UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMA DE INFORMACIÓN DEL POSGRADO DE LA UNAM, MÓDULO DE INFRAESTRUCTURA

T E S I N A

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A:

ALONSO BARRÓN GUILLERMO

ASESOR DE TESINA:
ING. ORLANDO ZALDÍVAR ZAMORATEGUI

MÉXICO, D.F. 2010

Í N D I C E G E N E R A L

Objetivo e introducción

Objetivo

Diseñar e implementar el Módulo de Infraestructura, que incluye tres módulos: altas, bajas y edición del Sistema de Información del Posgrado de la UNAM (SIPUMA).

Introducción

El presente trabajo de tesina se desarrolla para cubrir objetivos como son el llevar a la práctica los conocimientos adquiridos en la Facultad de Ingeniería en la carrera de Ingeniería en Computación y concluir el servicio social con el programa registrado ante la Dirección General de Orientación y Servicios Educativos (DGOSE) con la clave 2005-12/6-921 que tiene por objetivo el diseñar, desarrollar e implementar sistemas informáticos que permitan organizar y administrar diversos procesos en instancias dentro de la Dirección General de Servicios de Cómputo Académico (DGSCA), así como en otras dependencias universitarias.

Desarrollar un sistema para facilitar el control de la infraestructura[1] con la que cuentan los posgrados de la Universidad Nacional Autónoma de México (UNAM), con lo cual se podrá obtener un programa que gestione la información administrativa que se requiere.

[1] Conjunto de elementos o servicios que se consideran necesarios para el funcionamiento de una organización o para el desarrollo de una actividad, por ejemplo: computadoras, carros, etc.

Por tal motivo es necesario desarrollar un sistema hecho a la medida para cubrir las necesidades que tiene la Dirección General de Estudios de Posgrado (DGEP).

El sistema deberá contar con una base de datos relacional que permita manejar la información que un sistema de este tipo requiere; se pretende que tenga un ambiente visual donde, de forma intuitiva, el usuario pueda navegar de forma ágil y segura por cada uno de los menús.

Lo que se pretende lograr con un sistema intuitivo para el usuario es que éste ingrese la menor cantidad de información ya que ésta deberá estar contenida en catálogos.

Toda la información de los catálogos será actualizada constantemente, para que el sistema preste un mejor servicio al usuario.

Se espera que la solución generada, proporcione beneficios tangibles, para la DGEP y los usuarios, facilitando la administración y la operación de la misma, agilizando y potenciando la toma de decisiones, así como los tiempos de respuesta.

La tesina ha sido estructurada de la siguiente forma:

En el primer capítulo se presenta un marco teórico, donde se muestra brevemente una introducción a la ingeniería de software. También se incluye una

introducción a las bases de datos, en la que se expone una clasificación de las mismas y se explican las fases por las que hay que pasar para tener un buen diseño de la base de datos y, finalmente, se incluye una clasificación de los lenguajes de programación tomando en cuenta ciertos criterios.

Dentro del capítulo segundo se explica la forma de cómo se realizaron cada uno de los componentes que integran el Módulo de Infraestructura del SIPUMA, los que consideran tres etapas:

1. **Planteamiento y análisis del problema:** Es la etapa en la que se recopiló información para diseñar la interfaz, así como las restricciones y las facilidades que debe tener el sistema.
2. **Implementación:** Es donde se diseña y programan las funciones que realizará la interfaz.
3. **Evaluación:** Es cuando se realizan las pruebas pertinentes para verificar el funcionamiento del sistema y un vez depurado se hace la presentación del software al usuario final para comprobar que se cumple con los requerimientos planteados.

El tercero contiene dos apartados, cada uno de ellos referido a diferentes actividades que se realizaron en el tiempo que se prestó el servicio social, las cuales fueron:

1. **Videos informativos.** Consistió en la creación de una aplicación para mostrar videos informativos de algunos programas de posgrado.

2. **Mapa interactivo de la UNAM.** Es una aplicación que permite mostrar la ubicación de los diversos edificios que se encuentran en la UNAM.

Finalmente, dentro del cuarto capítulo se presentan los resultados, el impacto y conclusiones que se obtuvieron durante el diseño e implementación del SIPUMA.

Capítulo I

Marco teórico

I.1 Procesos del software

Un proceso del software es un conjunto de actividades que conduce a la creación de un producto. Estas actividades pueden consistir en el desarrollo de software desde cero; sin embargo, cada vez más, se desarrolla software ampliando y modificando los sistemas existentes. Los procesos del software aumentan de complejidad y dependen de las personas que toman decisiones y juicios. Debido a la necesidad de juzgar y crear, los intentos para automatizar estos procesos han hecho que surjan las herramientas de ingeniería asistida por computadora (CASE, Computer Aided Software Engineering), sin embargo, no existe la posibilidad de automatizar el diseño creativo del software.

Aunque existen muchos procesos diferentes de software, algunas de las actividades fundamentales son comunes para todos:

1. **Especificación del software.** Se debe definir la funcionalidad del software y las restricciones en su operación. Existen cuatro fases principales.

   a) Estudio de viabilidad. Se estima si las necesidades del cliente se pueden satisfacer con las tecnologías de software y hardware actuales. El estudio analiza si el sistema propuesto es rentable desde el punto de vista de negocios y si se puede desarrollar dentro de las restricciones del presupuesto existente. Existen cinco tipos:

   - Técnica. Determina si se puede desarrollar e implementar la solución con la tecnología existente.

- Económica. Determina si se cuenta con los fondos necesarios para desarrollar e implementar la solución.
- Legal. Determina que no exista ningún conflicto entre el sistema a desarrollar y la capacidad de la organización para descargar sus obligaciones legales.
- Operacional. Consiste en determinar si el análisis y el diseño están basados en los procedimientos existentes que operan la organización.
- De programa. El diseño del sistema debe ser capaz de ser operativo dentro de algún plazo de tiempo.

b) Obtención y análisis de requerimientos. Es el proceso de obtener requerimientos con base en la observación de sistemas existentes y la discusión con los usuarios potenciales.

c) Especificación de requerimientos. Es la actividad de traducir la información recopilada en la fase de análisis a un documento que define un conjunto de requerimientos.

d) Validación de requerimientos. Esta actividad comprueba la veracidad y consistencia de los requerimientos (ver figura 1).

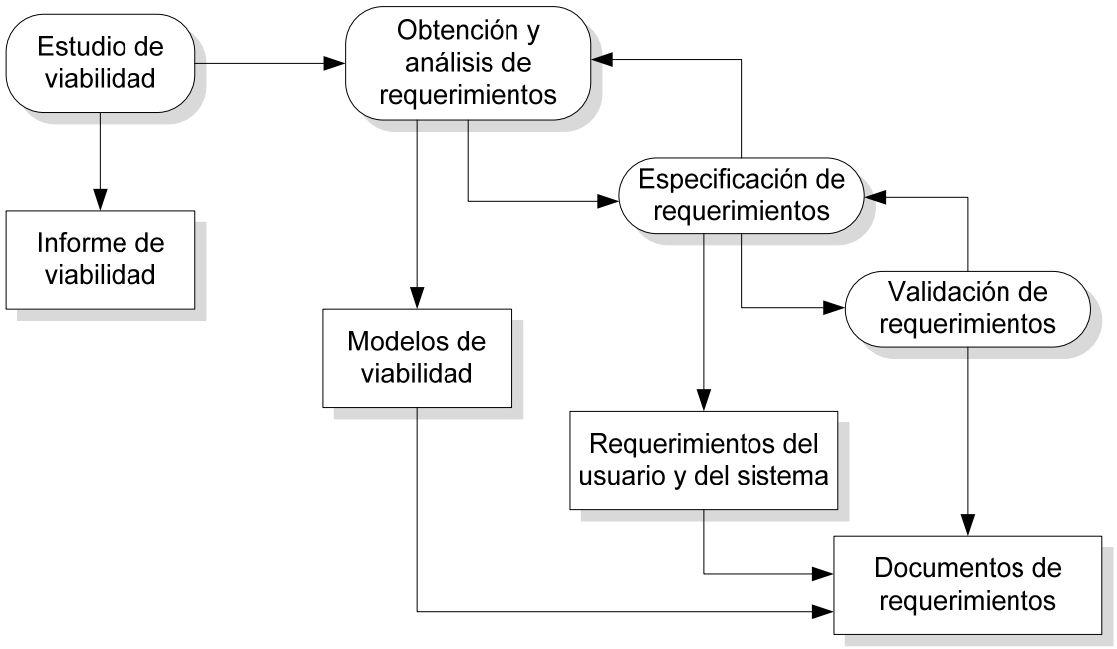

Fig. 1. El proceso de ingeniería de requerimientos

2. Diseño e implementación del software. Se debe producir un software que cumpla su especificación. Las actividades específicas del proceso de diseño son:

a) Diseño arquitectónico. Los subsistemas que forman el sistema y sus relaciones se identifican y documentan.

b) Especificación abstracta. Para cada subsistema se produce una especificación y las restricciones bajo las cuales deben funcionar.

c) Diseño de la interfaz. Para cada subsistema se diseña y documenta su interacción con otros sistemas.

d) Diseño de componentes. Se asignan servicios a los componentes y se diseñan sus interfaces.

e) Diseño de estructura de datos. Se diseñan en detalle y especifica la estructura de datos utilizada en la implementación del sistema.

f) Diseño de algoritmos. Se diseñan y especifican los algoritmos utilizados para proporcionar los servicios.

La figura 2 muestra de manera esquemática el modelo general del proceso del diseño.

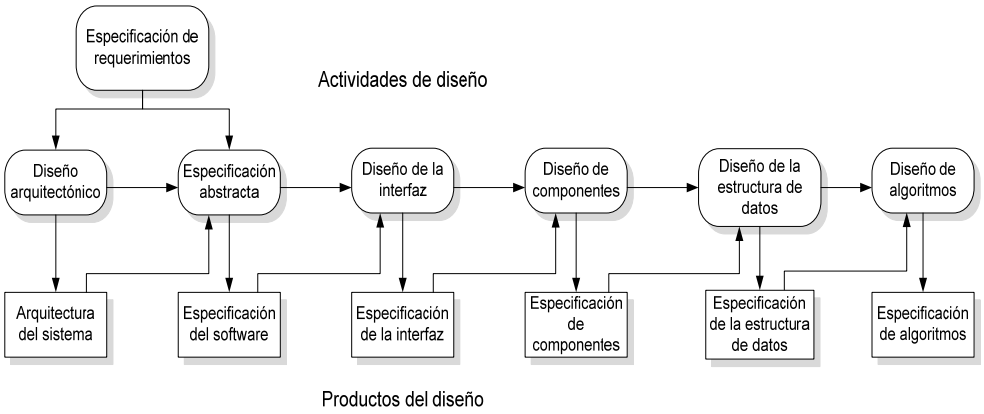

Fig. 2. Modelo general del proceso de diseño

3. Validación del software. Se debe validar el software para asegurar que hace lo que el cliente desea. Las etapas del proceso de pruebas son:

a) Prueba de componentes (o unidades). Se prueban los componentes individuales para asegurarse de que funcionan correctamente. Los componentes pueden ser unidades simples como funciones o clases de objetos o pueden ser agrupaciones coherentes de estas unidades.

b) *Prueba del sistema.* Los componentes se agrupan para formar el sistema. Este proceso comprende encontrar errores que resultan de interacciones no previstas entre los componentes y su interfaz.

c) *Prueba de aceptación.* Es la etapa final en el proceso de pruebas antes de poner en funcionamiento el sistema. Éste se prueba con datos reales proporcionados por el cliente; la prueba de aceptación puede revelar los errores y omisiones en la definición de requerimientos del sistema.

La figura 3 muestra el diagrama correspondiente a la validación del software.

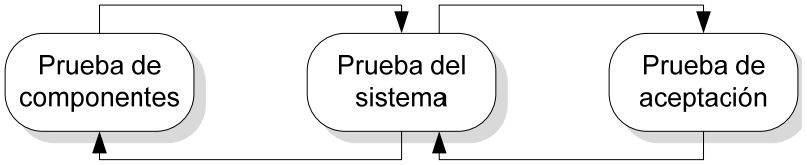

Fig. 3 Validación del software

I.1.1 Modelos del proceso de software

Es una descripción de un proceso del software que se presenta desde una perspectiva particular.

Estos modelos generales no son descripciones definitivas de los procesos del software, más bien, son abstracciones que se pueden utilizar para explicar diferentes enfoques del desarrollo de software.

Los modelos de procesos más usados son:

1. **El modelo en cascada.** Considera las actividades fundamentales del proceso de especificación, desarrollo, validación y los representa como fases separadas del proceso, tales como:

   a) *Análisis y definición de requerimientos.* Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios.

   b) *Diseño del sistema y del software.* El proceso de diseño del sistema divide los requerimientos en sistemas hardware o software. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.

   c) *Implementación y prueba de unidades.* El diseño de software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.

   d) *Integración y prueba del software.* Los programas o las unidades individuales del programa se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de las pruebas, el sistema de software se entrega al cliente.

   e) *Funcionamiento y mantenimiento.* El sistema es instalado y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores, mejorar la

implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos.

La figura 4 muestra de forma esquemática el ciclo de vida del software.

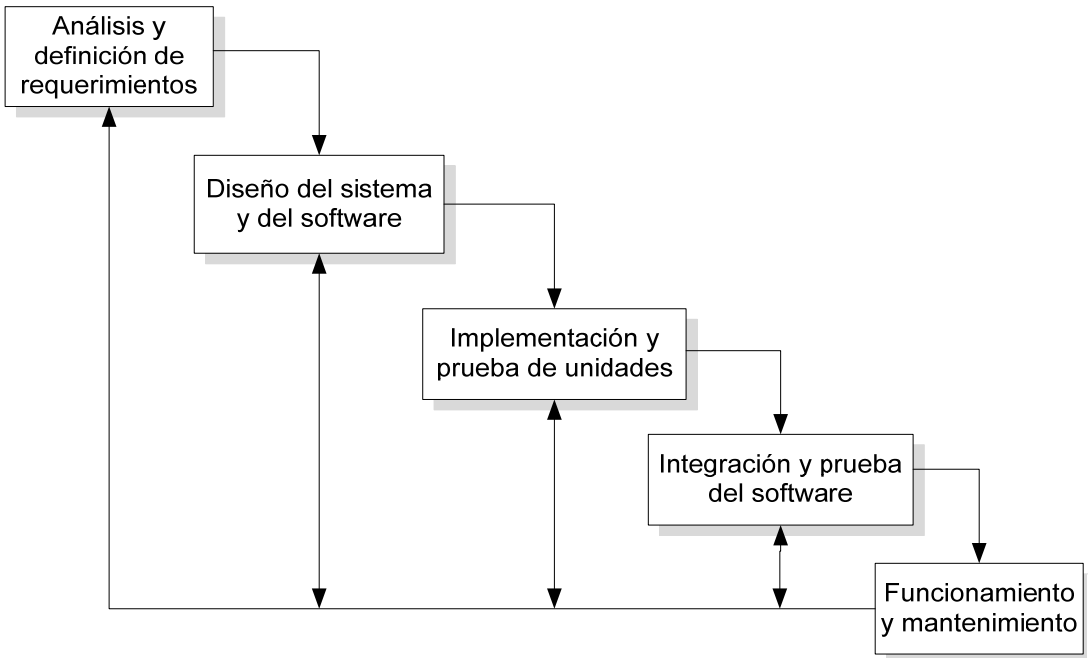

Fig. 4. El ciclo de vida del software

2. Desarrollo evolutivo. Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado (ver figura 5).

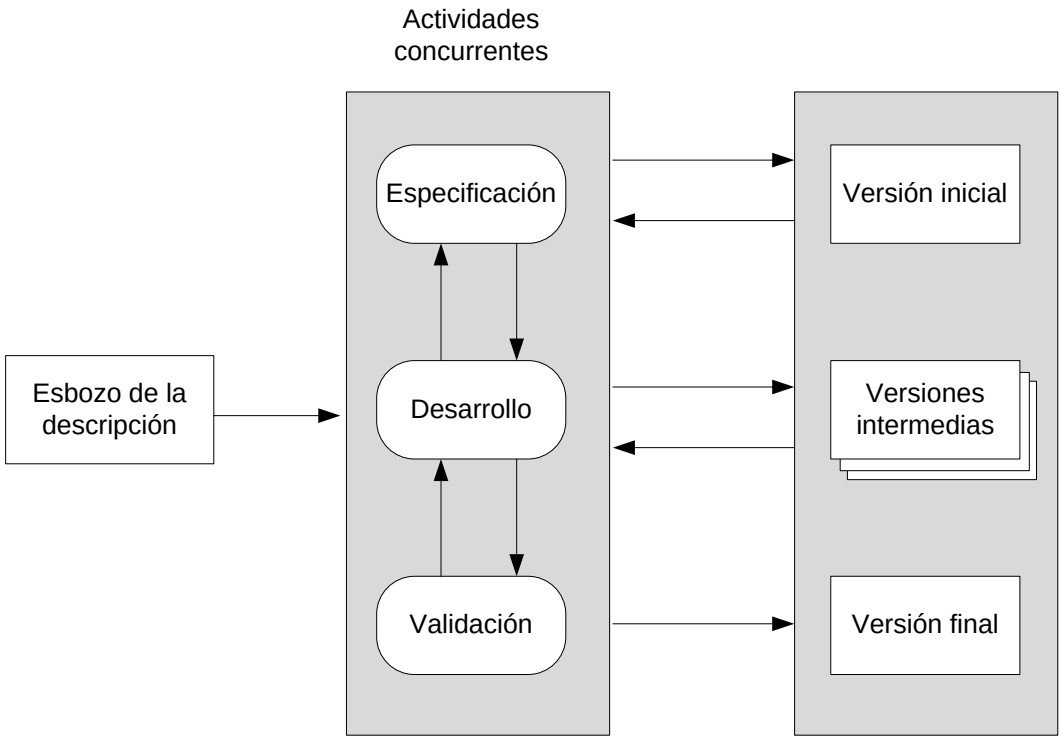

Fig. 5. Desarrollo evolutivo

Existen dos tipos de desarrollo evolutivo.

a) *Desarrollo exploratorio.* Donde el objetivo del proceso es trabajar con el cliente para explorar sus requerimientos y entregar un sistema final. El sistema evoluciona agregando nuevos atributos propuestos por el cliente.

b) *Prototipos desechables.* Donde el objetivo del proceso de desarrollo evolutivo es comprender los requerimientos del cliente y entonces desarrollar una definición mejorada de los requerimientos para

el sistema. El prototipo se centra en experimentar con los requerimientos del cliente que no se comprenden del todo.[2]

3. Desarrollo en espiral. Es un modelo del proceso de software evolutivo que conjuga la naturaleza iterativa y la reducción de riesgos del modelo de construcción de prototipos con los aspectos controlados y sistemáticos del modelo en cascada. El modelo en espiral se divide en un número de actividades de marco de trabajo, también llamadas regiones de tareas. La figura 6 representa un modelo en espiral que contiene cuatro regiones de tareas:

a) Definición de objetivos. Se definen los objetivos específicos; se establecen las restricciones del proceso y el producto, y se estipula un plan de administración. Se identifican los riesgos del proyecto y se plantean estrategias alternativas.

b) Evaluación y reducción de riesgos. Se realiza un análisis detallado para cada uno de los riesgos del problema.

c) Desarrollo y validación. Después del análisis de riesgos, se elige un modelo para el desarrollo del sistema.

d) Planear la siguiente fase. El proyecto se revisa y se toma la decisión de si se debe continuar con un ciclo posterior de la espiral.

[2] SOMMERVILLE, Ian. *Ingeniería de Software.* Pearson Educación, 7ª Edición, Madrid. 2005. Págs. 59–85.

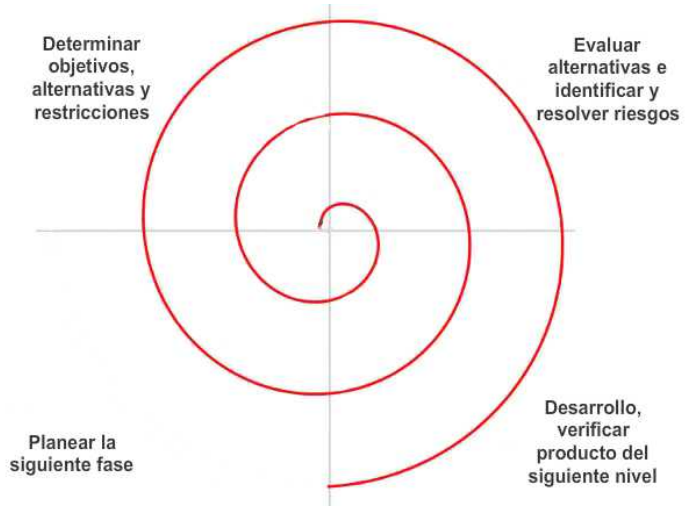

Fig.6 Desarrollo en espiral

I.2 Lenguajes de programación

Los lenguajes de programación comenzaron a surgir a partir de los años sesenta, atendiendo a diversos enfoques, características y propósitos. Se puede decir que día con día aparecen nuevos lenguajes de programación que prometen hacer mejor uso de los recursos computacionales y facilitar el trabajo de los programadores.

La existencia de tantos lenguajes se debe a que cada uno de ellos está encaminado a resolver ciertas tareas, dentro de la amplia problemática de la información, o bien, a que su arquitectura o forma de llevar a cabo la programación tienen un enfoque particular.

De acuerdo con el estilo de organización, podemos clasificarlos en:

1. **Imperativos.** Basan su funcionamiento en un conjunto de instrucciones secuenciales, las cuales, al ejecutarse, van alterando las regiones de memoria donde residen todos los valores de las variables involucradas en el problema a resolver.

2. **Declarativos.** En este paradigma, lo que nos interesa es el qué deseamos obtener a través del programa en lugar del cómo desarrollarlo.

3. **Orientados a objetos.** Este último paradigma se enfoca en los objetos que se van a manipular, y no en la lógica para manipularlos, cada uno tendrá ciertas funciones *(métodos)* y características que los identifican. Cabe mencionar, con un poco más de detalle, los elementos fundamentales que debe poseer este tipo de lenguajes:

a) Abstracción. Determinación de las características de los objetos, que sirven para identificarlos y hacerlos diferentes a los demás.

b) Encapsulación. Es el proceso que agrupa y almacena los elementos que definen la estructura y el comportamiento de una abstracción, en un mismo lugar.

c) Modularidad. Es la propiedad de agrupar las abstracciones que guardan cierta relación lógica y a la vez minimizar la interdependencia entre las diversas agrupaciones.

d) Jerarquía. Consiste en establecer un orden o una clasificación de las abstracciones.

Si se toma como referencia las herramientas usadas en el proceso de traducción y ejecución de los programas, se tiene la siguiente clasificación.

1. **Lenguajes ensamblados.** Se refiere al lenguaje ensamblador, que es una representación simbólica de las instrucciones correspondientes a cada arquitectura, por lo general la correspondencia entre este lenguaje y el de máquina es de uno a uno.

2. **Lenguajes compilados.** Son aquellos que se traducen de un lenguaje de alto nivel a lenguaje máquina, produciendo un programa objeto permanente.

3. **Lenguajes interpretados.** Estos lenguajes no generan ningún código objeto, sino que cada instrucción es analizada y ejecutada a la vez, lo que ofrece mucha interacción con los usuarios, pero a la vez resultan ineficientes cuando se desea ejecutar repetidamente un programa.

4. **Lenguajes preprocesados.** Como primer paso son traducidos a un lenguaje intermedio de más bajo nivel, para posteriormente volverlos a traducir y producir el programa objeto. Estos tipos de lenguajes fueron creados con la idea de proporcionar un lenguaje más potente que el lenguaje intermedio.

Si se toma en cuenta la evolución histórica y el entorno de programación, tenemos la siguiente clasificación:

1. **Lenguajes de cuarta generación (4GL).** Estos lenguajes se distinguen por formar parte de un entorno de desarrollo, que comprende el manejador de base de datos y todo lo que de esto se deriva.

2. **Lenguajes visuales.** Se les llama de esta manera a los lenguajes que forman parte de una aplicación que está dotada de una interface gráfica, la cual por medio de iconos y otras herramientas visuales y simbólicas, pretende facilitar las tareas de los programadores, como son el diseño y desarrollo de formularios e informes.

3. **Metalenguajes.** Son lenguajes que sirven para definir otros lenguajes.

4. **Lenguajes de propósito específico.** Son desarrollados con la finalidad de resolver problemas específicos, para cálculos científicos y de ingeniería.

5. **Lenguajes script.** Son lenguajes que se utilizan en ambientes cliente-servidor con la finalidad de permitir la programación del lado del cliente y de esta manera hacer más atractivas las interfaces gráficas.

I.3 Bases de datos

Una base de datos es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior.

Las bases de datos se pueden clasificar de acuerdo a su modelo de administración de datos. Un modelo de datos es básicamente una “descripción” de cómo contener los datos, los modelos no son cosas físicas, son abstracciones que

permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a algoritmos y conceptos matemáticos.

Los modelos de base de datos se pueden clasificar de la siguiente manera:

1. **Modelo primitivo o sistema basado en archivos.** Los objetos o entidades de interés están representadas por registros que a su vez se encuentran almacenados en archivos. Las relaciones entre los objetos son representadas utilizando archivos de diversos tipos.

2. **Modelo de datos tradicional.** A su vez se divide en:

a) Modelo jerárquico. Almacena su información jerárquicamente. Está organizado de manera similar a un árbol invertido, en donde un nodo padre de información puede tener varios nodos hijos. El nodo que no tiene padres es llamado raíz y los nodos que no tienen hijos se les conoce como hojas.

Las bases de datos jerárquicas son útiles en caso de manejar un gran volumen de información y datos muy compartidos, permitiendo crear estructuras estables y de gran rendimiento; su principal limitación es su incapacidad de representar eficientemente la redundancia de datos.

b) Modelo de red. Este modelo es ligeramente diferente al modelo jerárquico, ya que permite que un mismo nodo tenga varios nodos padre. Esta mejora reduce el problema de la redundancia de datos, pero la dificultad que significa administrar una base de datos de red ha significado que sea un modelo poco utilizado.

c) Modelo de datos relacional. Es utilizado para modelar problemas reales y administrar datos dinámicamente. Su idea fundamental es el uso de "relaciones". Estas relaciones pueden considerarse en forma lógica como conjuntos de datos llamados "tuplas". Para conceptualizar de una manera más sencilla, podemos imaginar cada relación como si fuese una tabla que está compuesta por registros, que representarían las tuplas, y los campos como las columnas de la tabla.

En este modelo no importa el lugar y la forma en que se almacenen los datos, esto presenta una ventaja considerable ya que es más fácil de entender y utilizar. El lenguaje más habitual para construir las consultas es SQL (*Structured Query Language)*.

d) Modelo orientado a objetos. Es un modelo reciente, el cual trata de almacenar en la base de datos los objetos completos (estado y comportamiento). Incorpora todos los conceptos importantes del paradigma de objetos:

- Encapsulación. Permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
- Herencia. A través de la cual los objetos heredan comportamientos y propiedades de las clases a las que pertenecen.
- Polimorfismo. Propiedad que permite tener procedimientos diferentes, asociados a objetos distintos, con el mismo nombre.

e) Modelo distribuido. Se puede definir como una colección de bases de datos situadas en diferentes partes bajo control de manejadores

de bases de datos separados, ejecutándose en computadoras diferentes. Todas las computadoras están interconectadas y cada una de ellas tienen capacidad de realizar procesos autónomos.

3. **Modelo de datos semántico.** Es capaz de expresar gran cantidad de interdependencias entre entidades de interés. Esta interdependencia consiste en dos tipos, exclusión e inclusión, permitiendo al modelo representar la semántica de los datos en la base de datos.[3]

I.3.1 Diseño de base de datos

El diseño de una base de datos es un proceso iterativo y existen una infinidad de metodologías para implementarla.

En la figura 7 se muestra el ciclo para el diseño de un sistema de base de datos.

[3] ELMASRI, Ramez y NAVATHE, Shamkant B. *Fundamentos de sistemas de bases de datos*. Pearson Addison Wesley, España, 2007.

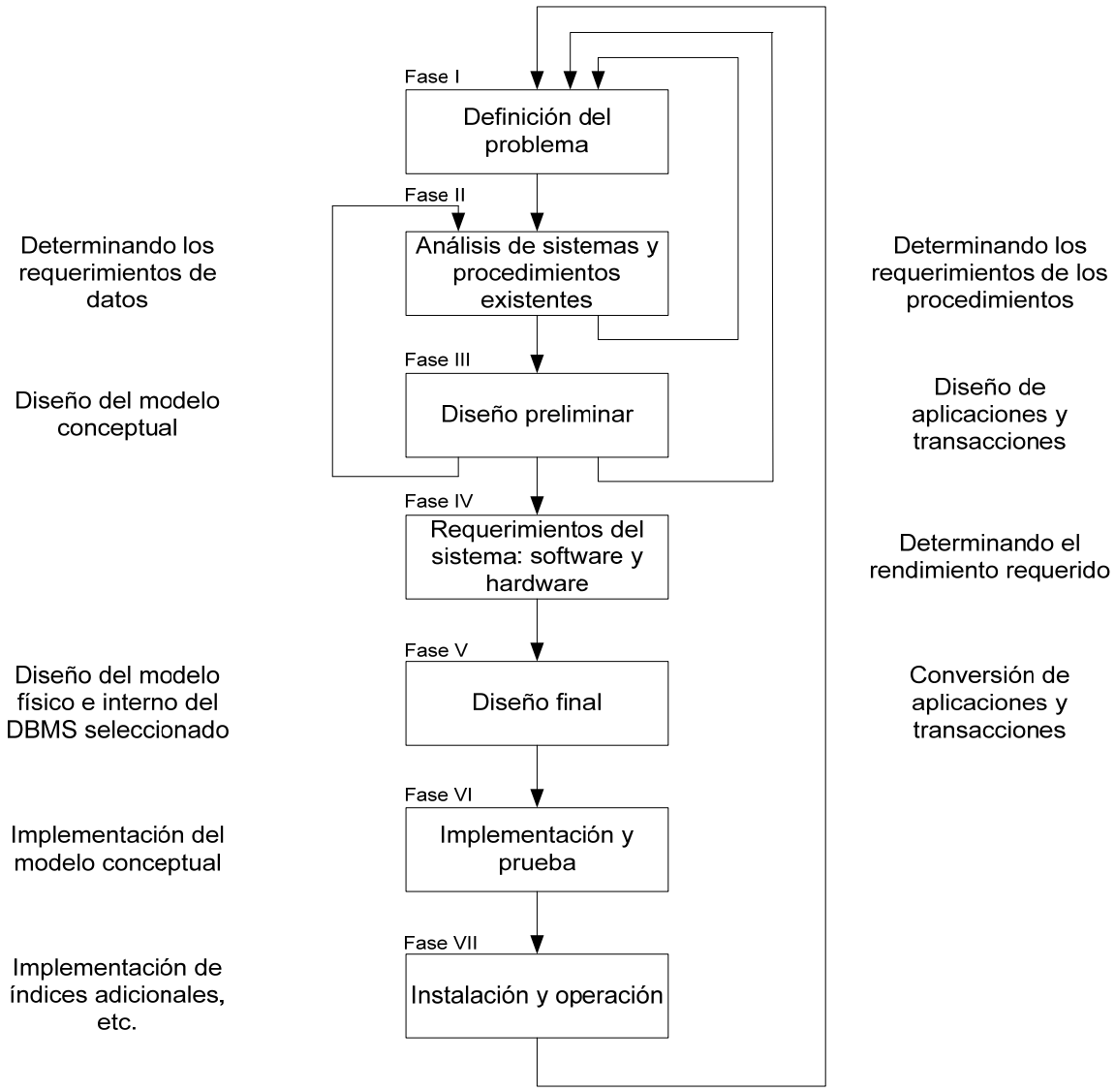

Fig. 7. Ciclo para el diseño de un sistema de base de datos

El diseño comienza con la definición del problema y va a través de un número de pasos, culminando en la instalación y operación del sistema. A continuación se da una breve explicación de cada una de las fases:

1. **Definición del problema.** Es para definir el alcance y el plan a seguir del proyecto. Las alternativas son examinadas y una de ellas es aceptada, se da también la estimación de costos, el riesgo y beneficio son

examinados. Una vez que se ha decidido y aprobado el plan de trabajo se comienza a diseñar el sistema de base de datos.

2. Análisis de sistemas y procedimientos existentes. Esta fase es importante ya que puede modificar el plan de trabajo, es aquí en donde se cuestiona a los trabajadores involucrados con el sistema para conocer su opinión sobre el mismo, a su vez hay que verificar la integridad del software existente y la viabilidad de modificarlo para mejorar su eficiencia. Lo que se obtiene en esta fase es:

- Requerimientos de datos.
- Propiedades e interrelaciones de los datos.
- Requerimientos de operación.

3. Diseño preliminar. El diseño es evaluado contra los requerimientos iniciales, los usuarios son consultados y si se requiere hacer cambios, éstos son reflejados en el diseño.

Estos tres puntos son repetitivos hasta obtener un diseño preliminar satisfactorio. El diseño de un modelo conceptual es independiente del DBMS (Database Manager System, Sistema Manejador de Bases de Datos) lo que permite un mejor entendimiento, que puede describir el contenido de la base de datos sin referencia a su implementación. Un modelo de datos como el diagrama E-R (Entity Relation, Entidad Relación) puede ser muy útil por su simplicidad.

La especificación de requerimientos habría establecido las entidades y las relaciones entre ellas, así como sus atributos. Las llaves primarias de las

entidades, la cardinalidad de las relaciones y las restricciones son especificados en el diseño conceptual de la base de datos.

4. **Requerimientos del sistema: software y hardware.** La decisión puede ser basada en el ambiente de trabajo existente. Si la base de datos es implementada en un sistema existente, la elección está limitada por el DBMS.

Las características de los sistemas son muy importantes, las principales a considerar son la facilidad de generar reportes, utilidades como menús e interfaces de usuario, facilidades de comunicación, etc., otra consideración muy importante en la elección del sistema es la experiencia del personal.

5. **Diseño final.** Una vez que se ha escogido el DBMS, el primer paso es traducir el modelo o esquema conceptual obtenido en el tercer punto al modelo requerido por el DBMS elegido, en este caso en la figura 8 se muestran las reglas a través de las cuales se puede hacer dicha transformación en un DBMS relacional.

Entidad	Relación
1:1	Para cada relación binaria se incluyen los atributos de la clave primaria de la entidad padre en la relación que representa a la entidad hijo, para actuar como una clave foránea.
1:N	Se incluyen los atributos de la clave primaria de la entidad padre en la relación que representa a la entidad hijo, para actuar como una clave foránea. Pero ahora, la entidad padre es la de “la parte de muchos”, mientras que la entidad hijo es la de “la parte del uno”.
M:N	Como una relación incluyendo las llaves primarias de las entidades de la relación.

Fig. 8. Reglas de conversión

6. Implementación y prueba. La implementación consiste en escribir y compilar el código del modelo (scheme) conceptual en el DDL (Data Definition Language, Lenguaje de Definición de Datos) del DBMS. La base física es creada y cargada con los datos de prueba. La aplicación y transacciones son escritas en lenguajes de alto nivel con sentencias de DML (Data Manipulation Language, Lenguaje de Manipulación de Datos) incrustadas.

Una vez implementado, el sistema es sometido a numerosas pruebas para verificar su funcionamiento. Estas pruebas deben ser planeadas cuidadosamente para facilitar la detección de errores.

Es aquí donde se prepara la documentación del sistema, los procedimientos que se deben seguir para una operación correcta y los pasos a seguir en caso de un error.

7. Instalación y operación. En esta fase el ciclo es completado y está listo para usarse. El sistema ha sido probado y el rendimiento actual puede medirse. Si el rendimiento no es satisfactorio se puede mejorar haciendo alguna de las siguientes opciones: incrementando el número de buffers, definiendo índices adicionales, particionando registros o agrupando registros que pueden ser accedidos juntos.[4]

I.3.2 Sistema manejador de bases de datos

El sistema manejador de bases de datos es la porción más importante del software de un sistema de bases de datos. Un DBMS es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de alguna tarea específica.

Las funciones principales de un DBMS son:

- Crear y organizar la base de datos.

[4] ELMASRI, Ramez y NAVATHE Shamkant B. *Fundamentos de sistemas de bases de datos*. Pearson Addison Wesley, España, 2007.

- Establecer y mantener las trayectorias de acceso a la base de datos de tal forma que los datos puedan ser rápidamente localizados.
- Manejar los datos de acuerdo a las peticiones de los usuarios.
- Registrar el uso de las bases de datos.
- Interacción con el manejador de archivos. Esto a través de las sentencias DML al comando del sistema de archivos. Así el manejador de bases de datos es el responsable del verdadero almacenamiento de los datos.
- Respaldo y recuperación. Consiste en contar con mecanismos implantados que permitan fácilmente la recuperación de los datos en caso de ocurrir fallas en el sistema de bases de datos.
- Control de concurrencia. Consiste en controlar la interacción entre los usuarios concurrentes para no afectar la consistencia de los datos.
- Seguridad e integridad. Consiste en contar con mecanismos que permitan el control de la consistencia de los datos evitando que éstos se vean perjudicados por cambios no autorizados o previstos.

El DBMS es conocido también como gestor de bases de datos.

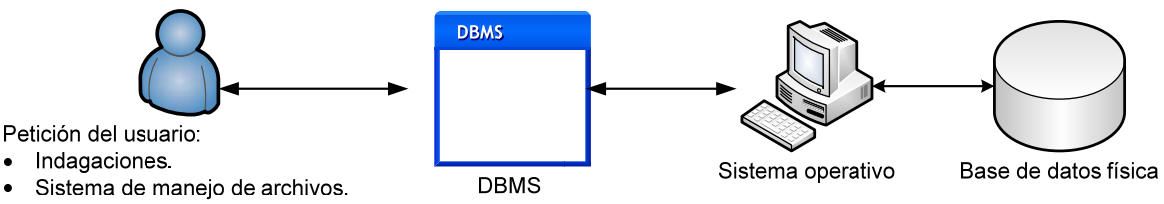

Fig. 9. Sistema manejador de bases de datos

La figura 9 muestra el DBMS como interfaz entre la base de datos física y las peticiones del usuario. El DBMS interpreta las peticiones de entrada/salida del usuario y las manda al sistema operativo para la transferencia de datos entre la unidad de memoria secundaria y la memoria principal.

En sí, un sistema manejador de bases de datos es el corazón de la base de datos ya que se encarga del control total de los posibles aspectos que la puedan afectar.

Como resultado de lo visto sobre base de datos es que se decidió usar en la aplicación el modelo de base de datos relacional y a su vez utilizar el ciclo del sistema de información ya que se observó que es mucho más sencillo realizar los bocetos de la base de datos cuando se conoce a fondo el problema a resolver, además que permite llevar una registro y reducir el tiempo en convertir el diseño lógico al físico y obtener un sistema de base de datos funcional.

I.4 Análisis de las herramientas utilizadas en el sistema

Primeramente hay que mencionar que el Sistema de Información del Posgrado de la UNAM (SIPUMA) fue solicitado por la Secretaría Académica a la Unidad de Sistemas de la Dirección General de Estudios de Posgrado y debido a que el sistema está dirigido para el personal de las Unidades del Posgrado encargados de proporcionar la infraestructura con la que cuenta su posgrado, éste va a ser instalado en un servidor web para que sea de fácil acceso a todas las unidades y a su vez a la información que proveerán; por lo que se realizó un análisis de las condiciones con las que se cuentan y poder tomar una decisión correcta de qué software y hardware se utilizaría para un funcionamiento adecuado.

Al realizar el análisis del problema se tomaron en cuenta los servicios que se quieren ofrecer, tratando de apegarse a las tendencias en los sistemas y servicios de información, se acordó realizar el sistema utilizando el lenguaje de programación Visual Basic .Net 2003 ya que cuenta con un ambiente completamente gráfico que facilita la creación de interfaces gráficas, así como la programación misma, además de trabajar sobre un marco común de librerías (.Net Framework) independiente del sistema operativo, lo que permite hacer una aplicación portable entre los sistemas operativos Windows y finalmente es lenguaje con el cual se han desarrollado sistemas dentro de la dependencia y como servidor de bases de datos SQL Server 2000 ya que cuenta con herramientas que facilitan la creación y administración de las bases de datos y es completamente compatible con lenguaje de programación elegido.

I.4.1 Visual Basic Studio .Net 2003

Visual Basic Studio .Net 2003 es un ambiente de desarrollo completamente gráfico que permite implementar aplicaciones tanto para web como para el sistema operativo Microsoft Windows 98, ME, NT, 2000, XP y 2003 Server. Las aplicaciones creadas en Visual Basic están basadas en objetos y son manejadas por eventos, se deriva de Basic, el cual es un lenguaje de programación estructurado. Sin embargo, Visual Basic .Net emplea un modelo de programación orientado a objetos.[5]

En este lenguaje de programación la ejecución no sigue una ruta predefinida, sino que se ejecutan diferentes secciones de código en respuesta a eventos. Los eventos se desencadenan por acciones del usuario, por mensajes del sistema o de otras aplicaciones. La secuencia de eventos determina el orden en que el código se ejecuta, por tal motivo la ruta que sigue el código es diferente cada vez que se inicia el programa.

Una parte esencial de la programación manejada por eventos es el escribir código que responda a los posibles eventos que pueden ocurrir en una aplicación. Visual Basic .Net facilita la implementación del modelo de programación manejada por eventos.

[5] DOBSON, Rick. *Programación de Microsoft SQL Server 2000 con Microsoft Visual Basic .NET*. McGraw Hill, España, 2002. Págs. 3-29.

Visual Basic Studio .Net 2003 es uno de los programas más populares en el desarrollo de aplicaciones, tanto para programadores expertos como para principiantes, dado que se pueden realizar grandes aplicaciones en poco tiempo y su aprendizaje es muy sencillo. Esto es debido a que es un producto con una interfaz gráfica de muy fácil manejo e intuitiva, además de poder integrar imágenes y multimedia.

La ventaja principal de este lenguaje de programación es su sencillez para programar aplicaciones de cierta complejidad para web y para Windows, además de contar con una capa de software intermedio que deja a un lado el uso de funciones exclusivas del sistema operativo tales como las librerías de enlace dinámico (DLL), y su desventaja es el costo de la licencia que es necesaria para poder crear aplicaciones. En la mayor parte de las aplicaciones, las herramientas aportadas por Visual Basic .Net son más que suficientes para lograr un programa fácil de realizar y de altas prestaciones.

I.4.2 Características, ventajas y desventajas de SQL Server 2000

SQL Server 2000 es el sistema manejador de bases de datos relacionales de Microsoft (RDBMS), que se encarga de mantener la relación entre la información y la base de datos, se asegura de que la información sea almacenada correctamente, es decir, que las reglas que definen las relaciones entre los datos no sean violadas y recupera toda la información en un punto conocido en caso de que el sistema falle.

SQL Server usa la arquitectura Cliente/Servidor para separar la carga de trabajo en tareas diseñadas para servidores y tareas diseñadas para computadoras cliente. La arquitectura cliente es responsable de la parte lógica y de presentar la información al usuario. Generalmente, el cliente se ejecuta en una o más computadoras, aunque también puede correr en una computadora servidor con SQL Server. SQL Server admite los siguientes clientes:

- Windows NT Workstation.
- Windows 2000 Professional.
- Windows 98.
- Windows 95.
- Apple Macintosh.
- OS/2.
- UNIX.

Los recursos disponibles del servidor y las bases de datos son distribuidos y administrados por SQL Server (tales como memoria, operaciones de disco, etc.) entre las múltiples peticiones. Las características de SQL Server 2000 incluyen:

- **Escalabilidad y disponibilidad.** Se puede utilizar en un intervalo de plataformas desde equipos portátiles con Windows 98, hasta grandes servidores con varios procesadores que tengan instalado Windows 2003.
- **Características de base de datos corporativas**. El motor de base de datos relacional de SQL Server admite las características necesarias para satisfacer los exigentes entornos de procesamiento de datos. El

motor de base de datos protege la integridad de los datos a la vez que minimiza la carga de trabajo que supone la administración de miles de usuarios modificando la base de datos simultáneamente.

- **Facilidad de instalación, distribución y utilización.** SQL Server 2000 incluye un conjunto de herramientas administrativas y de desarrollo que mejora el proceso de instalación, distribución, administración y su uso en varios sitios, lo que permite que el empleo de las bases de datos y de los almacenes de datos resulte una parte fluida de la creación de sistemas sólidos y escalables. Estas características permiten entregar con rapidez aplicaciones que los clientes pueden implementar sencillamente.
- **Almacenamiento de datos.** Incluye herramientas para extraer y analizar datos de resumen para el procesamiento analítico en línea. SQL Server incluye también herramientas para diseñar gráficamente las bases de datos y analizar los datos mediante consultas.
    - Soporta XML.
    - Soporte nativo ASP.
    - Vistas indexadas.
    - Vistas particionadas distribuidas.
    - English Query.
    - Servicios de metadata.
    - Data mining.[6]

[6] http://www.microsoft.com/spain/sql/2000/productinfo/caracteristicas.aspx

Microsoft SQL Server 2000 puede proporcionar los servicios de bases de datos necesarios para sistemas extremadamente grandes, por ejemplo: los servidores de gran tamaño pueden tener miles de usuarios conectados a una instancia de SQL Server 2000 al mismo tiempo, dispone también de protección total para estos entornos, cuenta con medidas de seguridad que evitan problemas al tener varios usuarios intentando actualizar los mismos datos al mismo tiempo, así mismo asigna también de manera muy eficaz los recursos disponibles, como memoria, ancho de banda de la red y E/S del disco, entre los distintos usuarios.

Los sitios de internet extremadamente grandes pueden dividir sus datos entre varios servidores, extendiendo la carga de procesamiento entre varios equipos y permitiendo que el sitio sirva a miles de usuarios simultáneos.

Las aplicaciones diseñadas para SQL Server 2000 se pueden ejecutar en el mismo equipo donde se encuentra instalado, lo que permite implementar SQL Server en sistemas pequeños en los que una aplicación debe almacenar los datos localmente.

Los sitios web de mayor tamaño y los sistemas de procesamiento de datos a nivel corporativo generan a menudo un mayor procesamiento de base de datos del que puede admitir un único equipo. En estos grandes sistemas, los servicios de base de datos vienen proporcionados por un grupo de servidores que forman una batería bien sincronizada para manejar altos volúmenes de información.

Ventajas de SQL Server 2000

Algunas de sus principales ventajas son:

- Facilidad de uso. Maneja un ambiente gráfico que facilita las tareas del administrador y del cliente.
- Manejo de múltiples plataformas de hardware.
- Manejo de múltiples aplicaciones de software.
- Familiar al usuario. Al estar basado en la plataforma Windows muchas de sus características visuales le son familiares al usuario.
- Expone los datos como servicios en la web.
- Integración con Internet. El motor de base de datos de SQL Server 2000 incluye compatibilidad integrada con XML.
- Escalabilidad y disponibilidad.

Desventajas de SQL Server 2000

Algunas de sus principales desventajas son:

- Costo de la licencia.
- El software y el alojamiento de aplicaciones desarrolladas en SQL Server son de un costo regular en relación a otros DBMS.
- La mayoría de las empresas de hosting no soportan SQL-Server.
- La licencia se vende por procesador o por máquina.
- Únicamente opera bajo Windows.

- Para varias de las tareas administrativas se requiere de la re inicialización del servidor.

Una vez que se analizó el lenguaje de programación y el DBMS, y que se conocieron sus ventajas y desventajas la Dirección General de Estudios de Posgrado sugirió que se utilizaran las herramientas de las cuales se disponen las licencias respectivas para evitar problemas legales de derechos de autor y mayores inversiones de software.

Capítulo II

Desarrollo

II.1 Módulo de altas

Para el diseño e implementación de este módulo se utilizaron dos modelos del proceso de software, el primero fue el modelo en espiral en el cual se dieron dos vueltas en las que se generaron dos prototipos del sistema y posteriormente se dio una vuelta más en la que se utilizó el modelo en cascada para realizar la implementación del producto final, es decir, el módulo de altas.

II.1.1 Planteamiento y análisis del problema

Se desea desarrollar un módulo que permita dar de alta la infraestructura con la que cuentan los posgrados de la UNAM, los datos en común son el programa de posgrado y la entidad académica a la que pertenece, los demás dependen de la infraestructura seleccionada.

El sistema deberá contar con una base de datos relacional que permita manejar la información que un sistema de este tipo requiere; se pretende que tenga un ambiente visual donde de forma intuitiva el usuario pueda navegar de forma ágil y segura por cada uno de los menús.

Una vez conocido el problema, como primer paso para realizar el sistema se obtuvieron los requerimientos y necesidades que demandaba el usuario para ello, junto con el responsable del proyecto, se realizaron entrevistas y se aplicaron

cuestionarios con algunas preguntas tales como: ¿Qué debe hacer el sistema? ¿Qué datos de entrada y salida intervienen en el proceso, cuáles van a ingresar los usuarios y qué restricción o límite tienen? ¿Quién será el usuario del sistema?

II.1.2 Implementación

Después de haber obtenido los requerimientos del sistema; se realizaron propuestas de interfaces gráficas en papel, mismas que se mostraron al responsable del proyecto para que con su experiencia en el área eligiera la más adecuada, es decir, de fácil comprensión para el usuario.

Así mismo, como resultado del análisis se vio la necesidad de incluir en la base de datos (ver figura 10) una tabla para insertar los datos que se capturan a través de la interfaz gráfica (ver figura 11); por tal motivo se comenzó a estudiar el sistema existente para adaptarlo al nuevo requerimiento. Como primer paso se realizó un diseño preliminar el cual se revisó en compañía del encargado del sistema. Una vez aprobado y de haber obtenido la versión final se implementó usando SQL Server 2000, como último paso se probó el buen funcionamiento del sistema.

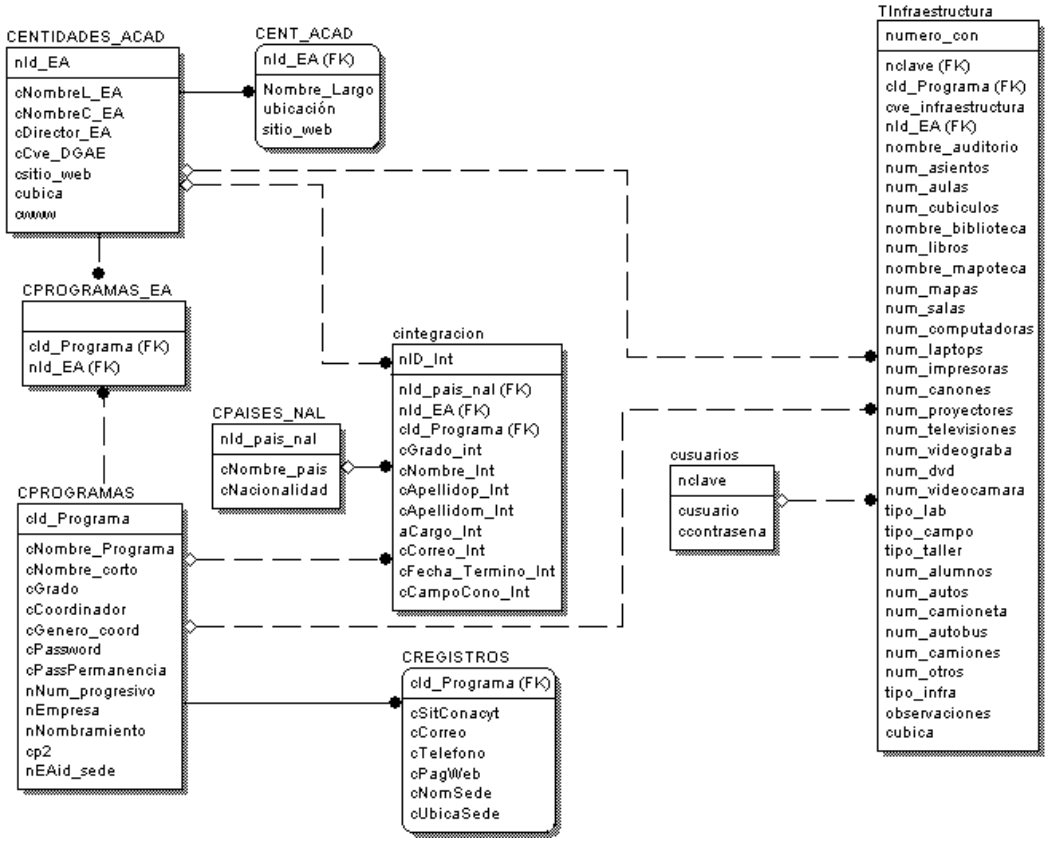

Fig. 10. Diagrama Entidad-Relación de la base de datos

Column Name	Data Type
numero_con	numeric(18, 0)
nclave	smallint
cId_Programa	varchar(3)
cve_infraestructura	int
nId_EA	int
nombre_auditorio	varchar(60)
num_asientos	int
num_aulas	int
num_cubiculos	int
nombre_biblioteca	varchar(60)
num_libros	int
nombre_mapoteca	varchar(60)
num_mapas	int
num_salas	int
num_computadoras	int
num_laptops	int
num_impresoras	int
num_canones	int
num_proyectores	int
num_televisiones	int
num_videograba	int
num_dvd	int
num_videocamara	int
tipo_lab	varchar(60)
tipo_campo	varchar(60)
tipo_taller	varchar(60)
num_alumnos	int
num_autos	int
num_camioneta	int
num_autobus	int
num_camiones	int
num_otros	int
tipo_infra	varchar(60)
observaciones	varchar(60)
cubica	varchar(60)

Fig. 11. Tabla de infraestructura

Posteriormente, antes de concluir con la implementación de la interfaz gráfica del módulo de altas, se tuvieron que presentar los avances del proyecto. En esta cita

se vio, junto con el responsable del proyecto, que se tenían que hacer algunos cambios relacionados con la base de datos, por lo tanto, fue necesario modificar la tabla y hacer las pruebas pertinentes para comprobar que recibía los caracteres requeridos, a su vez, se realizó una modificación más, ya que los datos debían estar ordenados con base a un número que va generando un código de *SQL* que se encuentra en un *Stored Procedure* (ver figura 12), al mismo tiempo, se utilizó esta presentación de avances para aclarar diferentes dudas con respecto a las dependencias de algunos datos que ya estaban incluidos en la base de datos y que eran necesarios para mostrarlos en pantalla al usuario.

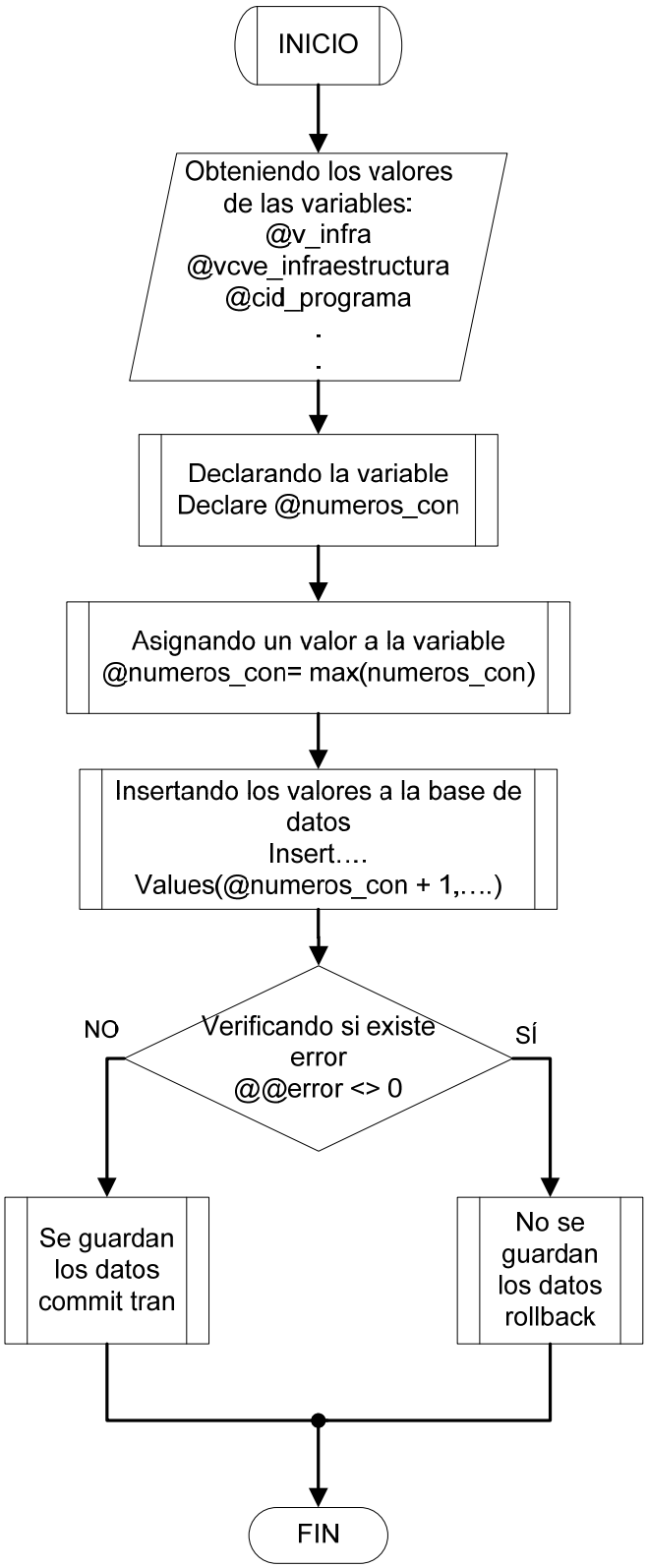

Fig. 12. Diagrama de flujo del stored procedure

Una vez elegida la interfaz gráfica del módulo de altas se comenzó a programarla haciendo uso de *Visual Basic Studio .Net 2003* para aplicaciones *web (ASP.Net)* basándonos en el diagrama de la figura 13 y de la figura 14 a la 17 se muestra a detalle la arquitectura del módulo. Cabe mencionar que además de los conocimientos adquiridos se tuvo que investigar algunas propiedades de las herramientas que se utilizaron para hacer la interfaz gráfica que había aprobado el responsable del proyecto.

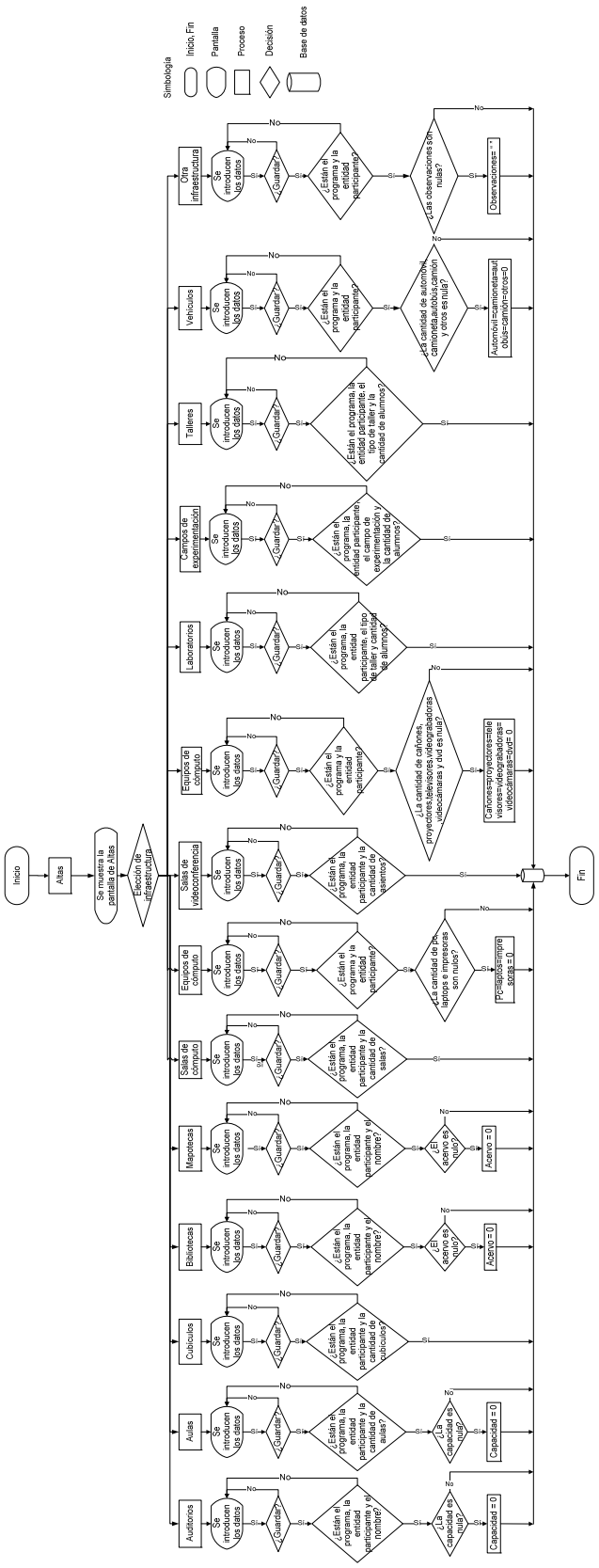

Fig. 13 Arquitectura del módulo de altas

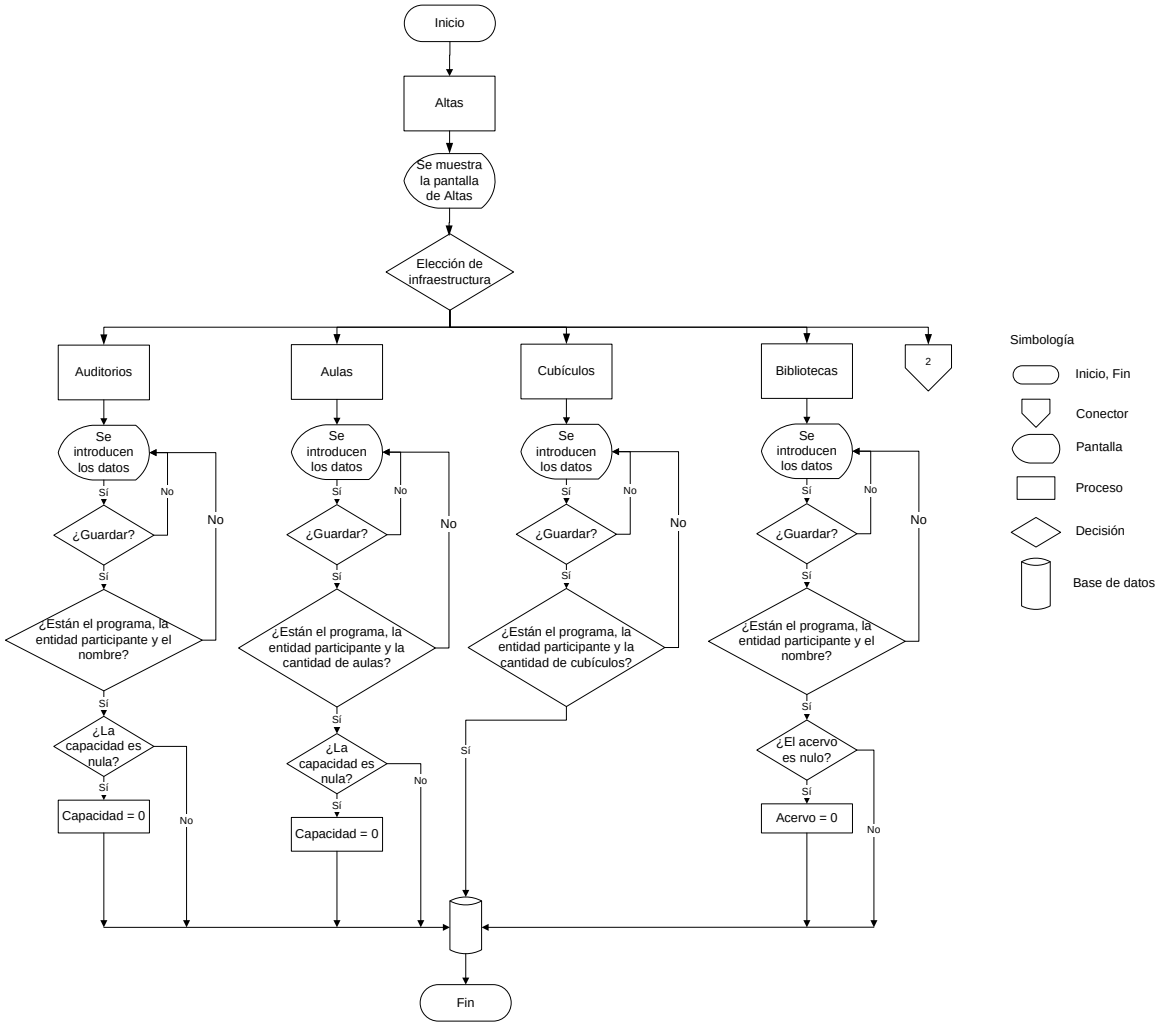

Fig. 14 Arquitectura del módulo de altas (auditorios, aulas, cubículos y bibliotecas)

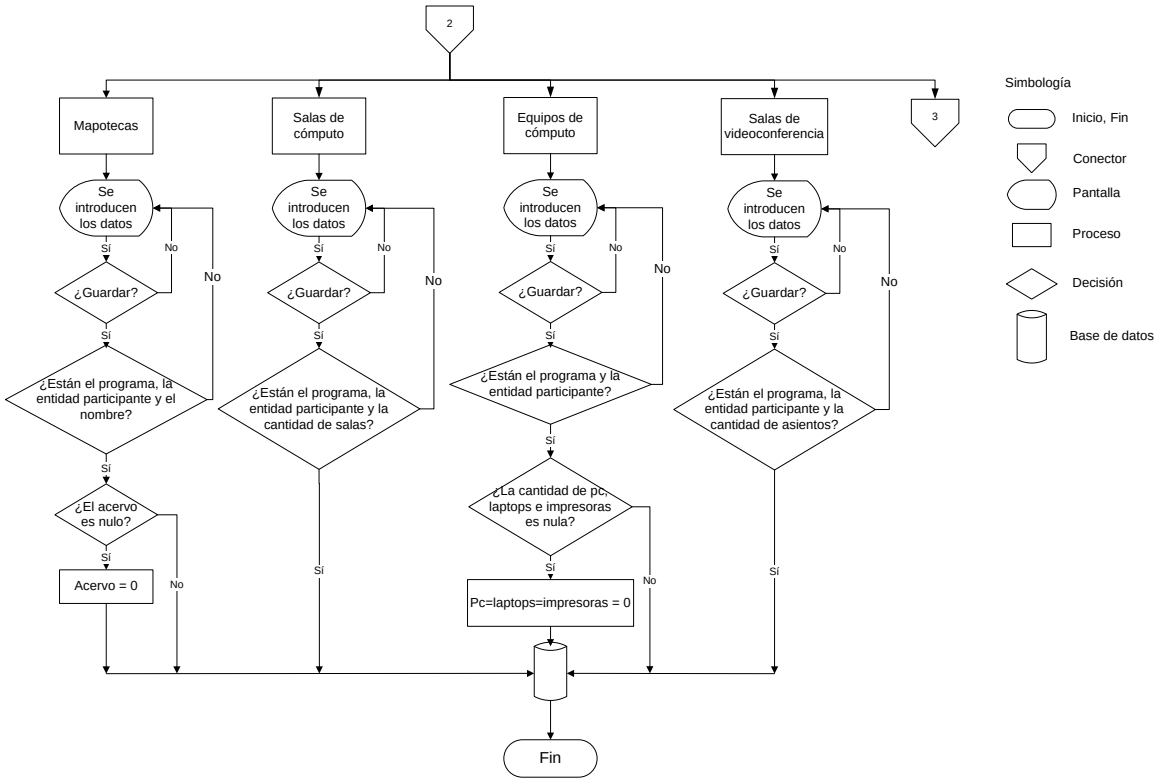

Fig.15 Arquitectura del módulo de altas (mapotecas, salas de cómputo, equipos de cómputo y salas de videoconferencias)

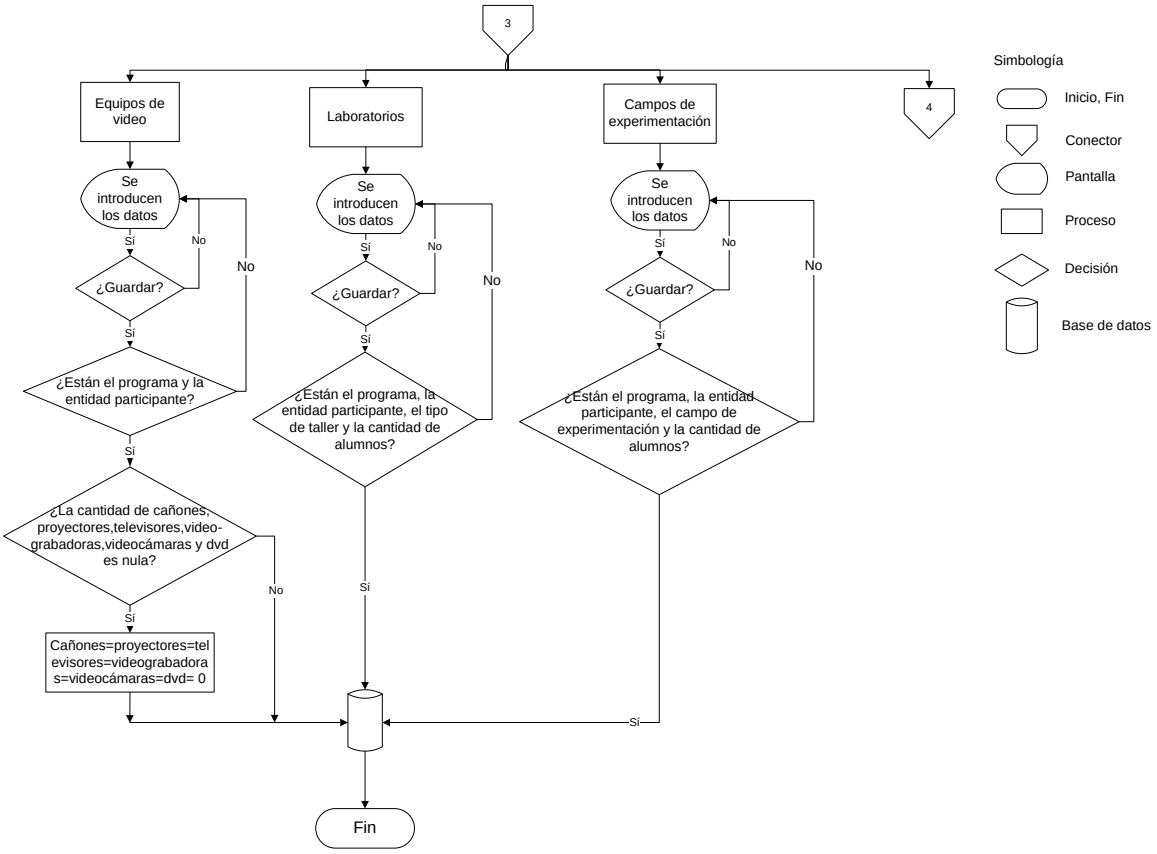

Fig. 16 Arquitectura del módulo de altas (equipos de video, laboratorios, campos de experimentación)

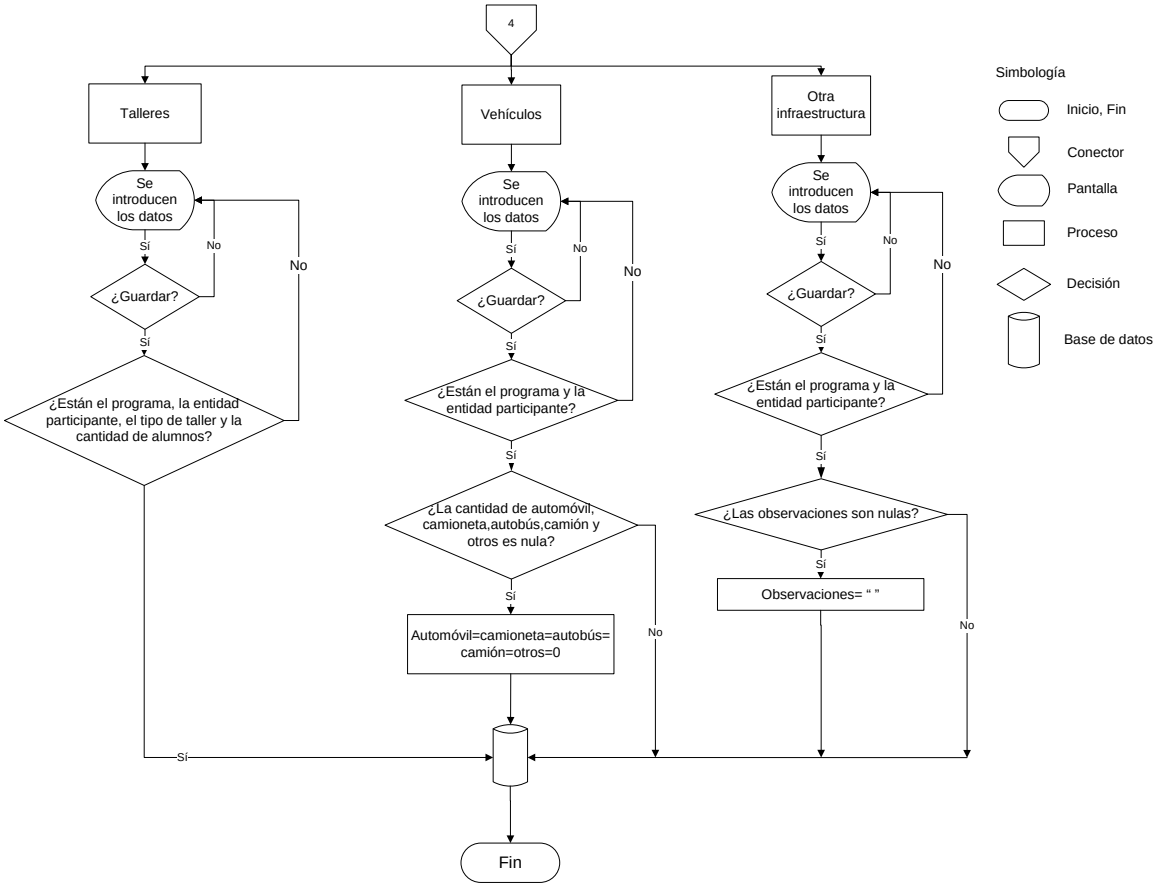

Fig. 17 Arquitectura del módulo de altas (talleres, vehículos, otra infraestructura)

Durante el tiempo en el que se estuvo programando este módulo, se realizaron algunas pruebas para comprobar que el sistema estaba haciendo lo que se había especificado en los requerimientos.

El siguiente fragmento de código forma parte del procedimiento encargado de insertar los datos, ingresados por el usuario, a la base de datos mediante el uso de *stored procedures*.

```
If (ddl_programa.SelectedItem.Text = "-------Seleccione una opción-------")
Then 'Verifica si el usuario ha seleccionado un programa; si no, le manda un
mensaje de aviso
            Response.Write("<Script  language=""javascript"">"  &  Chr(13)  &
"<!--" & Chr(13) & "alert('Hace falta escoger un programa')" & Chr(13) & "--
><" & "/" & "script>")
        Else
```

```
            If   ddl_entidad.SelectedItem.Text   =   "-------Seleccione   una
opción-------"   Then   'Verifica   si   el   usuario   ha   seleccionado   una
ubicación(entidad), si no, manda un mensaje de aviso
                Response.Write("<Script language=""javascript"">" & Chr(13)
& "<!--" & Chr(13) & "alert('Hace falta escoger una entidad')" & Chr(13) &
"--><" & "/" & "script>")
            Else
                Select Case ddlopcion_infra.SelectedItem.Value
                    Case 1 'Toma los valores de la intefaz de Auditorios y
los manda a la base de datos
                        rfv_aulas.Enabled = False
                        rfv_cubil.Enabled = False
                        rfv_salas.Enabled = False
'Se declaran las variables que van a recoger los datos de la interfaz
gráfica
                        Dim    vcve_infraestructura    As    SqlParameter    =
cmdtest1.Parameters.Add("@vcve_infraestructura", SqlDbType.Int, 4)
                        Dim     vcid_programa     As     SqlParameter     =
cmdtest1.Parameters.Add("@vcid_programa", SqlDbType.NVarChar, 3)
                        Dim      vnid_EA      As      SqlParameter      =
cmdtest1.Parameters.Add("@vnid_EA", SqlDbType.Int, 4)
                        Dim    vnombre_auditorio    As    SqlParameter    =
cmdtest1.Parameters.Add("@vnombre_auditorio", SqlDbType.VarChar, 60)
                        Dim     vnum_asientos     As     SqlParameter     =
cmdtest1.Parameters.Add("@vnum_asientos", SqlDbType.Int, 4)
'Se define la cadena de conexión y el procedimiento almacenado a utilizar
                        cmdtest1.Connection = conn
                        cmdtest1.CommandType = CommandType.StoredProcedure
                        cmdtest1.CommandText = "pa_insertaauditorios_infra"
                        RetValue.Direction = ParameterDirection.ReturnValue
                        vcve_infraestructura.Direction                    =
ParameterDirection.Input
                        vcid_programa.Direction = ParameterDirection.Input
                        vnid_EA.Direction = ParameterDirection.Input
                        vnombre_auditorio.Direction                       =
ParameterDirection.Input
                        vnum_asientos.Direction = ParameterDirection.Input
                        v_infra.Direction = ParameterDirection.Input
'Se asignan los valores a las variables
                        v_infra.Value = "Auditorio"
                        vcve_infraestructura.Value                        =
ddlopcion_infra.SelectedItem.Value
                        vcid_programa.Value                               =
ddl_programa.SelectedItem.Value
                        vnid_EA.Value = ddl_entidad.SelectedItem.Value
                        vnombre_auditorio.Value = txt_name_auditorio.Text
                        If txt_capacidad_auditorio.Text <> "" Then
                            vnum_asientos.Value                           =
CType(txt_capacidad_auditorio.Text, Integer)
                        Else
                            vnum_asientos.Value = 0
                        End If
                        conn.Open()
                        cmdtest1.ExecuteScalar()
                        errorfuntion(RetValue.Value)
```

II.1.3 Evaluación

Por último, ya con el módulo revisado y aprobado, se concertó una reunión con el usuario para mostrarle el sistema. En esta entrevista se explicó al usuario la utilización y el funcionamiento del módulo de altas. Como resultado de esta cita, se recibió la autorización para empezar a diseñar el siguiente módulo del sistema, que tiene por título bajas.

Las figuras 18,19 y 20 son imágenes que pertenecen al módulo de altas.

Fig. 18. Página de inicio del módulo de altas

Fig. 19. Agregando información de la infraestructura seleccionada

Fig. 20. Enviando la información a la base de datos

II.2 Módulo de bajas

Para realizar el diseño y la implementación de este módulo se aplicaron dos modelos del proceso de software, el primero que se utilizó fue el modelo en espiral en el cual se generaron dos prototipos del sistema y posteriormente utilizó el modelo en cascada para realizar la implementación del producto final, es decir, el módulo de bajas.

II.2.1 Planteamiento y análisis del problema

Se requiere un módulo que permita dar de baja la infraestructura deseada de la base de datos del sistema, los datos obligatorios que se mostrarán para identificar el registro son el programa de posgrado y la entidad académica a la que pertenece, los demás dependen de la infraestructura seleccionada.

Una vez que se planteó y analizó el problema, al igual que en el módulo anterior, se concertaron citas con los usuarios para recabar información y obtener los requerimientos y necesidades para este nuevo módulo.

II.2.2 Implementación

En este caso, el diseño de la interfaz gráfica fue de menor complejidad que en el módulo anterior, ya que sólo bastó con mostrar la información necesaria que

permitiera al usuario identificar la tupla de información que desea eliminar de la base de datos e incluir un botón o una liga para realizar la acción. Conociendo lo anterior se comenzó a programar el módulo usando como base el diagrama de la figura 21.

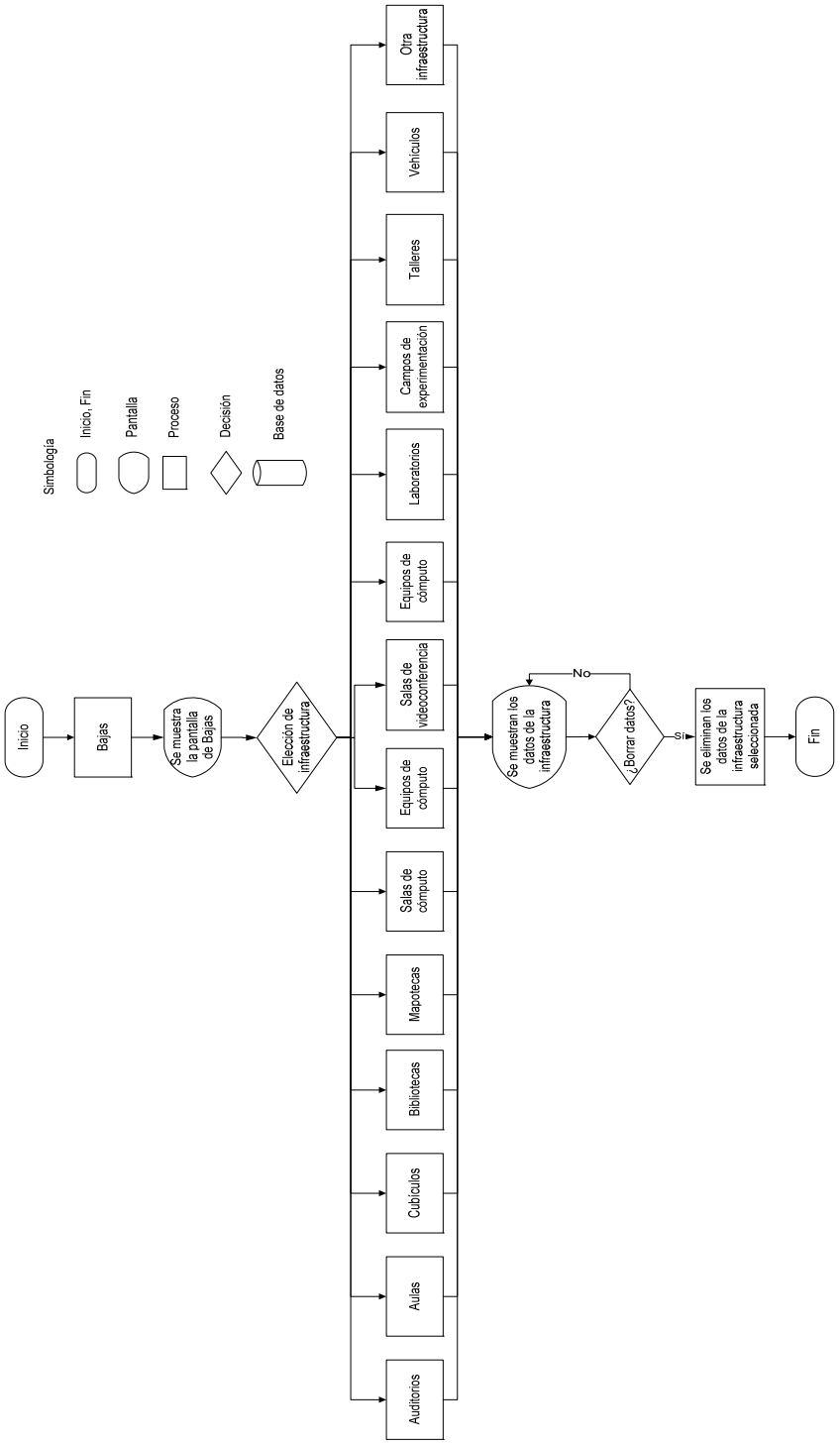

Fig.21 Arquitectura del módulo de bajas

Además se agregó un procedimiento almacenado (s*tored procedure*) dentro de la base de datos para realizar y facilitar la eliminación de la tupla seleccionada (ver figura 22) y otro para mostrar los datos de la infraestructura que el usuario haya seleccionado (ver figura 23).

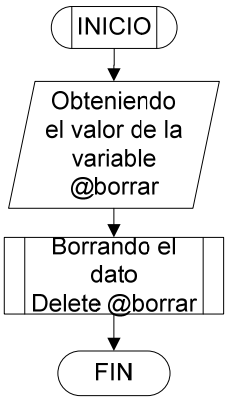

Fig. 22. Diagrama de flujo del procedimiento almacenado utilizado para borrar

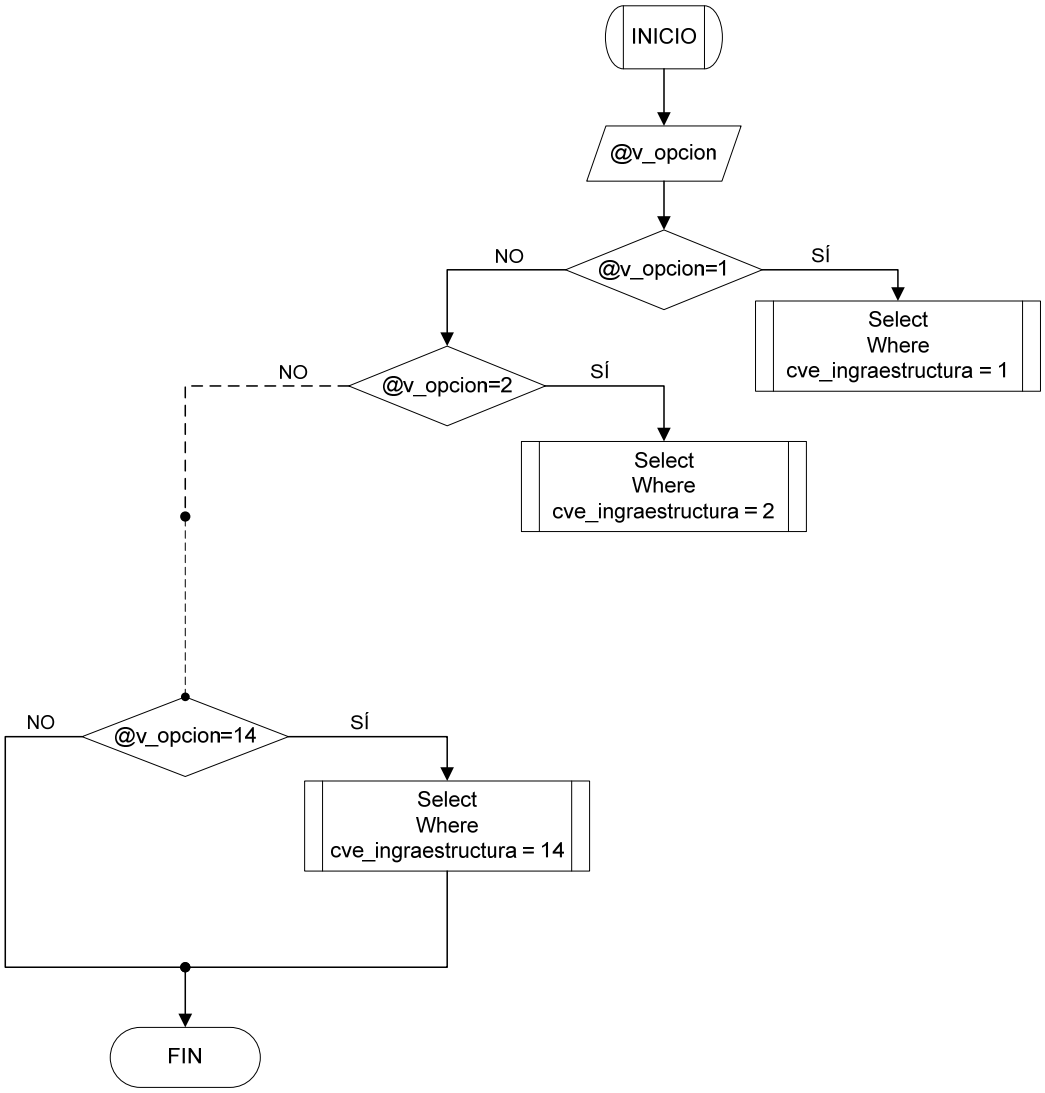

Fig. 23. Diagrama de flujo del procedimiento almacenado utilizado para mostrar la información

Esta implementación se logró colocando un objeto, incluido en *Visual Studio* llamado *Data Grid,* el cual en esencia es una especie de tabla que nos permite mostrar y ordenar la información recopilada de una tabla de la base de datos, además de poder agregar controles u objetos, tales como: botones, cajas de texto (*textbox*),etc.

A su vez se incluyó, como parte de la interfaz con el usuario, un mensaje de confirmación para proceder a eliminar la tupla seleccionada y un mensaje más

avisando que se ha borrado la tupla o en su caso si hubo algún error al momento del borrado de la tupla (ver figura 24). Cabe mencionar que este mensaje está hecho en *JavaScript,* ya que la versión 2003 de *Visual Basic Studio,* por implementar programación de lado del servidor, no incluía alguna función similar. A continuación se muestra un fragmento el código utilizado para realizar el borrado del dato seleccionado por el usuario:

```
'Este procedimiento es el encargado de eliminar el dato seleccionado de la
base de datos.
Private Sub dg_base_datos_DeleteCommand(ByVal source As Object, ByVal e As
System.Web.UI.WebControls.DataGridCommandEventArgs)                  Handles
dg_base_datos.DeleteCommand
'Se declara la cadena de conexión y el procedimiento almacenado encargado de
eliminar el dato
        Dim cmddelete As SqlCommand = New SqlCommand
        Dim  borrar  As  SqlParameter  =  cmddelete.Parameters.Add("@borrar",
SqlDbType.Int, 4)
        Dim numero_con As String = e.Item.Cells(1).Text
        cmddelete.Connection = conn
        cmddelete.CommandType = CommandType.StoredProcedure
        cmddelete.CommandText = "pa_borradato_infra"
        borrar.Direction = ParameterDirection.Input
        borrar.Value = numero_con
        conn.Open()
        cmddelete.ExecuteScalar()
        conn.Close()
        v_opcion.Value = ddlopcion_infra.SelectedItem.Value
        llamado(v_opcion.Value)
    End Sub
'Este es el procedimiento encargado de enviar el mensaje de advertencia al
usuario.
    Private Sub dg_base_datos_ItemDataBound(ByVal sender As Object, ByVal e
As          System.Web.UI.WebControls.DataGridItemEventArgs)          Handles
dg_base_datos.ItemDataBound
        If e.Item.ItemType <> ListItemType.Header And _
             e.Item.ItemType <> ListItemType.Footer Then
            'Se hace referencia al botón ubicado en el DataGrid
            Dim deleteButton As Button = e.Item.Cells(0).Controls(0)
            'Ahora se agrega un evento onclick que lanza el mensaje
            deleteButton.Attributes("onclick") = "javascript:return " & _
                      "confirm('Está seguro que quiere borrar el registro
#" & _
                      DataBinder.Eval(e.Item.DataItem,    "numero_con")    &
"?')"
        End If
    End Sub
```

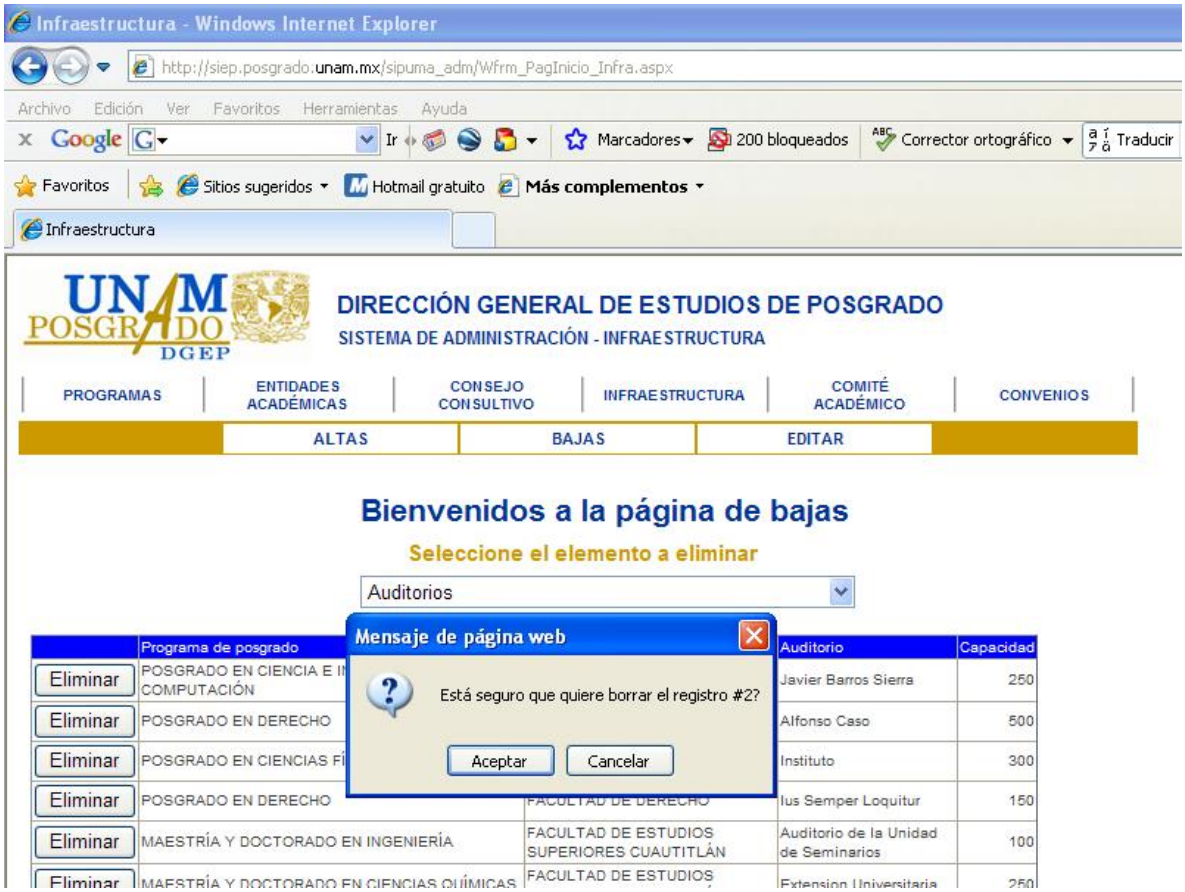

Fig. 24. Dando de baja un elemento de la infraestructura

II.2.3 Evaluación

Una vez que se concluyó con esta fase, fue presentada al usuario final para explicarle el funcionamiento de esta nueva etapa y a su vez se realizaran algunas observaciones, una de ellas consistió en mostrar al usuario páginas de sólo diez tuplas; esto permite ver parte de la información y de esta manera se facilita la búsqueda, a su vez se puede navegar entre las demás páginas en caso de tenerlas. Finalmente, se obtuvo la autorización para continuar con el siguiente módulo que tienen como título edición.

Las figuras 25 y 26 son imágenes que pertenecen al módulo de bajas.

Fig. 25. Inicio del módulo de bajas

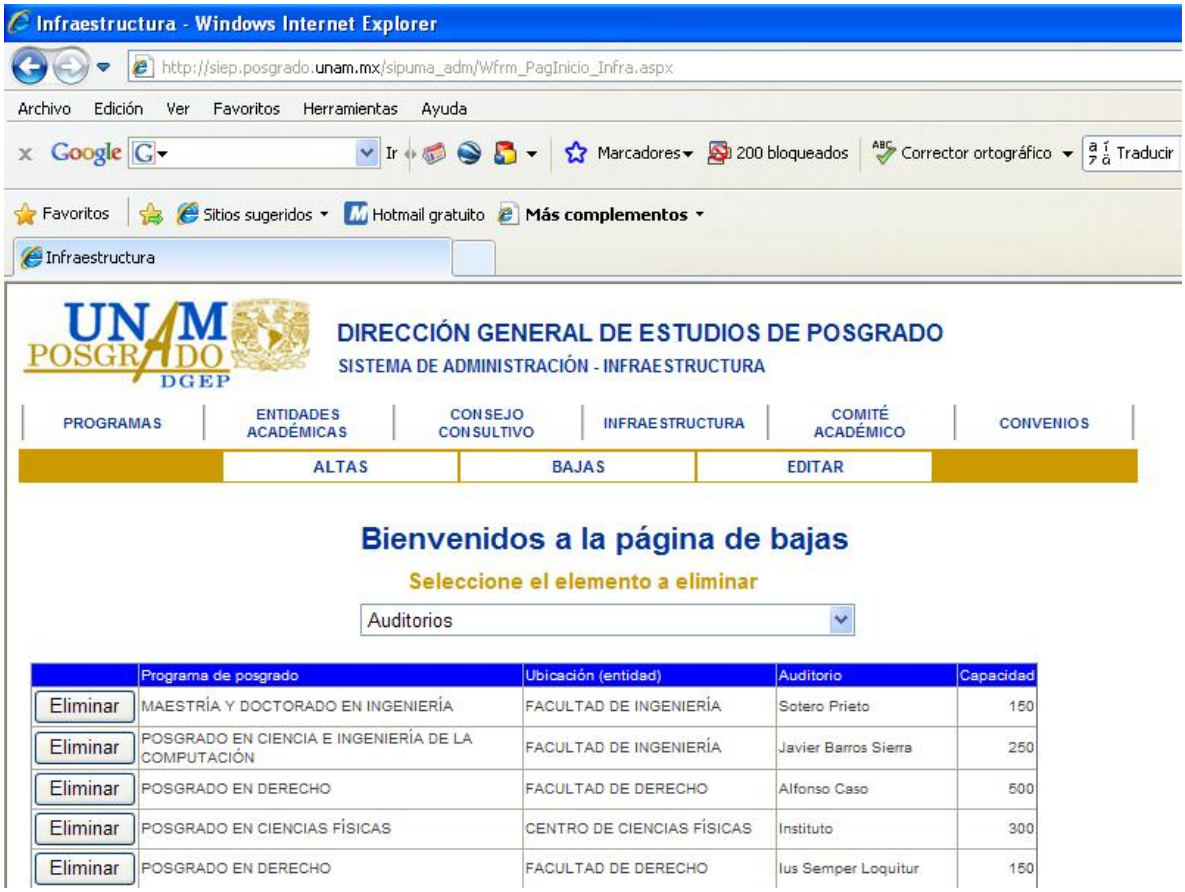

Fig. 26. Mostrando la información de la infraestructura seleccionada

La figura 26 muestra la interfaz aprobada por el usuario del sistema, incluyendo las modificaciones pedidas.

II.3 Módulo de edición

Para desarrollar este módulo se utilizaron dos modelos del proceso de software, el primero fue el modelo en espiral en el cual se dieron dos vueltas en las que se generaron dos prototipos del sistema y posteriormente en una tercer vuelta se utilizó el modelo en cascada para realizar la implementación del producto final, es decir, el módulo de edición.

II.3.1 Planteamiento y análisis del problema

Se requiere un módulo que permita editar la infraestructura deseada y guardar los cambios realizados en la base de datos del sistema, los datos que podrán modificarse del registro son el programa de posgrado, la entidad académica a la que pertenece y los otros campos que dependen de la infraestructura seleccionada.

Después de plantear el problema fue necesario concertar citas con los futuros usuarios del sistema para obtener los requerimientos de este nuevo módulo. Al igual que en los módulos anteriores se aplicaron cuestionarios con algunas de preguntas tales como: ¿Qué debe hacer el sistema? ¿Qué datos de entrada y salida intervienen en el proceso y qué campos pueden ser modificados?

Una vez que se obtuvieron los requerimientos se empezaron a realizar los diseños en papel y mostrarlos al responsable del proyecto para que señalara cuál

sería la mejor opción. Ya que se eligió el diseño se comenzó la implementación del módulo tomando en cuenta los campos que podrían ser modificados por el usuario.

II.3.2 Implementación

Una vez que se comprendió el problema y se diseñó la solución, se comenzó a programar, nuevamente nos ayudamos con el control llamado *Data Grid*, únicamente que en esta ocasión se tuvieron que hacer algunas modificaciones, tales como: agregar cada columna de la tabla de la base de datos, para poder dar formato a cada una de ellas, es decir, para cambiarles el ancho o el alto, agregarles un control diferente a los que el *Data Grid* nos da por defecto y poder ocultarlas a la vista del usuario.

Durante el desarrollo de esta etapa se conoció el concepto de *columna plantilla;* ésta tiene propiedades diferentes a las demás, ya que este tipo de columna nos permite agregar otros tipos de controles al *Data Grid* tales como: *Dropdown List* (*Listas Desplegables*) o *Etiquetas* (*Label*). Con este nuevo concepto se trató de hacer algo similar a lo que se programó en el módulo de altas, es decir, que al momento en que el usuario eligiera una opción dentro de un *Dropdown List,* colocado en una *columna plantilla,* se lanzará una función que llenará otro *Dropdown List*. Cabe señalar que esto es parte de la interfaz gráfica y que el responsable del proyecto lo solicitó.

Por tal motivo se buscó y encontró información al respecto y posteriormente se comenzó un proceso de análisis para entender qué era lo que hacía el código del ejemplo que se había encontrado, el cual utilizaba conceptos tales como la creación de nuevas clases y controles. Sin embargo, gracias a la educación integral que se imparte en la Facultad de Ingeniería y a los maestros que nos exhortan a comprender lenguajes de programación relativamente nuevos, se comenzó analizando parte por parte para, de esta manera, poder entender lo que hacía cada una de ellas.

Cuando se analizaron y probaron cada una de las partes del código se empezó a unirlas, y de esta forma observar, analizar y probar lo que hacían en conjunto.

Una vez concluido el análisis del código encontrado, se comenzó a tomar las partes que servían y con ellas se fue armando un código ejemplo, que se tuvo que seguir depurando hasta obtener lo que se necesitaba. Cabe mencionar que el ejemplo consta de un control de *Visual Studio .NET* llamado *Web User Control* (WUC), es decir un control que el usuario define dependiendo de las necesidades del sistema.

Tras haber realizado una serie de correcciones, se logró que el *Web User Control* funcionara, ahora sólo era necesario incluirlo dentro del módulo, para tal efecto se tuvo que hacer uso del código de *Visual Basic*. Dentro del *Web User Control* se tuvo que llenar el *Dropdown List* (Lista Desplegable) con ayuda de un procedimiento almacenado que devuelve los valores de la tabla de la base de datos de interés, además de algunas funciones del propio lenguaje tales como el uso de *Data Reader* (lectores de datos) y *SQLCommand* (comando de SQL). Este procedimiento lo utilizó

en ambos *Web User Control*. Una vez que el *Web User Control* funcionó se comenzó a programar el módulo utilizando el diagrama de la figura 27 y de la figura 28 a la 31 se ilustra en detalle la arquitectura del módulo.

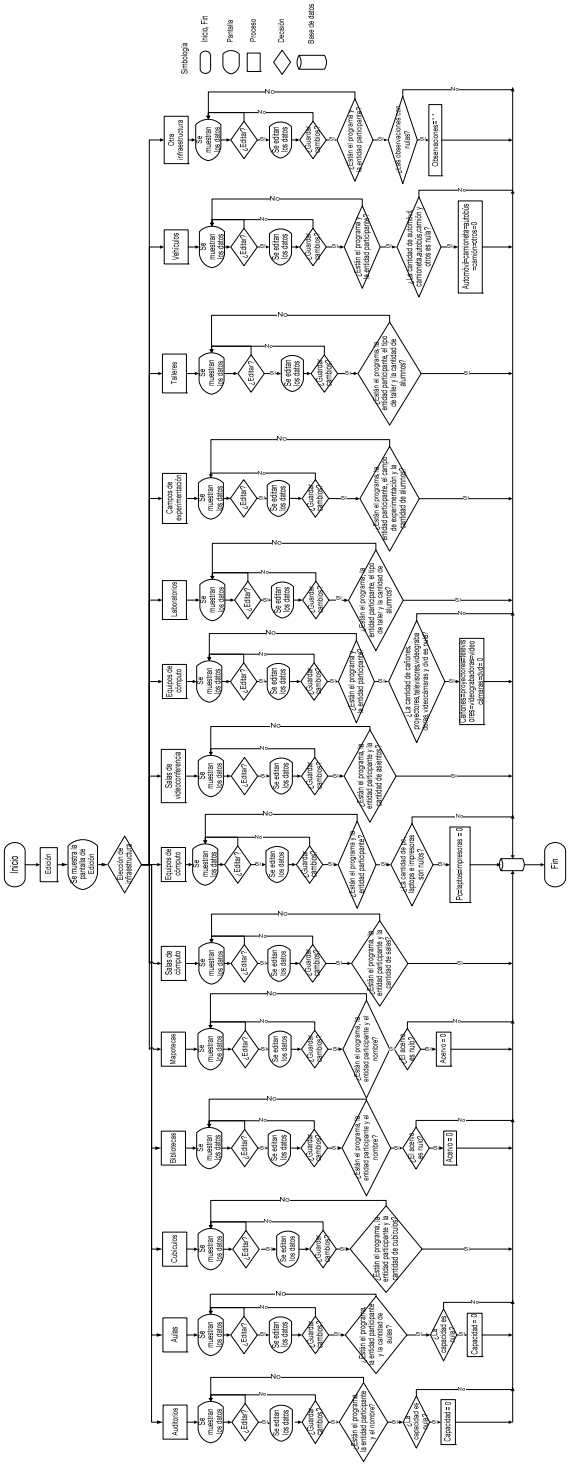

Fig. 27 Arquitectura del módulo de edición

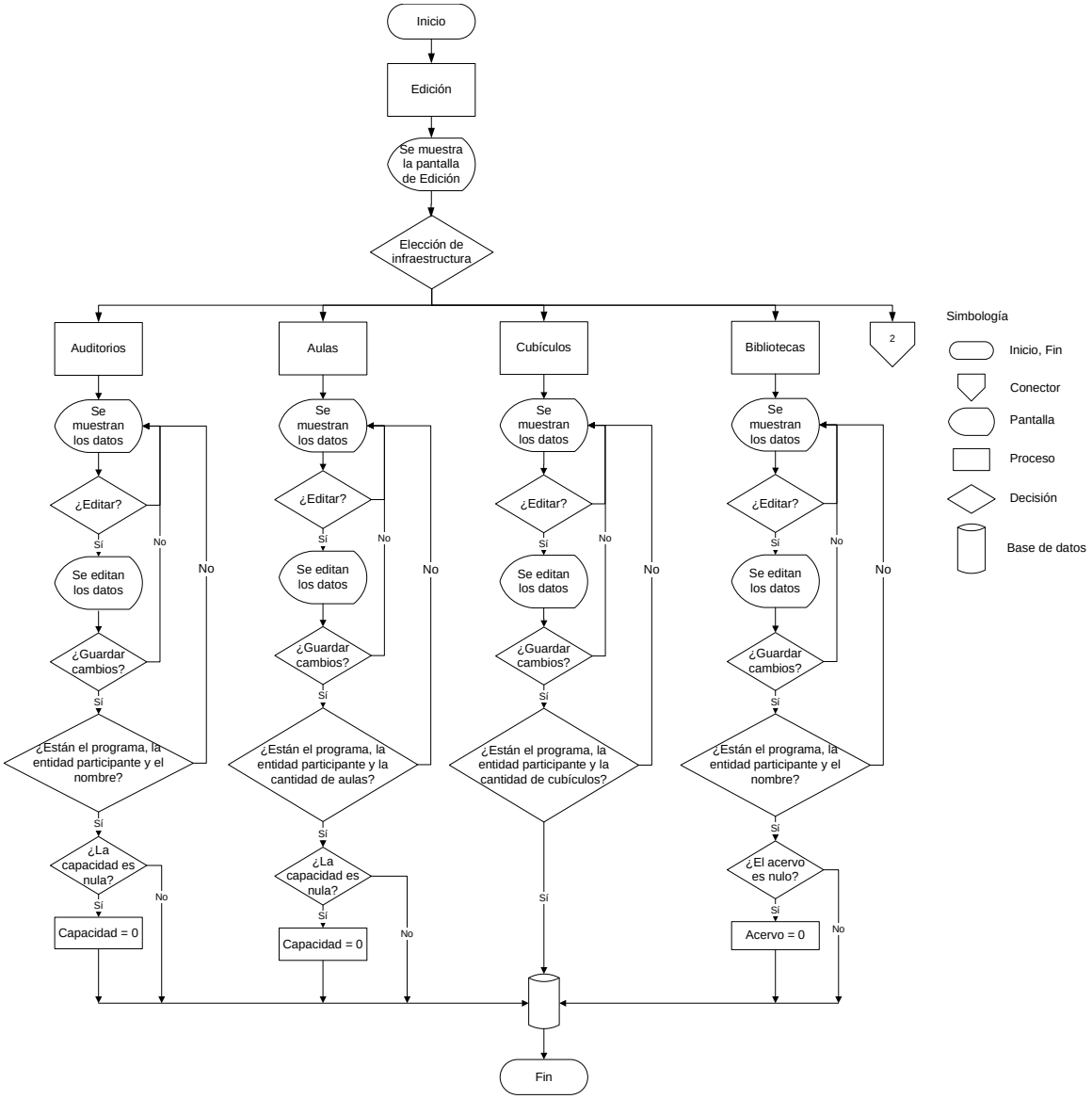

Fig. 29 Arquitectura del módulo de edición (auditorios, aulas, cubículos, bibliotecas)

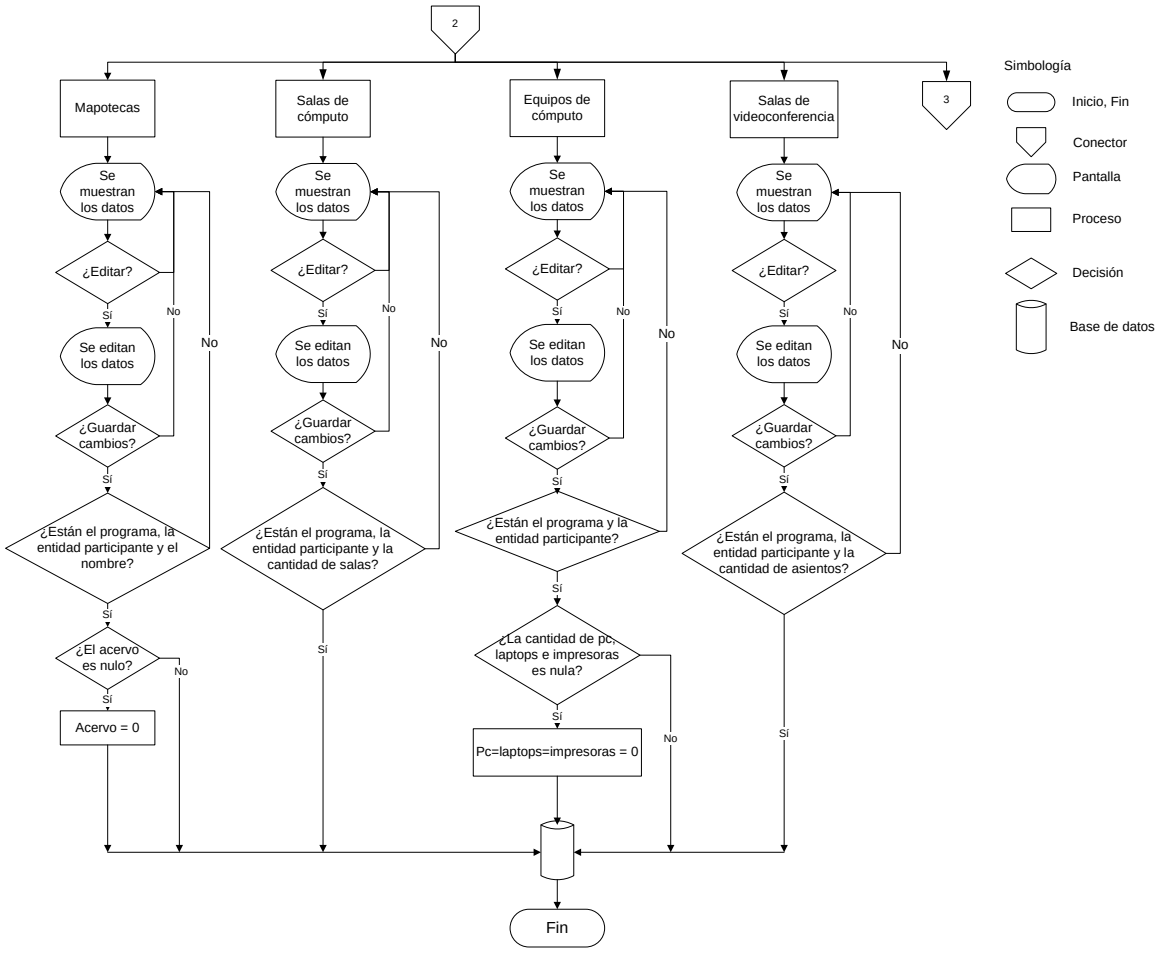

Fig. 30 Arquitectura del módulo de edición (mapotecas, salas de cómputo, equipos de cómputo y salas de videoconferencia)

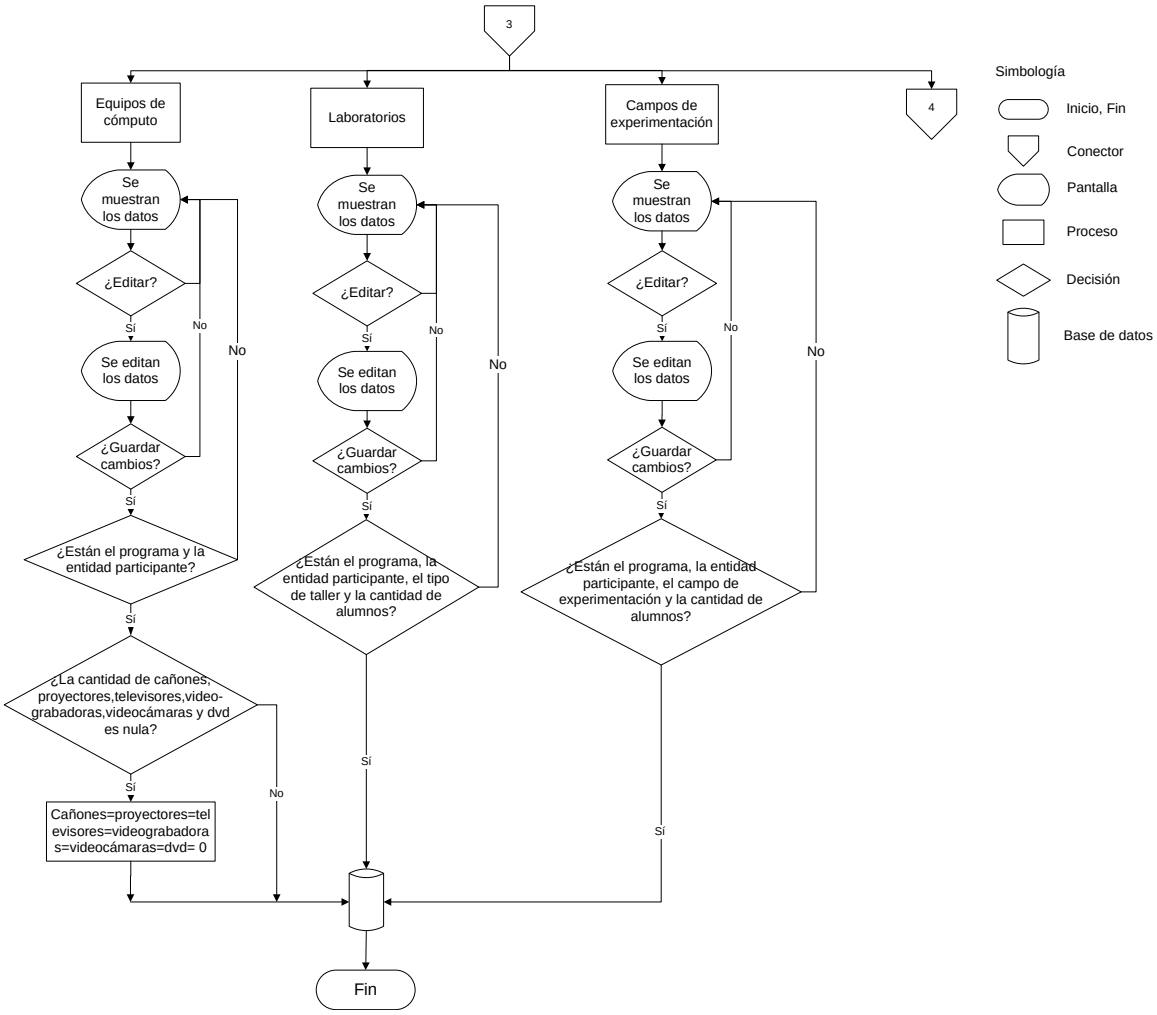

Fig. 31 Arquitectura del módulo de edición (equipo de video, laboratorios, campos de experimentación)

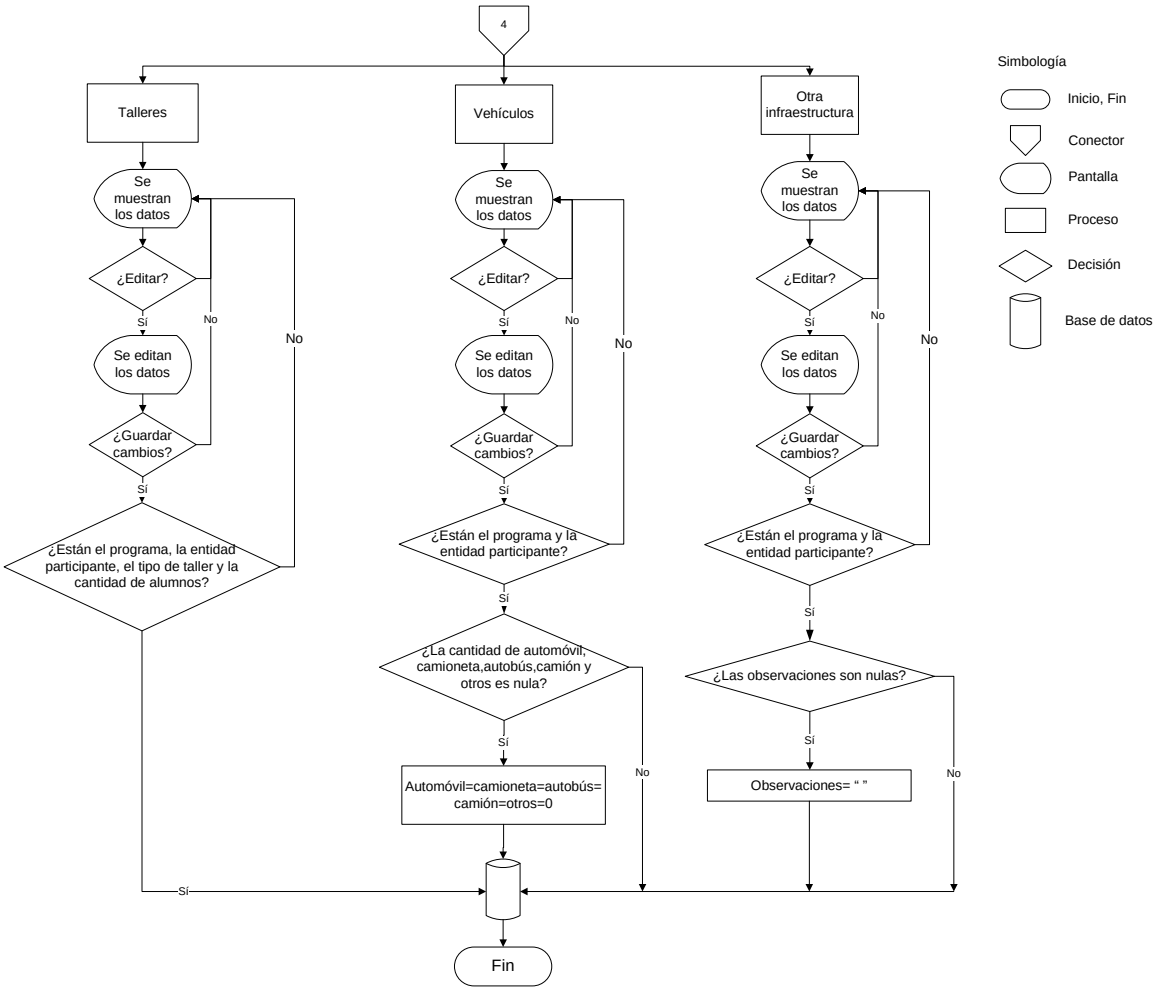

Fig. 32 Arquitectura del módulo de edición (talleres, vehículos y otra infraestructura)

Cabe mencionar que dentro del *Data Grid* se colocó una *Etiqueta* (*Label*) que contiene la información de la base de datos en forma de una cadena de caracteres, ésta se guardó en una variable de sesión, misma que se pasó al *Web User Control* y se utilizó para encontrar dentro de la *lista desplegable*, el valor de la base de datos que estaba seleccionado antes de querer editarlo (ver figura 33). El siguiente código es el procedimiento encargado de realizar lo que anteriormente se dijo:

```
        Private Sub dg_modifica_EditCommand(ByVal source As Object, ByVal
e As System.Web.UI.WebControls.DataGridCommandEventArgs) Handles
dg_modifica.EditCommand
```

```
        'Las variables de sesión que se utilizan para guardar la
información referente a la ubicación, al programa y al id del programa.
        Session("strubica") = CType(e.Item.FindControl("lblubica"),
Label).Text
        Session("strprog") = CType(e.Item.FindControl("lblprog"),
Label).Text
        Session("stridprog") = CType(e.Item.FindControl("lblidprog"),
Label).Text
        dg_modifica.Columns(4).Visible = False
        dg_modifica.Columns(5).Visible = False
        dg_modifica.EditItemIndex = e.Item.ItemIndex
        index = ddlopcion_infra.SelectedItem.Value
        DatasBind(index)
        End Sub
```

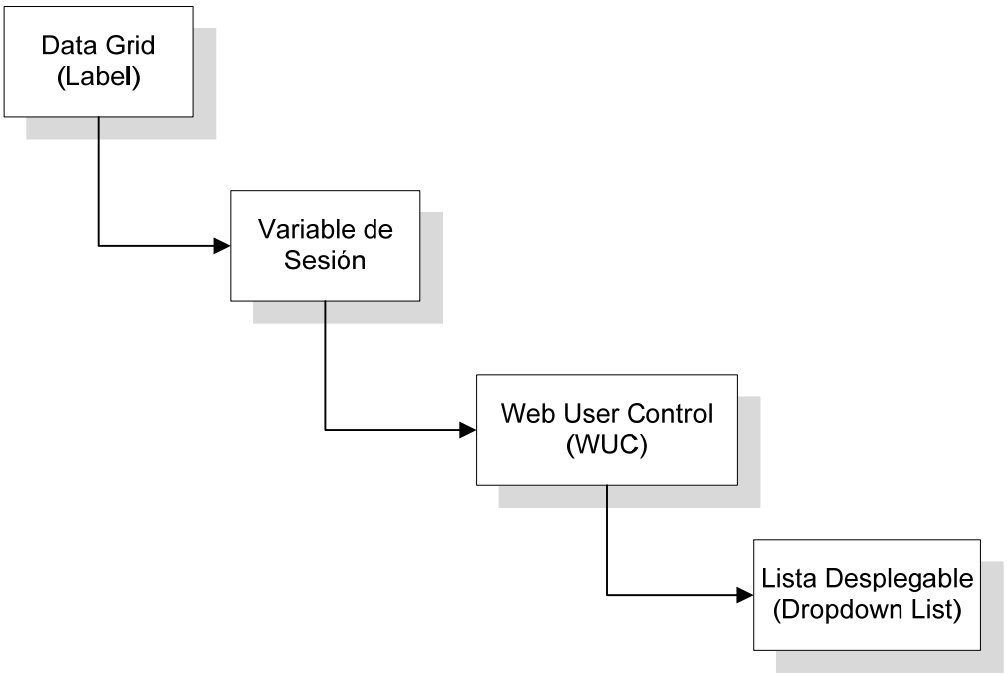

Fig. 33. Esquema de los elementos utilizados para realizar el módulo de edición

En el WUC se incluyó una variable de sesión 1 que almacenará la clave única que nos devolverá la lista desplegable 1 al momento de cambiar y de la misma manera, pasarla al procedimiento almacenado y de esta forma traer los datos relacionados con la clave para poblar la lista desplegable 2 y de igual manera se colocó una variable de sesión 2 para que en ella se guarde la clave que nos devuelve la lista desplegable 2 al momento de ser cambiada (ver figura 34) (ver Anexo 1).

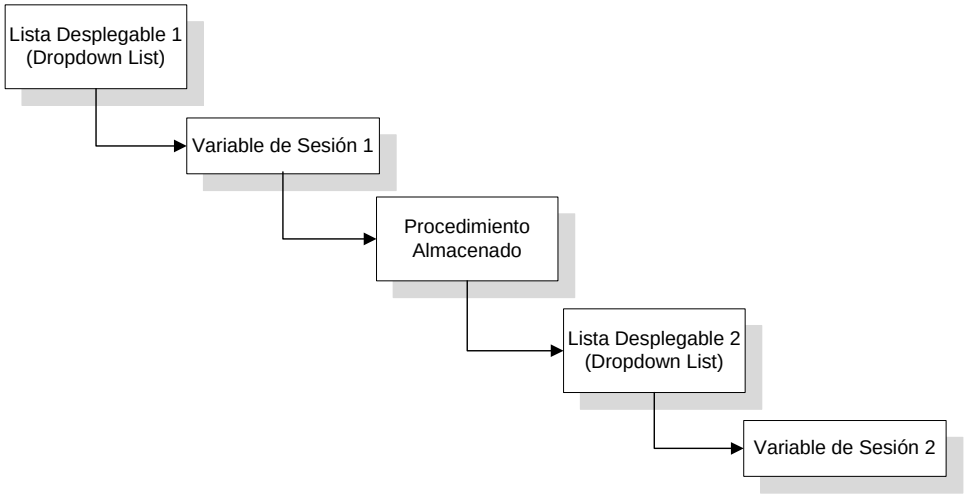

Fig. 34 Esquema de los elementos utilizados para llenar la lista desplegable 2 dependiendo de lo seleccionado en la lista desplegable 1

Una vez que el *WUC* y listas desplegables están con la información y ya se realizaron los cambios, el usuario tiene que presionar un botón y con esto se ejecuta un procedimiento que recopila los datos almacenados en las dos variables de sesión guardándolas en la tupla de la base de datos seleccionada para ser modificada. El siguiente código representa un fragmento del procedimiento antes mencionado:

```
'Se declaran las variables a utilizar para recopilar los datos
        Dim cmdupdate As SqlCommand = New SqlCommand
        Dim         vnumero_con         As         SqlParameter         =
cmdupdate.Parameters.Add("@vnumero_con", SqlDbType.Int, 4)
        Dim       vcve_infraestructura       As       SqlParameter       =
cmdupdate.Parameters.Add("@vcve_infraestructura", SqlDbType.Int, 4)
        Dim         vcid_programa         As         SqlParameter         =
cmdupdate.Parameters.Add("@vcid_programa", SqlDbType.NVarChar, 3)
        Dim          vnid_EA          As          SqlParameter          =
cmdupdate.Parameters.Add("@vnid_EA", SqlDbType.SmallInt)
        Dim          RetValue          As          SqlParameter          =
cmdupdate.Parameters.Add("RetValue", SqlDbType.Int)
        RetValue.Direction = ParameterDirection.ReturnValue
        vnumero_con.Direction = ParameterDirection.Input
        vcve_infraestructura.Direction = ParameterDirection.Input
        vcid_programa.Direction = ParameterDirection.Input
        vnid_EA.Direction = ParameterDirection.Input
        cmdupdate.Connection = conn
        cmdupdate.CommandType = CommandType.StoredProcedure
        index = ddlopcion_infra.SelectedItem.Value
        Select Case index
'Se insertan los datos dependiendo de la infraestructura que se trate
            Case 1
```

```
                cmdupdate.CommandText = "pa_actualizaauditorios_infra"
                Dim       vnum_asientos       As       SqlParameter       =
cmdupdate.Parameters.Add("@vnum_asientos", SqlDbType.Int, 4)
                vnum_asientos.Direction = ParameterDirection.Input
                vnumero_con.Value = e.Item.Cells(1).Text
                vcve_infraestructura.Value = e.Item.Cells(2).Text
                vcid_programa.Value = Session("idprog")
                vnid_EA.Value = Session("idubica")
                Dim      vnombre_auditorio      As      SqlParameter      =
cmdupdate.Parameters.Add("@vnombre_auditorio", SqlDbType.VarChar, 60)
                vnombre_auditorio.Value                                    =
CType(e.Item.Cells(6).Controls(0), TextBox).Text
                If CType(e.Item.Cells(7).Controls(0), TextBox).Text <> ""
Then
                    vnum_asientos.Value                                    =
CType(e.Item.Cells(7).Controls(0), TextBox).Text
                Else
                    vnum_asientos.Value = 0
                End If
```

Cuando se terminó de programar el *WUC*, se sometió a algunas pruebas al sistema, tales como: comenzar a modificar tuplas al azar y se observó que el código programado hacía lo que se necesitaba, además de ser eficiente y dar al usuario una interfaz sencilla y fácil de manejar.

II.3.3 Evaluación

Una vez implementado el módulo se sometió a pruebas tales como: respuesta del sistema con sobre carga de trabajo, es decir, se empezó a modificar tuplas al azar y cambiar la mayor parte del registro, guardarlo y así varias veces para observar la eficiencia del código, no se detectaron deficiencias.

Las figuras 35, 36, 37 y 38 son imágenes que pertenecen al módulo de edición.

Fig. 35. Inicio del módulo de edición

Fig. 36. Mostrando información de la infraestructura seleccionada

Fig. 37. Editando la tupla seleccionada

Fig. 38. Enviando la información actualizada

Posteriormente, junto con el responsable del proyecto se concertó una cita con el usuario, para que él observara el manejo del sistema y una vez que aprendiera a usarlo nos hiciera observaciones respecto a la interfaz o al funcionamiento del mismo. Al final, el usuario aceptó tanto la interfaz como el funcionamiento.

Por último, para poder ingresar al sistema se puede visitar la siguiente dirección electrónica:

- http://siep.posgrado.unam.mx/sipuma_adm para tener acceso a esta dirección se necesita contar con login y password ya que aquí se tiene permiso de creación, edición y borrado de datos.

Se tiene la siguiente dirección electrónica:

- http://siep.posgrado.unam.mx/sipuma en la cual únicamente se cuenta con permisos de lectura, es decir, sólo se pueden ver los datos que están alojados en la base de datos.

Capítulo III

Actividades adicionales

III.1 Videos informativos

Al concluir con el Módulo de Infraestructura y sus respectivos módulos, se comenzó a desarrollar una aplicación en donde se presentan videos informativos de algunos posgrados.

Los problemas que se presentaron fueron:

- El tamaño de los videos era aproximadamente de 200 MB.
- Se dependía de un reproductor de video comercial como *Windows Media Player* o *Real One Player.*

Después de analizar los problemas se acordó que la opción más viable era utilizar *Adobe Flash* ya que al generar la película con extensión *SWF*, ésta se puede incrustar en una página *web* sin que el video pierda calidad y que el archivo excediera los 100 MB.

Ya que se contaba con una parte de los requerimientos y una solución, se comenzó a trabajar con los videos que en algunos casos necesitaron un poco de edición, como redimensionamiento o recortar el tiempo de duración, y en otras ocasiones se hicieron conversiones de formato utilizando software de conversión de video.

Una vez que se terminó con el diseño preliminar de la aplicación, se presentó al responsable del proyecto, llegando a la conclusión de que se requería algo más cercano a un reproductor de video comercial, es decir, que tuviera botones de comenzar, adelantar, rebobinar, parar y que a su vez tuviera una línea de tiempo que fuese interactiva, o sea, que al momento de que el usuario diera un clic en alguna parte de ella, la reproducción del video se tenía que ir a ese punto del video.

Se pensó en modelar la línea de tiempo como si fuera una línea recta, es decir, se obtuvo la ecuación de la recta tomando dos puntos de la línea (ver figura 39), en este caso el punto inicial y final; como abscisa el punto donde el usuario daba clic sobre la línea y como ordenada el número de frame que corresponde a la abscisa dada (ver figura 40).

$$y-y_1 = m(x-x_1)$$

donde $m <> 0$

Fig.39 Ecuación de la recta

Fig.40 Modelado de la línea de tiempo como línea recta

Una vez que se terminó el diseño de la línea de tiempo se comenzó a programarlo y al mismo tiempo se hicieron las pruebas pertinentes tales como tiempo de respuesta y carga, pudiendo constatar que funcionó a la perfección. Posteriormente, ya con el primer video terminado y tomándolo como ejemplo, se terminaron los videos que se solicitaron y se entregaron al responsable del proyecto, en esta ocasión no hubo problemas ya que sistema hacía todo lo que se había solicitado.

La figura 41 muestra versión final de la aplicación.

Fig. 41. Aplicación desarrollada en flash

III.2 Mapa interactivo de la Universidad Nacional Autónoma de México (UNAM)

Una vez que se terminó el reproductor de videos informativos, se empezó a diseñar otro, referente a la elaboración de un mapa interactivo de la UNAM, aquí lo que se deseaba hacer es que cuando el usuario pasara el mouse sobre el dibujo y coincidiera con alguna Facultad, Escuela o Instituto, ésta iba a parpadear y cuando el usuario diera un clic se mostraría la información, contenida en una base de datos y una imagen del edificio seleccionado.

Se comenzó a analizar cómo hacer la conexión entre *flash* y la base de datos y sólo se encontró la manera de hacerlo utilizando código de *PHP,* sin embargo, dentro de los requerimientos se especificó que se hiciera utilizando *ASP .NET*, así que, se comenzó a adecuar el código que se había encontrado a las necesidades, hasta que se consiguió hacer la conexión de *flash* con *SQL Server* utilizando *ASP .NET*.

Una vez que se terminó de analizar el problema y con toda la información se comenzó a programarlo. Finalmente, ya con el sistema terminado, se mostró al responsable del proyecto para que lo sometiera a pruebas tales como tiempo de respuesta y veracidad en la información mostrada. En este caso no hubo cambios en el sistema ni requerimientos adicionales, así que se aprobó el proyecto.

En las figuras 42 a 43 se muestran la aplicación terminada.

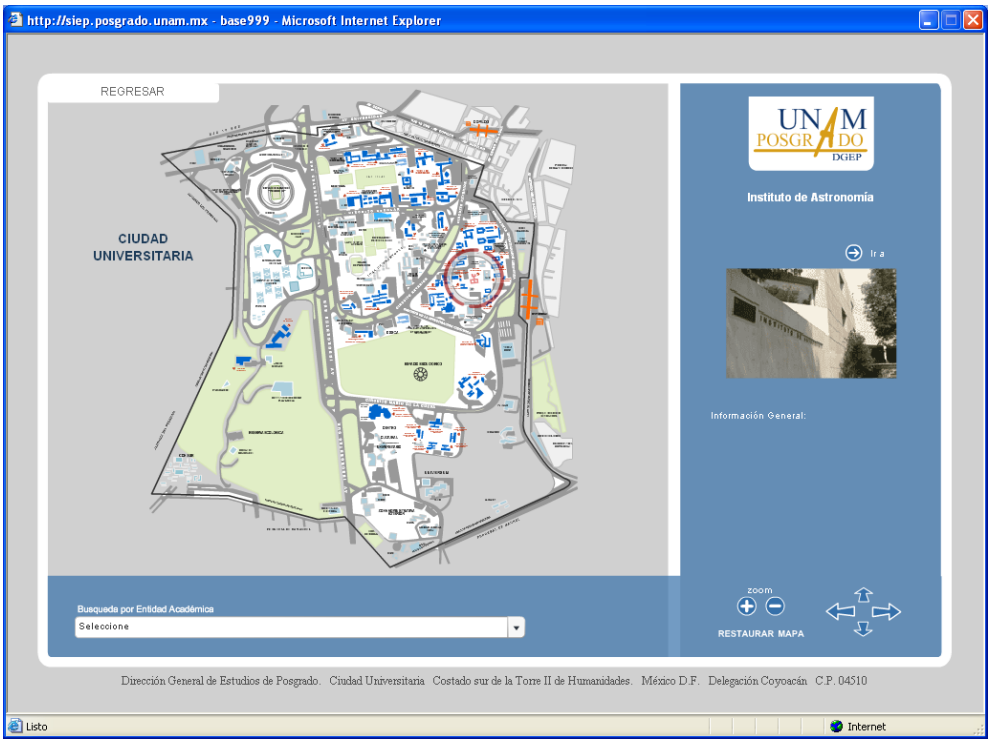

Fig. 42. Mapa de Ciudad Universitaria, UNAM

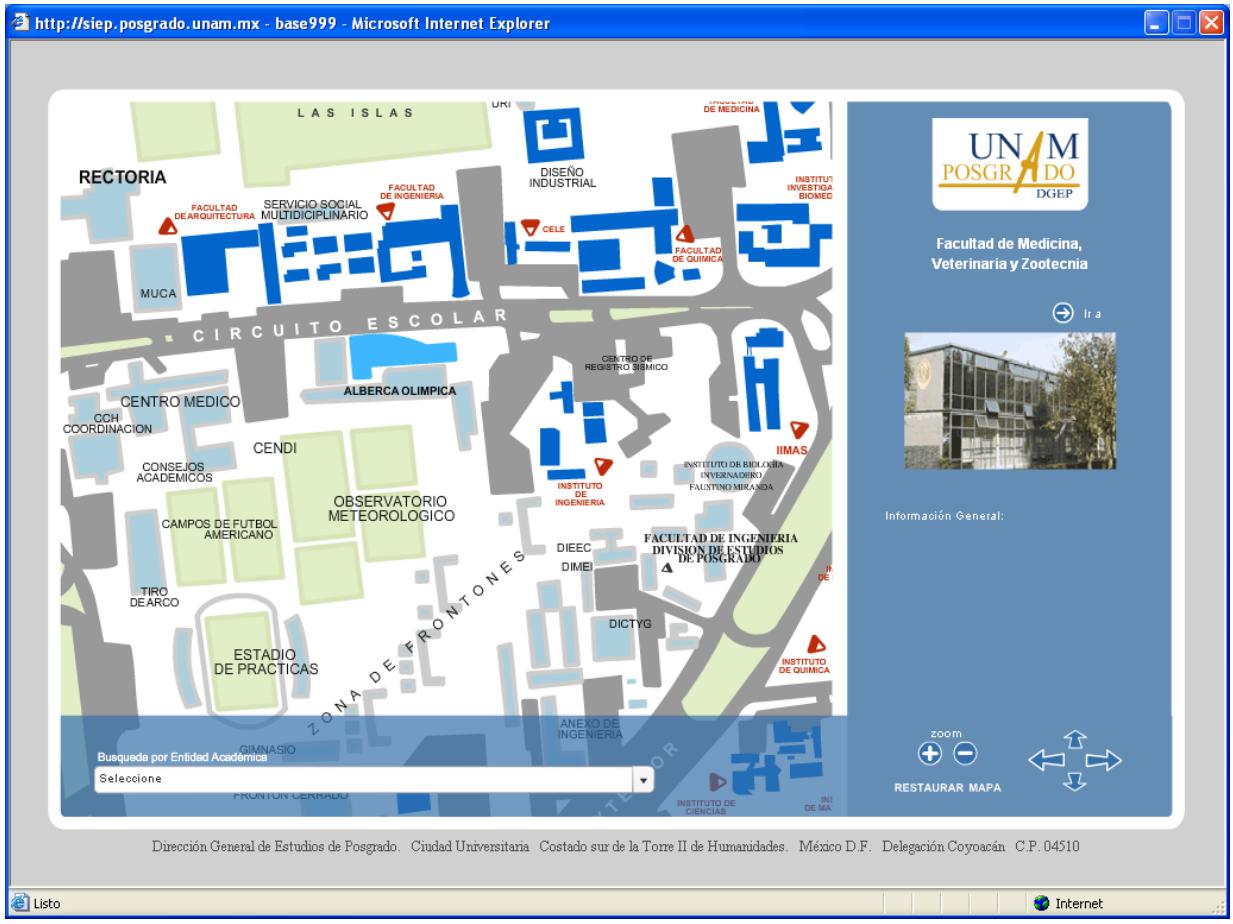

Fig. 43. Mapa con un zoom in aplicado

Capítulo IV

Resultados, impacto y conclusiones

Con base a los requerimientos adquiridos durante el desarrollo del SIPUMA, fueron obtenidos los siguientes resultados:

Requerimientos	Resultados
Proponer un sistema para automatizar y controlar la puesta en línea de la infraestructura de los posgrados de la UNAM.	El resultado fue satisfactorio, pues los posgrados ingresan su infraestructura.
Agilizar los procedimientos que se llevan a cabo en la búsqueda de una infraestructura y localización de posgrados.	El resultado fue satisfactorio, debido a que la búsqueda de una infraestructura se realiza de forma ágil.
Dar a conocer la localización de las instituciones dentro de Ciudad Universitaria, así como la información para contactarlas.	El resultado fue satisfactorio, debido a que el sistema puede ser consultado en cualquier momento y lugar.
Dar a conocer la infraestructura con la que cuentan los posgrados con el fin de identificarla y tomarla en cuenta para cualquier evento.	El resultado fue satisfactorio, debido a que a través de Internet el sistema puede ser accedido en cualquier momento y en cualquier lugar del mundo.
Construcción de una base de datos de la infraestructura con la que cuentan los posgrados de la UNAM.	Actualmente la base de datos se encuentra en crecimiento, debido a que no se contaba con una base actualizada de la infraestructura de los posgrados.

Facilitar el ingreso de información de la infraestructura a través de una interfaz de usuario amigable.	El resultado fue satisfactorio, ya que el ingreso de información es muy sencillo y ágil.
Los resultados obtenidos fueron así catalogados con base en la opinión del cliente y en las pruebas realizadas.	

Se logró desarrollar un sistema hecho a la medida, cubriendo los aspectos deseados. Hay que considerar que el sistema fue generado a partir de necesidades de la DGEP.

Se implementó un ambiente visual de tal forma que el aprendizaje del manejo del sistema se facilitó por parte del personal, generando con esto un menor tiempo de capacitación.

Se observó que con este sistema es posible automatizar un proceso administrativo reduciendo tiempos de respuesta entre las distintas entidades involucradas, además de contar con información actualizada al momento.

El realizar el desarrollo del SIPUMA me permitió valorar la necesidad e importancia de seguir una metodología para plantear los requerimientos y alcances de un sistema de forma adecuada.

Se pudo constatar que es necesario establecer alcances y requerimientos antes del desarrollo, pues de esta manera se evita tener cambios no contemplados y sustanciales que puedan retrasar las actividades planeadas del proyecto.

Existe en toda la Universidad una gran cantidad de sistemas por implementar y es necesario promocionarlos para que alumnos de la Universidad los realicen como servicio social o tesis. Esto llevaría a un beneficio de ambas partes donde los alumnos toman experiencia en el diseño profesional de aplicaciones y la Universidad satisface sus necesidades de sistemas de información.

Finalmente externo mi agradecimiento al apoyo recibido por parte de la Facultad de Ingeniera, de la Coordinación de Estudios de Posgrado y de la Dirección General de Servicios de Cómputo Académico, por prepararme para enfrentar los retos que la sociedad demanda.

Bibliografía

1. DAWES, Brendan. *Action Script para diseñadores gráficos*. Prentice Hall. México, 2002.

2. DOBSON, Rick. *Programación de Microsoft SQL Server 2000 con Microsoft Visual Basic .NET*. McGraw-Hill. España, 2002.

3. ELMASRI, Ramez y NAVATHE Shamkant B. *Fundamentos de Sistemas de Bases de Datos*. Pearson Addison Wesley, España, 2007.

4. HAWTHORNE, Rob. *Desarrollo de bases de datos en Microsoft SQL Server 2000. Desde el Principio.* Prentice Hall. México, 2002.

5. PEÑA, Jaime y VIDAL Ma. Carmen. *Flash MX. Guía de Aprendizaje*. McGraw-Hill. España, 2002.

6. PRESSMAN, Roger S. *Ingeniería de Software. Un enfoque práctico.* McGraw-Hill, Quinta edición. España, 2001.

7. SOMMERVILLE, Ian. *Ingeniería de Software*. Pearson Educación, Séptima edición. Madrid, 2005.

MESOGRAFÍA

8. Adding a Dropdown List to an Editable Data Grid, Matthew Rause, 08 Mayo 2001,<http://www.4guysfromrolla.com/web-tech/050801-1.shtml>

9. Adding Client-Side Message Boxes in your ASP.NET Web Pages, Tim Stall, 11 Febrero 2004, <http://aspnet.4guysfromrolla.com/articules/021104-1.aspx>

10. An Extensive Examination of Data Grid Web Control, Scott Mitchell, 5 Abril 2002, <http://aspnet.4guysfromrolla.com/articules/040502-1.aspx>

11. Java Scripts Prompts for Buttons in ASP Data Grid for Delete Column, Akash Kava, 14 Agosto 2004, <http://www.codeproject.com/KB/webforms/jscriptpromptdatagrid.aspx>

12. Multiple Column Dropdown List for ASP.NET Data Grid, Jayarajan S. Kulaindevelu, 1 Junio 2005, <http://www.codeproject.com/KB/webforms/MultiColDdList.aspx>

13. Información y carecterísticas de SQL Server 2000, 10 Junio 2005, <http://www.microsoft.com/spain/sql/2000/productinfo/caracteristicas.aspx>

Anexo 1

El siguiente código representa el funcionamiento del *Web User Control (WUC):*

```
'Esta función se encarga de cargar la ubicación dependiendo del id de la
entidad
    Function carga_cambio_ubica()
        Dim dr As SqlDataReader
        Dim cmdtest As SqlCommand = New SqlCommand
        Dim v_prog As SqlParameter = cmdtest.Parameters.Add("@prog",
SqlDbType.Int, 4)
        v_prog.Direction = ParameterDirection.Input
        v_prog.Value = variable
        cmdtest.CommandType = CommandType.StoredProcedure
        cmdtest.CommandText = "pa_cargaentidad_infra"
        Try
            cmdtest.Connection = conn
            cmdtest.Connection.Open()
            dr = cmdtest.ExecuteReader()
            ddl_ubicacion.Items.Clear()
            While dr.Read
                ddl_ubicacion.Items.Add(New ListItem(dr("cNombreL_EA"),
dr("nId_EA")))
            End While
            dr.Close()
        Catch ex As Exception
            Response.Write("<SCRIPT LANGUAGE=""JavaScript"">" _
            & Chr(13) & "<!--" & Chr(13) _
            & "alert('Sucedio un error: ')" & Chr(13) _
            & "--><" & "/" & "SCRIPT>")
        Finally
            cmdtest.Connection.Close()
        End Try
        Session("idubica") = ddl_ubicacion.SelectedItem.Value
    End Function

'Esta función se encarga de asignar el valor que se encuentra en la base
de datos a la lista desplegable que muestra la ubicación
    Function encuentra_ub()
        ddl_ubicacion.SelectedIndex =
ddl_ubicacion.Items.IndexOf(ddl_ubicacion.Items.FindByText(Session("strub
ica")))
        Session("idubica") = ddl_ubicacion.SelectedItem.Value
    End Function

'Esta función carga los datos a la lista desplegable de ubicación
    Function carga_ubicacion()
        Dim cook As HttpCookie = Request.Cookies("preference")
        Dim dr As SqlDataReader
        Dim cmdtest As SqlCommand = New SqlCommand
        Dim v_prog As SqlParameter = cmdtest.Parameters.Add("@prog",
SqlDbType.Int, 4)
        v_prog.Direction = ParameterDirection.Input
        v_prog.Value = CType(Session("stridprog"), Integer)
        cmdtest.CommandType = CommandType.StoredProcedure
        cmdtest.CommandText = "pa_cargaentidad_infra"
        Try
            cmdtest.Connection = conn
```

```
            cmdtest.Connection.Open()
            dr = cmdtest.ExecuteReader()
            ddl_ubicacion.Items.Clear()
            While dr.Read
                ddl_ubicacion.Items.Add(New ListItem(dr("cNombreL_EA"),
dr("nId_EA")))
            End While
            dr.Close()
        Catch ex As Exception
            Response.Write("<SCRIPT LANGUAGE=""JavaScript"">" _
            & Chr(13) & "<!--" & Chr(13) _
            & "alert('Sucedio un error: ')" & Chr(13) _
            & "--><" & "/" & "SCRIPT>")
        Finally
            cmdtest.Connection.Close()
        End Try
        encuentra_ub()
    End Function

'Esta función se encarga de asignar el valor que se encuentra en la base
de datos a la lista desplegable que muestra la ubicación.
    Function encuentra_prog()
        ddl_entidad.SelectedIndex =
ddl_entidad.Items.IndexOf(ddl_entidad.Items.FindByText(Session("strprog")
))
        Session("idprog") = ddl_entidad.SelectedItem.Value
    End Function

'Esta función carga los datos a la lista desplegable de las entidades
    Function carga_entidad() As DataSet
        Dim dr As SqlDataReader
        Dim cmdtest As SqlCommand = New SqlCommand
        cmdtest.CommandType = CommandType.StoredProcedure
        cmdtest.CommandText = "pa_cargaprog_infra"
        Try
            cmdtest.Connection = conn
            cmdtest.Connection.Open()
            dr = cmdtest.ExecuteReader()
            ddl_entidad.Items.Clear()
            While dr.Read
                ddl_entidad.Items.Add(New
ListItem(dr("cNombre_Programa"), dr("cId_Programa")))
            End While
            dr.Close()
        Catch ex As Exception
            Response.Write("<SCRIPT LANGUAGE=""JavaScript"">" _
            & Chr(13) & "<!--" & Chr(13) _
            & "alert('Sucedio un error: ')" & Chr(13) _
            & "--><" & "/" & "SCRIPT>")
        Finally
            cmdtest.Connection.Close()
        End Try
        encuentra_prog()
    End Function
```

```
'Este procedimiento se encarga de asignar el valor que se genera al
cambiar el valor en la lista desplegable 1 y además llama a la función
que llena la lista desplegable 2.
    Private Sub ddl_entidad_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ddl_entidad.SelectedIndexChanged
        variable = CType(ddl_entidad.SelectedItem.Value, Integer)
        Session("texto") = CType(ddl_entidad.SelectedItem.Text, String)
        Session("idprog") = CType(ddl_entidad.SelectedItem.Value, String)
        carga_cambio_ubica()
    End Sub

'Esta procedimiento se encarga de asignar el valor que se genera al
cambiar el valor en la lista desplegable 2.
    Private Sub ddl_ubicacion_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ddl_ubicacion.SelectedIndexChanged
        Session("idubica") = ddl_ubicacion.SelectedItem.Value
    End Sub
```