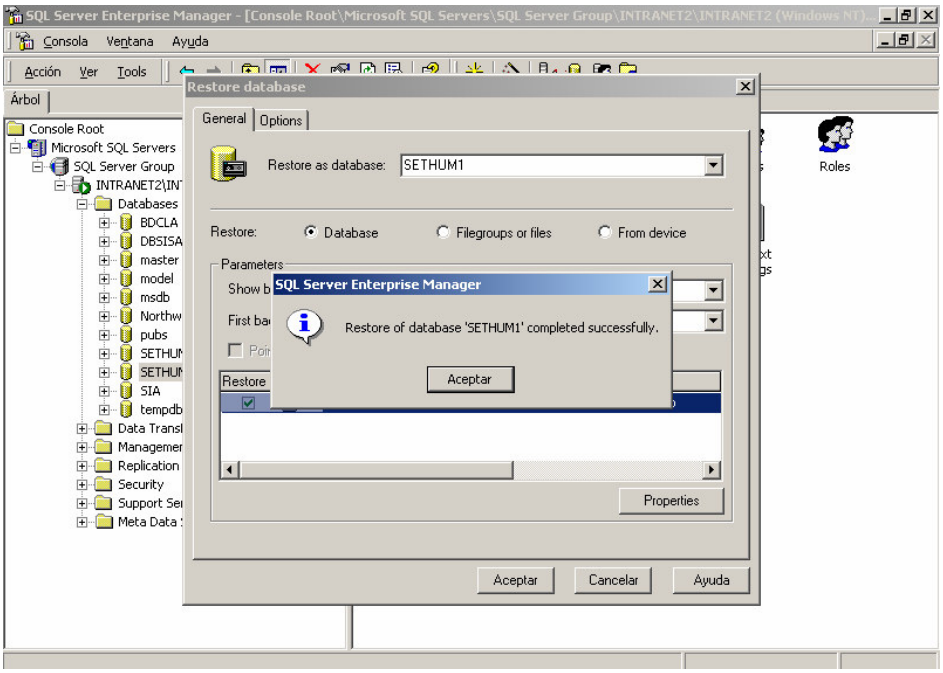
2. Se da un click derecho sobre la base de datos que se desea hacer la copia de seguridad. Del menú contextual que aparece se escoge la opción Todas las tareas y luego la opción Restore Database



3. Con esto aparece la siguiente pantalla seleccionamos por defecto la ubicación donde queda el backup y hacemos click en el botón aceptar.



4. La siguiente pantalla nos muestra que la restauración de la base de datos se realizo correctamente.



.

4. DICCIONARIO DE DATOS

PERSONA		
Tabla que contiene los datos personales de un empleado o un familiar del empleado.		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Campo autonumérico que se crea cuando se ingresa los datos	
CIUDAD_NACIMIENTO	Id de la ciudad donde nacio la persona	
CIUDAD_RESIDENCIA	Id de la ciudad donde vive la persona	
APELLIDO1	Primer apellido de la persona	
APELLIDO2	Segundo apellido de la persona	
NOMBRES	Nombres de la persona	
SEXO	Genero de la persona	
FECHA_NACIMIENTO	Fecha en la que nacio la pesona	
DIRECCION	Direccion donde reside la persona	
TELEFONO	Telefono donde se pueda ubicar a la persona	X
E-MAIL	Correo electronico perteneciente a la persona	X
TALLA_CAMISA	Talla de la camisa dela persona	X
TALLA_PANTALON	Talla de pantalón de la persona	X
TALLA_ZAPATO	Talla de zapato de la persona	X

ESTADO_CIVIL		
Tabla que contiene los diferentes estado civil.		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id del estado civil	
NOMBRE	Nombre del estado civil	

PARENTESCO		
Tabla que contiene los diferentes parentescos familiares		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id del parentesco	
NOMBRE	Nombre del parentesco	

DOCUMENTO		
Tabla que contiene los diferentes tipos de documentos que puede tener una persona.		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id del documento	
TIPO _DOCUMENTO	Nombre del documento	

COMPOSICION_FAMILIAR		
Tabla que relaciona a un empleado con las personas que conforman su familia.		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID_PERSONA	Id del empleado	
ID_FAMILIA	Id de la persona familiar	
ID_PARENTESCO	Id del tipo de relacion familiar	
OCUPACION	A que se dedica el familiar	
CONVIVEN	Si el empleado convive con el familiar	

PAIS		
Tabla que contiene los paises		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id del pais	
NOMBRE	Nombre del pais	

DEPARTAMENTO		
Tabla que contiene los departamentos		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id del departamento	
NOMBRE	Nombre del departamento	
ID_PAIS	Id del país al que pertenece el departamento	

CIUDAD		
Tabla que contiene todas las ciudades		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id de la ciudad	
NOMBRE	Nombre de la ciudad	
ID_DEPARTAMENTO	Id del departamento a que pertenece la ciudad	

RAZON_SOCIAL		
Tabla que contiene las razones social o el nombre de empresas		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id de la razon social	
NOMBRE	Nombre de la razon social	

CENTRO_COSTO		
Tabla que contiene los centro de costos		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id de la razon social	
NOMBRE	Nombre de la razon social	

VIVIENDA		
Tabla que contiene la información de vivienda del empleado		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Campo autonumerico que se incrementa cuando se crean los datos	
TENENCIA	Tipo de tenencia de la vivienda del empleado	
ADQUISICION	Como adquiero la vivienda	
ID_PERSONA	Id del empleado relacionado a la vivienda	
DOCUMENTO	Tipo de documento que tiene de la vivienda	

SERVICIOS		
Tabla que contiene los diferentes servicios domiciliarios con que cuenta una vivienda		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id del servicio	
NOMBRE	Nombre del servicio	

EMPLEADO		
Tabla que contiene la información del empleado relacionada con la empresa		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Campo autonumerico que se incrementa cuando se crean los datos	
ID_OFICIO	Id del oficio que realiza el empleado	
FECHA_INGRESO	Fecha en la que ingreso a la empresa	
FECHA_VENCIMIENTO	Fecha en la que termina su relacion laboral	X
ESTADO	Estado en el que encuentra el empleado (activo,inactivo,desempleado)	
ID_PERSONA	Id de la tabla persona que guarda datos personales	

OFICIO		
Tabla que guarda los cargos que puede realizar un empleado		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id del cargo	
NOMBRE	Nombre del cargo	

CAPACITACION		
Tabla que contiene los datos de la capacitación que realiza la empresa		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Campo autonumérico que se incrementa cuando se crean los datos.	
LUGAR	Lugar donde ser realizo la capacitación	
CONFERENCISTA	Persona que realizo la capacitación	
FECHA	Fecha en que se realizo la capacitación	
HORA_INCIAL	Hora que comenzó la capacitación	
HORA_FINAL	Hora que finalizo la capacitación	
ID_PROCEDIMIENTO	Id del procedimiento	
SISTEMA	Sistema a la que pertenece la capacitación	
SEMANA	Semana que se realizo la capacitación	
PERIODO	Periodo en que se realizo la capacitación	
ANO	Ano en que se realizo la capacitación	
HORAS	Total de horas que duro la capacitación	
MINUTOS	Minutos que duro la capacitación	
ID_RAZON	Id de la razon social que realizo la capacitación	

PROCEDIMIENTO		
Tabla que contiene los procedimientos iso		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id del procedimiento	
NOMBRE	Nombre del procedimiento	

SISTEMA		
Tabla que tiene los diferentes sistemas iso (9001,14001)		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id del sistema iso	
NOMBRE	Nombre del sistema (9001,14001)	

CUENTA_BANCARIA		
Tabla que tiene la información de la cuenta bancaria del empleado		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
NUMERO	Numero de la cuenta bancaria	
ID_PERSONA	Id del empleado relacionado con la cuenta	
TIPO	Tipo de cuenta (corriente, ahorro)	
BANCO	Nombre del banco donde esta registrado la cuenta	

USUARIOS		
Tabla que contiene los usuarios que pueden utilizar la aplicación		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Campo autonumérico que se incrementa cuando se crean los datos.	
NOMBRE	Nombre del usuario	
PASSWORD	Contraseña encriptada del usuario	
ID_RAZON	Id de la razon a la que pertenece	

ROLES		
Tabla que contiene los diferentes permisos de los usuraos		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Id del permiso	
NOMBRE	Nombre del permiso o rol	

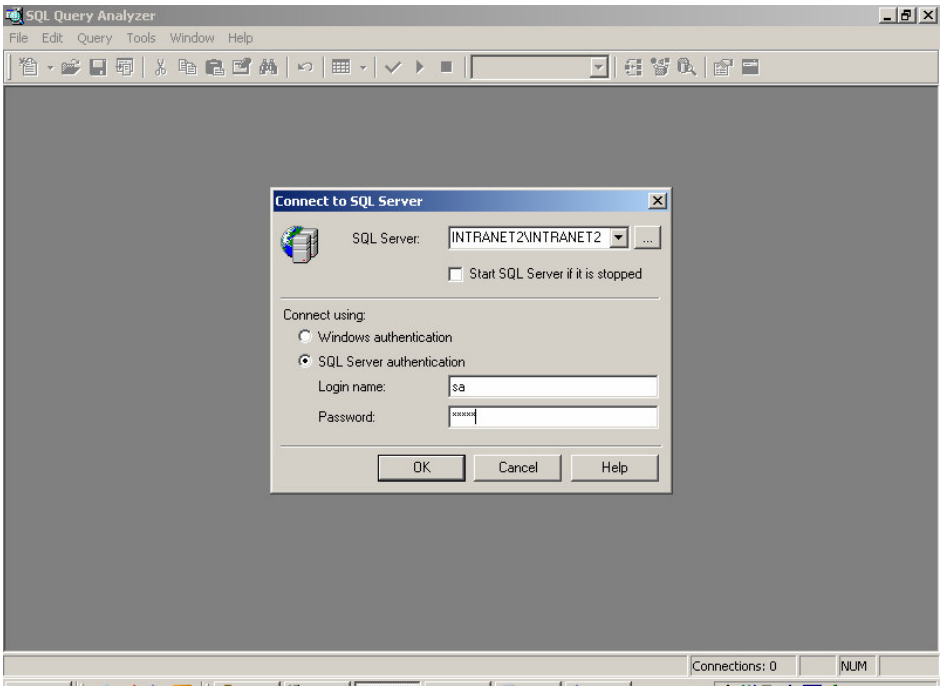
REGISTER_VACUNA		
Tabla que contiene los datos de los registros de vacunación		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID	Campo autonumérico que se incrementa cuando se crean los datos.	
ID_VACUNA	Id de la vacuna	
ID_RAZON	Id de la empresa que realizo la vacunación	
FECHA	Fecha en que se realizo la vacunación	
ANO	Ano en que se realizo la vacunacion	

PESONA_ESCOLARIDAD		
Tabla que contiene los datos del nivel escolar de una persona		
NOMBRE CAMPO	**DESCRIPCION**	**NULL**
ID_PERSONA	Id de la persona	
ID_ESCOLAR	Id del nivel escolar	
ESTUDIA	Si estudia actualmente	
DESCRIPCION	Que estudia actualmente	X
TITULO	Titulo que obtuvo	X

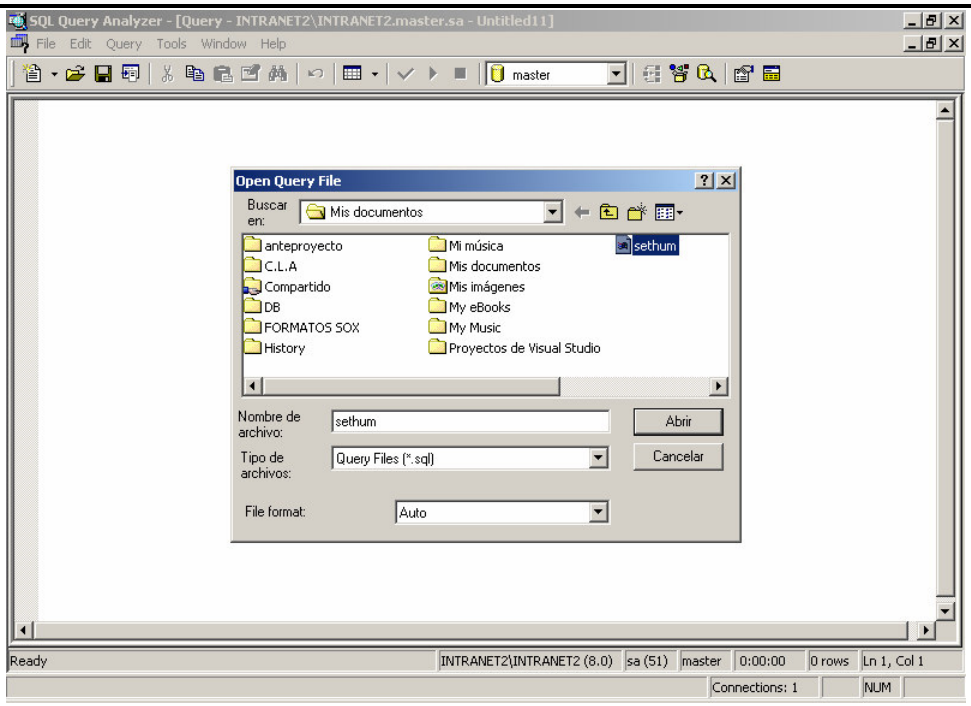
5. CREACION DE LA BASE DE DATOS

Si hemos perdido la base de datos, o si la base ha sufrido daños podemos restaurarla siguiendo los siguientes pasos:

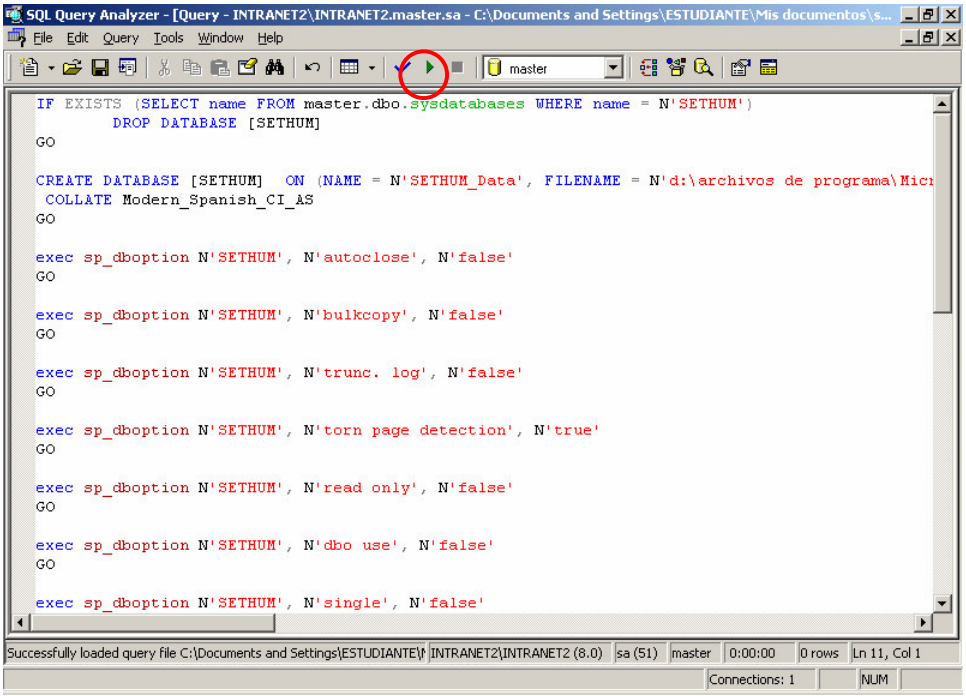
1. Ejecutamos el SQL Query Analizar, el cual es una consola de administración que permite crear tablas, procedimientos, vistas o Base de Datos mediante la ejecución del script de creación. En esta consola debemos autenticarnos como usuarios sa.



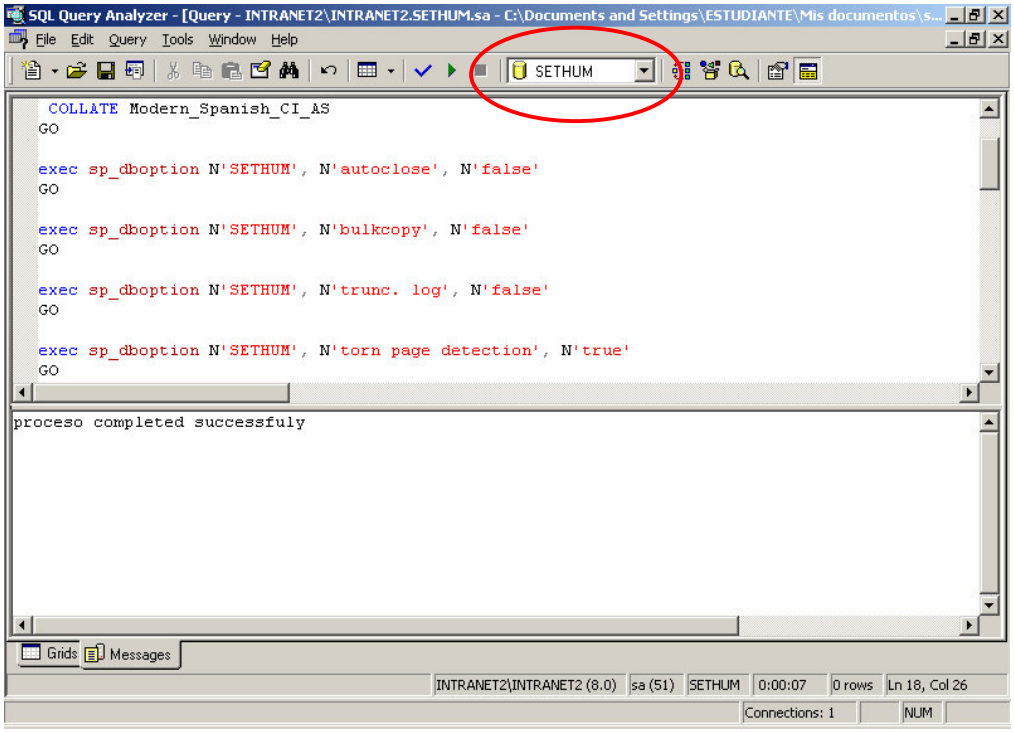
2. Luego de habernos autenticado debemos abrir el archivo (.sql) el cual contiene el script de creación de las tablas, vistas y procedimientos almacenados que utiliza la base de datos.



3. Cuando abrimos el archivo podemos ver el script en la consola, ahora debemos ejecutarlo para esto presionamos la tecla **F5** o hacemos click en el icono ejecutar de la barra de herramientas.



4. Si el proceso termina exitosamente hemos creado toda la estructura de la base de datos, ya que podemos ver que ahora nos aparece en la lista de base de datos del motor de SQL.



6. SCRIPT DE CREACION DE LA BASE DE DATOS

6.1. TABLAS

```
/*==============================================================*/
/* DBMS name:      Microsoft SQL Server 2000                    */
/* Created on:     20/07/2005 10:01:25 p.m.                     */
/*==============================================================*/

/*==============================================================*/
/* Table: ACCIDENTES                                            */
/*==============================================================*/
create table dbo.ACCIDENTES (
   ID                   int                  identity(1 , 1),
   ID_PERSONA           int                  not null,
   FECHA                datetime             not null,
   HORA                 datetime             null,
   DIAS_INCAPACIDAD     float                null,
   constraint PK_ACCIDENTES primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: AREA                                                  */
/*==============================================================*/
create table dbo.AREA (
   ID                   int                  identity(1 , 1),
   NOMBRE               sysname              not null,
   ID_RAZON             nvarchar(4)          not null,
   TIPO                 char(1)              not null,
   constraint PK_AREA primary key clustered (ID)
         on "PRIMARY"
)
go



/*==============================================================*/
/* Table: AUDIOMETRIA                                           */
/*==============================================================*/
create table dbo.AUDIOMETRIA (
   ID_EMPLEADO          int                  not null,
   ID_REGISTER          int                  not null,
   RESULTADO            nvarchar(12)         not null,
   constraint PK_AUDIOMETRIA primary key clustered (ID_EMPLEADO, ID_REGISTER)
         on "PRIMARY"
)
go



/*==============================================================*/
/* Table: CAJA_COMPENSACION                                     */
```

```
/*==============================================================*/
create table dbo.CAJA_COMPENSACION (
   ID                nvarchar(13)        not null,
   NOMBRE            nvarchar(32)        not null,
   constraint PK_CAJA_COMPENSACION primary key clustered (ID)
        on "PRIMARY"
)
go

/*==============================================================*/
/* Table: CAMISA                                                */
/*==============================================================*/
create table dbo.CAMISA (
   ID_DOTACION       int               not null,
   TMANGA            char(1)           not null,
   BOLSILLO          char(1)           not null,
   TIPO              nvarchar(10)      not null
)
on "PRIMARY"
go


/*==============================================================*/
/* Table: CAPACITACION                                          */
/*==============================================================*/
create table dbo.CAPACITACION (
   ID                int               identity(1 , 1),
   LUGAR             nvarchar(64)      not null,
   CONFERENCISTA     sysname           null,
   FECHA             datetime          not null,
   ID_RAZON          nvarchar(4)       not null,
   HORA_INICIO       datetime          null,
   HORA_FINAL        datetime          null,
   SISTEMA           char(10)          null,
   ID_PROCEDIMIENTO  nvarchar(10)      null,
   ID_AREA           int               null,
   ID_TEMA           int               null,
   ID_SISTEMA        nvarchar(10)      null,
   PRO_ID_PROCEDIMIENTO nvarchar(10)   null,
   PRO_ID_AREA       int               null,
   SEMANA            int               null,
   PERIODO           int               null,
   ANO               int               null,
   HORAS             int               null,
   MINUTOS           int               null,
   NDIAS             int               null,
   ESTADO            char(1)           not null,
   constraint PK_CAPACITACION primary key clustered (ID)
        on "PRIMARY"
)
go




/*==============================================================*/
/* Table: CENTRO_COSTO                                          */
```

```
/*==============================================================*/
create table dbo.CENTRO_COSTO (
   ID                int                 not null,
   NOMBRE            nvarchar(32)        not null,
   constraint PK_CENTRO_COSTO primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: CITOLOGIAS                                            */
/*==============================================================*/
create table dbo.CITOLOGIAS (
   ID_EMPLEADO       int                 not null,
   ID_REGISTER       int                 not null,
   constraint PK_CITOLOGIA primary key clustered (ID_EMPLEADO, ID_REGISTER)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: CIUDAD                                                */
/*==============================================================*/
create table dbo.CIUDAD (
   ID                nvarchar(5)         not null,
   NOMBRE            sysname             null,
   ID_DEPARTAMENTO   nvarchar(2)         not null,
   constraint PK_CIUDAD primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: COLINESTARASA                                         */
/*==============================================================*/
create table dbo.COLINESTARASA (
   ID_EMPLEADO       int                 not null,
   ID_REGISTER       int                 not null,
   TIPO              char(2)             not null,
   RESULTADO         int                 not null,
   constraint PK_COLINESTARASA primary key clustered (ID_EMPLEADO, ID_REGISTER)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: COMPOSICION_FAMILIAR                                  */
```

```
/*==============================================================*/
create table dbo.COMPOSICION_FAMILIAR (
   ID_PERSONA          int                 not null,
   ID_FAMILIA          int                 not null,
   ID_PARENTESCO       nvarchar(2)         not null,
   LLAMAR_EMERGENCIA   bit                 not null default 0,
   OCUPACION           nvarchar(32)        null,
   CONVIVEN            bit                 not null default 0,
   ID_CAJA             nvarchar(13)        null,
   constraint PK_COMPOSICION_FAMILIAR primary key clustered (ID_PERSONA, ID_FAMILIA)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: CUENTA_BANCARIA                                       */
/*==============================================================*/
create table dbo.CUENTA_BANCARIA (
   NUMERO              nvarchar(16)        not null,
   ID_PERSONA          int                 not null,
   TIPO                nvarchar(12)        not null,
   BANCO               nvarchar(64)        not null,
   constraint PK_CUENTA_BANCARIA primary key clustered (NUMERO)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: DEPARTAMENTO                                          */
/*==============================================================*/
create table dbo.DEPARTAMENTO (
   ID                  nvarchar(2)         not null,
   ID_PAIS             nvarchar(2)         not null,
   NOMBRE              nvarchar(64)        not null,
   constraint PK_DEPARTAMENTO primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: DOCUMENTO                                             */
/*==============================================================*/
create table dbo.DOCUMENTO (
   ID                  nvarchar(2)         not null,
   TIPO_DOCUMENTO      nvarchar(32)        not null,
   constraint PK_DOCUMENTO primary key clustered (ID)
         on "PRIMARY"
)
go




/*==============================================================*/
/* Table: DOCUMENTO_PERSONA                                     */
```

```
/*==============================================================*/
create table dbo.DOCUMENTO_PERSONA (
   ID                nvarchar(2)          not null,
   NUMERO            nvarchar(12)         not null,
   ID_PERSONA        int                  not null,
   LUGAR_EXPEDICION  nvarchar(5)          null,
   FECHA_EXPEDICION  datetime             null,
   constraint PK_DOCUMENTO_PERSONA primary key clustered (ID, NUMERO)
        on "PRIMARY"
)
go

/*==============================================================*/
/* Table: DOTACION                                              */
/*==============================================================*/
create table dbo.DOTACION (
   ID                int                  identity(1 , 1),
   COLOR             nvarchar(64)         null,
   DESCRIPCION       sysname              null,
   TIPO              char(1)              not null,
   constraint PK_DOTACION primary key clustered (ID)
        on "PRIMARY"
)
go

/*==============================================================*/
/* Table: DOTACION_OFICIO                                       */
/*==============================================================*/
create table dbo.DOTACION_OFICIO (
   ID_DOTACION       int                  not null,
   ID_OFICIO         int                  not null,
   DOTACION          sysname              not null,
   constraint PK_DOTACION_OFICIO primary key clustered (ID_DOTACION, ID_OFICIO)
        on "PRIMARY"
)
go

/*==============================================================*/
/* Table: EMPLEADO                                              */
/*==============================================================*/
create table dbo.EMPLEADO (
   ID                int                  identity(1 , 1),
   ID_CAJA           nvarchar(13)         null,
   ID_OFICIO         int                  null,
   FECHA_INGRESO     datetime             not null,
   TIPO_CONTRATO     nvarchar(1)          not null,
   FECHA_VENCIMIENTO datetime             null,
   ESTADO            nvarchar(1)          not null,
   ID_PERSONA        int                  not null,
   UNIFORME          bit                  null,
   constraint PK_EMPLEADO primary key clustered (ID)
        on "PRIMARY"
)
go

/*==============================================================*/
/* Table: EMPLEADO_COSTO                                        */
/*==============================================================*/
```

```
create table dbo.EMPLEADO_COSTO (
   ID_EMPLEADO          int                 not null,
   ID_COSTO             int                 not null,
   constraint PK_EMPLEADO_COSTO primary key clustered (ID_EMPLEADO, ID_COSTO)
         on "PRIMARY"
)
go
/*==============================================================*/
/* Table: EMPLEADO_PLAN                                         */
/*==============================================================*/
create table dbo.EMPLEADO_PLAN (
   ID_EMPLEADO          int                 not null,
   ID_PLAN              nvarchar(2)         not null,
   P_AHORRO             nvarchar(12)        not null,
   constraint PK_EMPLEADO_PLAN primary key clustered (ID_EMPLEADO, ID_PLAN)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: EMPLEADO_VACUNA                                       */
/*==============================================================*/
create table dbo.EMPLEADO_VACUNA (
   ID_EMPLEADO          int                 not null,
   ID_REGISTRO          int                 not null,
   constraint PK_VACUNA_EMPLEADO primary key clustered (ID_EMPLEADO, ID_REGISTRO)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: EPP                                                   */
/*==============================================================*/
create table dbo.EPP (
   ID                   nvarchar(10)        not null,
   NOMBRE               nvarchar(256)       not null,
   REFERENCIA           nvarchar(10)        null,
   ID_RAZON             nvarchar(4)         not null,
   constraint PK_EPP primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: EPP_EMPLEADO                                          */
/*==============================================================*/
create table dbo.EPP_EMPLEADO (
   ID_MOV               int                 not null,
   ID_EPP               nvarchar(10)        not null,
   CANT                 int                 not null,
   DESCRIPCION          nvarchar(512)       null,
   constraint PK_EPP_EMPLEADO primary key clustered (ID_MOV, ID_EPP)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: EPP_OFICIO                                            */
```

```
/*==============================================================*/
create table dbo.EPP_OFICIO (
   ID_OFICIO           int                 not null,
   ID_EPP              nvarchar(10)        not null,
   constraint PK_EPP_OFICIO primary key clustered (ID_OFICIO, ID_EPP)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: ESTADOCIVIL_PERSONA                                   */
/*==============================================================*/
create table dbo.ESTADOCIVIL_PERSONA (
   ID_PERSONA          int                 not null,
   ID_CIVIL            nvarchar(2)         not null,
   constraint PK_ESTADOCIVIL_PERSONA primary key clustered (ID_PERSONA, ID_CIVIL)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: ESTADO_CIVIL                                          */
/*==============================================================*/
create table dbo.ESTADO_CIVIL (
   ID                  nvarchar(2)         not null,
   NOMBRE              nvarchar(16)        not null,
   constraint PK_ESTADO_CIVIL primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: FONDO_PENSION_SALUD                                   */
/*==============================================================*/
create table dbo.FONDO_PENSION_SALUD (
   NIT                 nvarchar(10)        not null,
   TIPO_FONDO          nvarchar(3)         not null,
   ID_CIUDAD           nvarchar(5)         null,
   NOMBRE              nvarchar(64)        not null,
   DIRECCION           nvarchar(64)        not null,
   TELEFONO            nvarchar(10)        null,
   constraint PK_FONDO_PENSION_SALUD primary key clustered (NIT, TIPO_FONDO)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: FPS_PERSONA                                           */
```

```
/*==============================================================*/
create table dbo.FPS_PERSONA (
   ID_PERSONA          int                 not null,
   NIT                 nvarchar(10)        not null,
   TIPO_FONDO          nvarchar(3)         not null,
   TIPO_AFILIACION     nvarchar(1)         null,
   constraint PK_FPS_PERSONA primary key clustered (ID_PERSONA, NIT, TIPO_FONDO)
         on "PRIMARY"
)
go
/*==============================================================*/
/* Table: JEFES                                                 */
/*==============================================================*/
create table dbo.JEFES (
   ID_JEFE             int                 not null,
   ID_EMPLEADO         int                 not null,
   constraint PK_JEFES primary key clustered (ID_JEFE, ID_EMPLEADO)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: LADOS_ACCIDENTES                                      */
/*==============================================================*/
create table dbo.LADOS_ACCIDENTES (
   LADO                nvarchar(1)         not null,
   ID_ACCIDENTES       int                 not null,
   ID_LADO             nvarchar(2)         not null,
   constraint PK_LADOS_ACCIDENTES primary key clustered (ID_ACCIDENTES, ID_LADO)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: LADOS_AFECTADOS                                       */
/*==============================================================*/
create table dbo.LADOS_AFECTADOS (
   ID                  nvarchar(2)         not null,
   NOMBRE              nvarchar(32)        not null,
   constraint PK_LADOS_AFECTADOS primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: LICENCIA_CONDUCCION                                   */
/*==============================================================*/
create table dbo.LICENCIA_CONDUCCION (
   NUMERO_LICENCIA     nvarchar(12)        not null,
   CATEGORIA           nvarchar(1)         not null,
   FECHA_VENCIMIENTO   datetime            not null,
   ID_PERSONA          int                 null,
   constraint PK_LICENCIA_CONDUCCION primary key clustered (NUMERO_LICENCIA)
         on "PRIMARY"
)
go
/*==============================================================*/
/* Table: MOV_EPP                                               */
```

```
/*==============================================================*/
create table dbo.MOV_EPP (
   ID                int                identity(1 , 1),
   FECHA             datetime           not null,
   SEMANA            int                not null,
   PERIODO           int                not null,
   ANO               int                not null,
   ID_EMPLEADO       int                not null,
   constraint PK_MOV_EPP primary key clustered (ID)
        on "PRIMARY"
)
go
/*==============================================================*/
/* Table: NIVEL_ESCOLAR                                         */
/*==============================================================*/
create table dbo.NIVEL_ESCOLAR (
   ID                nvarchar(2)        not null,
   NOMBRE            nvarchar(32)       not null,
   constraint PK_NIVEL_ESCOLAR primary key clustered (ID)
        on "PRIMARY"
)
go
/*==============================================================*/
/* Table: OFICIO                                                */
/*==============================================================*/
create table dbo.OFICIO (
   ID                int                identity(1 , 1),
   ID_TIPO_EMPLEO    int                not null,
   ID_RAZON          nvarchar(4)        not null,
   NOMBRE            nvarchar(64)       not null,
   constraint PK_OFICIO primary key clustered (ID)
        on "PRIMARY"
)
go

/*==============================================================*/
/* Table: PAIS                                                  */
/*==============================================================*/
create table dbo.PAIS (
   ID                nvarchar(2)        not null,
   NOMBRE            nvarchar(64)       not null,
   constraint PK_PAIS primary key clustered (ID)
        on "PRIMARY"
)
go

/*==============================================================*/
/* Table: PARENTESCO                                            */
/*==============================================================*/
create table dbo.PARENTESCO (
   ID                nvarchar(2)        not null,
   NOMBRE            nvarchar(32)       not null,
   constraint PK_PARENTESCO primary key clustered (ID)
        on "PRIMARY"
)
go

/*==============================================================*/
```

```
/* Table: PARTES                                */
/*==============================================================*/
create table dbo.PARTES (
   ID                nvarchar(2)         not null,
   NOMBRE            nvarchar(32)        not null,
   constraint PK_PARTES primary key clustered (ID)
        on "PRIMARY"
)
go

/*==============================================================*/
/* Table: PERSONA                               */
/*==============================================================*/
create table dbo.PERSONA (
   ID                int                 identity(1 , 1),
   CIUDAD_NACIMIENTO    nvarchar(5)          null,
   CIUDAD_RESIDENCIA    nvarchar(5)          null,
   APELLIDO1            nvarchar(16)         not null,
   APELLIDO2            nvarchar(16)         null,
   NOMBRES              nvarchar(32)         not null,
   SEXO                 nvarchar(1)          not null,
   FECHA_NACIMIENTO     datetime             not null,
   TARJETA_PROFESIONAL  nvarchar(64)         null,
   DIRECCION            nvarchar(64)         null,
   TELEFONO             nvarchar(10)         null,
   CORREO               sysname              null,
   CELULAR              nvarchar(10)         null,
   TIPO_SANGRE          nvarchar(2)          null,
   FACTOR_SANGRE        nvarchar(1)          null,
   PESO                 smallint             null,
   ESTATURA             float                null,
   TALLA_CAMISA         nvarchar(2)          null,
   TALLA_PANTALON       nvarchar(2)          null,
   TALLA_ZAPATO         nvarchar(2)          null,
   constraint PK_PERSONA primary key clustered (ID)
        on "PRIMARY"
)
go


/*==============================================================*/
/* Table: PERSONA_ESCOLARIDAD                   */
/*==============================================================*/
create table dbo.PERSONA_ESCOLARIDAD (
   ID_PERSONA           int                 not null,
   ID_ESCOLAR           nvarchar(2)         not null,
   ESTUDIA              bit                 not null default 0,
   DESCRIPCION          sysname             null,
   TITULO               nvarchar(64)        null,
   ESTADO               nvarchar(1)         null,
   constraint PK_PERSONA_ESCOLARIDAD primary key clustered (ID_PERSONA, ID_ESCOLAR)
        on "PRIMARY"
)
go

/*==============================================================*/
/* Table: PERSONA_POLIZA                        */
/*==============================================================*/
```

```
create table dbo.PERSONA_POLIZA (
   TIPO_POLIZA          nvarchar(1)          not null,
   ID_PERSONA           int                  not null,
   ENTIDAD              nvarchar(32)         not null,
   constraint PK_PERSONA_POLIZA primary key clustered (TIPO_POLIZA, ID_PERSONA,
ENTIDAD)
         on "PRIMARY"
)
go
/*==============================================================*/
/* Table: PLAN_AHORRO                                           */
/*==============================================================*/
create table dbo.PLAN_AHORRO (
   ID                   nvarchar(2)          not null,
   NOMBRE               nvarchar(32)         not null,
   constraint PK_PLAN_AHORRO primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: POLIZAS                                               */
/*==============================================================*/
create table dbo.POLIZAS (
   TIPO_POLIZA          nvarchar(1)          not null,
   NIT                  nvarchar(14)         not null,
   ENTIDAD              nvarchar(32)         not null,
   constraint PK_POLIZAS primary key clustered (TIPO_POLIZA, ENTIDAD)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: PROCEDIMIENTO                                         */
/*==============================================================*/
create table dbo.PROCEDIMIENTO (
   ID_AREA              int                  not null,
   ID                   nvarchar(10)         not null,
   NOMBRE               nvarchar(256)        not null,
   constraint PK_PROCEDIMIENTO primary key clustered (ID, ID_AREA)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: PROCEDIMIENTO_SISTEMA                                 */
/*==============================================================*/
create table dbo.PROCEDIMIENTO_SISTEMA (
   ID_SISTEMA           nvarchar(10)         not null,
   ID_PROCEDIMIENTO     nvarchar(10)         not null,
   ID_AREA              int                  not null,
   constraint PK_PROCEDIMIENTO_SISTEMA primary key clustered (ID_SISTEMA,
ID_PROCEDIMIENTO, ID_AREA)
         on "PRIMARY"
)
go

/*==============================================================*/
```

```
/* Table: RAZON_COSTO                                    */
/*==============================================================*/
create table dbo.RAZON_COSTO (
   ID_RAZON            nvarchar(4)         not null,
   ID_COSTO            int                 not null,
   constraint PK_RAZON_COSTO primary key clustered (ID_RAZON, ID_COSTO)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: RAZON_EMPLEO                                   */
/*==============================================================*/
create table dbo.RAZON_EMPLEO (
   ID_RAZON            nvarchar(4)         not null,
   ID_TIPO_EMPLEO      int                 not null,
   constraint PK_RAZON_EMPLEO primary key clustered (ID_RAZON, ID_TIPO_EMPLEO)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: RAZON_SISTEMA                                  */
/*==============================================================*/
create table dbo.RAZON_SISTEMA (
   ID_RAZON            nvarchar(4)         not null,
   ID_SISTEMA          nvarchar(10)        not null,
   constraint PK_RAZON_SISTEMA primary key clustered (ID_RAZON, ID_SISTEMA)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: RAZON_SOCIAL                                   */
/*==============================================================*/
create table dbo.RAZON_SOCIAL (
   ID                  nvarchar(4)         not null,
   NOMBRE              sysname             not null,
   FINCA               bit                 not null default 0,
   CORREO              nvarchar(24)        null,
   constraint PK_RAZON_SOCIAL primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: REGISTER_EXAMEN                                */
/*==============================================================*/
create table dbo.REGISTER_EXAMEN (
   ID                  int                 identity(1 , 1),
   ID_RAZON            nvarchar(4)         not null,
   FECHA               datetime            not null,
   ANO                 int                 not null,
   TIPO                char(1)             not null,
   constraint PK_REGISTER_CITOLOGIA primary key clustered (ID)
         on "PRIMARY"
)
go
```

```
/*==============================================================*/
/* Table: REGISTER_VACUNAS                                      */
/*==============================================================*/
create table dbo.REGISTER_VACUNAS (
   ID                int                identity(1 , 1),
   ID_VACUNA         int                not null,
   ID_RAZON          nvarchar(4)        not null,
   FECHA             datetime           not null,
   ANO               int                not null,
   constraint PK_REGISTER_VACUNAS primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: REGISTRO                                              */
/*==============================================================*/
create table dbo.REGISTRO (
   ID_EMPLEADO       int                not null,
   ID_CAPACITACION   int                not null,
   constraint PK_REGISTRO primary key clustered (ID_EMPLEADO, ID_CAPACITACION)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: ROLES                                                 */
/*==============================================================*/
create table dbo.ROLES (
   ID                nvarchar(2)        not null,
   NOMBRE            nvarchar(16)       not null,
   constraint PK_ROLES primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: SERVICIOS                                             */
/*==============================================================*/
create table dbo.SERVICIOS (
   ID                nvarchar(2)        not null,
   NOMBRE            nvarchar(32)       not null,
   constraint PK_SERVICIOS primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: SISTEMA                                               */
/*==============================================================*/
create table dbo.SISTEMA (
   ID                nvarchar(10)       not null,
   NOMBRE            nvarchar(32)       not null,
   constraint PK_SISTEMA primary key clustered (ID)
         on "PRIMARY"
)
go
```

```
/*==============================================================*/
/* Table: TAMIZAJE                                              */
/*==============================================================*/
create table dbo.TAMIZAJE (
   ID_EMPLEADO          int                 not null,
   ID_REGISTER          int                 not null,
   TIPO                 char(1)             not null,
   RESULTADO            int                 not null,
   constraint PK_TAMIZAJE primary key clustered (ID_EMPLEADO, ID_REGISTER)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: TEMA_LIBRE                                            */
/*==============================================================*/
create table dbo.TEMA_LIBRE (
   ID                   int                 identity(1 , 1),
   ID_AREA              int                 not null,
   NOMBRE               sysname             not null,
   constraint PK_TEMA_LIBRE primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: TIPO_EMPLEO                                           */
/*==============================================================*/
create table dbo.TIPO_EMPLEO (
   ID                   int                 identity(1 , 1),
   NOMBRE               nvarchar(32)        not null,
   constraint PK_TIPO_EMPLEO primary key clustered (ID)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: USUARIOS                                              */
/*==============================================================*/
create table dbo.USUARIOS (
   ID                   int                 identity(1 , 1),
   NOMBRE               nvarchar(14)        not null,
   PASSWORD             nvarchar(14)        not null,
   ID_RAZON             nvarchar(4)         not null,
   constraint PK_USUARIOS primary key clustered (ID)
         on "PRIMARY"
)
go




/*==============================================================*/
/* Table: USUARIOS_ROLES                                        */
```

```
/*==============================================================*/
create table dbo.USUARIOS_ROLES (
   ID_USUARIO          int                 not null,
   ID_ROL              nvarchar(2)         not null
)
on "PRIMARY"
go
/*==============================================================*/
/* Table: VACUNAS                                               */
/*==============================================================*/
create table dbo.VACUNAS (
   ID                  int                 identity(1 , 1),
   NOMBRE              nvarchar(64)        not null,
   constraint PK_VACUNAS primary key clustered (ID)
        on "PRIMARY"
)
go

/*==============================================================*/
/* Table: VISIOMETRIA                                           */
/*==============================================================*/
create table dbo.VISIOMETRIA (
   ID_EMPLEADO         int                 not null,
   ID_REGISTER         int                 not null,
   RESULTADO           nvarchar(12)        not null,
   constraint PK_VISIOMETRIA primary key clustered (ID_EMPLEADO, ID_REGISTER)
        on "PRIMARY"
)
go

/*==============================================================*/
/* Table: VIVIENDA                                              */
/*==============================================================*/
create table dbo.VIVIENDA (
   ID                  int                 identity(1 , 1),
   TENENCIA            nvarchar(32)        not null,
   ADQUISICION         nvarchar(32)        null,
   ID_PERSONA          int                 not null,
   DOCUMENTO           nvarchar(32)        not null,
   constraint PK_VIVIENDA primary key clustered (ID)
        on "PRIMARY"
)
go
/*==============================================================*/
/* Table: VIVIENDA_PARTES                                       */
/*==============================================================*/
create table dbo.VIVIENDA_PARTES (
   ID_VIVIENDA         int                 not null,
   ID_PARTES           nvarchar(2)         not null,
   CANTIDAD            int                 null,
   ESTADO              nvarchar(32)        not null,
   INTERNO             bit                 not null default 0,
   constraint PK_VIVIENDA_PARTES primary key clustered (ID_VIVIENDA, ID_PARTES)
        on "PRIMARY"
)
go

/*==============================================================*/
```

```
/* Table: VIVIENDA_SERVICIOS                                    */
/*==============================================================*/
create table dbo.VIVIENDA_SERVICIOS (
   ID_SERVICIO          nvarchar(2)          not null,
   ID_VIVIENDA          int                  not null,
   constraint PK_VIVIENDA_SERVICIOS primary key clustered (ID_SERVICIO, ID_VIVIENDA)
         on "PRIMARY"
)
go

/*==============================================================*/
/* Table: ZAPATO                                                */
/*==============================================================*/
create table dbo.ZAPATO (
   ID_DOTACION          int                  not null,
   P_HIERRO             bit                  not null
)
on "PRIMARY"
go
```

6.2. VISTAS

```
/*==============================================================*/
/* View: DATOSEMPLEADO                                          */
/*==============================================================*/
CREATE VIEW dbo.DATOSEMPLEADO
AS
SELECT P.ID_RAZON, P.ID_COSTO,P.NUMERO,
MAX(CASE SEXO WHEN 'F' THEN'FEMENINO' ELSE 'MASCULINO'END) AS SEXO,
MAX(CASE TIPO_CONTRATO WHEN 'I' THEN 'INDEFINIDO' ELSE 'DEFINIDO' END) AS
TIPO_CONTRATO
FROM PEREMPLEADO P
GROUP BY P.ID_RAZON,P.ID_COSTO,P.NUMERO

/*==============================================================*/
/* View: PERFAMILIA                                             */
/*==============================================================*/
CREATE VIEW dbo.PERFAMILIA
AS
SELECT     O.ID_RAZON, EO.ID_COSTO, COUNT(F.ID_FAMILIA)  + 1 AS FAMILIA,
CC.NOMBRE AS COSTOS
FROM         dbo.COMPOSICION_FAMILIAR F INNER JOIN
                  dbo.EMPLEADO E ON F.ID_PERSONA = E.ID_PERSONA INNER JOIN
                  dbo.EMPLEADO_COSTO EO ON EO.ID_EMPLEADO =  E.ID INNER JOIN
                  dbo.OFICIO O  ON O.ID = E.ID_OFICIO INNER JOIN
                  dbo.CENTRO_COSTO CC ON EO.ID_COSTO = CC.ID
WHERE     (F.CONVIVEN = 1)
GROUP BY F.ID_PERSONA, EO.ID_COSTO, O.ID_RAZON, CC.NOMBRE




/*==============================================================*/
```

```
/* View: ESTADOPARTES
/*============================================================*/
CREATE VIEW dbo.ESTADOPARTES
AS
SELECT ID_VIVIENDA,
MAX(CASE ID_PARTES WHEN '1' THEN ESTADO END) AS COCINA,
MAX(CASE ID_PARTES WHEN '2' THEN ESTADO END) AS SANITARIO,
MAX(CASE ID_PARTES WHEN '3' THEN ESTADO END) AS SALA,
MAX(CASE ID_PARTES WHEN '4' THEN ESTADO END) AS COMEDOR,
MAX(CASE ID_PARTES WHEN '5' THEN ESTADO END) AS PATIO,
MAX(CASE ID_PARTES WHEN '6' THEN ESTADO END) AS PISOS,
MAX(CASE ID_PARTES WHEN '7' THEN ESTADO END) AS CUARTOS,
MAX(CASE ID_PARTES WHEN '7' THEN CANTIDAD END) AS CANTIDAD
FROM VIVIENDA_PARTES
GROUP BY ID_VIVIENDA

/*============================================================*/
/* View: .ESTADOSERVICIOS
/*============================================================*/
CREATE VIEW dbo.ESTADOSERVICIOS
AS
SELECT ID_VIVIENDA,
MAX(CASE ID_SERVICIO WHEN '1' THEN 1 END) AS LUZ,
MAX(CASE ID_SERVICIO WHEN '2' THEN 1 END) AS GAS,
MAX(CASE ID_SERVICIO WHEN '3' THEN 1 END) AS POZO_ARTESANAL,
MAX(CASE ID_SERVICIO WHEN '4' THEN 1 END) AS ACUEDUCTO_PUBLICO,
MAX(CASE ID_SERVICIO WHEN '5' THEN 1 END) AS CONEXION_ALCANTARILLADO,
MAX(CASE ID_SERVICIO WHEN '7' THEN 1 END) AS ACUEDUCTO_COMUNAL,
MAX(CASE ID_SERVICIO WHEN '8' THEN 1 END) AS CARROTANQUE,
MAX(CASE ID_SERVICIO WHEN '9' THEN 1 END) AS RIO,
MAX(CASE ID_SERVICIO WHEN '10' THEN 1 END) AS TELEFONO,
MAX(CASE ID_SERVICIO WHEN '11' THEN 1 END) AS PILA_PUBLICA,
MAX(CASE ID_SERVICIO WHEN '12' THEN 1 END) AS POZO_SEPTICO
FROM VIVIENDA_SERVICIOS
GROUP BY ID_VIVIENDA


/*============================================================*/
/* View: . FAMILIA
/*============================================================*/
CREATE VIEW dbo.FAMILIA
AS
SELECT ID_COSTO,ID_RAZON, COSTOS,
        COUNT(CASE FAMILIA WHEN 1 THEN 1 END) AS '1',
        COUNT(CASE FAMILIA WHEN 2 THEN 1 END) AS '2',
        COUNT(CASE FAMILIA WHEN 3 THEN 1 END) AS '3',
        COUNT(CASE FAMILIA WHEN 4 THEN 1 END) AS '4',
        COUNT(CASE FAMILIA WHEN 5 THEN 1 END) AS '5',
        COUNT(CASE FAMILIA WHEN 6 THEN 1 END) AS '6',
        COUNT(CASE FAMILIA WHEN 7 THEN 1 END) AS '7',
        COUNT(CASE FAMILIA WHEN 8 THEN 1 END) AS '8',
        COUNT(CASE FAMILIA WHEN 9 THEN 1 END) AS '9',
        COUNT(CASE FAMILIA WHEN 10 THEN 1 END) AS '10',
        COUNT(CASE FAMILIA WHEN 11 THEN 1 END) AS '11',
        COUNT(CASE FAMILIA WHEN 12 THEN 1 END) AS '12',
        COUNT(CASE FAMILIA WHEN 13 THEN 1 END) AS '13'
FROM dbo.PERFAMILIA
GROUP BY ID_COSTO, ID_RAZON, COSTOS
```

```
/*==============================================================*/
/* View: . GRPARTES
/*==============================================================*/
CREATE VIEW dbo.GRPARTES
AS
SELECT O.ID_RAZON, EO.ID_COSTO,ID_VIVIENDA,P.NOMBRE, VP.ID_PARTES,
MAX(CASE VP.ESTADO WHEN 'NO EXISTE' THEN'1' END) AS NO_EXISTE,
MAX(CASE VP.ESTADO WHEN 'DEFICIENTE' THEN '1' END) AS DEFICIENTE,
MAX(CASE VP.ESTADO WHEN 'REGULAR' THEN '1' END) AS REGULAR,
MAX(CASE VP.ESTADO WHEN 'BUENO' THEN '1' END) AS BUENO,
MAX(CASE VP.ID_PARTES WHEN '2' THEN VP.ESTADO END) AS SANITARIO,
MAX(CASE VP.ID_PARTES WHEN '6' THEN VP.ESTADO END) AS PISOS
FROM VIVIENDA_PARTES VP  INNER JOIN
VIVIENDA V ON V.ID=VP.ID_VIVIENDA INNER JOIN
PARTES P ON VP.ID_PARTES= P.ID INNER JOIN
EMPLEADO E ON V.ID_PERSONA= E.ID_PERSONA INNER JOIN
EMPLEADO_COSTO EO ON EO.ID_EMPLEADO =  E.ID INNER JOIN
OFICIO O ON O.ID = E.ID_OFICIO
 WHERE (E.ESTADO='a')
GROUP BY O.ID_RAZON,EO.ID_COSTO,ID_VIVIENDA, P.NOMBRE, VP.ID_PARTES

/*==============================================================*/
/* View: . HIJOS
/*==============================================================*/
CREATE VIEW dbo.HIJOS
AS
SELECT     O.ID_RAZON, EO.ID_COSTO, E.ID_PERSONA AS ID, H.EMPLEADO,
COUNT(F.ID_FAMILIA) AS NHIJO , H.HIJO, H.SEXO, H.FECHA
FROM         dbo.COMPOSICION_FAMILIAR F INNER JOIN
                  dbo.EMPLEADO E ON F.ID_PERSONA = E.ID_PERSONA INNER JOIN
                  dbo.EMPLEADO_COSTO EO ON EO.ID_EMPLEADO =  E.ID INNER JOIN
                  dbo.OFICIO O ON O.ID = E.ID_OFICIO INNER JOIN
                  dbo.PERHIJOS H ON H.ID_PERSONA = E.ID_PERSONA
WHERE     (F.ID_PARENTESCO = 5)
GROUP BY F.ID_PERSONA, EO.ID_COSTO, O.ID_RAZON, E.ID_PERSONA, H.HIJO,
H.EMPLEADO, H.SEXO, H.FECHA

/*==============================================================*/
/* View: . JEFE
/*==============================================================*/
CREATE VIEW dbo.JEFE
AS
SELECT     DISTINCT J.ID_JEFE,P.CORREO , J.ID_EMPLEADO, PE.CARGO,PE.NUMERO
,PE.NOMBRES,PE.APELLIDO1,PE.APELLIDO2,PE.COSTO
FROM         JEFES J INNER JOIN
                     dbo.EMPLEADO E ON E.ID = J.ID_JEFE INNER JOIN
                     dbo.PERSONA P ON P.ID =  E.ID_PERSONA INNER JOIN
                     dbo.PEREMPLEADO PE ON J.ID_EMPLEADO = PE.ID
WHERE     (E.ESTADO = 'a')




/*==============================================================*/
```

```
/* View: . PERCAPACITACION
/*==============================================================*/
CREATE VIEW dbo.PERCAPACITACION
AS
SELECT     TOP 100 PERCENT O.ID_RAZON,EO.ID_COSTO AS COSTO, CC.NOMBRE AS
COSTOS,DP.NUMERO, P.NOMBRES, P.APELLIDO1, P.APELLIDO2, A.NOMBRE AS AREA,
TL.NOMBRE AS TEMA,C.SISTEMA AS SISTEMAS, C.ID_PROCEDIMIENTO AS
PROCEDIMIENTO,AP.NOMBRE AS ID_AREA, PR.NOMBRE AS TEMAPROCE,C.FECHA,
C.SEMANA, C.PERIODO, C.ANO, C.HORAS, C.MINUTOS, C.ID, E.ID AS ID_EMPLEADO
FROM        REGISTRO R INNER JOIN
                  EMPLEADO E ON R.ID_EMPLEADO = E.ID INNER JOIN
                  CAPACITACION C ON R.ID_CAPACITACION = C.ID LEFT JOIN
                  TEMA_LIBRE TL ON C.ID_TEMA = TL.ID LEFT JOIN
                  AREA A ON TL.ID_AREA = A.ID LEFT JOIN
                  PROCEDIMIENTO PR ON C.ID_PROCEDIMIENTO = PR.ID INNER JOIN
                  PERSONA P ON E.ID_PERSONA = P.ID INNER JOIN
                  DOCUMENTO_PERSONA DP ON DP.ID_PERSONA = P.ID INNER JOIN
                 EMPLEADO_COSTO EO ON EO.ID_EMPLEADO=E.ID INNER JOIN
                  CENTRO_COSTO AS CC ON CC.ID = EO.ID_COSTO INNER JOIN
                  OFICIO O ON O.ID = E.ID_OFICIO LEFT JOIN
                  AREA AP ON AP.ID = C.ID_AREA
WHERE     (E.ESTADO = 'a' )
GROUP BY O.ID_RAZON, EO.ID_COSTO, CC.NOMBRE,DP.NUMERO,P.NOMBRES,
P.APELLIDO1, P.APELLIDO2, A.NOMBRE, TL.NOMBRE,C.SISTEMA,C.ID_PROCEDIMIENTO,
PR.NOMBRE, C.FECHA, C.SEMANA, C.PERIODO,C.ANO,
C.HORAS,C.MINUTOS,C.ID,E.ID,AP.NOMBRE

/*==============================================================*/
/* View: . PEREMPLEADO
/*==============================================================*/
CREATE VIEW dbo.PEREMPLEADO
AS
SELECT     TOP 100 PERCENT O.ID_RAZON, EO.ID_COSTO, O.NOMBRE AS CARGO,
DP.NUMERO, P.NOMBRES, P.APELLIDO1, P.APELLIDO2, P.SEXO,P.CORREO, CC.NOMBRE
AS COSTO, E.ID,E.FECHA_INGRESO AS
FECHA,E.TIPO_CONTRATO,P.TALLA_CAMISA,P.TALLA_PANTALON,P.TALLA_ZAPATO,O.ID_
TIPO_EMPLEO
FROM        dbo.PERSONA P INNER JOIN
                  DOCUMENTO_PERSONA DP ON DP.ID_PERSONA = P.ID INNER JOIN
                  EMPLEADO E ON P.ID = E.ID_PERSONA INNER JOIN
                  OFICIO O ON O.ID = E.ID_OFICIO INNER JOIN
                  EMPLEADO_COSTO EO ON EO.ID_EMPLEADO=E.ID INNER JOIN
                  CENTRO_COSTO CC ON EO.ID_COSTO = CC.ID
WHERE     (E.ESTADO = 'a')
ORDER BY O.ID_RAZON, EO.ID_COSTO

/*==============================================================*/
/* View: . PEREPP
/*==============================================================*/
CREATE VIEW dbo.PEREPP
AS
SELECT     TOP 100 PERCENT EP.ID_RAZON,O.ID_RAZON AS RAZON ,EO.ID_COSTO AS
COSTO, DP.NUMERO, P.NOMBRES, P.APELLIDO1, P.APELLIDO2, ME.FECHA, ME.SEMANA,
ME.PERIODO, ME.ANO, EP.ID AS ID_EPP,EP.NOMBRE AS EPP,
EP.REFERENCIA,EE.CANT,EE.DESCRIPCION
FROM        MOV_EPP ME INNER JOIN
                  EMPLEADO E ON ME.ID_EMPLEADO = E.ID INNER JOIN
                  EPP_EMPLEADO EE ON  EE.ID_MOV = ME.ID INNER JOIN
```

```
                EPP EP ON EP.ID = EE.ID_EPP INNER JOIN
                PERSONA P ON E.ID_PERSONA = P.ID INNER JOIN
                DOCUMENTO_PERSONA DP ON DP.ID_PERSONA = P.ID INNER JOIN
                 EMPLEADO_COSTO EO ON EO.ID_EMPLEADO=E.ID INNER JOIN
                OFICIO O ON O.ID = E.ID_OFICIO
WHERE    (E.ESTADO = 'a' )
ORDER BY O.ID_RAZON, EO.ID_COSTO

/*============================================================*/
/* View: . PERFACTOR
/*============================================================*/
CREATE VIEW dbo.PERFACTOR
AS
SELECT    TOP 100 PERCENT O.ID_RAZON, EO.ID_COSTO, CC.NOMBRE AS COSTOS,
P.NOMBRES, P.APELLIDO1, P.APELLIDO2, P.TIPO_SANGRE + P.FACTOR_SANGRE  AS
SANGRE, P.DIRECCION, P.TELEFONO
FROM       dbo.PERSONA P INNER JOIN
             dbo.EMPLEADO E ON P.ID = E.ID_PERSONA INNER JOIN
             dbo.EMPLEADO_COSTO EO ON EO.ID_EMPLEADO =  E.ID INNER JOIN
             dbo.OFICIO O ON O.ID = E.ID_OFICIO INNER JOIN
             dbo.CENTRO_COSTO CC ON EO.ID_COSTO = CC.ID
ORDER BY O.ID_RAZON, EO.ID_COSTO

/*============================================================*/
/* View: . PERHIJOS
/*============================================================*/
CREATE VIEW dbo.PERHIJOS
AS
SELECT    F.ID_PERSONA, PE.NOMBRES+' '+PE.APELLIDO1 AS EMPLEADO, P.NOMBRES+'
'+P.APELLIDO1+' '+P.APELLIDO2 AS HIJO , P.SEXO,  P.FECHA_NACIMIENTO AS FECHA
FROM       dbo.COMPOSICION_FAMILIAR F INNER JOIN
             dbo.PERSONA P ON P.ID = F.ID_FAMILIA INNER JOIN
             dbo.PERSONA PE ON PE.ID = F.ID_PERSONA INNER JOIN
             dbo.EMPLEADO E ON F.ID_PERSONA = E.ID_PERSONA
WHERE    (F.ID_PARENTESCO = 5)

/*============================================================*/
/* View: . PERIODOEPP
/*============================================================*/
CREATE VIEW dbo.PERIODOEPP
AS
SELECT ID_RAZON,ANO,EPP,
   SUM(CASE PERIODO WHEN '1' THEN CANT ELSE 0 END) AS '1',
   SUM(CASE PERIODO WHEN '2' THEN CANT ELSE 0 END) AS '2',
   SUM(CASE PERIODO WHEN '3' THEN CANT ELSE 0 END) AS '3',
   SUM(CASE PERIODO WHEN '4' THEN CANT ELSE 0 END) AS '4',
   SUM(CASE PERIODO WHEN '5' THEN CANT ELSE 0 END) AS '5',
   SUM(CASE PERIODO WHEN '6' THEN CANT ELSE 0 END) AS '6',
   SUM(CASE PERIODO WHEN '7' THEN CANT ELSE 0 END) AS '7',
   SUM(CASE PERIODO WHEN '8' THEN CANT ELSE 0 END) AS '8',
   SUM(CASE PERIODO WHEN '9' THEN CANT ELSE 0 END) AS '9',
   SUM(CASE PERIODO WHEN '10' THEN CANT ELSE 0 END) AS '10',
   SUM(CASE PERIODO WHEN '11' THEN CANT ELSE 0 END) AS '11',
   SUM(CASE PERIODO WHEN '12' THEN CANT ELSE 0 END) AS '12',
   SUM(CASE PERIODO WHEN '13' THEN CANT ELSE 0 END) AS '13'
FROM dbo.PEREPP
/*============================================================*/
/* View: . PERSERVICIOS
```

```
/*============================================================*/
CREATE VIEW dbo.PERSERVICIOS
AS
SELECT     TOP 100 PERCENT O.ID_RAZON, EO.ID_COSTO, DP.NUMERO, P.NOMBRES,
P.APELLIDO1, P.APELLIDO2, P.DIRECCION, C.NOMBRE AS CIUDAD,
                  V.TENENCIA, V.DOCUMENTO, ES.LUZ, ES.GAS, ES.ACUEDUCTO_PUBLICO,
ES.ACUEDUCTO_COMUNAL, ES.CARROTANQUE, ES.POZO_ARTESANAL,
                  ES.PILA_PUBLICA, ES.RIO, ES.CONEXION_ALCANTARILLADO,
ES.POZO_SEPTICO, ES.TELEFONO
FROM         dbo.PERSONA P INNER JOIN
                    dbo.DOCUMENTO_PERSONA DP ON P.ID = DP.ID_PERSONA INNER JOIN
                    dbo.CIUDAD C ON P.CIUDAD_RESIDENCIA = C.ID INNER JOIN
                    dbo.EMPLEADO E ON P.ID = E.ID_PERSONA INNER JOIN
                    dbo.VIVIENDA V ON P.ID = V.ID_PERSONA INNER JOIN
                    dbo.ESTADOSERVICIOS ES ON ES.ID_VIVIENDA = V.ID INNER JOIN
                   dbo.EMPLEADO_COSTO EO ON EO.ID_EMPLEADO =  E.ID INNER JOIN
                    dbo.OFICIO O ON O.ID = E.ID_OFICIO
WHERE     (E.ESTADO = 'a')
ORDER BY O.ID_RAZON, EO.ID_COSTO

/*============================================================*/
/* View: . PERTALLAS
/*============================================================*/
CREATE VIEW dbo.PERTALLAS
AS
SELECT     TOP 100 PERCENT  O.ID_RAZON, EO.ID_COSTO, CC.NOMBRE AS COSTOS,
P.TALLA_CAMISA, P.TALLA_PANTALON, P.TALLA_ZAPATO
            FROM         dbo.PERSONA P INNER JOIN
                  dbo.EMPLEADO E ON P.ID = E.ID_PERSONA INNER JOIN
                  dbo.EMPLEADO_COSTO EO ON EO.ID_EMPLEADO =  E.ID INNER JOIN
                  dbo.OFICIO O ON O.ID = E.ID_OFICIO INNER JOIN
                  dbo.CENTRO_COSTO CC ON EO.ID_COSTO = CC.ID
ORDER BY O.ID_RAZON, EO.ID_COSTO

/*============================================================*/
/* View: . PERVACUNAS
/*============================================================*/
CREATE VIEW dbo.PERVACUNAS
AS
SELECT     TOP 100 PERCENT P.NOMBRES, P.APELLIDO1, P.APELLIDO2, EO.ID_COSTO,
O.ID_RAZON, RV.ID_VACUNA, RV.FECHA , V.NOMBRE,RV.ANO
FROM         dbo.PERSONA P INNER JOIN
                    dbo.EMPLEADO E ON P.ID = E.ID_PERSONA INNER JOIN
                    dbo.EMPLEADO_COSTO EO ON EO.ID_EMPLEADO =  E.ID INNER JOIN
                    dbo.OFICIO O ON O.ID = E.ID_OFICIO INNER JOIN
                    dbo.EMPLEADO_VACUNA EV ON EV.ID_EMPLEADO = E.ID INNER JOIN
                    dbo.REGISTER_VACUNAS RV ON EV.ID_REGISTRO = RV.ID INNER JOIN
                    dbo.VACUNAS V ON RV.ID_VACUNA = V.ID
ORDER BY P.NOMBRES




/*============================================================*/
```

```
/* View: . PERVIVIENDA
/*============================================================*/
CREATE VIEW dbo.PERVIVIENDA
AS
SELECT     TOP 100 PERCENT O.ID_RAZON, EO.ID_COSTO, DP.NUMERO, P.NOMBRES,
P.APELLIDO1, P.APELLIDO2, P.DIRECCION, C.NOMBRE AS CIUDAD,
                  V.TENENCIA, V.DOCUMENTO,EP.COCINA, EP.SANITARIO, EP.SALA, EP.PATIO,
EP.PISOS, EP.COMEDOR, EP.CUARTOS,EP.CANTIDAD
FROM         dbo.PERSONA P INNER JOIN
                    dbo.DOCUMENTO_PERSONA DP ON P.ID = DP.ID_PERSONA INNER JOIN
                    dbo.CIUDAD C ON P.CIUDAD_RESIDENCIA = C.ID INNER JOIN
                    dbo.EMPLEADO E ON P.ID = E.ID_PERSONA INNER JOIN
                    dbo.VIVIENDA V ON P.ID = V.ID_PERSONA INNER JOIN
                    dbo.ESTADOPARTES EP ON EP.ID_VIVIENDA = V.ID INNER JOIN
                   dbo.EMPLEADO_COSTO EO ON EO.ID_EMPLEADO =  E.ID INNER JOIN
                    dbo.OFICIO O ON O.ID = E.ID_OFICIO
WHERE     (E.ESTADO = 'a')
ORDER BY O.ID_RAZON, EO.ID_COSTO

/*============================================================*/
/* View: . TALLAS
/*============================================================*/
CREATE VIEW dbo.TALLAS
AS
SELECT ID_RAZON, COSTOS,
   COUNT(CASE TALLA_CAMISA WHEN 'L' THEN 1 END) AS L,
   COUNT(CASE TALLA_CAMISA WHEN 'M' THEN 1 END) AS M,
   COUNT(CASE TALLA_CAMISA WHEN 'S' THEN 1 END) AS S,
   COUNT(CASE TALLA_CAMISA WHEN 'XL' THEN 1 END) AS XL,
   COUNT(CASE TALLA_PANTALON WHEN '28' THEN 1 END) AS P28,
   COUNT(CASE TALLA_PANTALON WHEN '30' THEN 1 END) AS P30,
   COUNT(CASE TALLA_PANTALON WHEN '32' THEN 1 END) AS P32,
   COUNT(CASE TALLA_PANTALON WHEN '34' THEN 1 END) AS P34,
   COUNT(CASE TALLA_PANTALON WHEN '36' THEN 1 END) AS P36,
   COUNT(CASE TALLA_PANTALON WHEN '38' THEN 1 END) AS P38,
   COUNT(CASE TALLA_PANTALON WHEN '40' THEN 1 END) AS P40,
   COUNT(CASE TALLA_PANTALON WHEN '42' THEN 1 END) AS P42,
   COUNT(CASE TALLA_ZAPATO WHEN '35' THEN 1 END) AS Z35,
   COUNT(CASE TALLA_ZAPATO WHEN '36' THEN 1 END) AS Z36,
   COUNT(CASE TALLA_ZAPATO WHEN '37' THEN 1 END) AS Z37,
   COUNT(CASE TALLA_ZAPATO WHEN '38' THEN 1 END) AS Z38,
   COUNT(CASE TALLA_ZAPATO WHEN '39' THEN 1 END) AS Z39,
   COUNT(CASE TALLA_ZAPATO WHEN '40' THEN 1 END) AS Z40,
   COUNT(CASE TALLA_ZAPATO WHEN '41' THEN 1 END) AS Z41,
   COUNT(CASE TALLA_ZAPATO WHEN '42' THEN 1 END) AS Z42,
   COUNT(CASE TALLA_ZAPATO WHEN '43' THEN 1 END) AS Z43,
   COUNT(CASE TALLA_ZAPATO WHEN '44' THEN 1 END) AS Z44,
   COUNT(CASE TALLA_ZAPATO WHEN '45' THEN 1 END) AS Z45
FROM dbo.PERTALLAS
GROUP BY ID_RAZON, COSTOS




/*============================================================*/
```

```
/* View: . TCAPACITACION
/*==============================================================*/
CREATE VIEW dbo.TCAPACITACION
AS
SELECT     TOP 100 PERCENT C.ID_RAZON,C.PERIODO,C.SEMANA,COUNT(C.ID) AS
TOTAL,C.ID,C.LUGAR,C.CONFERENCISTA,C.HORA_INICIO,C.HORA_FINAL,C.SISTEMA,PR.ID
AS ID_PROCE,C.ID_AREA AS PROCE,PR.NOMBRE AS TEMAP,TL.ID_AREA,TL.NOMBRE AS
TEMAA,C.ID_TEMA,C.FECHA,C.HORAS,C.MINUTOS,C.ANO,C.NDIAS,C.ESTADO
FROM         REGISTRO R INNER JOIN
                      EMPLEADO E ON R.ID_EMPLEADO = E.ID INNER JOIN
                      CAPACITACION C ON R.ID_CAPACITACION = C.ID LEFT JOIN
                      TEMA_LIBRE TL ON C.ID_TEMA = TL.ID LEFT JOIN
                      PROCEDIMIENTO PR ON C.ID_PROCEDIMIENTO = PR.ID AND C.ID_AREA =
PR.ID_AREA
WHERE (E.ESTADO = 'a')
GROUP BY
C.ID_RAZON,C.PERIODO,C.SEMANA,C.CONFERENCISTA,PR.NOMBRE,C.FECHA,C.HORAS,
C.MINUTOS,C.ANO,C.LUGAR,C.ID,C.ESTADO,PR.ID,C.SISTEMA,TL.ID_AREA,C.HORA_INICIO,
C.NDIAS,C.ID_TEMA,TL.NOMBRE,C.HORA_FINAL,C.ID_AREA
```

6.3. PROCEDIMIENTOS

```
/*==============================================================*/
/* Procedure:  ACAPA
/*==============================================================*/
CREATE PROCEDURE ACAPA @CEDULA NVARCHAR(12), @RAZON NVARCHAR(4) AS
DECLARE @ATEMA NVARCHAR(64), @TEMA NVARCHAR(256),@FECHA DATETIME,
@HORAS NVARCHAR(12), @MINUTOS NVARCHAR(12),@CONT INT
DECLARE CUR CURSOR FOR SELECT DISTINCT TEMA FROM PERCAPACITACION WHERE
NUMERO=@CEDULA AND ID_RAZON=@RAZON AND TEMA IS NOT NULL
OPEN CUR
FETCH NEXT FROM CUR
INTO  @ATEMA
WHILE @@FETCH_STATUS = 0
BEGIN
DECLARE CUR1 CURSOR FOR SELECT TEMA, FECHA  FROM PERCAPACITACION WHERE
ID_RAZON=@RAZON AND NUMERO=@CEDULA AND TEMA=@ATEMA ORDER BY FECHA
DESC
OPEN CUR1
FETCH NEXT FROM CUR1
INTO @TEMA, @FECHA
SELECT @CONT = COUNT(*),  @HORAS = SUM(HORAS)+(SUM(MINUTOS)/60),@MINUTOS =
SUM(MINUTOS)-((SUM(MINUTOS)/60)*60) FROM PERCAPACITACION WHERE
ID_RAZON=@RAZON AND NUMERO=@CEDULA AND TEMA=@TEMA
IF LEN(@MINUTOS) = 1
SET @MINUTOS = '0' + @MINUTOS
IF @CONT > 1
INSERT INTO RCAPACITACION (TEMA,FECHA, DURACION,RECAPA) VALUES(@TEMA,
@FECHA, @HORAS + ':' + @MINUTOS,'1')
ELSE
INSERT INTO RCAPACITACION (TEMA,FECHA, DURACION,RECAPA) VALUES(@TEMA,
@FECHA, @HORAS + ':' + @MINUTOS,'0')
CLOSE CUR1
DEALLOCATE CUR1
FETCH NEXT FROM CUR
INTO @ATEMA
```

```
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure:  ACAPAC
/*============================================================*/
CREATE PROCEDURE ACAPAC @TEMA NVARCHAR(256), @RAZON NVARCHAR(4),
@COSTO NVARCHAR(2), @ANO NVARCHAR(4)=' ' AS
DECLARE @CEDULA NVARCHAR(12), @FECHA DATETIME, @HORAS NVARCHAR(12),
@MINUTOS NVARCHAR(12),@CONT INT,@NOMBRES NVARCHAR(32), @APELLIDOS
NVARCHAR(64)
IF @ANO=' '
DECLARE CUR CURSOR FOR SELECT DISTINCT NUMERO FROM PERCAPACITACION
WHERE TEMA=@TEMA AND ID_RAZON=@RAZON AND COSTO=@COSTO
ELSE
DECLARE CUR CURSOR FOR SELECT DISTINCT NUMERO FROM PERCAPACITACION
WHERE TEMA=@TEMA AND ID_RAZON=@RAZON AND COSTO=@COSTO AND ANO=@ANO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @CEDULA
WHILE @@FETCH_STATUS = 0
BEGIN
IF @ANO=' '
DECLARE CUR1 CURSOR FOR SELECT NOMBRES, APELLIDO1 + ' ' + APELLIDO2, FECHA
FROM PERCAPACITACION WHERE ID_RAZON=@RAZON AND COSTO=@COSTO AND
NUMERO=@CEDULA AND TEMA=@TEMA ORDER BY FECHA DESC
ELSE
DECLARE CUR1 CURSOR FOR SELECT NOMBRES, APELLIDO1 + ' ' + APELLIDO2, FECHA
FROM PERCAPACITACION WHERE ID_RAZON=@RAZON AND COSTO=@COSTO AND
NUMERO=@CEDULA AND TEMA=@TEMA AND ANO=@ANO ORDER BY FECHA DESC
OPEN CUR1
FETCH NEXT FROM CUR1
INTO @NOMBRES,@APELLIDOS, @FECHA
IF @ANO=' '
SELECT @CONT = COUNT(*),  @HORAS = SUM(HORAS)+(SUM(MINUTOS)/60),@MINUTOS =
SUM(MINUTOS)-((SUM(MINUTOS)/60)*60) FROM PERCAPACITACION WHERE
ID_RAZON=@RAZON AND NUMERO=@CEDULA AND TEMA=@TEMA
ELSE
SELECT @CONT = COUNT(*),  @HORAS = SUM(HORAS)+(SUM(MINUTOS)/60),@MINUTOS =
SUM(MINUTOS)-((SUM(MINUTOS)/60)*60) FROM PERCAPACITACION WHERE
ID_RAZON=@RAZON AND NUMERO=@CEDULA AND TEMA=@TEMA AND ANO=@ANO
IF LEN(@MINUTOS) = 1
SET @MINUTOS = '0' + @MINUTOS
IF @CONT > 1
INSERT INTO RCAPACITACION (CEDULA,NOMBRES,APELLIDOS,FECHA,
DURACION,RECAPA) VALUES(@CEDULA,@NOMBRES,@APELLIDOS, @FECHA, @HORAS +
':' + @MINUTOS,'1')
ELSE
INSERT INTO RCAPACITACION (CEDULA,NOMBRES,APELLIDOS,FECHA,
DURACION,RECAPA) VALUES(@CEDULA,@NOMBRES,@APELLIDOS, @FECHA, @HORAS +
':' + @MINUTOS,'0')
CLOSE CUR1
DEALLOCATE CUR1
FETCH NEXT FROM CUR
INTO @CEDULA
END
```

```
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure:  ACAPAR
/*============================================================*/
CREATE PROCEDURE ACAPAR @TEMA NVARCHAR(256), @RAZON NVARCHAR(4), @ANO
NVARCHAR(4)=' ' AS
DECLARE @CEDULA NVARCHAR(12), @FECHA DATETIME, @HORAS NVARCHAR(12),
@MINUTOS NVARCHAR(12),@CONT INT,@NOMBRES NVARCHAR(32), @APELLIDOS
NVARCHAR(64)
IF @ANO=' '
DECLARE CUR CURSOR FOR SELECT DISTINCT NUMERO FROM PERCAPACITACION
WHERE TEMA=@TEMA AND ID_RAZON=@RAZON
ELSE
DECLARE CUR CURSOR FOR SELECT DISTINCT NUMERO FROM PERCAPACITACION
WHERE TEMA=@TEMA AND ID_RAZON=@RAZON AND ANO=@ANO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @CEDULA
WHILE @@FETCH_STATUS = 0
BEGIN
IF @ANO=' '
DECLARE CUR1 CURSOR FOR SELECT NOMBRES, APELLIDO1 + ' ' + APELLIDO2, FECHA
FROM PERCAPACITACION WHERE ID_RAZON=@RAZON AND NUMERO=@CEDULA AND
TEMA=@TEMA ORDER BY FECHA DESC
ELSE
DECLARE CUR1 CURSOR FOR SELECT NOMBRES, APELLIDO1 + ' ' + APELLIDO2, FECHA
FROM PERCAPACITACION WHERE ID_RAZON=@RAZON AND NUMERO=@CEDULA AND
TEMA=@TEMA AND ANO=@ANO ORDER BY FECHA DESC
OPEN CUR1
FETCH NEXT FROM CUR1
INTO @NOMBRES,@APELLIDOS, @FECHA
IF @ANO=' '
SELECT @CONT = COUNT(*),  @HORAS = SUM(HORAS)+(SUM(MINUTOS)/60),@MINUTOS =
SUM(MINUTOS)-((SUM(MINUTOS)/60)*60) FROM PERCAPACITACION WHERE
ID_RAZON=@RAZON AND NUMERO=@CEDULA AND TEMA=@TEMA
ELSE
SELECT @CONT = COUNT(*),  @HORAS = SUM(HORAS)+(SUM(MINUTOS)/60),@MINUTOS =
SUM(MINUTOS)-((SUM(MINUTOS)/60)*60) FROM PERCAPACITACION WHERE
ID_RAZON=@RAZON AND NUMERO=@CEDULA AND TEMA=@TEMA AND ANO=@ANO
IF LEN(@MINUTOS) = 1
SET @MINUTOS = '0' + @MINUTOS
IF @CONT > 1
INSERT INTO RCAPACITACION (CEDULA,NOMBRES,APELLIDOS,FECHA,
DURACION,RECAPA) VALUES(@CEDULA,@NOMBRES,@APELLIDOS, @FECHA, @HORAS +
':' + @MINUTOS,'1')
ELSE
INSERT INTO RCAPACITACION (CEDULA,NOMBRES,APELLIDOS,FECHA,
DURACION,RECAPA) VALUES(@CEDULA,@NOMBRES,@APELLIDOS, @FECHA, @HORAS +
':' + @MINUTOS,'0')
CLOSE CUR1
DEALLOCATE CUR1
FETCH NEXT FROM CUR
INTO @CEDULA
END
CLOSE CUR
```

```
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: AGEC
/*============================================================*/
CREATE PROCEDURE AGEC @RAZON NVARCHAR(4), @COSTO NVARCHAR(2) AS
DECLARE @MES NVARCHAR(6), @AGE NVARCHAR(2), @MONTF INT, @MONT INT, @DAYF
INT, @DAY INT,@EMPLOYE NVARCHAR(50), @NHIJO CHAR(2), @HIJO NVARCHAR(50),
@SEXO CHAR(1)
SET @MONT = DATEPART(MONTH,GETDATE())
SET @DAY  = DATEPART(DAY, GETDATE())
DECLARE CUR CURSOR FOR SELECT  DATEDIFF(YEAR, FECHA, GETDATE()) ,
DATEPART(MONTH,FECHA) ,  DATEPART(DAY,FECHA), EMPLEADO, NHIJO, HIJO, SEXO
FROM HIJOS WHERE ID_RAZON=@RAZON AND ID_COSTO=@COSTO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @AGE, @MONTF, @DAYF, @EMPLOYE, @NHIJO, @HIJO, @SEXO
WHILE @@FETCH_STATUS = 0
BEGIN
IF @MONTF < @MONT
SET @MES = @MONT - @MONTF
IF @MONTF =  @MONT
BEGIN
IF @DAY < @DAYF
BEGIN
SET @MES = 11
SET @AGE = @AGE - 1
END
ELSE
SET @MES = 0
END
IF @MONT< @MONTF
BEGIN
SET @AGE = @AGE -1
SET @MES = (12-@MONTF) + @MONT
END
IF @DAY < @DAYF
SET @MES = @MES - 1
IF LEN(@MES) = 1
SET @MES = '0' + @MES
SET @MES =  @AGE+'.'+ @MES
INSERT INTO EDADHIJOS (EMPLEADO, NHIJO, HIJO, SEXO, EDAD) VALUES(@EMPLOYE,
@NHIJO, @HIJO, @SEXO,@MES)
FETCH NEXT FROM CUR
INTO  @AGE, @MONTF, @DAYF, @EMPLOYE, @NHIJO, @HIJO, @SEXO
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . AGEFC
```

```
/*============================================================*/
CREATE PROCEDURE AGEFC @RAZON NVARCHAR(4), @COSTO NVARCHAR(2), @FECHA
NVARCHAR(12) AS
DECLARE @MES NVARCHAR(6), @AGE NVARCHAR(2), @MONTF INT, @MONT INT, @DAYF
INT, @DAY INT,@EMPLOYE NVARCHAR(50), @NHIJO CHAR(2), @HIJO NVARCHAR(50),
@SEXO CHAR(1)
SET @MONT = DATEPART(MONTH,@FECHA)
SET @DAY  = DATEPART(DAY, @FECHA)
DECLARE CUR CURSOR FOR SELECT  DATEDIFF(YEAR, FECHA, @FECHA) ,
DATEPART(MONTH,FECHA) ,  DATEPART(DAY,FECHA), EMPLEADO, NHIJO, HIJO, SEXO
FROM HIJOS WHERE ID_RAZON=@RAZON AND ID_COSTO=@COSTO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @AGE, @MONTF, @DAYF, @EMPLOYE, @NHIJO, @HIJO, @SEXO
WHILE @@FETCH_STATUS = 0
BEGIN
IF @MONTF < @MONT
SET @MES = @MONT - @MONTF
IF @MONTF =  @MONT
BEGIN
IF @DAY < @DAYF
BEGIN
SET @MES = 11
SET @AGE = @AGE - 1
END
ELSE
SET @MES = 0
END
IF @MONT< @MONTF
BEGIN
SET @AGE = @AGE -1
SET @MES = (12-@MONTF) + @MONT
END
IF @DAY < @DAYF
SET @MES = @MES - 1
IF LEN(@MES) = 1
SET @MES = '0' + @MES
SET @MES = @AGE+'.'+ @MES
INSERT INTO EDADHIJOS (EMPLEADO, NHIJO, HIJO, SEXO, EDAD) VALUES(@EMPLOYE,
@NHIJO, @HIJO, @SEXO,@MES)
FETCH NEXT FROM CUR
INTO  @AGE, @MONTF, @DAYF, @EMPLOYE, @NHIJO, @HIJO, @SEXO
END
CLOSE CUR
DEALLOCATE CUR
GO
```

```
/*============================================================*/
```

```
/* Procedure: . AGEFR
/*============================================================*/
CREATE PROCEDURE AGEFR @RAZON NVARCHAR(4), @FECHA NVARCHAR(12) AS
DECLARE @MES NVARCHAR(6), @AGE NVARCHAR(2), @MONTF INT, @MONT INT, @DAYF
INT, @DAY INT,@EMPLOYE NVARCHAR(50), @NHIJO CHAR(2), @HIJO NVARCHAR(50),
@SEXO CHAR(1)
SET @MONT = DATEPART(MONTH,@FECHA)
SET @DAY  = DATEPART(DAY, @FECHA)
DECLARE CUR CURSOR FOR SELECT  DATEDIFF(YEAR, FECHA, @FECHA) ,
DATEPART(MONTH,FECHA) ,  DATEPART(DAY,FECHA), EMPLEADO, NHIJO, HIJO, SEXO
FROM HIJOS WHERE ID_RAZON=@RAZON
OPEN CUR
FETCH NEXT FROM CUR
INTO  @AGE, @MONTF, @DAYF, @EMPLOYE, @NHIJO, @HIJO, @SEXO
WHILE @@FETCH_STATUS = 0
BEGIN
IF @MONTF < @MONT
SET @MES = @MONT - @MONTF
IF @MONTF =  @MONT
BEGIN
IF @DAY < @DAYF
BEGIN
SET @MES = 11
SET @AGE = @AGE - 1
END
ELSE
SET @MES = 0
END
IF @MONT< @MONTF
BEGIN
SET @AGE = @AGE -1
SET @MES = (12-@MONTF) + @MONT
END
IF @DAY < @DAYF
SET @MES = @MES - 1
IF LEN(@MES) = 1
SET @MES = '0' + @MES
SET @MES = @AGE+'.'+ @MES
INSERT INTO EDADHIJOS (EMPLEADO, NHIJO, HIJO, SEXO, EDAD) VALUES(@EMPLOYE,
@NHIJO, @HIJO, @SEXO,@MES)
FETCH NEXT FROM CUR
INTO  @AGE, @MONTF, @DAYF, @EMPLOYE, @NHIJO, @HIJO, @SEXO
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
```

```
/* Procedure: . AGER
/*============================================================*/
CREATE PROCEDURE AGER @RAZON NVARCHAR(4) AS
DECLARE @MES NVARCHAR(6), @AGE NVARCHAR(2), @MONTF INT, @MONT INT, @DAYF
INT, @DAY INT,@EMPLOYE NVARCHAR(50), @NHIJO CHAR(2), @HIJO NVARCHAR(50),
@SEXO CHAR(1)
SET @MONT = DATEPART(MONTH,GETDATE())
SET @DAY  = DATEPART(DAY, GETDATE())
DECLARE CUR CURSOR FOR SELECT  DATEDIFF(YEAR, FECHA, GETDATE()) ,
DATEPART(MONTH,FECHA) ,  DATEPART(DAY,FECHA), EMPLEADO, NHIJO, HIJO, SEXO
FROM HIJOS WHERE ID_RAZON=@RAZON
OPEN CUR
FETCH NEXT FROM CUR
INTO  @AGE, @MONTF, @DAYF, @EMPLOYE, @NHIJO, @HIJO, @SEXO
WHILE @@FETCH_STATUS = 0
BEGIN
IF @MONTF < @MONT
SET @MES = @MONT - @MONTF
IF @MONTF =  @MONT
BEGIN
IF @DAY < @DAYF
BEGIN
SET @MES = 11
SET @AGE = @AGE - 1
END
ELSE
SET @MES = 0
END
IF @MONT< @MONTF
BEGIN
SET @AGE = @AGE -1
SET @MES = (12-@MONTF) + @MONT
END
IF @DAY < @DAYF
SET @MES = @MES - 1
IF LEN(@MES) = 1
SET @MES = '0' + @MES
SET @MES =  @AGE+'.'+ @MES
INSERT INTO EDADHIJOS (EMPLEADO, NHIJO, HIJO, SEXO, EDAD) VALUES(@EMPLOYE,
@NHIJO, @HIJO, @SEXO,@MES)
FETCH NEXT FROM CUR
INTO  @AGE, @MONTF, @DAYF, @EMPLOYE, @NHIJO, @HIJO, @SEXO
END
CLOSE CUR
DEALLOCATE CUR
GO
```

```
/*============================================================*/
```

```
/* Procedure: . CCAPA
/*============================================================*/
CREATE PROCEDURE CCAPA @RAZON NVARCHAR(4), @ANO NVARCHAR(4)=' ' AS
DECLARE @PERIODO NVARCHAR(4), @CAPA INT, @REAL INT
IF @ANO=' '
DECLARE CUR CURSOR FOR SELECT DISTINCT ANO AS PERIODO,COUNT(ID) AS CAPA
FROM TCAPACITACION WHERE ID_RAZON=@RAZON AND ESTADO='P' GROUP BY ANO
ELSE
DECLARE CUR CURSOR FOR SELECT DISTINCT PERIODO,COUNT(ID) AS CAPA  FROM
TCAPACITACION WHERE ID_RAZON=@RAZON AND ANO=@ANO AND ESTADO='P' GROUP
BY PERIODO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @PERIODO,@CAPA
WHILE @@FETCH_STATUS = 0
BEGIN
IF @ANO=' '
BEGIN
SELECT @REAL=COUNT(ID) FROM TCAPACITACION WHERE ESTADO='R' AND
PERIODO=@PERIODO AND ID_RAZON=@RAZON
INSERT INTO TOTALC (ANO,TCAPA) VALUES(@PERIODO,(@CAPA*100)/@REAL)
END
ELSE
BEGIN
SELECT @REAL=COUNT(ID) FROM TCAPACITACION WHERE ESTADO='R' AND
PERIODO=@PERIODO AND ID_RAZON=@RAZON AND ANO=@ANO
INSERT INTO TOTALC (PERIODO,TCAPA) VALUES(@PERIODO,(@CAPA*100)/@REAL)
END
FETCH NEXT FROM CUR
INTO  @PERIODO,@CAPA
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . FCAPA
/*============================================================*/
CREATE PROCEDURE FCAPA @FECHA1 DATETIME, @FECHA2 DATETIME, @RAZON
NVARCHAR(4) AS
DECLARE @TEMAP NVARCHAR(256), @TEMAA NVARCHAR(256), @LUGAR NVARCHAR(64),
@CONFERENCISTA NVARCHAR(128),@FECHA DATETIME, @HORAS NVARCHAR(12),
@MINUTOS NVARCHAR(12),@TOTAL INT
DECLARE CUR CURSOR FOR SELECT TEMAP,
TEMAA,LUGAR,CONFERENCISTA,FECHA,HORAS,MINUTOS,TOTAL FROM TCAPACITACION
WHERE ID_RAZON=@RAZON AND FECHA>=@FECHA1 AND FECHA<=@FECHA2 AND
ESTADO!='P'
OPEN CUR
FETCH NEXT FROM CUR
INTO  @TEMAP,@TEMAA, @LUGAR, @CONFERENCISTA, @FECHA, @HORAS,
@MINUTOS,@TOTAL
WHILE @@FETCH_STATUS = 0
BEGIN
IF @TEMAP = NULL
INSERT INTO TOTALC (TEMA,LUGAR,CONFERENCISTA,FECHA,HORAS,MINUTOS,TCAPA)
VALUES(@TEMAA,@LUGAR,@CONFERENCISTA,@FECHA,@HORAS,@MINUTOS,@TOTAL)
ELSE
```

```
INSERT INTO TOTALC (TEMA,LUGAR,CONFERENCISTA,FECHA,HORAS,MINUTOS,TCAPA)
VALUES(@TEMAP,@LUGAR,@CONFERENCISTA,@FECHA,@HORAS,@MINUTOS,@TOTAL)
FETCH NEXT FROM CUR
INTO  @TEMAP,@TEMAA, @LUGAR, @CONFERENCISTA, @FECHA, @HORAS,
@MINUTOS,@TOTAL
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . FOCAPA
/*============================================================*/
CREATE PROCEDURE FOCAPA @RAZON NVARCHAR(4), @ANO NVARCHAR(4)=' ' AS
DECLARE @PERIODO NVARCHAR(4), @HORAS INT
IF @ANO=' '
DECLARE CUR CURSOR FOR SELECT DISTINCT ANO AS
PERIODO,(SUM(HORAS)+(SUM(MINUTOS)/60))*SUM(TOTAL) AS HORAS  FROM
TCAPACITACION WHERE ID_RAZON=@RAZON AND ESTADO<>'P' GROUP BY ANO
ELSE
DECLARE CUR CURSOR FOR SELECT DISTINCT
PERIODO,(SUM(HORAS)+(SUM(MINUTOS)/60))*SUM(TOTAL) AS HORAS  FROM
TCAPACITACION WHERE ID_RAZON=@RAZON AND ANO=@ANO AND ESTADO<>'P' GROUP
BY PERIODO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @PERIODO,@HORAS
WHILE @@FETCH_STATUS = 0
BEGIN
IF @ANO=' '
INSERT INTO TOTALC (ANO,HORAS) VALUES(@PERIODO,@HORAS)
ELSE
INSERT INTO TOTALC (PERIODO,HORAS) VALUES(@PERIODO,@HORAS)
FETCH NEXT FROM CUR
INTO  @PERIODO,@HORAS
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . FPROG
/*============================================================*/
CREATE PROCEDURE FPROG @FECHA1 DATETIME, @FECHA2 DATETIME, @RAZON
NVARCHAR(4) AS
DECLARE @TEMAP NVARCHAR(256), @TEMAA NVARCHAR(256), @LUGAR NVARCHAR(64),
@CONFERENCISTA NVARCHAR(128),@FECHA DATETIME, @HORAS NVARCHAR(12),
@MINUTOS NVARCHAR(12),@TOTAL INT,@PERIODO INT,@SEMANA INT
DECLARE CUR CURSOR FOR SELECT TEMAP,
TEMAA,PERIODO,SEMANA,LUGAR,CONFERENCISTA,FECHA,HORAS,MINUTOS,TOTAL
FROM TCAPACITACION WHERE ID_RAZON=@RAZON AND FECHA>=@FECHA1 AND
FECHA<=@FECHA2 AND ESTADO='P'
OPEN CUR
FETCH NEXT FROM CUR
INTO  @TEMAP,@TEMAA, @PERIODO,@SEMANA,@LUGAR, @CONFERENCISTA, @FECHA,
@HORAS, @MINUTOS,@TOTAL
WHILE @@FETCH_STATUS = 0
BEGIN
```

```
IF @TEMAP = NULL
INSERT INTO TOTALC
(TEMA,PERIODO,SEMANA,LUGAR,CONFERENCISTA,FECHA,HORAS,MINUTOS,TCAPA)
VALUES(@TEMAA,@PERIODO,@SEMANA,@LUGAR,@CONFERENCISTA,@FECHA,@HORA
S,@MINUTOS,@TOTAL)
ELSE
INSERT INTO TOTALC
(TEMA,PERIODO,SEMANA,LUGAR,CONFERENCISTA,FECHA,HORAS,MINUTOS,TCAPA)
VALUES(@TEMAP,@PERIODO,@SEMANA,@LUGAR,@CONFERENCISTA,@FECHA,@HORA
S,@MINUTOS,@TOTAL)
FETCH NEXT FROM CUR
INTO  @TEMAP,@TEMAA,@PERIODO,@SEMANA, @LUGAR, @CONFERENCISTA, @FECHA,
@HORAS, @MINUTOS,@TOTAL
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . HCAPA
/*============================================================*/
CREATE PROCEDURE HCAPA @RAZON NVARCHAR(4), @ANO NVARCHAR(4)=' ' AS
DECLARE @PERIODO NVARCHAR(4), @HORAS INT
IF @ANO=' '
DECLARE CUR CURSOR FOR SELECT DISTINCT ANO AS
PERIODO,SUM(HORAS)+(SUM(MINUTOS)/60) AS HORAS  FROM TCAPACITACION WHERE
ID_RAZON=@RAZON AND ESTADO<>'P' GROUP BY ANO
ELSE
DECLARE CUR CURSOR FOR SELECT DISTINCT
PERIODO,SUM(HORAS)+(SUM(MINUTOS)/60) AS HORAS  FROM TCAPACITACION WHERE
ID_RAZON=@RAZON AND ANO=@ANO AND ESTADO<>'P' GROUP BY PERIODO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @PERIODO,@HORAS
WHILE @@FETCH_STATUS = 0
BEGIN
IF @ANO=' '
INSERT INTO TOTALC (ANO,HORAS) VALUES(@PERIODO,@HORAS)
ELSE
INSERT INTO TOTALC (PERIODO,HORAS) VALUES(@PERIODO,@HORAS)
FETCH NEXT FROM CUR
INTO  @PERIODO,@HORAS
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . PCAPA
/*============================================================*/
CREATE PROCEDURE PCAPA @CEDULA NVARCHAR(12), @RAZON NVARCHAR(4) AS
DECLARE @PTEMA NVARCHAR(64), @TEMA NVARCHAR(256),@FECHA DATETIME,
@HORAS NVARCHAR(12), @MINUTOS NVARCHAR(12),@CONT INT
DECLARE CUR CURSOR FOR SELECT DISTINCT TEMAPROCE FROM PERCAPACITACION
WHERE NUMERO=@CEDULA AND ID_RAZON=@RAZON AND TEMAPROCE IS NOT NULL
OPEN CUR
FETCH NEXT FROM CUR
INTO  @PTEMA
```

```
WHILE @@FETCH_STATUS = 0
BEGIN
DECLARE CUR1 CURSOR FOR SELECT TEMAPROCE, FECHA FROM PERCAPACITACION
WHERE ID_RAZON=@RAZON AND NUMERO=@CEDULA AND TEMAPROCE=@PTEMA
ORDER BY FECHA DESC
OPEN CUR1
FETCH NEXT FROM CUR1
INTO @TEMA, @FECHA
SELECT @CONT = COUNT(*), @HORAS = SUM(HORAS)+(SUM(MINUTOS)/60), @MINUTOS =
SUM(MINUTOS)-((SUM(MINUTOS)/60)*60)  FROM PERCAPACITACION WHERE
ID_RAZON=@RAZON AND NUMERO=@CEDULA AND TEMAPROCE=@TEMA
IF LEN(@MINUTOS) = 1
SET @MINUTOS = '0' + @MINUTOS
IF @CONT > 1
INSERT INTO RCAPACITACION (TEMA,FECHA, DURACION,RECAPA) VALUES( @TEMA,
@FECHA, @HORAS + ':' + @MINUTOS,'1')
ELSE
INSERT INTO RCAPACITACION (TEMA,FECHA, DURACION,RECAPA) VALUES( @TEMA,
@FECHA, @HORAS + ':' + @MINUTOS,'0')
CLOSE CUR1
DEALLOCATE CUR1
FETCH NEXT FROM CUR
INTO @PTEMA
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . PCAPAC
/*============================================================*/
CREATE PROCEDURE PCAPAC @PROCE NVARCHAR(8), @RAZON NVARCHAR(4),
@COSTO NVARCHAR(2), @ANO NVARCHAR(4)=' ' AS
DECLARE @CEDULA NVARCHAR(12), @FECHA DATETIME, @HORAS NVARCHAR(12),
@MINUTOS NVARCHAR(12),@CONT INT,@NOMBRES NVARCHAR(32), @APELLIDOS
NVARCHAR(64)
IF @ANO=' '
DECLARE CUR CURSOR FOR SELECT DISTINCT NUMERO FROM PERCAPACITACION
WHERE PROCEDIMIENTO=@PROCE AND ID_RAZON=@RAZON AND COSTO=@COSTO
ELSE
DECLARE CUR CURSOR FOR SELECT DISTINCT NUMERO FROM PERCAPACITACION
WHERE PROCEDIMIENTO=@PROCE AND ID_RAZON=@RAZON AND COSTO=@COSTO AND
ANO=@ANO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @CEDULA
WHILE @@FETCH_STATUS = 0
BEGIN
IF @ANO=' '
DECLARE CUR1 CURSOR FOR SELECT NOMBRES, APELLIDO1 + ' ' + APELLIDO2, FECHA
FROM PERCAPACITACION WHERE ID_RAZON=@RAZON AND COSTO=@COSTO AND
NUMERO=@CEDULA AND PROCEDIMIENTO=@PROCE ORDER BY FECHA DESC
ELSE
DECLARE CUR1 CURSOR FOR SELECT NOMBRES, APELLIDO1 + ' ' + APELLIDO2, FECHA
FROM PERCAPACITACION WHERE ID_RAZON=@RAZON AND COSTO=@COSTO AND
NUMERO=@CEDULA AND PROCEDIMIENTO=@PROCE AND ANO=@ANO ORDER BY FECHA
DESC
OPEN CUR1
```

```
FETCH NEXT FROM CUR1
INTO @NOMBRES,@APELLIDOS, @FECHA
IF @ANO=' '
SELECT @CONT = COUNT(*),  @HORAS = SUM(HORAS)+(SUM(MINUTOS)/60),@MINUTOS =
SUM(MINUTOS)-((SUM(MINUTOS)/60)*60) FROM PERCAPACITACION WHERE
ID_RAZON=@RAZON AND NUMERO=@CEDULA AND PROCEDIMIENTO=@PROCE
ELSE
SELECT @CONT = COUNT(*),  @HORAS = SUM(HORAS)+(SUM(MINUTOS)/60),@MINUTOS =
SUM(MINUTOS)-((SUM(MINUTOS)/60)*60) FROM PERCAPACITACION WHERE
ID_RAZON=@RAZON AND NUMERO=@CEDULA AND PROCEDIMIENTO=@PROCE AND
ANO=@ANO
IF LEN(@MINUTOS) = 1
SET @MINUTOS = '0' + @MINUTOS
IF @CONT > 1
INSERT INTO RCAPACITACION (CEDULA,NOMBRES,APELLIDOS,FECHA,
DURACION,RECAPA) VALUES(@CEDULA,@NOMBRES,@APELLIDOS, @FECHA, @HORAS +
':' + @MINUTOS,'1')
ELSE
INSERT INTO RCAPACITACION (CEDULA,NOMBRES,APELLIDOS,FECHA,
DURACION,RECAPA) VALUES(@CEDULA,@NOMBRES,@APELLIDOS, @FECHA, @HORAS +
':' + @MINUTOS,'0')
CLOSE CUR1
DEALLOCATE CUR1
FETCH NEXT FROM CUR
INTO @CEDULA
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . PCAPAR
/*============================================================*/
CREATE PROCEDURE PCAPAR @PROCE NVARCHAR(8), @RAZON NVARCHAR(4), @ANO
NVARCHAR(4)=' ' AS
DECLARE @CEDULA NVARCHAR(12), @FECHA DATETIME, @HORAS NVARCHAR(12),
@MINUTOS NVARCHAR(12),@CONT INT,@NOMBRES NVARCHAR(32), @APELLIDOS
NVARCHAR(64)
IF @ANO=' '
DECLARE CUR CURSOR FOR SELECT DISTINCT NUMERO FROM PERCAPACITACION
WHERE PROCEDIMIENTO=@PROCE AND ID_RAZON=@RAZON
ELSE
DECLARE CUR CURSOR FOR SELECT DISTINCT NUMERO FROM PERCAPACITACION
WHERE PROCEDIMIENTO=@PROCE AND ID_RAZON=@RAZON AND ANO=@ANO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @CEDULA
WHILE @@FETCH_STATUS = 0
BEGIN
IF @ANO=' '
DECLARE CUR1 CURSOR FOR SELECT NOMBRES, APELLIDO1 + ' ' + APELLIDO2, FECHA
FROM PERCAPACITACION WHERE ID_RAZON=@RAZON AND NUMERO=@CEDULA AND
PROCEDIMIENTO=@PROCE ORDER BY FECHA DESC
ELSE
DECLARE CUR1 CURSOR FOR SELECT NOMBRES, APELLIDO1 + ' ' + APELLIDO2, FECHA
FROM PERCAPACITACION WHERE ID_RAZON=@RAZON AND NUMERO=@CEDULA AND
PROCEDIMIENTO=@PROCE AND ANO=@ANO ORDER BY FECHA DESC
OPEN CUR1
```

```
FETCH NEXT FROM CUR1
INTO @NOMBRES,@APELLIDOS, @FECHA
IF @ANO=' '
SELECT @CONT = COUNT(*),  @HORAS = SUM(HORAS)+(SUM(MINUTOS)/60),@MINUTOS =
SUM(MINUTOS)-((SUM(MINUTOS)/60)*60) FROM PERCAPACITACION WHERE
ID_RAZON=@RAZON AND NUMERO=@CEDULA AND PROCEDIMIENTO=@PROCE
ELSE
SELECT @CONT = COUNT(*),  @HORAS = SUM(HORAS)+(SUM(MINUTOS)/60),@MINUTOS =
SUM(MINUTOS)-((SUM(MINUTOS)/60)*60) FROM PERCAPACITACION WHERE
ID_RAZON=@RAZON AND NUMERO=@CEDULA AND PROCEDIMIENTO=@PROCE AND
ANO=@ANO
IF LEN(@MINUTOS) = 1
SET @MINUTOS = '0' + @MINUTOS
IF @CONT > 1
INSERT INTO RCAPACITACION (CEDULA,NOMBRES,APELLIDOS,FECHA,
DURACION,RECAPA) VALUES(@CEDULA,@NOMBRES,@APELLIDOS, @FECHA, @HORAS +
':' + @MINUTOS,'1')
ELSE
INSERT INTO RCAPACITACION (CEDULA,NOMBRES,APELLIDOS,FECHA,
DURACION,RECAPA) VALUES(@CEDULA,@NOMBRES,@APELLIDOS, @FECHA, @HORAS +
':' + @MINUTOS,'0')
CLOSE CUR1
DEALLOCATE CUR1
FETCH NEXT FROM CUR
INTO @CEDULA
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . RANGOC
/*============================================================*/
CREATE PROCEDURE RANGOC @RAZON NVARCHAR(4), @COSTO NVARCHAR(2) AS
DECLARE @TIME INT, @MONTF INT, @MONT INT, @DAYF INT, @DAY INT
SET @MONT = DATEPART(MONTH,GETDATE())
SET @DAY  = DATEPART(DAY, GETDATE())
DECLARE CUR CURSOR FOR SELECT  DATEDIFF(YEAR, FECHA, GETDATE()) ,
DATEPART(MONTH,FECHA) ,  DATEPART(DAY,FECHA)  FROM PEREMPLEADO WHERE
ID_RAZON=@RAZON AND ID_COSTO=@COSTO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @TIME, @MONTF, @DAYF
WHILE @@FETCH_STATUS = 0
BEGIN
IF @TIME<>0
BEGIN
IF @MONT<@MONTF
SET @TIME= @TIME -1
IF @MONT =  @MONTF
BEGIN
IF @DAY < @DAYF
SET @TIME = @TIME-1
END
END
IF @TIME=0
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('0-1', 1)
IF @TIME>=1 AND @TIME<3
```

```
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('1-3', 2)
IF @TIME>=3 AND @TIME<5
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('3-5', 3)
IF @TIME>=5 AND @TIME<10
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('5-10', 4)
IF @TIME>=10 AND @TIME<15
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('10-15', 5)
IF @TIME>=15 AND @TIME<20
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('15-20', 6)
IF @TIME>=20 AND @TIME<25
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('20-25', 7)
IF @TIME>=25 AND @TIME<30
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('25-30', 8)
IF @TIME>=30 AND @TIME<35
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('30-35', 9)
IF @TIME>=35 AND @TIME<40
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('35-40', 10)
IF @TIME>=40 AND @TIME<45
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('40-45', 11)
IF @TIME>=45 AND @TIME<50
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('45-50', 12)
IF @TIME>=50
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('>50', 13)
FETCH NEXT FROM CUR
INTO  @TIME, @MONTF, @DAYF
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . RANGOR
/*============================================================*/
CREATE PROCEDURE RANGOR @RAZON NVARCHAR(4) AS
DECLARE @TIME INT, @MONTF INT, @MONT INT, @DAYF INT, @DAY INT
SET @MONT = DATEPART(MONTH,GETDATE())
SET @DAY  = DATEPART(DAY, GETDATE())
DECLARE CUR CURSOR FOR SELECT  DATEDIFF(YEAR, FECHA, GETDATE()) ,
DATEPART(MONTH,FECHA) ,  DATEPART(DAY,FECHA)  FROM PEREMPLEADO WHERE
ID_RAZON=@RAZON
OPEN CUR
FETCH NEXT FROM CUR
INTO  @TIME, @MONTF, @DAYF
WHILE @@FETCH_STATUS = 0
BEGIN
IF @TIME<>0
BEGIN
IF @MONT<@MONTF
SET @TIME= @TIME -1
IF @MONT =  @MONTF
BEGIN
IF @DAY < @DAYF
SET @TIME = @TIME-1
END
END
IF @TIME=0
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('0-1', 1)
IF @TIME>=1 AND @TIME<3
```

```
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('1-3', 2)
IF @TIME>=3 AND @TIME<5
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('3-5', 3)
IF @TIME>=5 AND @TIME<10
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('5-10', 4)
IF @TIME>=10 AND @TIME<15
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('10-15', 5)
IF @TIME>=15 AND @TIME<20
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('15-20', 6)
IF @TIME>=20 AND @TIME<25
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('20-25', 7)
IF @TIME>=25 AND @TIME<30
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('25-30', 8)
IF @TIME>=30 AND @TIME<35
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('30-35', 9)
IF @TIME>=35 AND @TIME<40
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('35-40', 10)
IF @TIME>=40 AND @TIME<45
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('40-45', 11)
IF @TIME>=45 AND @TIME<50
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('45-50', 12)
IF @TIME>=50
INSERT INTO ANTIGUEDAD (RANGOS, ANTIGUEDAD) VALUES('>50', 13)
FETCH NEXT FROM CUR
INTO  @TIME, @MONTF, @DAYF
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . REGIMENC
/*============================================================*/
CREATE PROCEDURE REGIMENC @RAZON NVARCHAR(4), @COSTO NVARCHAR(2) AS
DECLARE @MES NVARCHAR(2), @TIME NVARCHAR(2), @MONTF INT, @MONT INT, @DAYF
INT, @DAY INT,@NOMBRES NVARCHAR(32), @APELLIDO1 NVARCHAR(16), @APELLIDO2
NVARCHAR(16)
SET @MONT = DATEPART(MONTH,'27/09/2002')
SET @DAY  = DATEPART(DAY, '27/09/2002')
DECLARE CUR CURSOR FOR SELECT  DATEDIFF(YEAR, FECHA, '27/09/2002') ,
DATEPART(MONTH,FECHA) ,  DATEPART(DAY,FECHA), NOMBRES, APELLIDO1, APELLIDO2
FROM PEREMPLEADO WHERE ID_RAZON=@RAZON AND ID_COSTO=@COSTO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @TIME, @MONTF, @DAYF, @NOMBRES, @APELLIDO1, @APELLIDO2
WHILE @@FETCH_STATUS = 0
BEGIN
IF @MONTF < @MONT
SET @MES = @MONT - @MONTF
IF @MONTF =  @MONT
BEGIN
IF @DAY < @DAYF
BEGIN
SET @MES = 11
SET @TIME = @TIME - 1
END
ELSE
SET @MES = 0
```

```
END
IF @MONT< @MONTF
BEGIN
SET @TIME = @TIME -1
SET @MES = (12-@MONTF) + @MONT
END
IF @DAY < @DAYF
SET @MES = @MES - 1
IF LEN(@MES) = 1
SET @MES = '0' + @MES
SET @MES = @TIME+'.'+ @MES
INSERT INTO ANTIGUEDAD (NOMBRES, APELLIDO1, APELLIDO2, ANTIGUEDAD)
VALUES(@NOMBRES, @APELLIDO1, @APELLIDO2, @MES)
FETCH NEXT FROM CUR
INTO  @TIME, @MONTF, @DAYF, @NOMBRES, @APELLIDO1, @APELLIDO2
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . REGIMENR
/*============================================================*/
CREATE PROCEDURE REGIMENR @RAZON NVARCHAR(4) AS
DECLARE @MES NVARCHAR(2), @TIME NVARCHAR(2), @MONTF INT, @MONT INT, @DAYF
INT, @DAY INT,@NOMBRES NVARCHAR(32), @APELLIDO1 NVARCHAR(16), @APELLIDO2
NVARCHAR(16)
SET @MONT = DATEPART(MONTH,'27/09/2002')
SET @DAY  = DATEPART(DAY, '27/09/2002')
DECLARE CUR CURSOR FOR SELECT  DATEDIFF(YEAR, FECHA, '27/09/2002') ,
DATEPART(MONTH,FECHA) ,  DATEPART(DAY,FECHA), NOMBRES, APELLIDO1, APELLIDO2
FROM PEREMPLEADO WHERE ID_RAZON=@RAZON
OPEN CUR
FETCH NEXT FROM CUR
INTO  @TIME, @MONTF, @DAYF, @NOMBRES, @APELLIDO1, @APELLIDO2
WHILE @@FETCH_STATUS = 0
BEGIN
IF @MONTF < @MONT
SET @MES = @MONT - @MONTF
IF @MONTF =  @MONT
BEGIN
IF @DAY < @DAYF
BEGIN
SET @MES = 11
SET @TIME = @TIME - 1
END
ELSE
SET @MES = 0
END
IF @MONT< @MONTF
BEGIN
SET @TIME = @TIME -1
SET @MES = (12-@MONTF) + @MONT
END
IF @DAY < @DAYF
SET @MES = @MES - 1
IF LEN(@MES) = 1
SET @MES = '0' + @MES
```

```
SET @MES = @TIME+'.'+ @MES
INSERT INTO ANTIGUEDAD (NOMBRES, APELLIDO1, APELLIDO2, ANTIGUEDAD)
VALUES(@NOMBRES, @APELLIDO1, @APELLIDO2, @MES)
FETCH NEXT FROM CUR
INTO  @TIME, @MONTF, @DAYF, @NOMBRES, @APELLIDO1, @APELLIDO2
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . TCAPA
/*============================================================*/
CREATE PROCEDURE TCAPA @RAZON NVARCHAR(4), @ANO NVARCHAR(4)=' ' AS
DECLARE @PERIODO NVARCHAR(4), @CAPA INT
IF @ANO=' '
DECLARE CUR CURSOR FOR SELECT DISTINCT ANO AS PERIODO,COUNT(ID) AS CAPA
FROM TCAPACITACION WHERE ID_RAZON=@RAZON AND ESTADO<>'P' GROUP BY ANO
ELSE
DECLARE CUR CURSOR FOR SELECT DISTINCT PERIODO,COUNT(ID) AS CAPA  FROM
TCAPACITACION WHERE ID_RAZON=@RAZON AND ANO=@ANO AND ESTADO<>'P' GROUP
BY PERIODO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @PERIODO,@CAPA
WHILE @@FETCH_STATUS = 0
BEGIN
IF @ANO=' '
INSERT INTO TOTALC (ANO,TCAPA) VALUES(@PERIODO,@CAPA)
ELSE
INSERT INTO TOTALC (PERIODO,TCAPA) VALUES(@PERIODO,@CAPA)
FETCH NEXT FROM CUR
INTO  @PERIODO,@CAPA
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . TEMAR
/*============================================================*/
CREATE PROCEDURE TEMAR @RAZON NVARCHAR(4), @ANO NVARCHAR(4) AS
DECLARE @SISTEM NVARCHAR(12), @AREA NVARCHAR(64)
DECLARE CUR CURSOR FOR SELECT SISTEMAS,AREA FROM PERCAPACITACION WHERE
ID_RAZON=@RAZON AND ANO=@ANO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @SISTEM,@AREA
WHILE @@FETCH_STATUS = 0
BEGIN
IF @SISTEM=NULL
INSERT INTO TOTALC (TEMA) VALUES(@AREA)
ELSE
INSERT INTO TOTALC (TEMA) VALUES(@SISTEM)
FETCH NEXT FROM CUR
INTO  @SISTEM,@AREA
END
CLOSE CUR
```

```
DEALLOCATE CUR
GO

/*============================================================*/
/* Procedure: . TIMEC
/*============================================================*/
CREATE PROCEDURE TIMEC @RAZON NVARCHAR(4), @COSTO NVARCHAR(2) AS
DECLARE @MES NVARCHAR(6), @TIME NVARCHAR(2), @MONTF INT, @MONT INT, @DAYF
INT, @DAY INT,@NOMBRES NVARCHAR(32), @APELLIDO1 NVARCHAR(16), @APELLIDO2
NVARCHAR(16)
SET @MONT = DATEPART(MONTH,GETDATE())
SET @DAY  = DATEPART(DAY, GETDATE())
DECLARE CUR CURSOR FOR SELECT  DATEDIFF(YEAR, FECHA, GETDATE()) ,
DATEPART(MONTH,FECHA) ,  DATEPART(DAY,FECHA), NOMBRES, APELLIDO1, APELLIDO2
FROM PEREMPLEADO WHERE ID_RAZON=@RAZON AND ID_COSTO=@COSTO
OPEN CUR
FETCH NEXT FROM CUR
INTO  @TIME, @MONTF, @DAYF, @NOMBRES, @APELLIDO1, @APELLIDO2
WHILE @@FETCH_STATUS = 0
BEGIN
IF @MONTF < @MONT
SET @MES = @MONT - @MONTF
IF @MONTF =  @MONT
BEGIN
IF @DAY < @DAYF
BEGIN
SET @MES = 11
SET @TIME = @TIME - 1
END
ELSE
SET @MES = 0
END
IF @MONT< @MONTF
BEGIN
SET @TIME = @TIME -1
SET @MES = (12-@MONTF) + @MONT
END
IF @DAY < @DAYF
SET @MES = @MES - 1
IF LEN(@MES) = 1
SET @MES = '0' + @MES
SET @MES = @TIME+'.'+ @MES
INSERT INTO ANTIGUEDAD (NOMBRES, APELLIDO1, APELLIDO2, ANTIGUEDAD)
VALUES(@NOMBRES, @APELLIDO1, @APELLIDO2, @MES)
FETCH NEXT FROM CUR
INTO  @TIME, @MONTF, @DAYF, @NOMBRES, @APELLIDO1, @APELLIDO2
END
CLOSE CUR
DEALLOCATE CUR
GO

/*============================================================*/
```

```
/* Procedure: . TIMER
/*=============================================================*/
CREATE PROCEDURE TIMER @RAZON NVARCHAR(4) AS
DECLARE @MES NVARCHAR(6), @TIME NVARCHAR(2), @MONTF INT, @MONT INT, @DAYF
INT, @DAY INT,@NOMBRES NVARCHAR(32), @APELLIDO1 NVARCHAR(16), @APELLIDO2
NVARCHAR(16)
SET @MONT = DATEPART(MONTH,GETDATE())
SET @DAY  = DATEPART(DAY, GETDATE())
DECLARE CUR CURSOR FOR SELECT  DATEDIFF(YEAR, FECHA, GETDATE()) ,
DATEPART(MONTH,FECHA) ,  DATEPART(DAY,FECHA), NOMBRES, APELLIDO1, APELLIDO2
FROM PEREMPLEADO WHERE ID_RAZON=@RAZON
OPEN CUR
FETCH NEXT FROM CUR
INTO  @TIME, @MONTF, @DAYF, @NOMBRES, @APELLIDO1, @APELLIDO2
WHILE @@FETCH_STATUS = 0
BEGIN
IF @MONTF < @MONT
SET @MES = @MONT - @MONTF
IF @MONTF =  @MONT
BEGIN
IF @DAY < @DAYF
BEGIN
SET @MES = 11
SET @TIME = @TIME - 1
END
ELSE
SET @MES = 0
END
IF @MONT< @MONTF
BEGIN
SET @TIME = @TIME -1
SET @MES = (12-@MONTF) + @MONT
END
IF @DAY < @DAYF
SET @MES = @MES - 1
IF LEN(@MES) = 1
SET @MES = '0' + @MES
SET @MES =  @TIME+'.'+ @MES
INSERT INTO ANTIGUEDAD (NOMBRES, APELLIDO1, APELLIDO2, ANTIGUEDAD)
VALUES(@NOMBRES, @APELLIDO1, @APELLIDO2, @MES)
FETCH NEXT FROM CUR
INTO  @TIME, @MONTF, @DAYF, @NOMBRES, @APELLIDO1, @APELLIDO2
END
CLOSE CUR
DEALLOCATE CUR
GO
```

7. REQUERIMIENTOS DE HARDWARE Y SOFTWARE

La aplicación exige unos requerimientos mínimos de hardware y software, tanto en el servidor como en los equipos clientes para poder operar en óptimas condiciones, a continuación se especifican estos requerimientos:

7.1. REQUERIMIENTOS DEL SERVIDOR:

Procesador: 350 Mhz
Memoria: 256 Mb
Espacio en Disco Duro: 10 Gb
Tarjeta de Red: Cualquiera compatible con el sistema operativo.
Sistema operativo: Win 2000 Server/NT Server
Base de datos: SQL Server 200
Programas: Microsoft .NET Framework 1.1, Internet Explorer 6.0
Servidor de aplicación: Internet Information Services(IIS)

7.2. REQUERIMIENTOS DE LA ESTACION CLIENTE:

Procesador: 1.2 GHz
Memoria: 128 Mb
Espacio en Disco Duro: 10 Gb
Tarjeta de Red: Cualquiera compatible con el sistema operativo.
Sistema operativo: Win 98/2000/NT Server/XP.
Programas: Microsoft Internet Explorer 6.0

MANUAL DE USUARIO

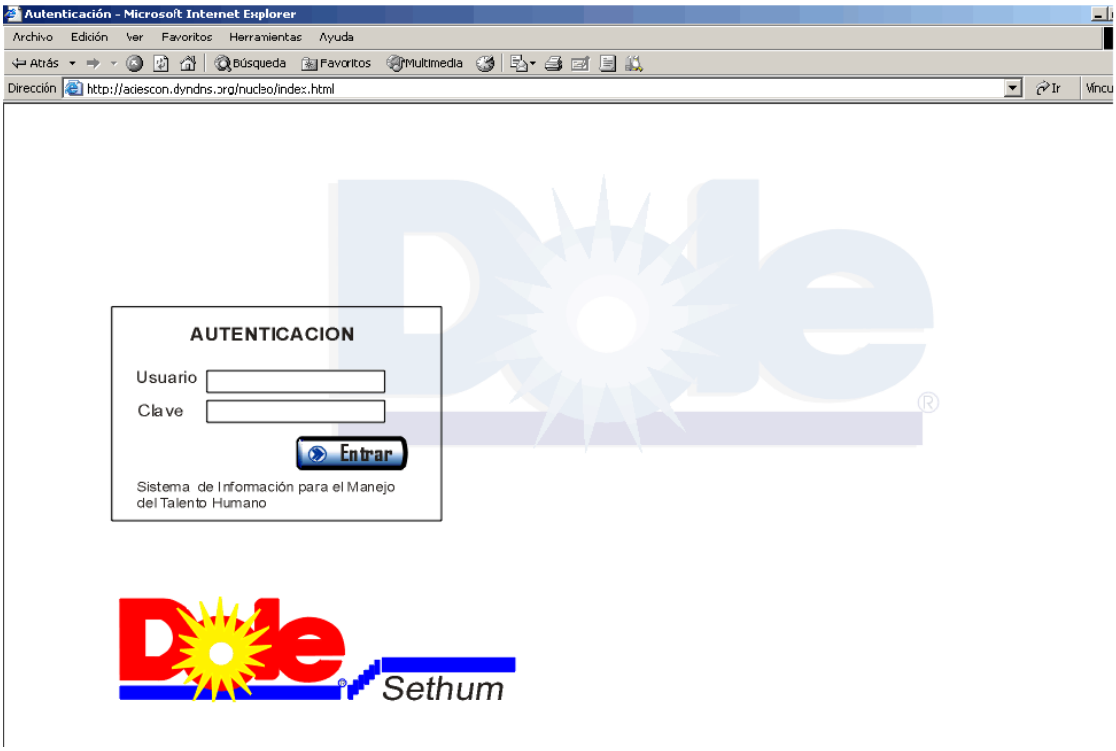
SISTEMA DE INFORMACIÓN PARA EL MANEJO DEL TALENTO HUMANO EN C.I. TÉCNICAS BALTIME DE COLOMBIA (SETHUM)

Autores:	Jaime Abella Wilman Rincón
Fecha Creación:	19/06//2005
Ultima Actualización:	18/07/2005
Versión:	1.00

TABLA DE CONTENIDO

1. AUTENTICACION DE USUARIOS

La autenticación de usuarios es la primera pantalla que aparece en el sistema en este se ingresa el nombre del usuario y la contraseña.

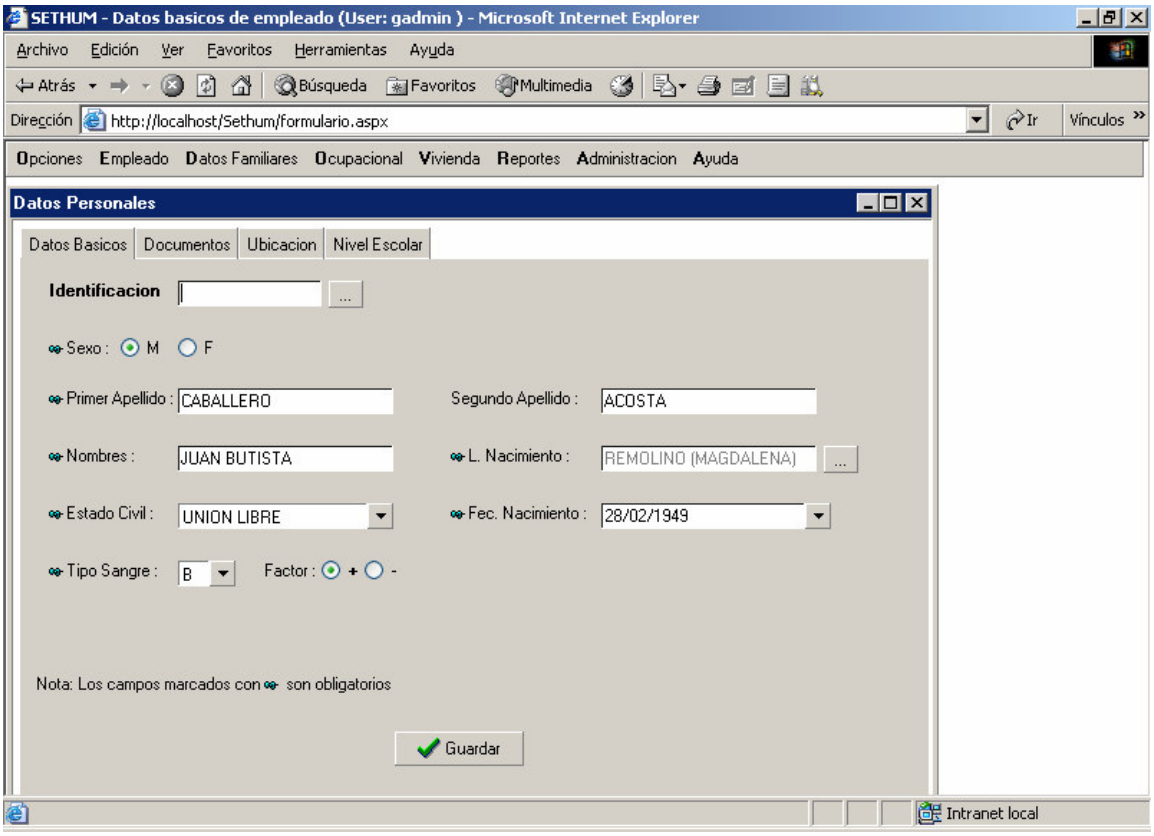


2. INGRESAR INFORMACION PERSONAL

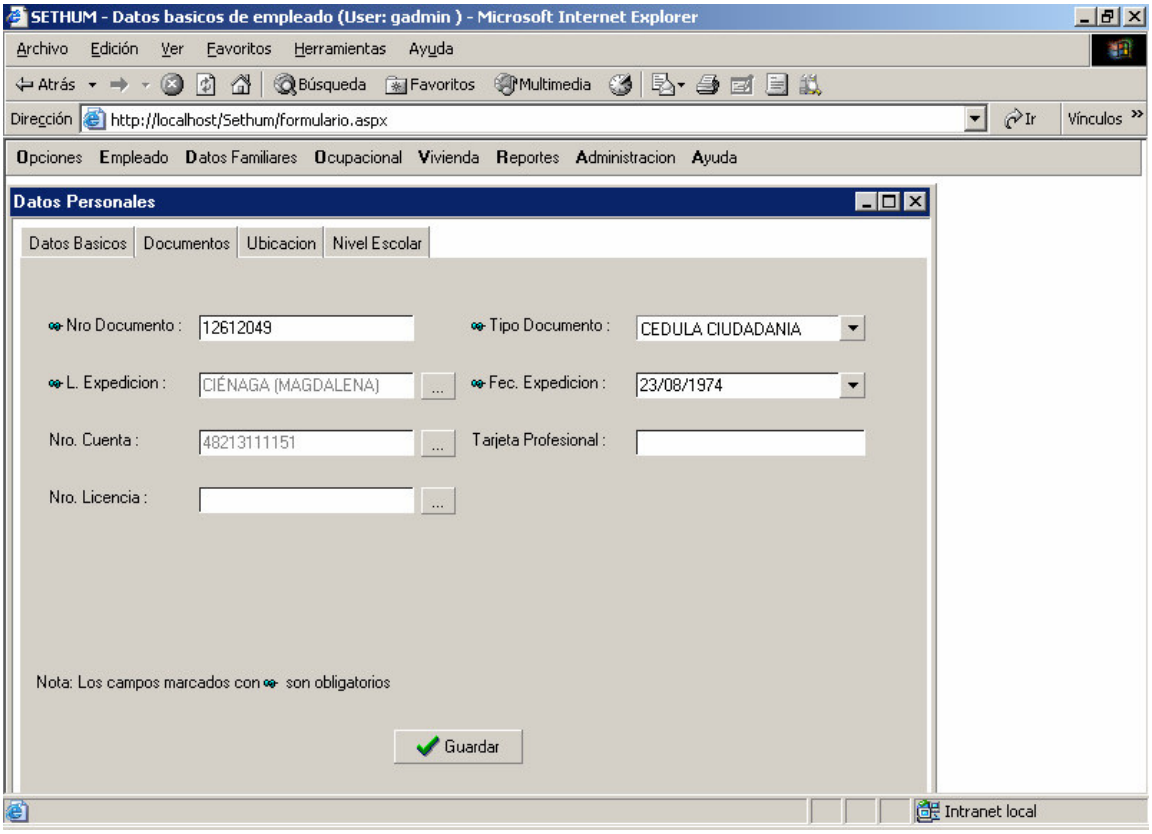
La información personal del empleado es la primera información que se ingresa, este modulo esta dividida en cuatro pestañas:

- Datos Básicos.
- Documentos.
- Ubicación.
- Nivel Escolar.

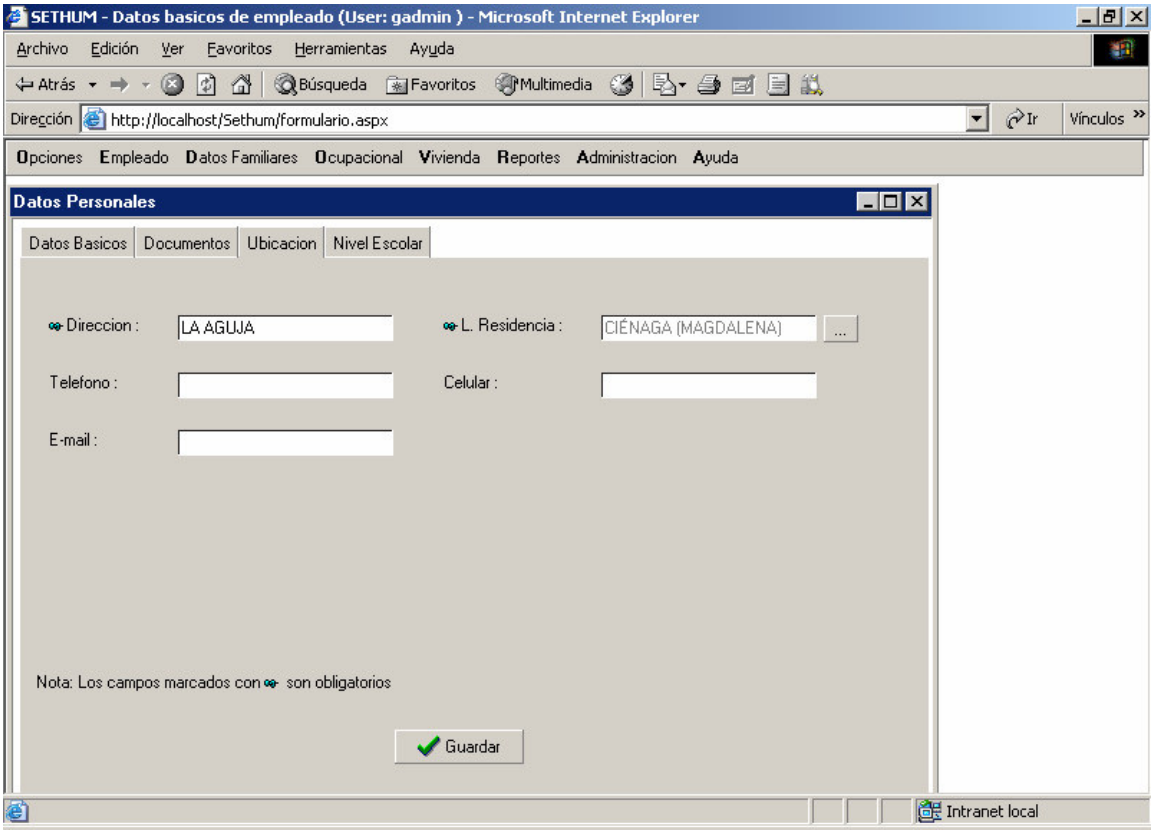
2.1. DATOS BASICOS



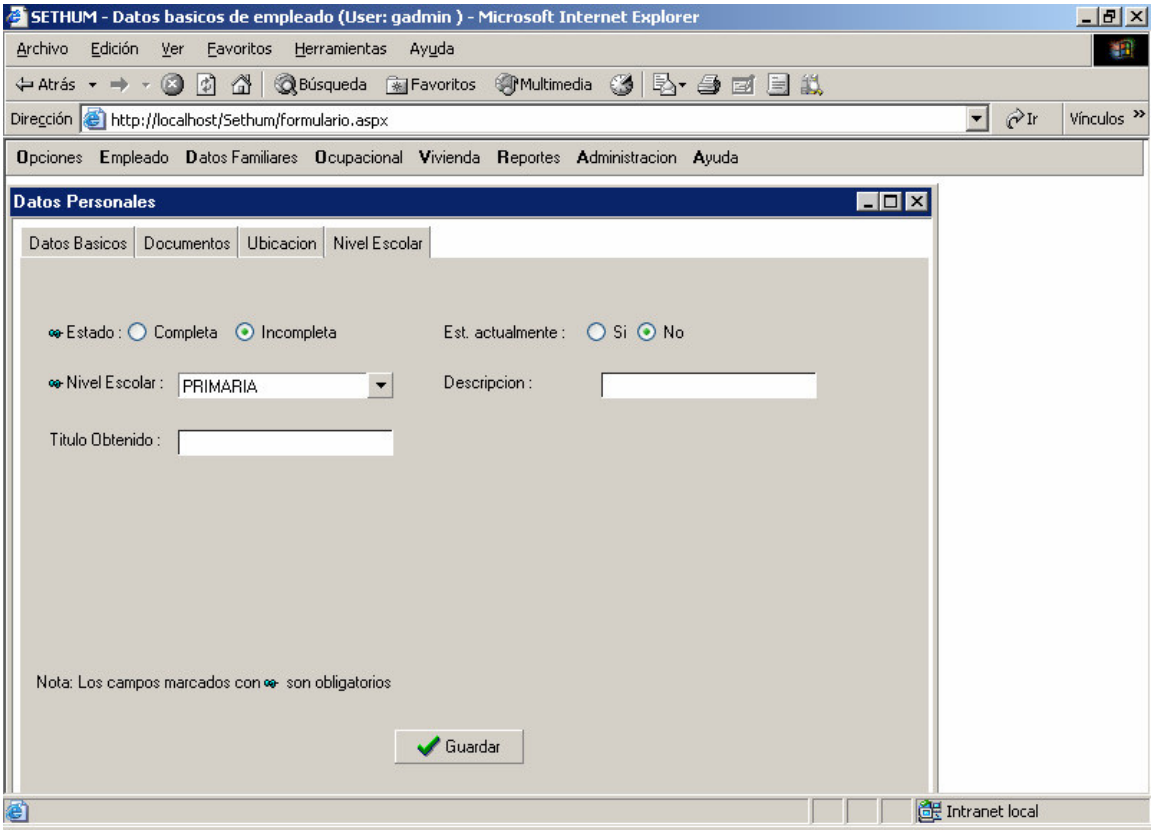
2.2. DOCUMENTOS



2.3. UBICACION

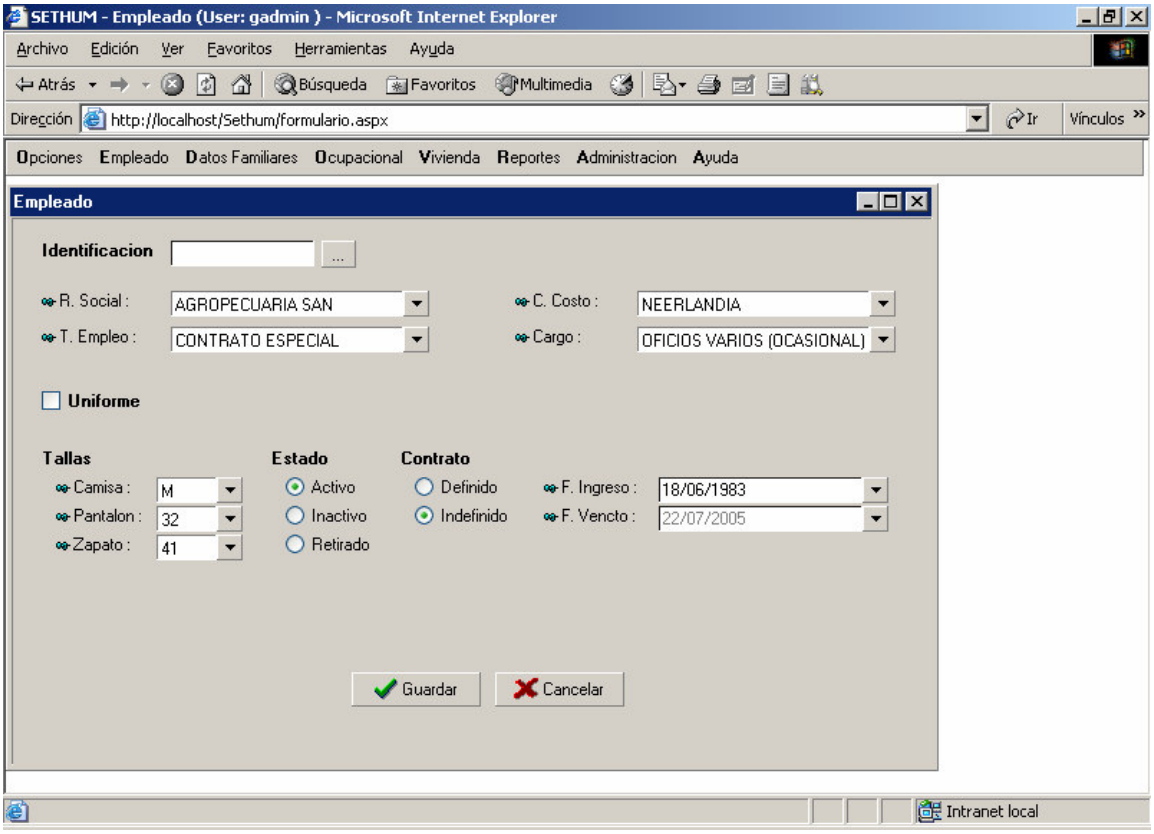


2.4. NIVEL ESCOLAR



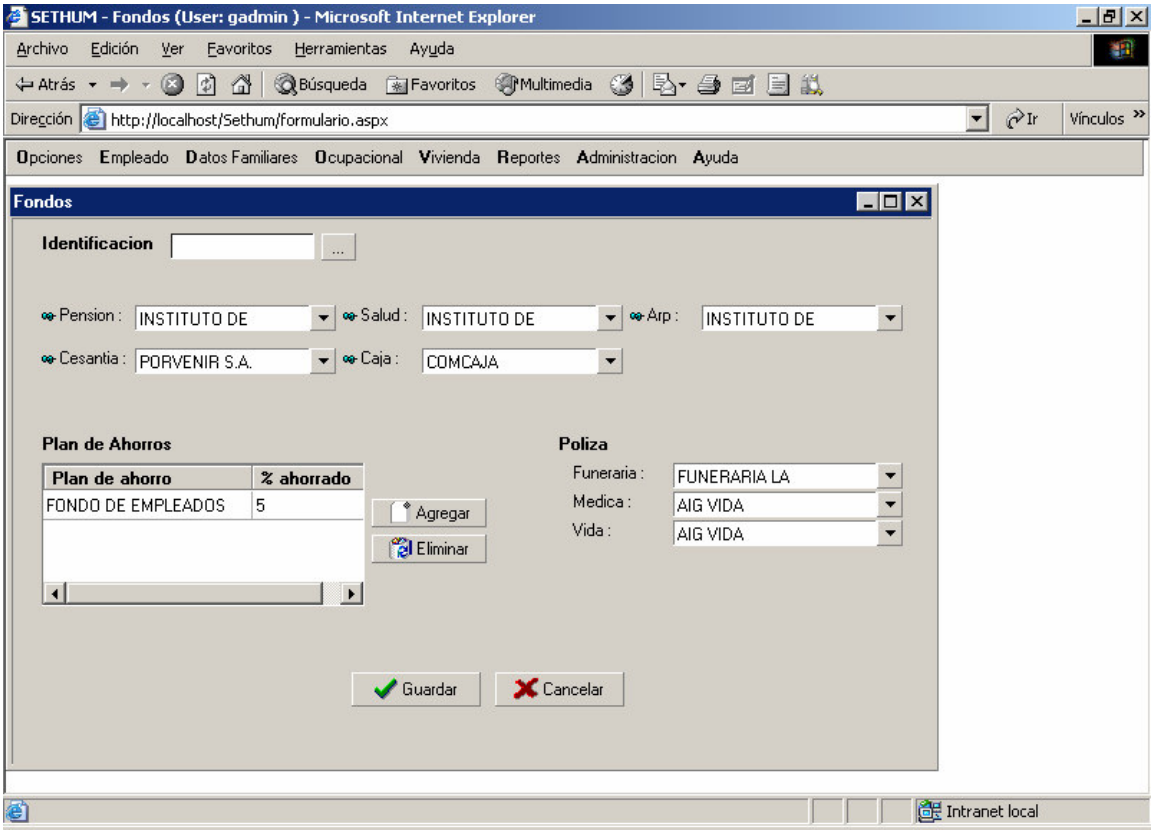
3. INGRESAR INFORMACION EMPLEADO

En este modulo ingresamos la información referente al empleado con la Compañía en la que trabaja. Para esto seleccionamos el menú empleado y seleccionamos la opción empleado.



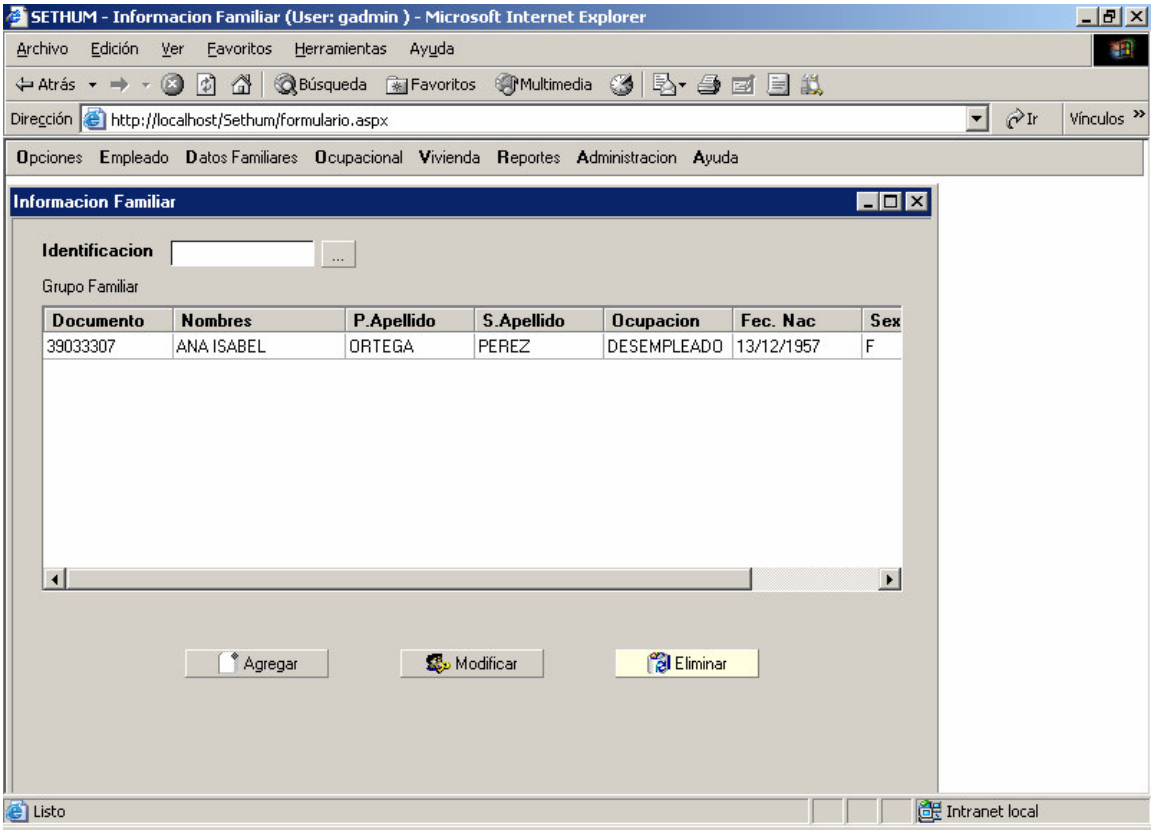
4. INGRESAR INFORMACION FODOS EMPLEADO

En este modulo ingresamos la información de las entidades a la que esta afiliado el empleado. Para esto seleccionamos el menú empleado y seleccionamos la opción fondo.

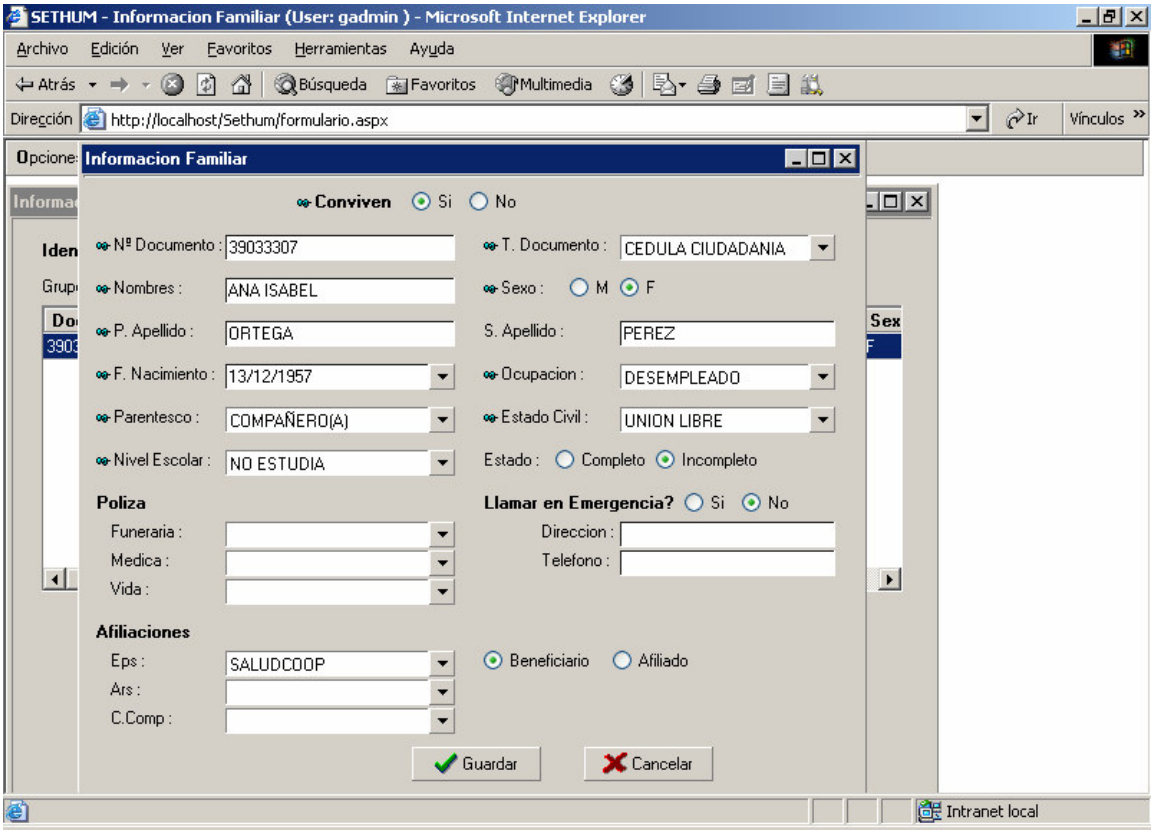


5. INGRESAR INFORMACION FAMILIAR

En este modulo ingresamos la información referente a la familia del empleado. Para esto seleccionamos el menú Datos Familiares y seleccionamos la opción Datos Familiares.

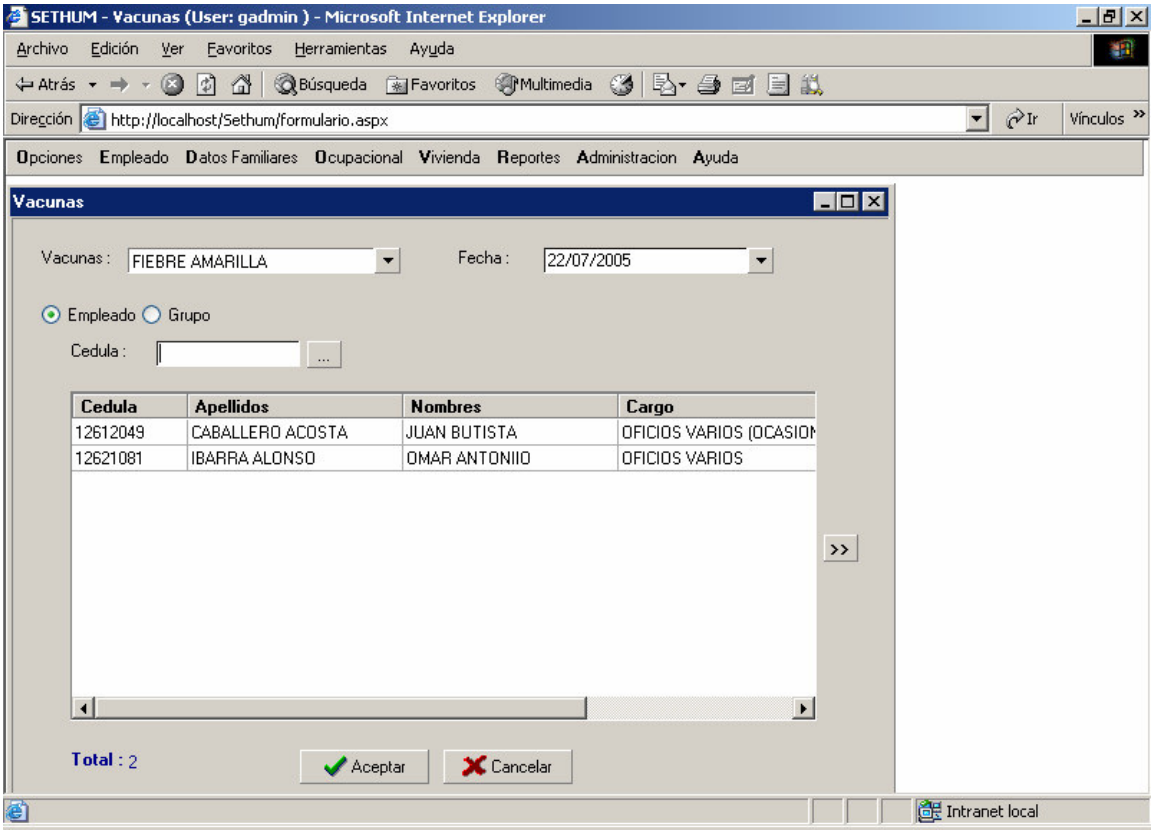


5.1. Registro de Captura del familiar

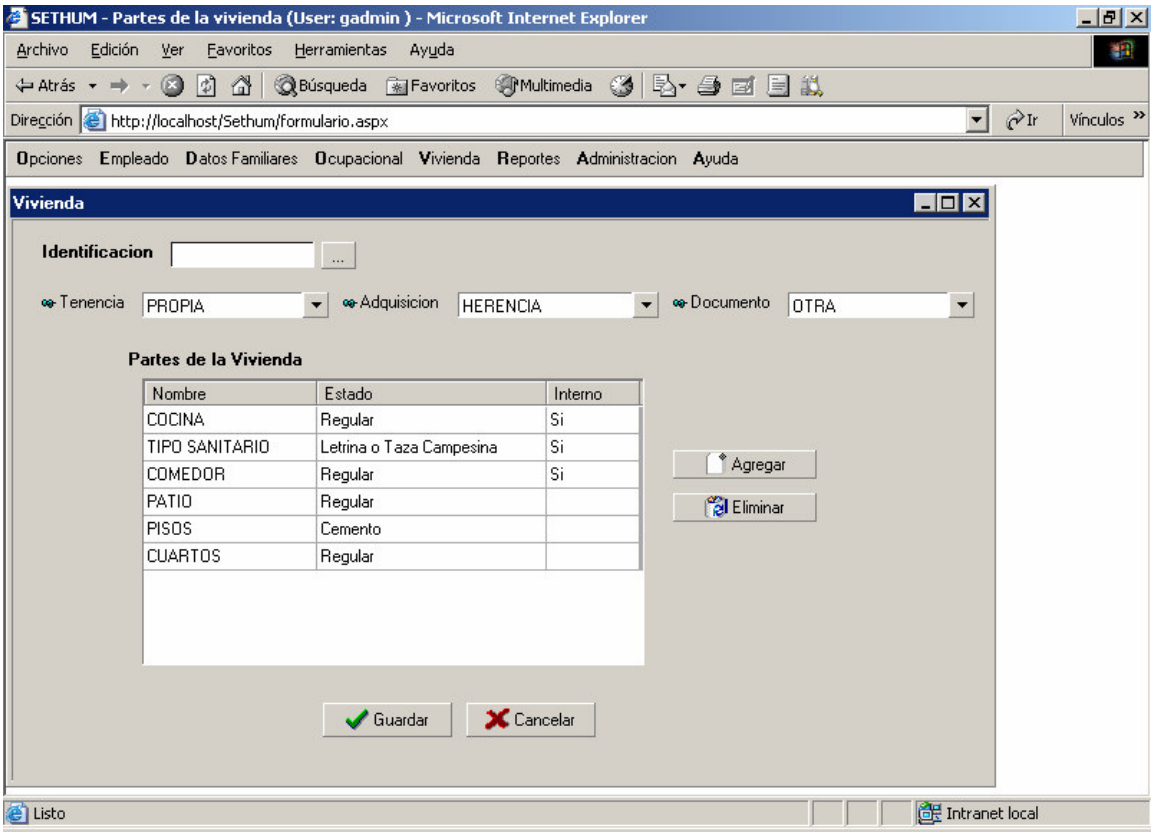


6. INGRESO DEL REGISTRO DE VACUNAS

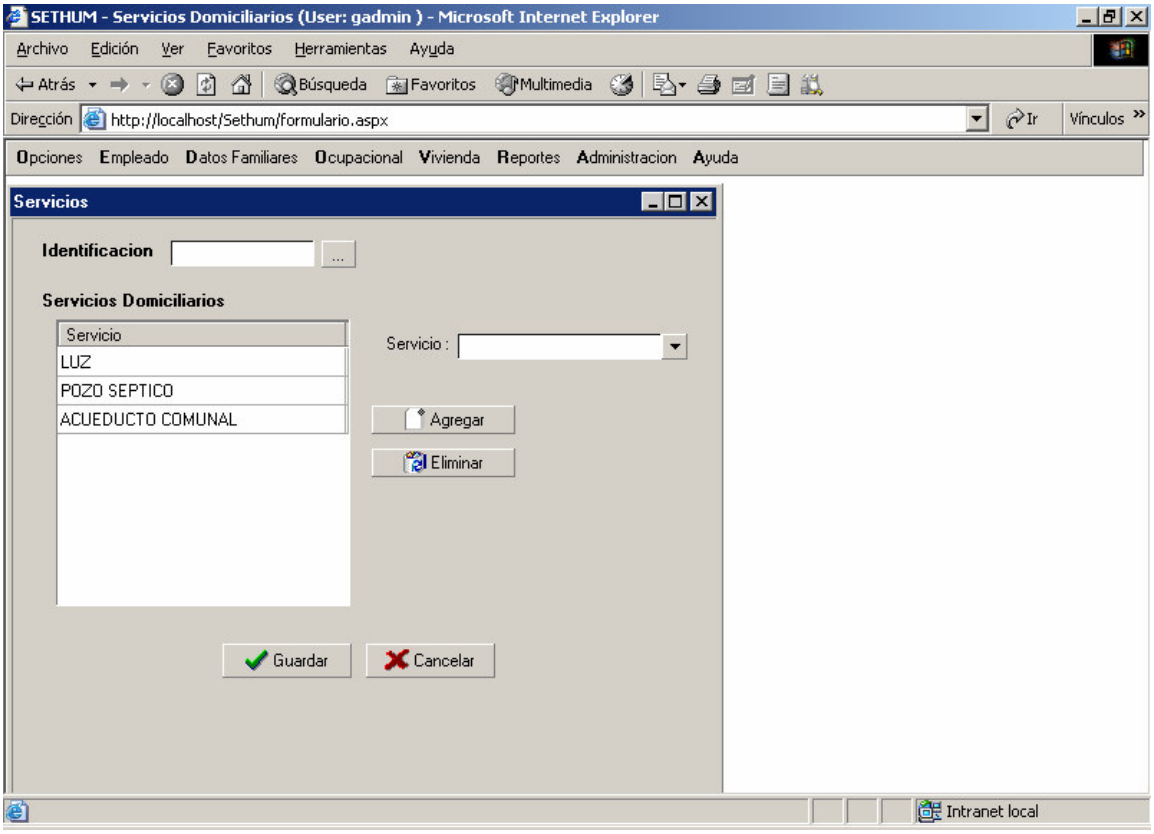
En este modulo registramos las jornadas de vacuna de los empleados. Para estos seleccionamos la opción Ocupacional luego la opción vacunas.



7. INGRESO DE LA INFORMACION DE VIVIENDA

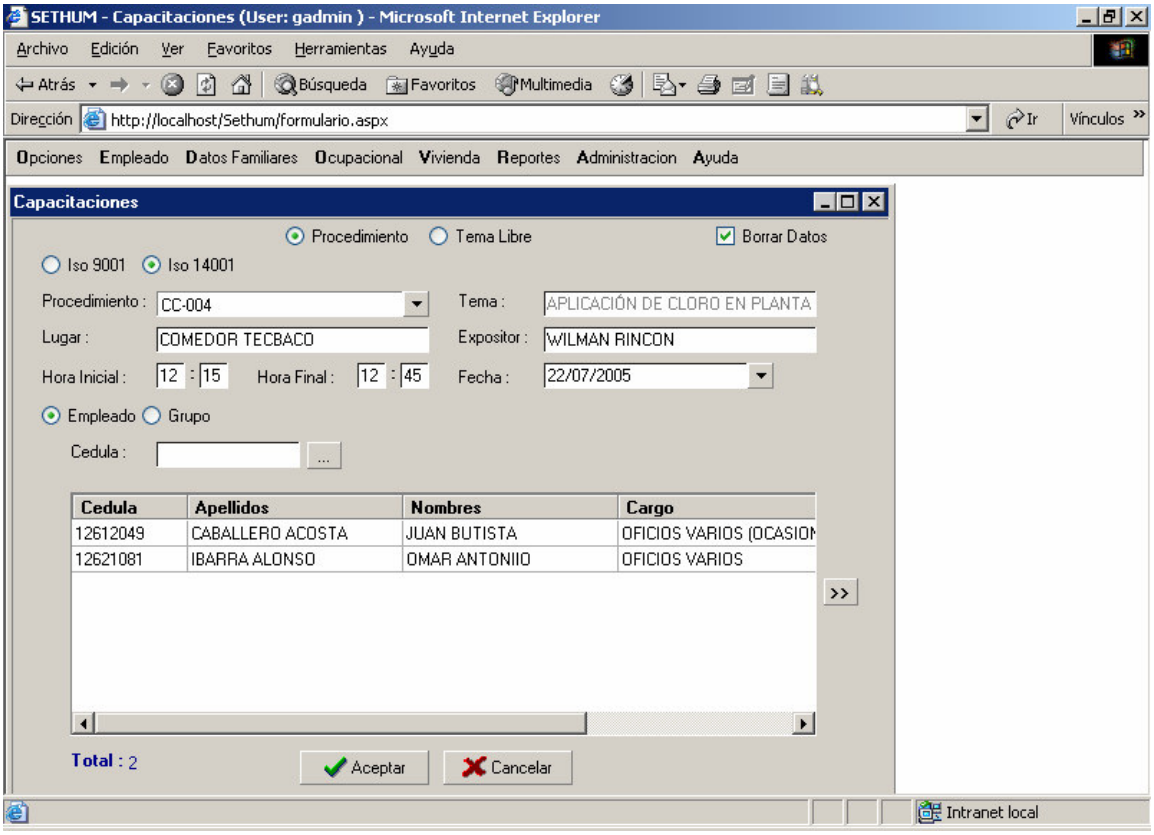


7.1. SERVICIOS DE LA VIVIENDA



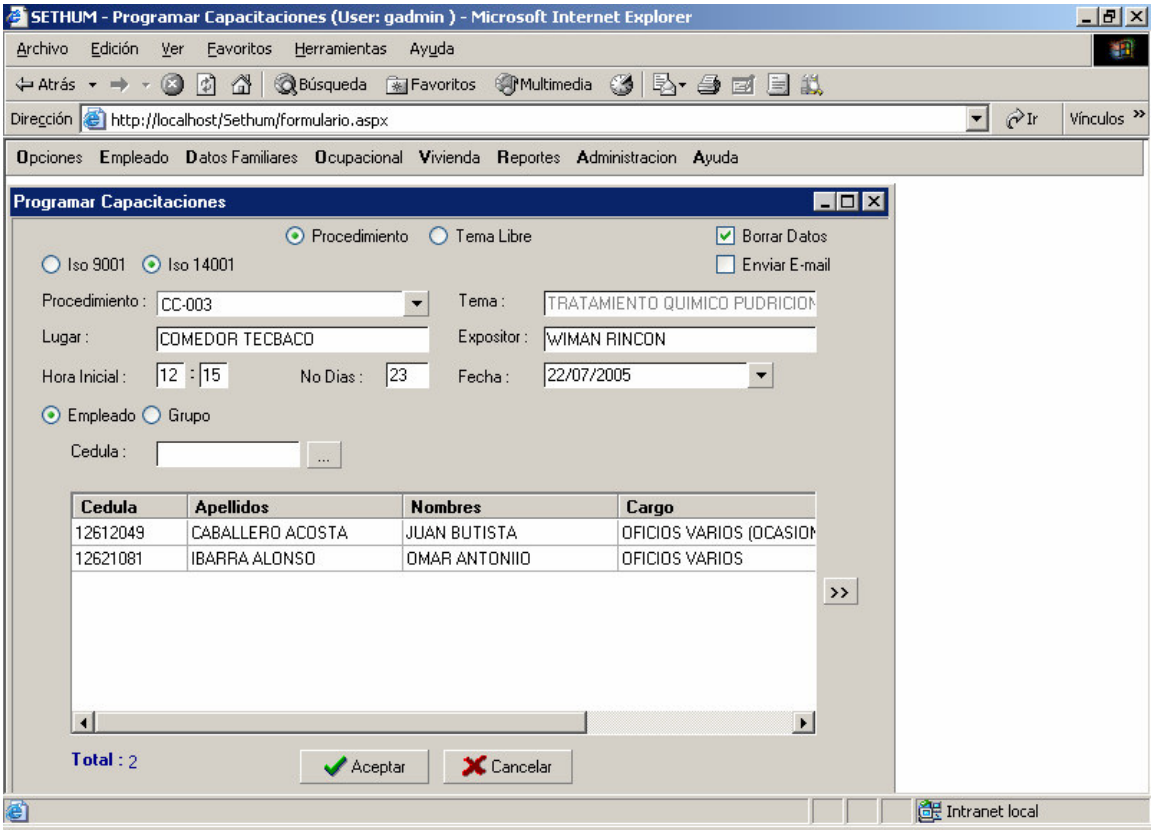
8. INGRESO DE CAPACITACIONES

En este modulo registramos las capacitaciones realizadas por la compañía. Para esto seleccionamos el menú ocupacional luego la opción Capacitaciones.



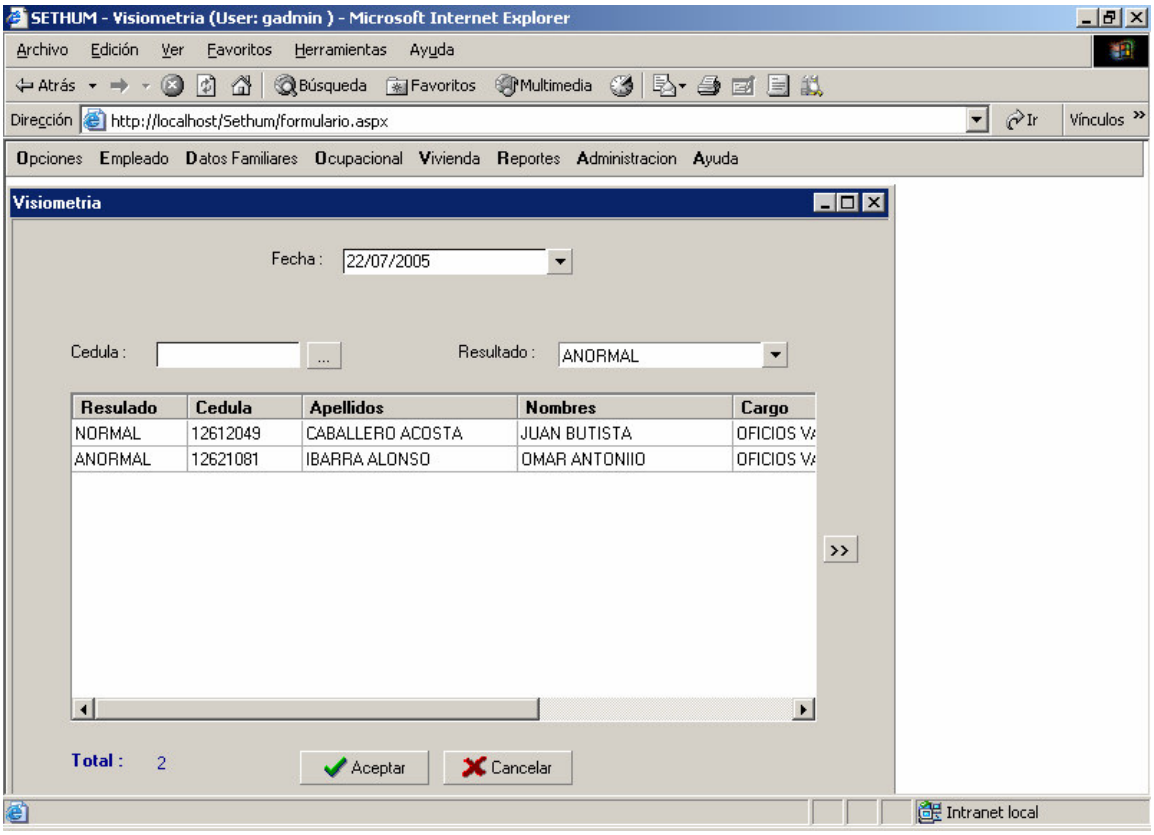
9. PROGRAMACION DE CAPACITACIONES

En este modulo registramos las capacitaciones que serán programadas por el personal de recursos humanos. Para esto seleccionamos el menú Ocupacional luego escojamos la opción Capacitación, programar.

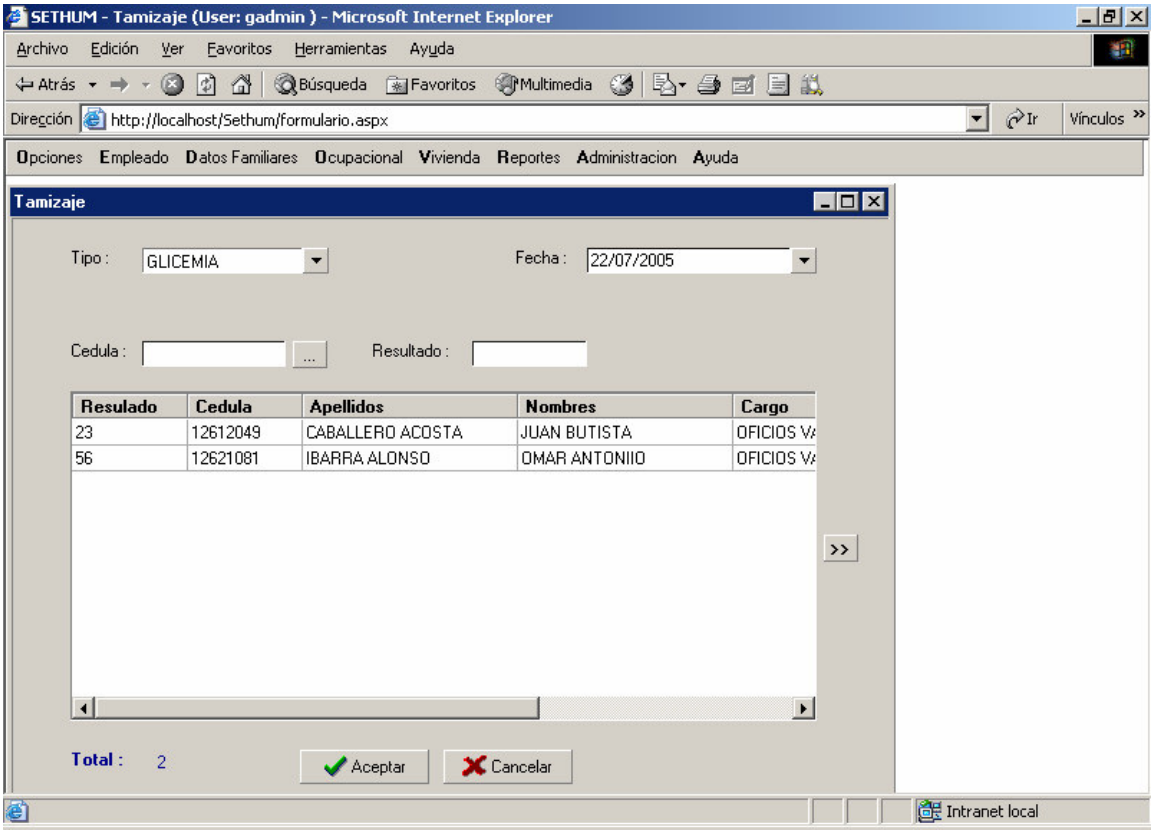


10. EXAMENES DE VISIOMETRIA

10.1. EXAMENES DE VISIOMETRIA

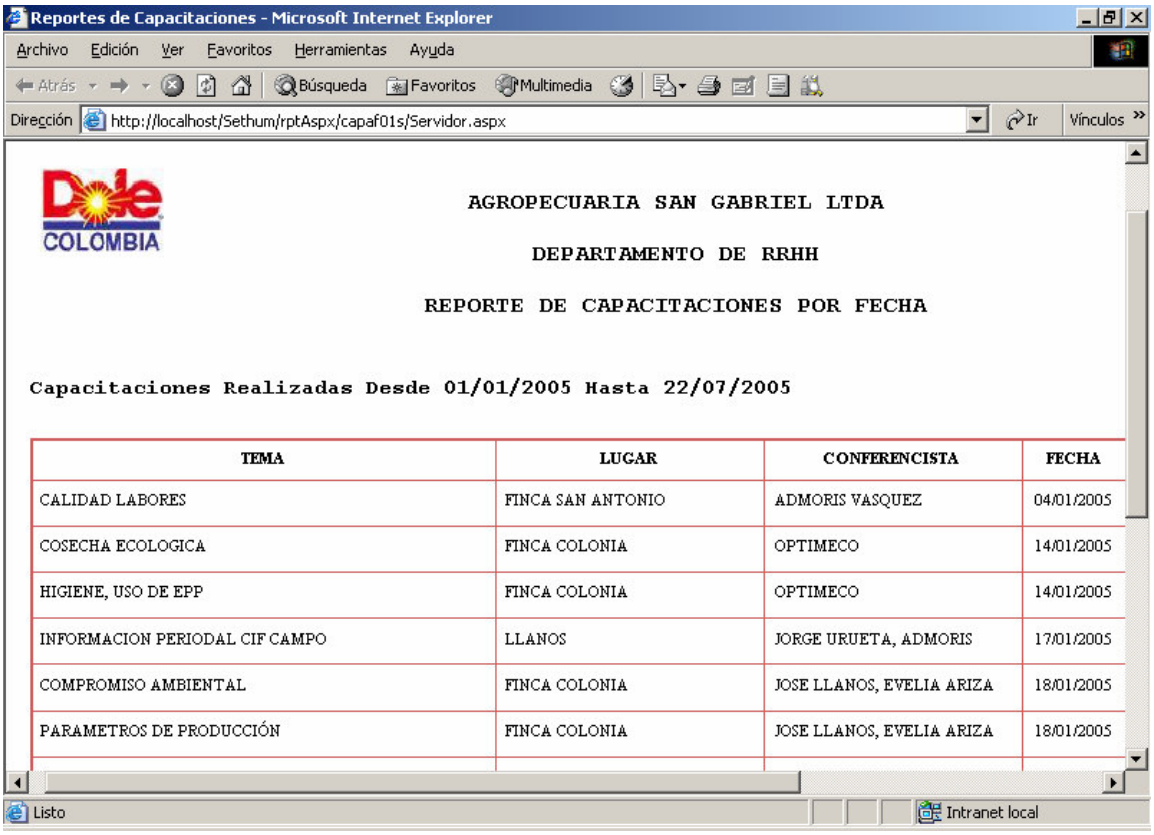


10.2. EXAMENES DE TAMIZAJE



11. REPORTES

11.1. REPORTES DE CAPACITACION POR FECHA



TEMA	LUGAR	CONFERENCISTA	FECHA
CALIDAD LABORES	FINCA SAN ANTONIO	ADMORIS VASQUEZ	04/01/2005
COSECHA ECOLOGICA	FINCA COLONIA	OPTIMECO	14/01/2005
HIGIENE, USO DE EPP	FINCA COLONIA	OPTIMECO	14/01/2005
INFORMACION PERIODAL CIF CAMPO	LLANOS	JORGE URUETA, ADMORIS	17/01/2005
COMPROMISO AMBIENTAL	FINCA COLONIA	JOSE LLANOS, EVELIA ARIZA	18/01/2005
PARAMETROS DE PRODUCCIÓN	FINCA COLONIA	JOSE LLANOS, EVELIA ARIZA	18/01/2005

11.2. REPORTE DE HIJOS POR EDADES