Capítulo 6

Implementación e implantación del sistema

En el capítulo 4 se identificaron cada uno de los casos de uso a partir de los cuales se fundamenta el diseño lógico del sistema y en el capítulo 5 se hace el análisis de las tecnologías de desarrollo y almacenamiento de información consideradas como opciones para lograr concretar la herramienta SIGED.

En este capítulo se procede a describir el desarrollo de los módulos, así como a las partes complementarias que permiten el buen funcionamiento, proporcionando previamente una explicación de las opciones tecnológicas que exige el sistema para controlar el flujo, procesamiento y almacenamiento de la información.

6.1 Arquitectura cliente / servidor

La arquitectura cliente / servidor significa que los equipos clientes contactan a un servidor, un equipo generalmente muy potente en materia de capacidad de entrada / salida que proporciona servicios a los equipos solicitantes.

Los servicios son utilizados por programas denominados programas clientes que se ejecutan en los equipos periféricos, de ahí que se utilice el término "cliente" cuando se hace referencia a un programa que ha sido diseñado para ser ejecutado en un equipo que es capaz de procesar los datos recibidos de un servidor. Por ejemplo, en el caso del cliente FTP se trata de un *software* que procesa archivos en un servidor remoto, mientras que para el cliente de correo se trata del *software* que descarga los mensajes de correo electrónico recibidos en el equipo servidor que nos otorga dicha prestación.

Un sistema cliente / servidor funciona tal como se detalla en la figura 6.1:

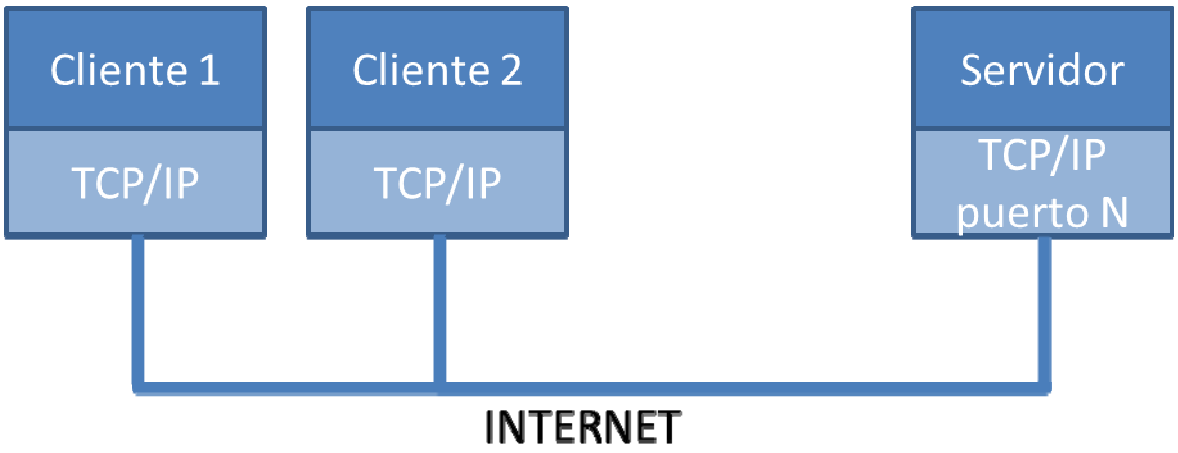

Figura 6.1 Diagrama de un sistema cliente / servidor.

El cliente envía una solicitud al servidor mediante su dirección IP y el puerto que está reservado para un servicio en particular que se ejecuta en el servidor. El servidor recibe la solicitud y responde con la dirección IP del equipo cliente y su puerto.

6.1.1 Arquitectura multicapa

La arquitectura multicapa se utiliza para describir a los sistemas cliente / servidor en donde el cliente solicita y el servidor (A) no responde directamente a la solicitud, sino que recurre a otro servidor (B) y efectúa una nueva solicitud y este tercero responde al segundo quien administra la información y responde finalmente al cliente. Esto significa que el servidor requiere de otra aplicación para proporcionar parte del servicio, tal y como se observa en la figura 6.2:

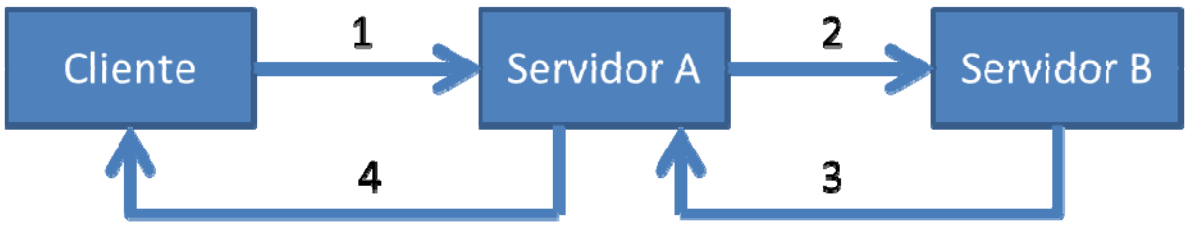

Figura 6.2 Arquitectura multicapa

Las aplicaciones de multiniveles distribuyen muchas de las funciones de los sistemas autónomos: presentan una interfaz de usuario, realizan el

proceso de pedido e informan el estado de la petición. Esta secuencia de comandos puede repetirse tantas veces como sea necesario.

La facilidad y la flexibilidad de los productos multiniveles es la base de muchas aplicaciones empresariales a gran escala. Esto responde a la necesidad de contar con un acceso más rápido a los datos y una reducción de tiempo al momento de desarrollar nuevas aplicaciones.

6.2 Modelos OSI y TCP/IP

El modelo OSI fue desarrollado en 1984 por la organización internacional de estándares llamada ISO, la cual es una federación global de organizaciones que representa a 130 países aproximadamente. El núcleo de este estándar es el modelo de referencia OSI, una normativa formada de siete capas que define las diferentes fases por las que deben pasar los datos para viajar de un dispositivo a otro sobre una red de comunicaciones.

La utilidad de esta normativa estandarizada viene al haber muchas tecnologías, fabricantes y compañías dentro del mundo de las comunicaciones y al estar en continua expansión, se tuvo que crear un método para que todos pudieran entenderse de algún modo, incluso cuando las tecnologías no coincidieran. De tal forma que no importa la localización geográfica o el lenguaje utilizado. Todo el mundo debe atenerse a unas normas mínimas para poder comunicarse entre sí. Esto es, sobre todo, importante cuando hablamos de la red de redes, es decir, Internet.

Se debe pensar en las siete capas que componen el modelo OSI como una línea de ensamblaje en un ordenador, como se ve en la primera parte de la Figura 6.3. En cada una de las capas ciertas cosas pasan a los datos que se preparan para ir a la siguiente capa. Las siete capas se pueden separar en dos grupos bien definidos: grupo de aplicación y grupo de transporte.

En el grupo de aplicación tenemos:

a) Capa 7: Aplicación. Esta es la capa que interactúa con el sistema operativo o aplicación cuando el usuario decide transferir archivos, leer mensajes, o realizar otras actividades de red. Por ello, en esta capa se incluyen tecnologías tales como http, DNS, SMTP, SSH, Telnet, etc.
b) Capa 6: Presentación. Esta capa tiene la misión de coger los datos que han sido entregados por la capa de aplicación, y convertirlos en un formato estándar que otras capas puedan entender. En esta capa tenemos como ejemplo los formatos MP3, MPG, GIF, etc.
c) Capa 5: Sesión. Esta capa establece, mantiene y termina las comunicaciones que se forman entre dispositivos. Se pueden poner como ejemplo, las sesiones SQL, RPC, NetBIOS, etc.

En el grupo de transporte tenemos:

d) Capa 4: Transporte. Esta capa mantiene el control de flujo de datos y provee de verificación de errores y recuperación de datos entre dispositivos. Control de flujo significa que la capa de transporte vigila si los datos vienen de más de una aplicación e integra cada uno de los datos de aplicación en un solo flujo dentro de la red física. Como ejemplos más claros tenemos TCP y UDP.
e) Capa 3: Red. Esta capa determina la forma en que serán mandados los datos al dispositivo receptor. Aquí se manejan los protocolos de enrutamiento y el manejo de direcciones IP. En esta capa hablamos de IP, IPX, X.25, etc.
f) Capa 2: Datos. También llamada capa de enlaces de datos. En esta capa, el protocolo físico adecuado es asignado a los datos. Se asigna el tipo de red y la secuencia de paquetes utilizada. Los ejemplos más claros son *Ethernet*, ATM, *Frame Relay*, etc.
g) Capa 1: Física. Este es el nivel de lo que llamamos llanamente hardware. Define las características físicas de la red, como las conexiones, niveles de voltaje, cableado, etc. Como habrás

supuesto, podemos incluir en esta capa la fibra óptica, el par trenzado, cable cruzados, etc.

Seguramente se habrá oído hablar de otro modelo paralelo al modelo OSI, llamado capas TCP/IP. Lo cierto es que son muy parecidas y las capas se entremezclan, como se aprecia en la Figura 6.3, solo que este último modelo solo utiliza niveles para explicar la funcionalidad de red. Las capas son las siguientes:

a) Capa 1: Red. Esta capa combina la capa física y la capa de enlaces de datos del modelo OSI. Se encarga de dirigir los datos entre dispositivos en la misma red. También maneja el intercambio de datos entre la red y otros dispositivos.

b) Capa 2: Internet. Esta capa corresponde a la capa de red. El protocolo de Internet utiliza direcciones IP, las cuales consisten en un identificador de red y un identificador de host, para determinar la dirección del dispositivo con el que se está comunicando.

c) Capa 3: Transporte. Corresponde directamente a la capa de transporte del modelo OSI y donde podemos encontrar al protocolo TCP. El protocolo TCP funciona preguntando a otro dispositivo en la red si está deseando aceptar información de un dispositivo local.

d) Capa 4: Aplicación. La capa 4 combina las capas de sesión, presentación y aplicación del modelo OSI. Protocolos con funciones específicas como correo o transferencia de archivos, residen en este nivel.

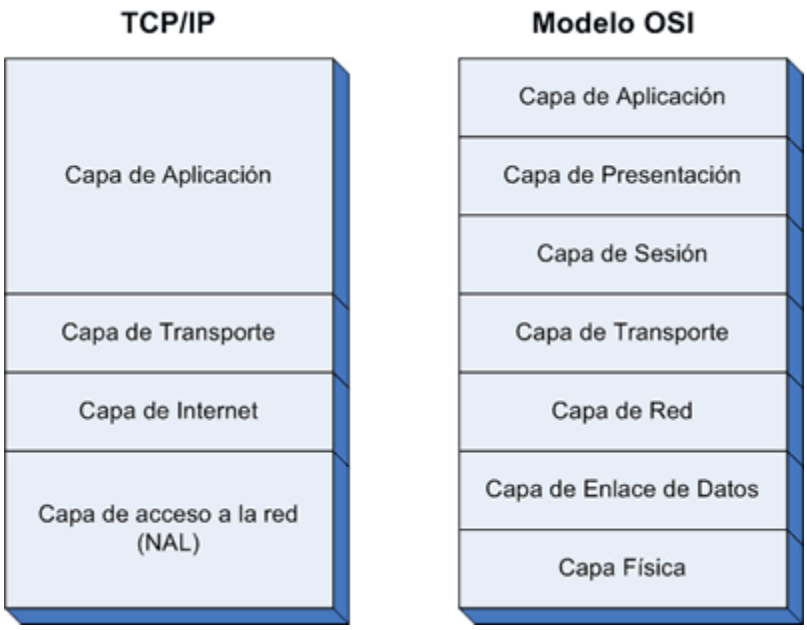

Figura 6.3 Modelos TCP/IP y OSI

6.3 Herramientas de Desarrollo

Microsoft C# es un lenguaje de programación diseñado para crear un amplio número de aplicaciones empresariales que se ejecutan en .NET Framework. Supone una evolución de Microsoft C y Microsoft C++ con un estilo sencillo y moderno, proporciona seguridad de tipos y está orientado a objetos. El código creado mediante C# se compila como código administrado, lo cual significa que se beneficia de los servicios de Common Language Runtime lo que se traduce en interoperabilidad entre lenguajes, recolección de elementos no utilizados, mejora de la seguridad y mayor compatibilidad entre versiones.

Visual C# utiliza plantillas de proyecto, diseñadores, páginas de propiedades, asistentes de código, un modelo de objetos y otras características del entorno de desarrollo. La biblioteca para programar en Visual C# es .NET Framework.

6.3.1 Microsoft SQL Server

SQL Server es un sistema administrador de bases de datos relacionales basadas en arquitectura cliente / servidor (RDBMS) que usa Transact - SQL para mandar peticiones entre un cliente y el SQL Server.

El *software* cliente es responsable de la parte lógica y de presentar la información al usuario y generalmente corre en una o más computadoras cliente, aunque también puede correr en un Servidor con SQL Server.

SQL Server administra bases de datos y distribuye los recursos disponibles del servidor (tales como memoria, operaciones de disco, etsc.) entre las múltiples peticiones.

La figura 6.4 muestra la arquitectura Cliente / Servidor de SQL server:

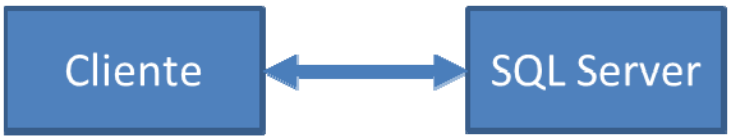

Figura 6.4 Arquitectura cliente / servidor de SQL Server

6.3.2 Servidor IIS

IIS, Servicios de Información de Internet (*Internet Information Services*, en inglés) es parte de la arquitectura de Windows Server. La función más importante de IIS es vincular los clientes que tienen acceso al sistema mediante el Protocolo de Transferencia de Hipertexto (HTTP) con los restantes servicios de Windows, como DHTML, ASP, etc. Además, IIS incluye un conjunto básico de funciones que los desarrolladores de sistemas pueden ampliar para definir una arquitectura personalizada de aplicaciones.

Define una funcionalidad básica que se puede utilizar para crear aplicaciones web. *Active Server Pages* (ASP) y otras tecnologías de Microsoft amplían esta funcionalidad básica para crear un entorno rico para el desarrollo de aplicaciones. La funcionalidad básica del servidor

se expone mediante la interfaz de programación de aplicaciones de servidor de Internet (ISAPI). Estas son las funciones básicas proporcionadas por IIS:

a) Establecer y mantener conexiones HTTP.
b) Leer peticiones HTTP y escribir respuestas HTTP.
c) Modificar encabezados HTTP.
d) Obtener la información del certificado del cliente.
e) Administrar conexiones asincrónicas.
f) Asignar localizadores de recursos universales (direcciones URL) a rutas físicas.
g) Administrar y ejecutar aplicaciones.
h) Transmitir archivos.

IIS y Servicios de componentes funcionan conjuntamente para formar la arquitectura básica para la creación de aplicaciones web y utiliza la funcionalidad proporcionada por Servicios de componentes para:

a) Aislar las aplicaciones en procesos diferentes.
b) Administrar la comunicación entre componentes COM (incluidos los objetos integrados de ASP).
c) Coordina el proceso de transacciones para las aplicaciones ASP transaccionales.

IIS define las aplicaciones web como una colección de archivos de recursos agrupados en un espacio de nombres lógico. Al agrupar los recursos en aplicaciones, es posible compartir los datos en el espacio de nombres y ejecutar la aplicación en un procesos aislado.

Internamente, IIS coordina las aplicaciones aisladas mediante un objeto llamado administrador de aplicaciones web. Este objeto incluye una interfaz pública (IWAMAdmin) que puede utilizar para crear programas de administración de aplicaciones web. Al ejecutar una aplicación web en un proceso aislado, IIS utiliza los servicios de componentes para

coordinar el acceso simultáneo a los recursos y pasar información de contexto entre los componentes COM.

IIS utiliza el objeto ObjectContext de Servicios de componentes para proporcionar acceso a los objetos integrados de ASP.

6.3.3 Plataforma .NET

Es un conjunto de tecnologías diseñadas para transformar Internet en un sistema informático distribuido a escala completa. Proporciona nuevas formas de desarrollar aplicaciones a partir de colecciones de servicios web. La plataforma cuenta con un completo soporte de tecnologías de Internet independientes y basadas en estándares, incluyendo *Hypertext Transfer Protocol* (HTTP), *Extensible Markup Language* (XML) y *Simple Object Access Protocol* (SOAP).

Además de lo anterior, la plataforma .NET proporciona:

a) Un modelo de programación coherente e independiente del lenguaje para todas las capas o niveles de una aplicación.
b) Una interoperabilidad transparente entre tecnologías.
c) Una fácil migración desde tecnologías existentes.

Los componentes principales del entorno .NET se describen a continuación:

Common Language Runtime (CLR). Es el mecanismo de ejecución para las aplicaciones .NET. Proporciona varios servicios, incluyendo la carga y ejecución del código, aislamiento de la memoria de las aplicaciones, administración de memoria, manejo de excepciones, acceso a metadata (información mejorada de tipos) y la conversión de MSIL (Lenguaje Intermedio Microsoft) a código nativo.

La *Base Class Library* (BCL). Proporciona un amplio conjunto de clases, lógicamente agrupadas en espacios de nombres jerárquicos que proporcionan acceso a las características más importantes del sistema operativo.

ADO.NET. Esta es una actualización evolutiva para la tecnología de acceso a datos *ActiveX Data Objects* (ADO) con mejoras importantes destinadas a la naturaleza desconectada del web.

ASP.NET. Esta es una versión avanzada de *Active Server Pages* (ASP) para el desarrollo de aplicaciones web (utilizando Formas web) y desarrollo de servicios web.

Common Language Specification (CSL). Esta especificación es responsable de hacer que muchas de las tecnologías antes mencionadas estén disponibles para todos los lenguajes que soportan .NET Framework. CLS no es una tecnología y no hay un código fuente para ella. Define un conjunto de reglas que proporcionan un contrato que rige la interoperabilidad entre los compiladores de lenguaje y las bibliotecas.

Win Forms. Modelo de programación y conjunto de controles. Proporciona una arquitectura sólida para el desarrollo de aplicaciones basada en Windows.

Visual Studio .NET. Ambiente de desarrollo, ofrece las herramientas que le permiten explotar las características del Framework .NET para crear aplicaciones concretas de una forma rápida y amigable.

6.4 Arquitectura del sistema

Debido a que este sistema va dirigido a una institución gubernamental a nivel nacional se optó por seguir una arquitectura de capas múltiples, con un servidor web que contendrá la aplicación, y con uno diferente donde estará la base de datos. Esto permite que la administración de los sistemas sea independiente, de tal manera que se pueden hacer mejoras de rendimiento tanto en las peticiones web como en las consultas a la base de datos.

Por otra parte, el protocolo utilizado entre los navegadores de los clientes y el servidor web es el http. De tal manera que el usuario final tiene acceso únicamente a pantallas de vista y de ninguna manera a los

archivos fuentes, ya que estos son procesados en el servidor de aplicación y con permiso de acceso local y por lo tanto tampoco tendrá acceso a la base de datos.

6.5 Interfaz de usuario

En cualquier sitio web intervienen diferentes tecnologías. Es fundamental el uso de páginas HTML, lenguajes *script* para realizar algunas operaciones y validaciones en el lado del cliente, acceso a bases de datos y servidor web entre otras. También se tomó como herramienta de presentación la utilización de Hojas de Estilos (CSS) que permiten estandarizar la apariencia en todas las pantallas y optimizar líneas de código.

Para realizar las interfaces de cada módulo del sistema primeramente se construyeron los prototipos de éstas de acuerdo a las características y funcionalidades con las que deberían contar. Para el diseño de estos prototipos se utilizó Microsoft Visual Studio, esta herramienta permite realizar de manera sencilla el diseño de interfaces con todos los elementos convencionales (cuadros de texto, botones, *combo box*, *frames*, etc.) y crear *Master Pages*. La característica que las *Master Pages* nos proporcionan es la habilidad de definir una estructura y elementos de interfaz comunes para nuestro sitio, tales como la cabecera de página o la barra de navegación, en una ubicación común para ser compartidos por varias páginas del sitio. Esto mejora la administración y mantenimiento de nuestro sitio y evita la duplicación innecesaria de código para estructuras o comportamientos del sitio que son comunes.

A partir de la idea establecida en los prototipos de las interfaces gráficas se construyeron los formularios tanto del menú principal como de los diferentes submódulos que constituyen el sistema. Con el código HTML de cada formulario y hojas de estilo se transformaron los formularios en interfaces gráficas amigables y fáciles de usar para el usuario.

Con base a lo mencionado anteriormente a continuación se muestran los prototipos realizados para los submódulos del sistema.

6.6 Aplicación web en .NET

La aplicación web con Visual Studio .NET (o proyecto) está integrada por los siguientes elementos:

a) La configuración de red de la aplicación (puerto, dominio, seguridad, etc.)
b) Un directorio virtual en IIS llamado SIGED, en el cual están los archivos que conforman la aplicación y controlan el acceso a los mismos. La creación de este se explica de forma más detallada en el Anexo I "Manual Técnico".
c) Un archivo de configuración llamado web.config en el cual se describen las variables públicas dentro de la aplicación y la cadena de conexión a la base de datos del sistema.

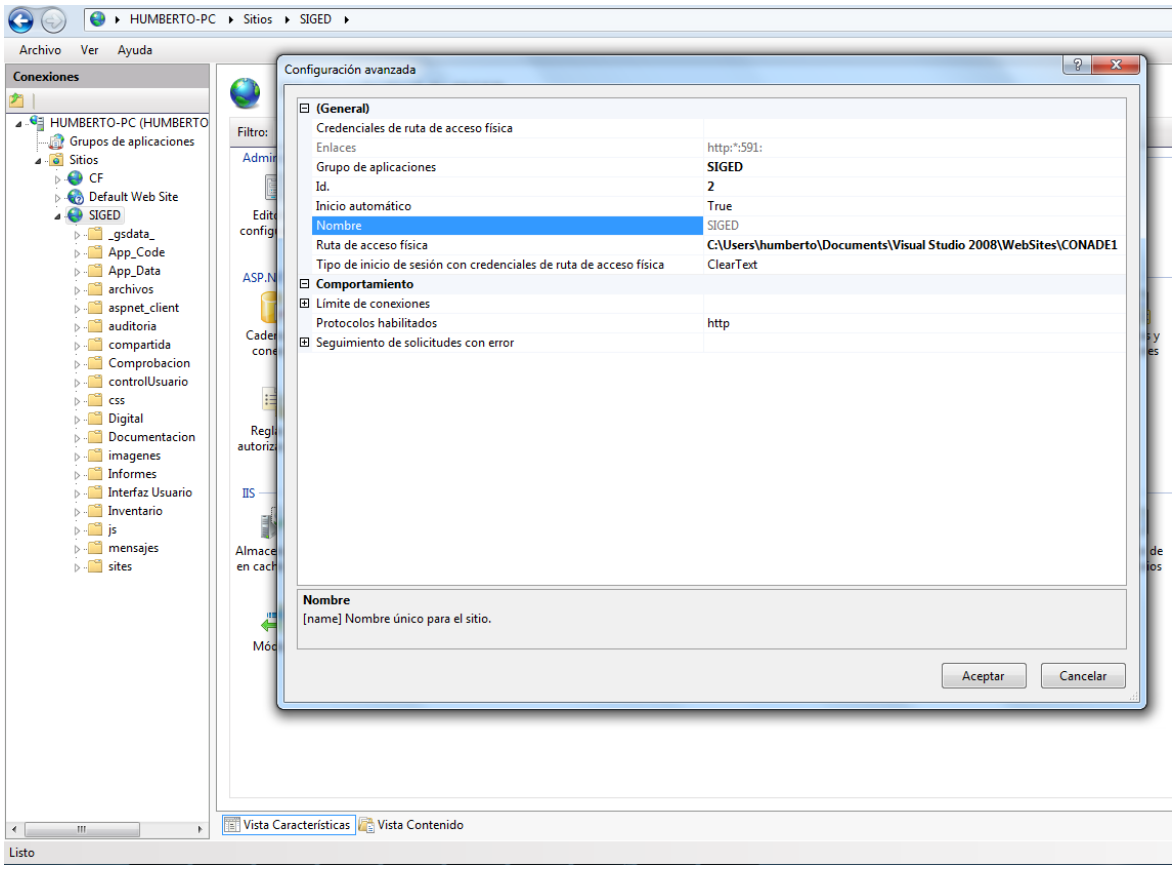

Figura 6.5 Configuración avanzada de IIS 7

El proyecto está dividido en diferentes directorios que representan cada uno de los módulos del sistema. Dentro de cada subdirectorio se encuentran los archivos *webform* que contienen la interfaz de usuario, así como los archivos de Código que describen el funcionamiento del sistema. Estos directorios se describen en el Anexo I "Manual Técnico".

6.7 Base de Datos

El diseño de una base de datos se compone de tres etapas: diseño conceptual, lógico y físico. La etapa del diseño lógico es independiente de los detalles de implementación que depende del tipo de Sistema de gestión de base de datos (SGBD o DBMS) que se vaya a utilizar. La salida de esta etapa es el esquema lógico y la documentación que lo describe. Todo ello es la entrada para la etapa que viene a continuación, el diseño físico.

Mientras que en el diseño lógico se especificó qué se guarda, en el diseño físico se especifica cómo se va a guardar. Por ello, es importante que se tenga y se conozca muy bien toda la funcionalidad del DBMS concreto que se va a utilizar así como la plataforma de desarrollo sobre la que se va a trabajar. El diseño físico no es una etapa aislada, ya que algunas decisiones que se tomen durante su desarrollo, por ejemplo para mejorar las prestaciones, pueden provocar una reestructuración del esquema lógico.

Desde capítulos anteriores se hizo referencia al uso del paradigma de base de datos relacional para la operación del SIGED e, incluso, ya se ha abordado el diagrama entidad-relación y esta etapa del proceso de desarrollo traducirá ese modelo a una estructura lógica administrada por SQL Server 2008, acompañándolo de un soporte de procedimientos almacenados que darán funcionalidad a las pantallas y rejillas desplegadas en todos los módulos del sistema.

6.8 Seguridad

En la actualidad todo desarrollo de sistemas debe comprender lineamientos y parámetros mínimos de seguridad que respondan a mantener los principios de confidencialidad, disponibilidad e integridad de la información que almacenarán y administrarán y siguiendo esta línea es sistema incluye las siguientes líneas de defensa:

a) Contraseñas no almacenadas en la base como texto plano, sino como un valor hash creado a partir de la contraseña asignada al usuario y un algoritmo conocido para tal fin.
b) Bloqueo de cuentas de usuario tras repetidos intentos fallidos de ingreso.
c) Validación de longitud máxima de caracteres y de no inserción de caracteres especiales en cada uno de los elementos de ingreso de información al sistema.

d) Cada página busca y verifica los datos de sesión iniciada y ante cualquier movimiento en los datos del sistema, registra la dirección IP origen de la petición, fecha, hora y navegador desde el cual se hizo la consulta y/o petición al servidor.
e) Política de respaldos frecuentes de la aplicación.
f) Variables de sesión y cache con tiempo límite de vida de 30 minutos.

Estas medidas de seguridad buscan otorgar parámetros mínimos de seguridad y son complementarias a las ya existentes dentro de las instalaciones de la Dirección de Planeación y Tecnologías de la Información de la CONADE.