Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

“Sistema Generador de Constancias Estudiantiles”

TESIS PROFESIONAL
PARA OBTENER EL TÍTULO DE
LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:
Othoniel González Campos

ASESOR:
M.C. Meliza Contreras González

Puebla, Pue. **Enero 2009**

ÍNDICE

CAPITULO1

MARCO TEÓRICO 1

CAPITULO 2

ANALISIS DEL SISTEMA

CAPITULO I
MARCO TEÓRICO

1 Ingeniería de software

1.1 Herramientas CASE y conceptos básicos

Para desarrollar software se necesita de otros tipos de software, es decir, para construir sistemas de cómputo se utilizan otros sistemas de cómputo. Los medios sistematizados que se utilizaron por mucho tiempo estaban limitados a los tradicionales editores de texto para la codificación, y los compiladores del lenguaje respectivo. Fuera de éstos era poco el soporte que un programador o desarrollador de sistemas obtenía por parte de su ambiente de trabajo. [PRE98]

Debido a esta escasez de herramientas adecuadas para el desarrollo de sistemas surgió la lógica necesidad de crear sistemas que se pudieran utilizar verdaderamente como herramientas de soporte en la construcción de software. De ahí surge la Ingeniería de Software Asistida por Computadora, o en inglés, Computer-Aided Software Engineering (CASE). Así, una herramienta CASE es un producto computacional enfocado a apoyar una o mas técnicas dentro un método de desarrollo de software.

A pesar de que las herramientas CASE no tienen una historia extremadamente larga, pues empiezan a surgir a partir de principios de la década de los ochenta, ya se han extendido a la mayor parte de las fases y actividades involucradas en el desarrollo de software. Existen diversas taxonomías de las herramientas CASE, que utilizan varios criterios para su clasificación. Una clasificación por función se divide en dos grandes áreas: CASE superiores (U-CASE) y CASE inferiores (L-CASE). Los U-CASE abarcan las etapas de planeación, análisis y diseño, mientras que los L-CASE comprenden las de codificación, pruebas y mantenimiento. De esta manera se cubren las grandes áreas del desarrollo de software. [ABR07]

Las herramientas CASE individuales pueden estar enfocadas a un área de ingeniería de software más específica, como lo puede ser la ingeniería de información, el modelado de procesos, planificación y administración de proyectos, análisis de riesgos, seguimiento de requisitos, métricas, documentación, control de calidad, gestión de bases de datos, de desarrollo de interfaz o de generación de prototipos entre otros El tipo específico de herramienta que se utilice depende de los requerimientos tanto del sistema a implementar como de los desarrolladores [ABR07].

1.2 Modelos de desarrollo de software

La ingeniería de software tiene varios modelos o paradigmas de desarrollo en los cuales se puede apoyar para la realización de software, de los cuales podemos destacar a éstos por ser los más utilizados y los más completos:

- Modelo en cascada o Clásico (modelo tradicional)
- Modelo en espiral (modelo evolutivo)
- Modelo de prototipos
- Desarrollo por etapas

[SOM05]

1.2.1 Modelo en cascada

En Ingeniería de software el desarrollo en cascada, también llamado modelo en cascada, es el enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma tal que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior.

Un ejemplo de una metodología de desarrollo en cascada es:

1. Análisis de requisitos
2. Diseño del Sistema
3. Diseño del Programa
4. Codificación
5. Pruebas
6. Implantación
7. Mantenimiento

De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costes del desarrollo. La palabra *cascada* sugiere, mediante la metáfora de la fuerza de la gravedad, el esfuerzo necesario para introducir un cambio en las fases más avanzadas de un proyecto.
Si bien ha sido ampliamente criticado desde el ámbito académico y la industria, sigue siendo el paradigma más seguido al día de hoy.

Análisis de requisitos
Se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir. De esta fase surge una memoria llamada SRD (documento de especificación de requisitos), que contiene la especificación completa de lo que debe hacer el sistema sin entrar en detalles internos.

Es importante señalar que en esta etapa se deben consensuar todo lo que se requiere del sistema y será aquello lo que seguirá en las siguientes etapas, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software.

Diseño del Sistema
Se descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

Diseño del Programa
Es la fase en donde se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario así como también los análisis necesarios para saber que herramientas usar en la etapa de Codificación.
Codificación
Es la fase de programación o implementación propiamente dicha. Aquí se implementa el código fuente, haciendo uso de prototipos así como pruebas y ensayos para corregir

errores.

Dependiendo del lenguaje de programación y su versión se crean las librerías y componentes reutilizables dentro del mismo proyecto para hacer que la programación sea un proceso mucho más rápido.

Pruebas

Los elementos, ya programados, se ensamblan para componer el sistema y se comprueba que funciona correctamente antes de ser puesto en explotación.

Implantación

El software obtenido se pone en producción. Se implantan los niveles software y hardware que componen el proyecto. La implantación es la fase con más duración y con más cambios en el ciclo de elaboración de un proyecto. Es una de las fases finales del proyecto.

Durante la explotación del sistema pueden surgir cambios, bien para corregir errores o bien para introducir mejoras. Todo ello se recopila en los Documentos de Cambios.

Variantes

Existen variantes de este modelo; especialmente destacamos la que hace uso de prototipos y en la que se establece un ciclo antes de llegar a la fase de mantenimiento, verificando que el sistema final este libre de fallos.

Desventajas

En la vida real, un proyecto rara vez sigue una secuencia lineal, esto crea una mala implementación del modelo, lo cual hace que lo lleve al fracaso.

Difícilmente un cliente va a establecer al principio todos los requerimientos necesarios, por lo que provoca un gran atraso trabajando en este modelo, ya que este es muy restrictivo y no permite movilizarse entre fases.

Los resultados y/o mejoras no son visibles, el producto se ve recién cuando este esté finalizado, lo cual provoca una gran inseguridad por parte del cliente que anda ansioso de ver avances en el producto. Esto también implica toparse con requerimientos que no se habían tomado en cuenta, y que surgieron al momento de la implementación, lo cual provocara que se regrese nuevamente a la fase de requerimientos.

Ventajas

Se tiene todo bien organizado y no se mezclan las fases.

Es perfecto para proyectos que son rígidos, y además donde se especifiquen muy bien los

requerimientos y se conozca muy bien la herramienta a utilizar
[SOM05]

1.2.2 Modelo en espiral

El Desarrollo en Espiral es un modelo de ciclo de vida desarrollado por Barry Boehm en 1985, utilizado generalmente en la Ingeniería de software. Las actividades de este modelo son una espiral, cada bucle es una actividad. Las actividades no están fijadas a prioridad, sino que las siguientes se eligen en función del análisis de riesgo, comenzando por el bucle interior.

Para cada actividad habrá cuatro tareas:

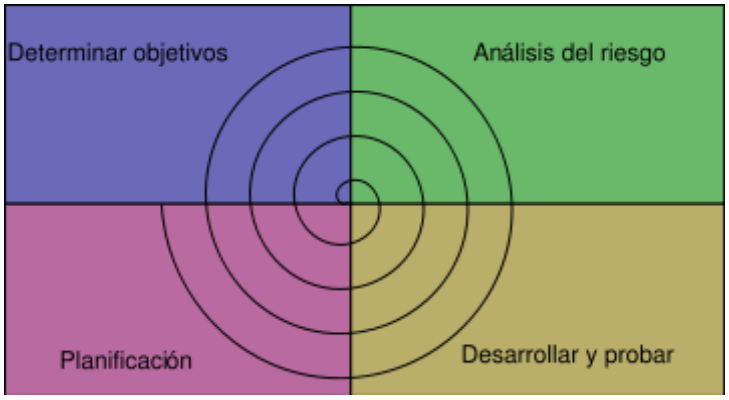

Figura 1.1 Modelo espiral

Determinar o fijar objetivos

- Fijar también los productos definidos a obtener: requerimientos, especificación, manual de usuario.
- Fijar las restricciones.
- Identificación de riesgos del proyecto y estrategias alternativas para evitarlos.
- Hay una cosa que solo se hace una vez: planificación inicial o previa.

Análisis del riesgo

- Se estudian todos los riesgos potenciales y se seleccionan una o varias alternativas propuestas para reducir o eliminar los riesgos.

Desarrollar, verificar y validar (probar)

- Tareas de la actividad propia y de prueba.
- Análisis de alternativas e identificación resolución de riesgos.

- Dependiendo del resultado de la evaluación de los riesgos, se elige un modelo para el desarrollo, el que puede ser cualquiera de los otros existentes, como formal, evolutivo, cascada, etc. Así si por ejemplo si los riesgos en la interfaz de usuario son dominantes, un modelo de desarrollo apropiado podría ser la construcción de prototipos evolutivos. Si lo riesgos de protección son la principal consideración, un desarrollo basado en transformaciones formales podría ser el más apropiado.

Planificar

- Revisamos todo lo hecho, evaluándolo, y con ello decidimos si continuamos con las fases siguientes y planificamos la próxima actividad.

Mecanismos de control

- La dimensión radial mide el costo.
- La dimensión angular mide el grado de avance del proyecto.

Ventajas

El análisis del riesgo se hace de forma explícita y clara. Une los mejores elementos de los restantes modelos. - Reduce riesgos del proyecto - Incorpora objetivos de calidad - Integra el desarrollo con el mantenimiento, etc. Además es posible tener en cuenta mejoras y nuevos requerimientos sin romper con la metodología, ya que este ciclo de vida no es rígido ni estático.

Desventajas

- Genera mucho tiempo en el desarrollo del sistema - Modelo costoso - Requiere experiencia en la identificación de riesgos

1.3 Bases de Datos

1.3.1 Definición

Una base de datos es un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta.

Las bases de datos no son un fenómeno nuevo. De hecho, Herman Hollerith mecanizó el almacenamiento del Censo de EEUU de 1890 en lo que de alguna forma se considera la primera base de datos significativa “computarizada” [PRE01]

En la actualidad, y gracias al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos tienen formato electrónico, que ofrece un amplio rango de soluciones al problema de almacenar datos.

En informática existen los sistemas gestores de bases de datos (SGBD), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de los sistemas gestores de bases de datos se estudian en informática.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

Aunque las bases de datos pueden contener muchos tipos de datos, algunos de ellos se encuentran protegidos por las leyes de varios países. Por ejemplo en España, los datos personales se encuentran protegidos por la Ley Orgánica de Protección de Datos de Carácter Personal.

1.3.2 Tipos de Bases de Datos

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al criterio elegido para su clasificación:

1.3.2.1 Según la variabilidad de los datos almacenados

a. **Bases de datos estáticas**

   Éstas son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.

b. **Bases de datos dinámicas**

   Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un videoclub, etc.

1.3.3. Sistema gestor de base de datos

Un sistema gestor de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto *práctica* como *eficiente*.

Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o los intentos de acceso sin autorización. Si los datos van a ser compartidos entre diversos usuarios, el sistema debe evitar posibles resultados anómalos.

Dado que la información es tan importante en la mayoría de las organizaciones, los científicos informáticos han desarrollado un amplio conjunto de conceptos y técnicas para la gestión de los datos. En este capítulo se presenta una breve introducción a los principios de los sistemas de bases de datos.

1.3.4 Modelo de bases de datos Relacionales

Éste es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se

conceptualiza de una manera más fácil de imaginar. Esto es pensando en cada relación como si fuese una tabla que está compuesta por registros (las filas de una tabla), que representarían las tuplas, y campos (las columnas de una tabla).

En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

El lenguaje más habitual para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Durante su diseño, una base de datos relacional pasa por un proceso al que se le conoce como normalización de una base de datos.

1.3.4.1 Normalización

Concepto

Normalización es un conjunto de reglas que sirven para ayudar a los diseñadores a desarrollar un esquema que minimice los problemas de lógica. Cada regla está basada en la que le antecede. La normalización se adoptó porque el viejo estilo de poner todos los datos en un solo lugar, como un archivo o una tabla de la base de datos, era ineficiente y conducía a errores de lógica cuando se trataba de manipular los datos.

Ejemplo

Veamos la base de datos MiTienda. Si se almacenan todos los datos en la tabla Clientes,

ésta podría verse como se muestra a continuación:

Clientes
ID_Cliente
Nombre
Apellidos
Nombre_Producto1
Costo_Producto1
Imagen_Producto1
Nombre_Producto2
Costo_Producto2
Imagen_Producto2
Fecha_Pedido
Cantidad_Pedido
Nombre_Cia_Envios

Figura 1.2 Tabla clientes

La tabla se ha descrito de manera abreviada pero aun así representa la idea general.

¿Cómo podría añadir un nuevo cliente en la tabla Clientes? Debería añadir un producto y un pedido también. ¿Qué tal si quisiera emitir un informe de todos los productos que se venden? No podría separar fácilmente los productos de los clientes con una simple instrucción SQL. Lo bueno de las bases de datos relacionales, si están bien diseñadas, es que estos reportes se pueden hacer fácilmente.

La normalización también hace las cosas fáciles de entender. Los seres humanos tenemos la tendencia de simplificar las cosas al máximo. Lo hacemos con casi todo, desde los animales hasta con los automóviles. Vemos una imagen de gran tamaño y la hacemos menos compleja agrupando cosas similares juntas. Las guías que la normalización provee crean el marco de referencia para simplificar la estructura. En la base de datos de ejemplo

es fácil detectar que tenemos tres diferentes grupos: clientes, productos y pedidos. Si seguimos las guías de la normalización, podríamos crear las tablas basándonos en estos grupos.

El proceso de normalización tiene un nombre y una serie de reglas para cada fase. Esto puede parecer un poco confuso al principio, pero poco a poco se entiende el proceso, así como las razones para hacerlo de esta manera. A la mayoría de la gente le encantan las hojas de cálculo por la forma en la que manejan sus datos. El tiempo que lleva reconfigurar un esquema para ajustarlo al proceso de normalización, siempre será bien invertido. Al fin y al cabo, esto nos tomará menos tiempo que el que tendríamos que invertir, para cortar y pegar sus columnas de datos para generar el informe adecuado a las necesidades de la empresa.

Otra ventaja de la normalización de base de datos es el consumo de espacio. Una base de datos normalizada puede ocupar menos espacio en disco que una no normalizada. Hay menos repetición de datos, lo que tiene como consecuencia un mucho menor uso de espacio en disco.

Grados de normalización

Existen básicamente tres niveles de normalización: Primera Forma Normal (1NF), Segunda Forma Normal (2NF) y Tercera Forma Normal (3NF). En general, son suficientes para cubrir las necesidades de la mayoría de las bases de datos. El creador de estas 3 primeras formas normales (o reglas) fue Edgar F. Codd, éste introdujo la normalización en un artículo llamado *A Relational Model of Data for Large Shared Data Banks* Communications of the ACM. Cada una de estas formas tiene sus propias reglas. Cuando una base de datos se conforma a un nivel, se considera normalizada a esa forma de normalización. Por ejemplo, supongamos que su base de datos cumple con todas las reglas del segundo nivel de normalización. Se considera que está en la Segunda Forma Normal. No siempre es una buena idea tener una base de datos conformada en el nivel más alto de normalización. Puede llevar a un nivel de complejidad que pudiera ser evitado si estuviera en un nivel más bajo de normalización.

Primera Forma Normal

La regla de la Primera Forma Normal establece que las columnas repetidas deben eliminarse y colocarse en tablas separadas. Ésta es una regla muy fácil de seguir. Observe el esquema de la tabla Clientes de la base de datos.

Clientes
ID_Cliente
Nombre
Apellidos
Nombre_Producto1
Costo_Producto1
Imagen_Producto1
Nombre_Producto2
Costo_Producto2
Imagen_Producto2
Fecha_Pedido
Cantidad_Pedido
Nombre_Cia_Envios

Figura 1.3 Tabla clientes

La tabla tiene varias columnas repetidas. Éstas se refieren principalmente a los productos. De acuerdo con la regla, debe eliminar las columnas repetidas y crearles su propia tabla.

Eliminación de datos repetidos en la base de datos:

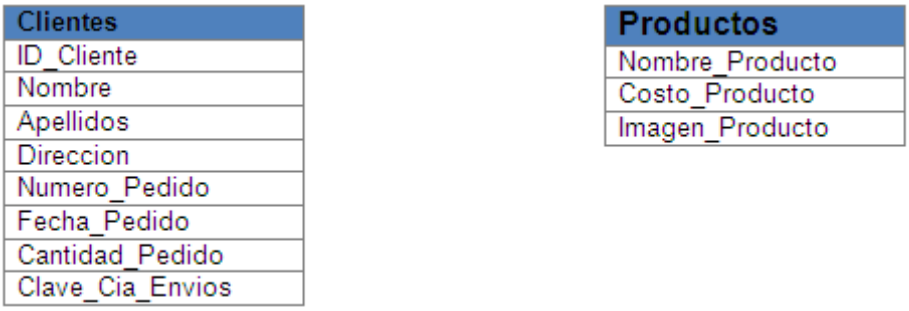

Figura 1.4 Primera forma normal

Ahora tiene dos tablas. Pero todavía hay un problema. No hay forma de relacionar los datos de la tabla original con los de la nueva tabla. Para hacerlo, debe añadir un campo clave a la segunda tabla de forma que se establezca la relación. Añada a la tabla Productos una clave primaria que se llame ID_Producto y añada una clave a la tabla Clientes que la relacione con la tabla Productos. El campo ID_Producto es el candidato ideal.

Figura 1.5 Primera forma normal

Así, se ha establecido una relación uno a varios. Ésta representa lo que la base de datos estará haciendo en la vida real. El cliente tendrá muchos productos que podrá comprar, sin importar cuántos otros clientes quieran comprarlos también. Además, el cliente necesitará haber pedido un producto para ser un cliente. Usted ya no está obligado a añadir un cliente cada vez que añade un nuevo producto a su inventario.

Poner la base de datos en la Primera Forma Normal resuelve el problema de los encabezados de columna múltiples. Muy a menudo, los diseñadores de bases de datos inexpertos harán algo similar a la tabla no normalizada. Una y otra vez, crearán columnas que representen los mismos datos. En una empresa de servicios de electricidad, había una base de datos para el control de refacciones de una planta nuclear. La tabla de su base de datos, la cual contenía los números de parte de las refacciones, tenía una columna repetida más de treinta veces. Cada vez que una nueva parte se tenía que dar de alta, se creaba una nueva columna para almacenar la información. Obviamente, el diseño de la base de datos era bastante pobre y, por lo mismo, resultaba una pesadilla para sus programadores/administradores.

La normalización ayuda a clarificar la base de datos y a organizarla en partes más pequeñas y más fáciles de entender. En lugar de tener que entender una tabla gigantesca y monolítica que tiene muchos diferentes aspectos, usted sólo tiene que entender objetos pequeños y más tangibles, así como las relaciones que guardan con otros objetos también pequeños. No es necesario mencionar que un mejor entendimiento del funcionamiento de nuestra base de datos conducirá a un mejor aprovechamiento de sus activos.

Segunda Forma Normal

La regla de la Segunda Forma Normal establece que todas las dependencias parciales se deben eliminar y separar dentro de sus propias tablas. Una dependencia parcial es un término que describe a aquellos datos que no dependen de la clave de la tabla para identificarlos. En la base de datos de muestra, la información de pedidos está en cada uno de los registros. Sería mucho más simple utilizar únicamente el número del pedido. El resto de la información podría residir en su propia tabla. Una vez que haya organizado la información de pedidos.

Eliminación de las dependencias parciales - Segunda Forma Normal

Clientes
ID Producto
ID_Cliente
Nombre
Apellidos
Direccion
Clave_Cia_Envios

Pedidos
ID Producto
Numero_Pedido
Fecha_Pedido
Cantidad_Pedido

Productos
ID Producto
Nombre_Producto
Costo_Producto
Imagen_Producto

Figura 1.6 Segunda forma normal

De nuevo, al organizar el esquema de esta forma podemos reflejar el mundo real en nuestra base de datos. Tendría que hacer algunos cambios en las reglas del negocio para que esto fuera aplicable, pero para ilustrar la normalización, así está bien.

Una de las mayores desventajas de la normalización es el tiempo que lleva hacerlo. La mayoría de la gente está demasiado ocupada, y emplear tiempo para asegurarse de que sus datos están normalizados cuando todo funciona más o menos bien, parece ser un desperdicio de tiempo. Pero no es así. De hecho tendríamos que emplear más tiempo arreglando una base de datos no normalizada que el que emplearía en una normalizada.

Al haber alcanzado la Segunda Forma Normal, podemos disfrutar de algunas de las ventajas de las bases de datos relacionales. Por ejemplo, podemos añadir nuevas columnas a la tabla Clientes sin afectar a las tablas Productos y Pedidos. Lo mismo aplica para las otras tablas. Alcanzar este nivel de normalización permite que los datos se acomoden de una manera natural dentro de los límites esperados.

Una vez que ha alcanzado el nivel de la Segunda Forma Normal, se han controlado la mayoría de los problemas de lógica. Podemos insertar un registro sin un exceso de datos en la mayoría de las tablas. Observando un poco más de cerca la tabla Clientes, vemos la columna Clave_Cia_Envios. Ésta no es dependiente del cliente. El siguiente nivel de normalización explicará cómo solucionar esto.

Tercera Forma Normal

La regla de la Tercera Forma Normal señala que hay que eliminar y separar cualquier dato que no sea clave. El valor de esta columna debe depender de la clave. Todos los valores deben identificarse únicamente por la clave. En la base de datos de muestra, la tabla Clientes contiene la columna Clave_Cia_Envios, la cual no se identifica únicamente por la clave. Podríamos separar estos datos de la tabla y ponerlos en una tabla aparte.

Eliminación de los datos que no son claves para la Tercera Forma Normal

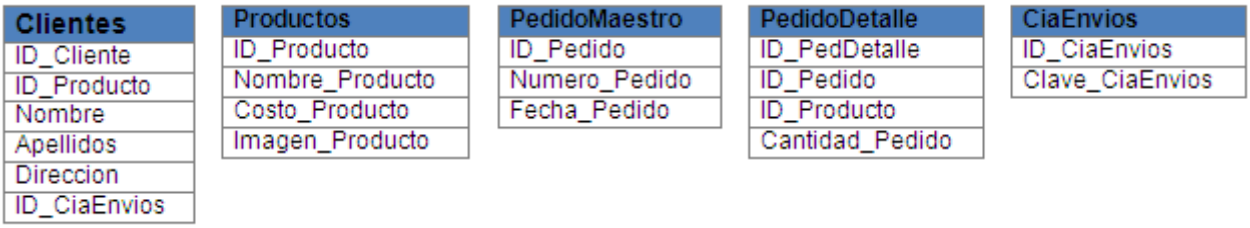

Figura 1.7 Tercera forma normal

Ahora todas las tablas están en la Tercera Forma Normal. Esto le da más flexibilidad y previene errores de lógica cuando se inserta o borra registros. Cada columna en la tabla está identificada de manera única por la clave, y no hay datos repetidos. Esto provee un esquema limpio y elegante, que es fácil de trabajar y expandir.

1.4 Tecnologías de Conectividad

1.4.1 ASP

ASP es una tecnología desarrollada por MS para crear páginas web de contenido dinámico apoyándose en scripts ejecutados en el servidor. Básicamente una página ASP es una mezcla entre una página HTML y un programa que da como resultado una página HTML que es enviada al cliente (navegador).

Estos scripts o programas pueden en ASP ser escritos en uno de estos dos lenguajes de programación VBScript o JavaScript, pero el más extendido es VBScript.

ASP es una tecnología que pertenece a la parte servidor, por esto no es necesario que el cliente o navegador la soporte ya que se ejecuta en el servidor, sí que deberemos buscar un servidor que nos soporte este tipo de tecnología para que nuestras páginas corran correctamente. En la figura 1.8 se describe el procesamiento de una página

Figura 1.8 Procesamiento de una página ASP

Hay que destacar que ASP es una tecnología propietaria de Microsoft, y que el uso de esta tecnología implica el uso de los productos de Microsoft: MS Internet Information System y MS Windows en el servidor.

1.4.2 PHP

El lenguaje PHP es un lenguaje de programación de estilo clásico, con esto quiero decir que es un lenguaje de programación con variables, sentencias condicionales, bucles, funciones.... No es un lenguaje de marcas como podría ser HTML, XML o WML. Está más cercano a JavaScript o a C, para aquellos que conocen estos lenguajes.

Pero a diferencia de Java o JavaScript que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso nos permite acceder a los recursos que tenga el servidor como por ejemplo podría ser una base de datos. El programa PHP es ejecutado en el servidor y el

resultado enviado al navegador. El resultado es normalmente una página HTML pero igualmente podría ser una pagina WML.

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador, pero sin embargo para que sus páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP. A continuación se muestra en la Figura 1.9, el procesamiento de una página PHP.

Figura 1.9 Procesamiento de una página PHP

1.4.3 MySQL

1.4.3.1. ¿Qué es MySQL?

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. MySQL fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca.

Aunque MySQL es software libre, MySQL AB distribuye una versión comercial de MySQL, que no se diferencia de la versión libre más que en el soporte técnico que se ofrece, y la posibilidad de integrar este gestor en un software propietario, ya que de no ser así, se vulneraría la licencia GPL.

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a

través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

1.4.3.2. Características de MySQL

Las principales características de este gestor de bases de datos son las siguientes:

1. Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.
2. Soporta gran cantidad de tipos de datos para las columnas.
3. Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
4. Gran portabilidad entre sistemas.
5. Soporta hasta 32 índices por tabla.
6. Gestión de usuarios y passwords, manteniendo un muy buen nivel de seguridad en los datos.

Capitulo II

Análisis del sistema

2.1 Planteamiento del problema.

Durante toda su vida universitaria los estudiantes necesitan la elaboración de distintos documentos tanto para la solicitud de becas, seguro social, condonación de pagos, apoyo a congresos, entre otros trámites.

Actualmente la DAE (Dirección de Administración Escolar) proporciona constancias, kárdex, pólizas, certificados y otros documentos necesarios para el alumno.

Dado que un trámite en la DAE mínimo tarda desde 5 días hasta algunos meses hábiles, esto ha provocado que los estudiantes acudan a las Secretarias Académicas de las Unidades Académicas para solicitar algunas de las constancias emitidas por la DAE con el objetivo de realizar su trámite más rápido.

Es por esta razón que surge la idea de generar un sistema que agilice el proceso de emisión de constancias estudiantiles mejorando el tiempo de entrega de los documentos al alumnado, así como permitir consultar la proporción de las demandas de determinados documentos.

2.2 Especificación de requerimientos

- Se necesita una base de datos que contenga todos los campos necesarios para almacenar la información de las constancias para el estudiante.
- El sistema de constancias estudiantiles contará con varios formularios en donde el estudiante colocará los datos necesarios para cada documento.
- Debe contar con un panel de administración para el personal.
- El sistema contará con un sistema de comandos de impresión para cada documento realizado.
- Cuando la constancia este terminada y firmada, el sistema enviará un correo electrónico al alumno informándole que ya puede recoger su constancia.

2.3 Alcances

Mediante el sistema de constancias estudiantiles el alumno solicitara un documento por medio de una página Web. Cuando un estudiante solicite una constancia, esta se almacenará en la base de datos, de tal forma que si el estudiante requiere realizar otra constancia, ya no volverá a capturar la información.

Cuando el estudiante termina la solicitud, los datos son revisados en el panel de control, y posteriormente se imprimen para ser firmados y enviados a la dirección de la Facultad de Ciencias de la Computación en donde el alumno recibirá su constancia.

2.4 Metodología

Se usará el modelo en espiral de la Ingeniería de Software.

El modelo en espiral es una de las metodologías más recomendables para el desarrollo y creación de un programa, ya que consta de pocas etapas o fases, las cuales se van realizando en una manera continua y cíclica. :

2.4.1.- Planificación:

Lo primero que se debe hacer es definir los objetivos, las alternativas y las restricciones y planificar el proyecto inicial.

Identificación de requisitos:

Para conocer los requisitos de cada documento, es necesario analizar los datos que debe contener cada constancia.

Después de comprender lo que se debe realizar, se procederá a identificar los problemas actuales, es decir, las características que deben agregarse o modificarse en el método actual de emitir constancias, los cuales son:

- Mayor confiabilidad y consistencia en el almacenamiento de los datos para eliminar errores al momento de realizar la constancia.
- Disminuir los errores de procesamiento de la información.
- Mayor velocidad de procesamiento para la realización de una constancia.

2.4.2.- Análisis de riesgo:

En este paso se identifican los diferentes riesgos para cada una de las alternativas:

Riesgos:

- Perdida de los datos almacenados en la base de datos.
- La pérdida de los avances en el proyecto.

- Luego de terminado el desarrollo, el programador no proporciona asesoramiento en caso de alguna falla inesperada del sistema.
- La persona encargada del software se encuentre inhabilitada.
- El usuario no se adapte al nuevo sistema.
- Aumento de costo para el asesoramiento del personal

2.4.3.- Ingeniería:

En esta etapa se desarrolla un prototipo inicial del software, en donde se muestra el modelo del software a construir. Este prototipo está compuesto por las diferentes pantallas, las cuales permiten describir la interacción hombre – máquina, facilitando de esta manera que el usuario comprenda como se llevará a cabo dicha interacción.

2.4.4.- Evaluación del cliente:

En esta etapa se le presenta al cliente el programa. En el caso de que el cliente no quede satisfecho, habrá que realizar nuevos arreglos hasta que el producto cumpla con sus expectativas.

2.5 Descripción de la información

2.5.1 Casos de uso

El modelado de Casos de Uso es la técnica más efectiva y a la vez la más simple para modelar los requisitos del sistema desde la perspectiva del usuario. Los Casos de Uso se utilizan para modelar cómo un sistema o negocio funciona actualmente, o cómo los usuarios desean que funcione. Es decir, se está diciendo lo que tiene que hacer un sistema y cómo.

El modelo de casos de uso consiste en actores y casos de uso. Los actores representan usuarios y otros sistemas que interaccionan con el sistema. Se dibujan como "muñecos" de palo. Actualmente representan el tipo de usuario, no una instancia de usuario. Los casos de uso representan el comportamiento del sistema, los escenarios que el sistema atraviesa en respuesta a un estímulo desde un actor. Se dibujan como elipses.

2.5.2 Identificación de casos de uso del sistema

Casos de Uso	**Registro usuario**
Actores	Usuario
Descripción	El usuario coloca sus datos básicos de registro.

Casos de Uso	**Realizar constancia**
Actores	Usuario o Administrador
Descripción	El usuario o administrador, puede emitir una constancia con los formatos disponibles.

Casos de Uso	**Editar constancia**
Actores	Usuario o Administrador
Descripción	El usuario o administrador puede modificar las constancias realizadas anteriormente.

Casos de Uso	**Ver constancias**
Actores	Usuario o Administrador
Descripción	El usuario o administrador pueden ver todas las constancias realizadas, posteriormente volver a solicitarla, borrarla o editarla. El administrador tiene acceso a todas las constancias realizadas por todos los alumnos, mientras que el alumno, solo a las realizadas por el.

Casos de Uso	**Iniciar Sesión**
Actores	Usuario o Administrador
Descripción	El usuario o administrador coloca su usuario y password para poder utilizar el sistema.

Casos de Uso	Enviar mensaje al administrador
Actores	Usuario
Descripción	El usuario puede enviar comentarios o sugerencias al administrador.

Casos de Uso	Administrar Usuarios
Actores	Administrador
Descripción	El administrador puede dar de baja usuarios registrados, y también enviarles mensajes privados.

Casos de Uso	Administrar Periodos
Actores	Administrador
Descripción	El administrador puede crear, editar o borrar periodos.

Casos de Uso	Administrar Formatos
Actores	Administrador
Descripción	El administrador puede crear, editar o borrar formatos de constancia.

Casos de Uso	Administrar Facultades
Actores	Administrador
Descripción	El administrador puede crear, editar o borrar facultades.

Casos de Uso	Administrar Semestres
Actores	Administrador
Descripción	El administrador puede crear, editar o borrar semestres.

Figura 2.1 Identificacion de casos de uso

2.6 Diagrama de Casos de Uso

En el diagrama se muestra como una segunda aproximación el sistema de forma general, en este diagrama, se presentan los actores que van a interactuar directamente con el sistema y sus casos de usos sobre los cuales interactúa:

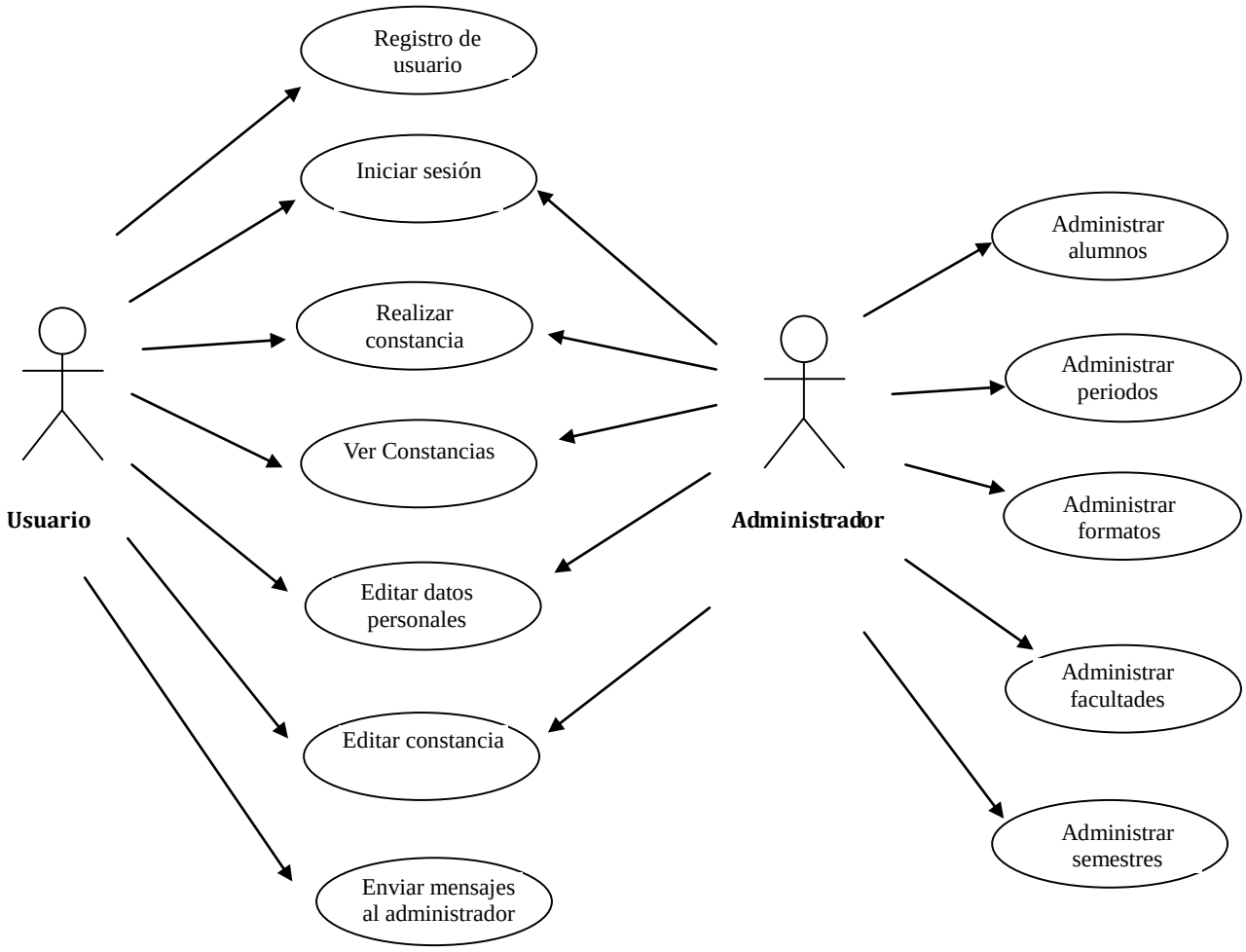

Figura 2.1 Diagrama de casos de uso

2.7 Esquema Entidad-Relación

El modelado entidad-relación Figura 2.4 es una técnica para el modelado de datos utilizando diagramas entidad relación. No es la única técnica pero sí la más utilizada.

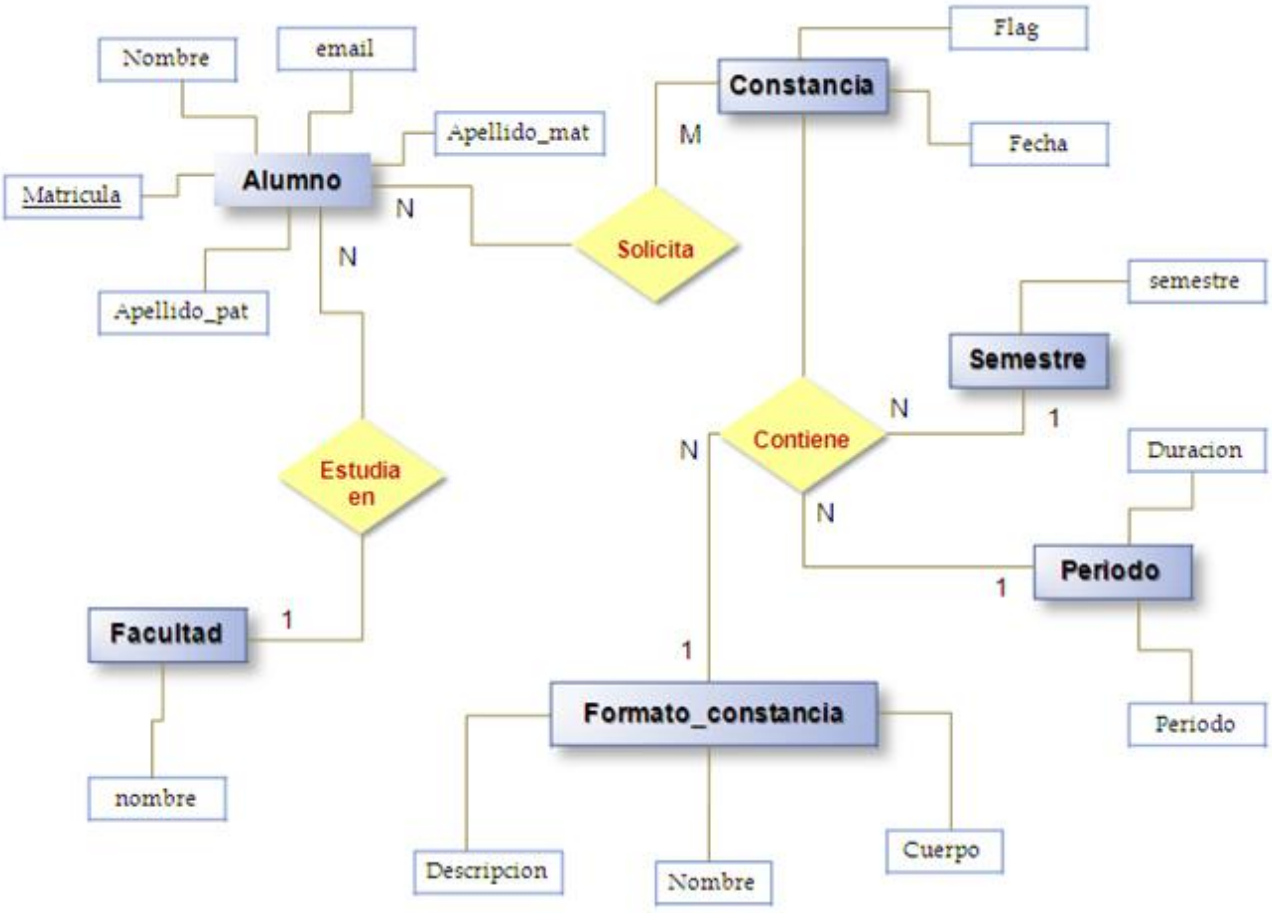

Figura 2.2 Esquema Entidad-Relación

2.8 Descripción de las entidades

En la entidad alumno se almacenan los datos necesarios para realizar las constancias.

En la entidad Facultades se almacenan todas las facultades de las cuales se pueden emitir constancias.

En la entidad Periodo se almacenan los periodos desde 1990 hasta hoy, la cual automáticamente se llenara en cada periodo nuevo.

En la entidad Formato, se encontraran los diferentes tipos de constancias que se pueden elaborar en el sistema, así como también la descripción de las mismas.

En la entidad administrador, se almacenan los datos de acceso de los administradores y el tipo de acceso que tenga (ver, actualizar, borrar, etc.)

2.9 Descripción del esquema relacional

Alumno

Atributo	Simple	Compuesto	Derivado	Univalorado	Multivalorado	Nulo
Nombre	X				X	
Ap_pat	X				X	
Ap_mat	X				X	
Matricula	X			X		
Fecha	X					
Semestre	X				X	
Periodo	X			X		
facultad	X			X		
Descripción	X				X	
Formato	X			X		

Usuarios

Atributo	Simple	Compuesto	Derivado	Univalorado	Multivalorado	Nulo
Id	X			X		
Usuario	X				X	
Password	X				X	
Tipo	X			X		

Facultades

Atributo	Simple	Compuesto	Derivado	Univalorado	Multivalorado	Nulo
Id_facultad	X			X		
facultad	X				X	

Periodos

Atributo	Simple	Compuesto	Derivado	Univalorado	Multivalorado	Nulo
Id_periodo	X			X		
Periodo	X				X	

Formato

Atributo	Simple	Compuesto	Derivado	Univalorado	Multivalorado	Nulo
Id	X			X		
Tipo	X				X	

2.10 Descripción entidad-relación

La entidad Alumno se relaciona con la entidad Formato, debido a que el alumno selecciona alguno de los tipos de formatos especificados en dicha tabla, ese valor será almacenado en la entidad alumno.

La entidad Alumno se relaciona con la entidad Periodo, ya que las constancias realizadas requieren saber el periodo en que se encuentra el estudiante para hacerla vigente.

La entidad alumno se relaciona con la entidad Facultades. De esta forma el sistema de constancias puede ser utilizado para todas las facultades de la universidad, almacenando el id de la facultad en la entidad alumno.

2.11 Modelo Funcional

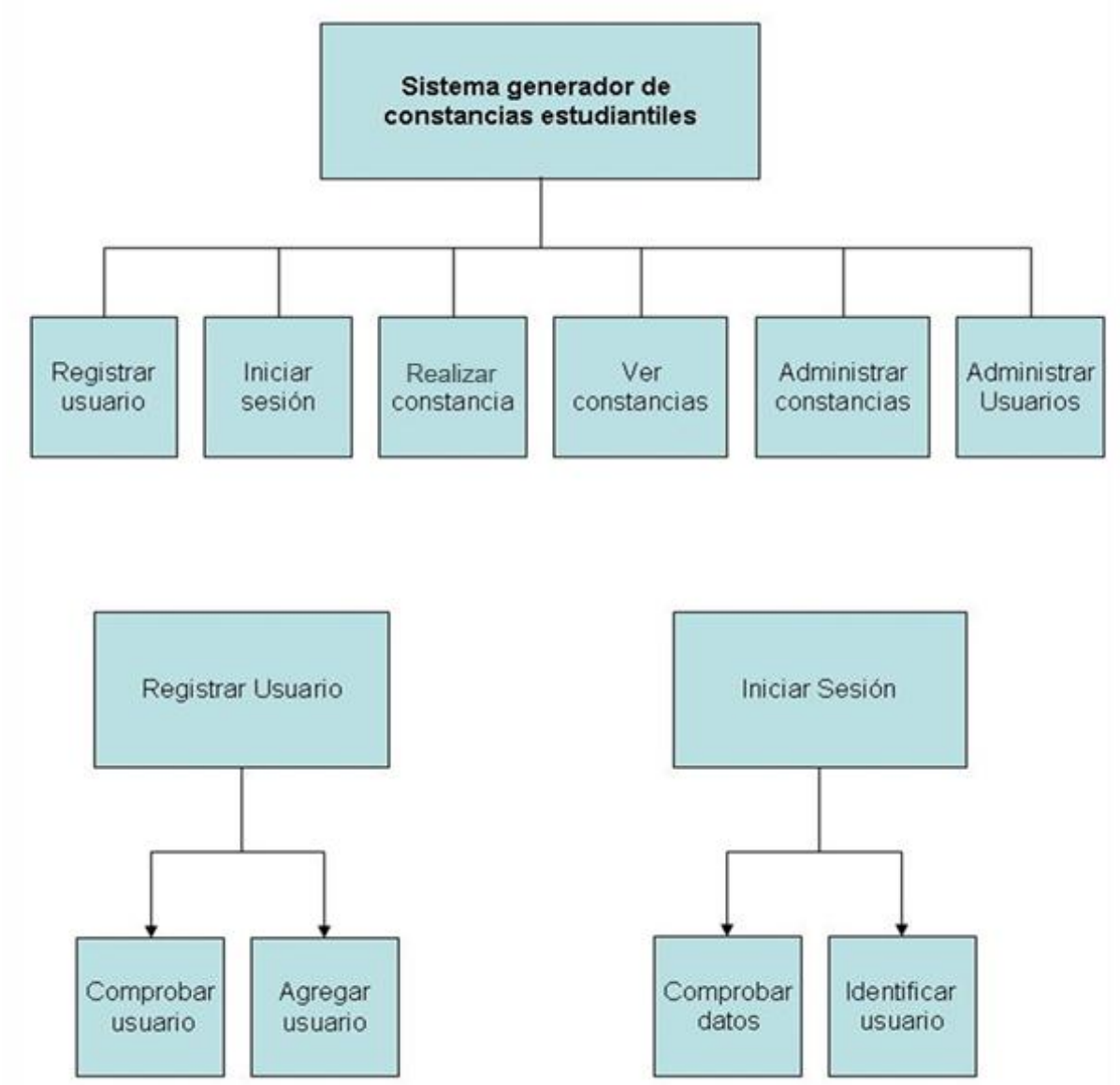

Figura 2.2 Modelo funcional del sistema generador de constancias

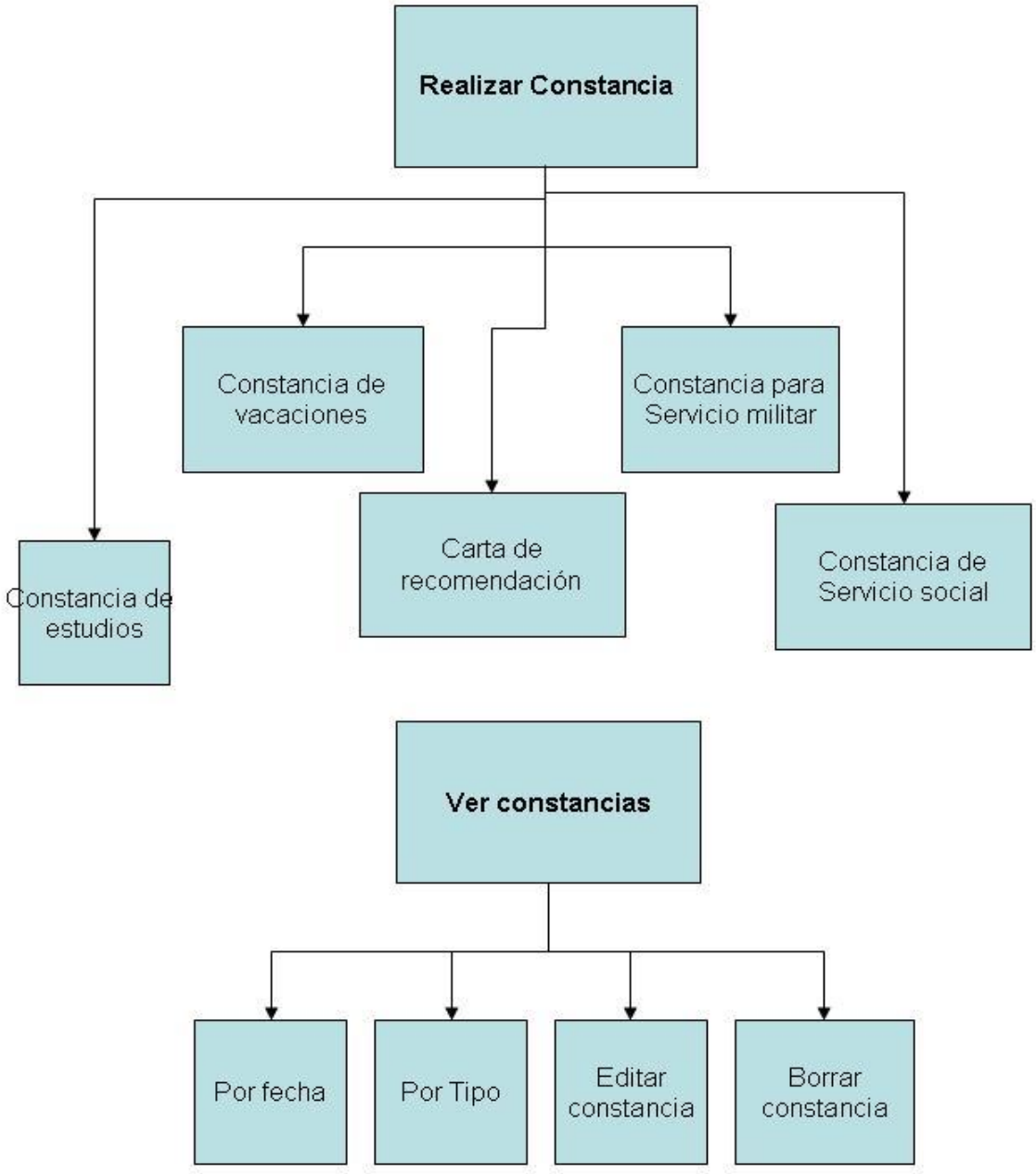

Figura 2.3 Realizar y ver constancias

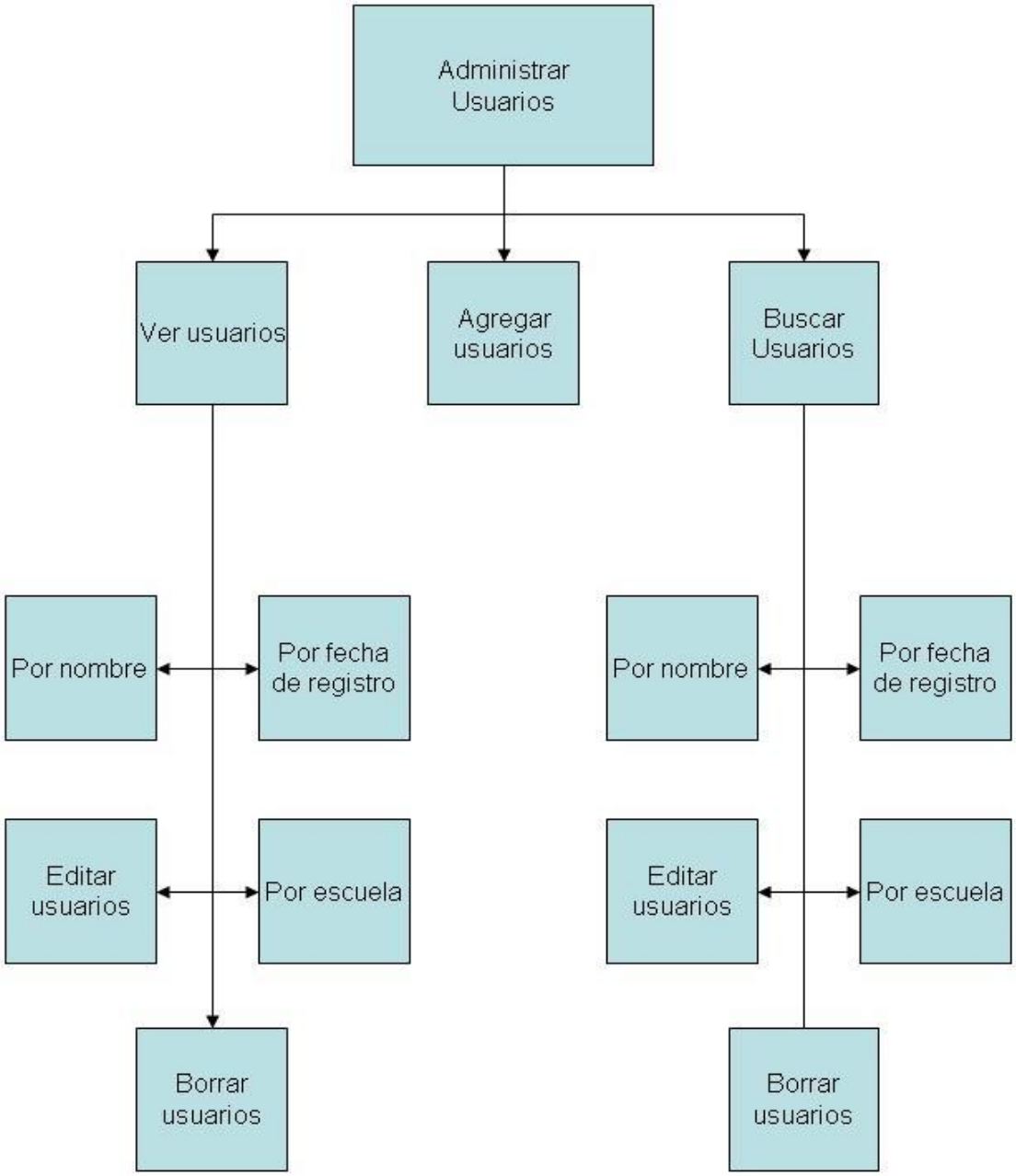

Figura 2.4 Administrar usuarios

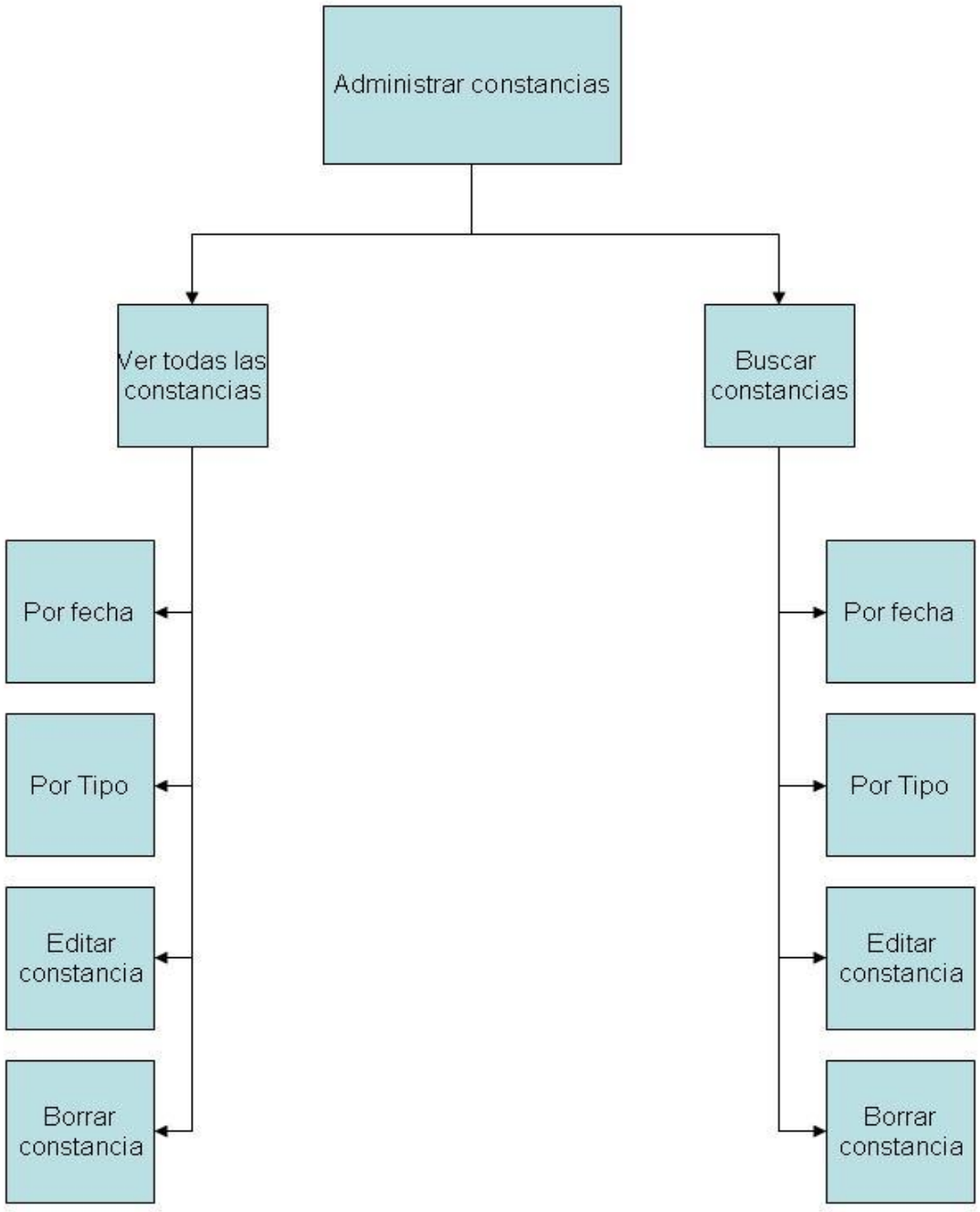

Figura 2.5 Administrar Constancias

2.12 Narrativa de las funciones

El sistema generador de constancias esta representado por las siguientes funciones.

1.- Registrar usuario

En este proceso el sistema solicitara

Nombre completo del usuario

Matricula del usuario

Semestre actual

Periodo actual

Facultad en la que estudia

Nombre de usuario

Clave

2.- Iniciar sesión

- Se requiere que se coloquen los datos de acceso.
- Se comprueba que sean correctos.
- Se le permite el acceso al sistema.

3.- Realizar constancia

Tiene varias opciones:

- Constancia de estudios
- Constancia de vacaciones
- Carta de recomendación
- Constancia para servicio militar
- Constancia para servicio social

Una vez que elige alguna de las opciones, aparece el documento seleccionado.

4.- Ver constancias

Cuando el usuario selecciona esta opción, se mostrara una lista con todas las constancias que ha realizado, posteriormente puede ordenarlas por tipo o fecha.

También puede seleccionar alguna constancia de la lista, editarla o borrarla.

5.- Administrar constancias.

En esta sección se encuentran 2 opciones:

- Ver todas las constancias.
    - En esta opción puede ver todas las constancias emitidas, las cuales puede mostrar por fecha o tipo. También las puede editar o borrar.

- Buscar constancias.
    - El administrador puede buscar constancias por usuario, periodo, matricula, tipo o usuario.
    - Los resultados los puede editar o borrar.

6.- Administrar usuarios.

El administrador cuenta con un menú en el cual puede buscar usuarios, agregar usuarios y borrar usuarios.

Se podrá editar la información personal de los usuarios.

El administrador puede ver todas las constancias que ha realizado algún usuario.

En esta sección el administrador puede borrar o agregar nuevos usuarios.

Capítulo III
Diseño del sistema

3.1 Diseño lógico

El diseño lógico es el proceso de construir un esquema de la información, basándose en un modelo de base de datos específico, independiente del SGBD concreto que se vaya a utilizar y de cualquier otra consideración física.

En esta etapa, se transforma el esquema conceptual en el esquema lógico que utilizará las estructuras de datos del modelo de base de datos en el que se basa el SGBD que se vaya a utilizar, como puede ser el modelo relacional, el modelo de red, el modelo jerárquico o el modelo orientado a objetos. Conforme se va desarrollando el esquema lógico, éste se va probando y validando con los requisitos de usuario.

Para llevar a cabo la transformación del modelo entidad-relación al modelo relacional se siguen las siguientes reglas:

1. Toda entidad se convierte en una tabla.

2. Toda relación **N: M** se genera una nueva tabla que contenga cada una de las llaves primarias de las entidades involucradas en la relación.

3. Toda relación **1: M** se incluye en la tabla **M** la llave primaria del la tabla **1** como llave foránea.

3.2 Modelo relacional

La estructura fundamental del modelo relacional es precisamente esa, "relación", es decir una tabla bidimensional constituida por líneas (tuplas) y columnas (atributos). Las relaciones representan las entidades que se consideran interesantes en la base de datos.

Cada instancia de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representarán las propiedades de la entidad. Por ejemplo, si en la base de datos se tienen que representar usuarios, se podrá definir una relación llamada "Usuarios", cuyos atributos describen las características de los Usuarios. Cada tupla de la relación "Usuarios" representará un usuario en concreto, en la siguiente figura mostramos el modelo relacional de nuestro sistema.

Alumno (matricula **PK,** password, clave_facultad **FK**, nombre, apellido_paterno, apellido_materno, email**)**

Facultad (clave_facultad **PK**, nombre**)**

Formato_Constancia (codigo **PK**, nombre, descripción cuerpo**)**

Periodo (id_periodo **PK**, periodo, duracion**)**

Semestre (id **PK**, semestre**)**

Constancia (id **PK**, matricula **FK**, codigo **FK**, id_periodo **FK**, id_semestre **FK**, fecha, flag**)**

3.3 Normalización de la BD.

La normalización es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional, ya que asegura que las relaciones (tablas) obtenidas no tienen datos redundantes, en seguida aplicaremos la normalización al esquema lógico.

Primera Forma Normal

La regla de la Primera Forma Normal establece que las columnas repetidas deben eliminarse y colocarse en tablas separadas.

Formato
Matricula
Nombre
Password
Ap_paterno
Ap_materno
Email
Fecha
Semestre
Periodo
Facultad
Descripción
Tipo_formato1
Tipo_formato2

Figura 3.1 Tabla original

Ahora tiene dos tablas. Pero todavía hay un problema. No hay forma de relacionar los datos de la tabla original con los de la nueva tabla. Para hacerlo, añadiremos un campo clave a la segunda tabla de forma que se establezca la relación. Para esto se añade a la

tabla Productos una clave primaria que se llame ID_formato y una clave a la tabla Alumno que la relacione con la tabla formato.

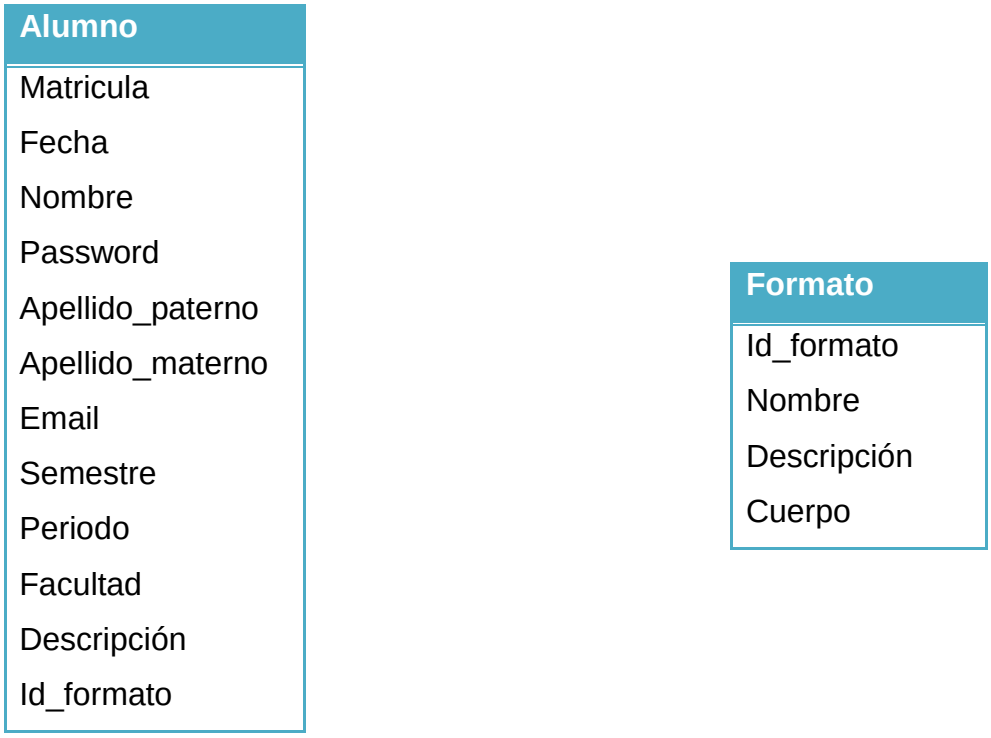

Figura 3.2 Primera forma normal

Así, se ha establecido una relación uno a varios. Ésta representa lo que la base de datos estará haciendo en la vida real. El alumno puede realizar varios formatos, sin tener que volver a capturar sus datos. De esta forma ya tenemos las tablas en la primera forma normal.

Segunda Forma Normal

La regla de la Segunda Forma Normal establece que todas las dependencias parciales se deben eliminar y separar dentro de sus propias tablas. Una dependencia parcial es un

término que describe a aquellos datos que no dependen de la clave de la tabla para identificarlos.

- Determinar cuáles columnas que no son llave no dependen de la llave primaria de la tabla.
- Eliminar esas columnas de la tabla base.
- Crear una segunda tabla con esas columnas y la(s) columna(s) de la PK de la cual dependen.

Las columnas periodo, facultad y semestre no son dependientes de matricula por lo que eliminaremos esas columnas y crearemos una tabla para cada una de ellas y la llave primaria de la que dependen.

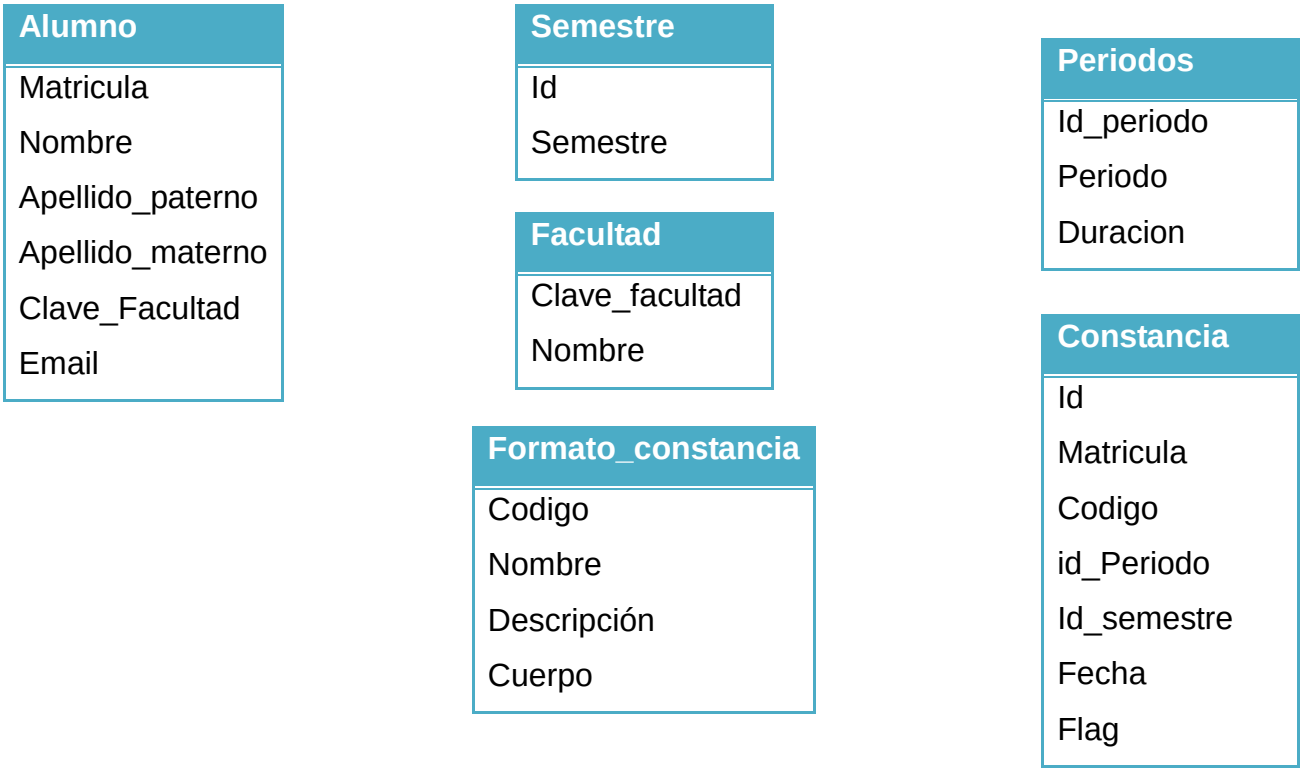

Figura 3.3 Segunda forma normal

La tercera forma normal nos dice que tenemos que eliminar cualquier columna no llave

que sea dependiente de otra columna no llave.

- Determinar las columnas que son dependientes de otra columna no llave.
- Eliminar esas columnas de la tabla base.
- Crear una segunda tabla con esas columnas y con la columna no llave de la cual son dependientes.

Al revisar las tablas creadas a partir de la segunda forma normal, observamos que ya se encuentran en la tercera forma normal.

Tercera forma normal

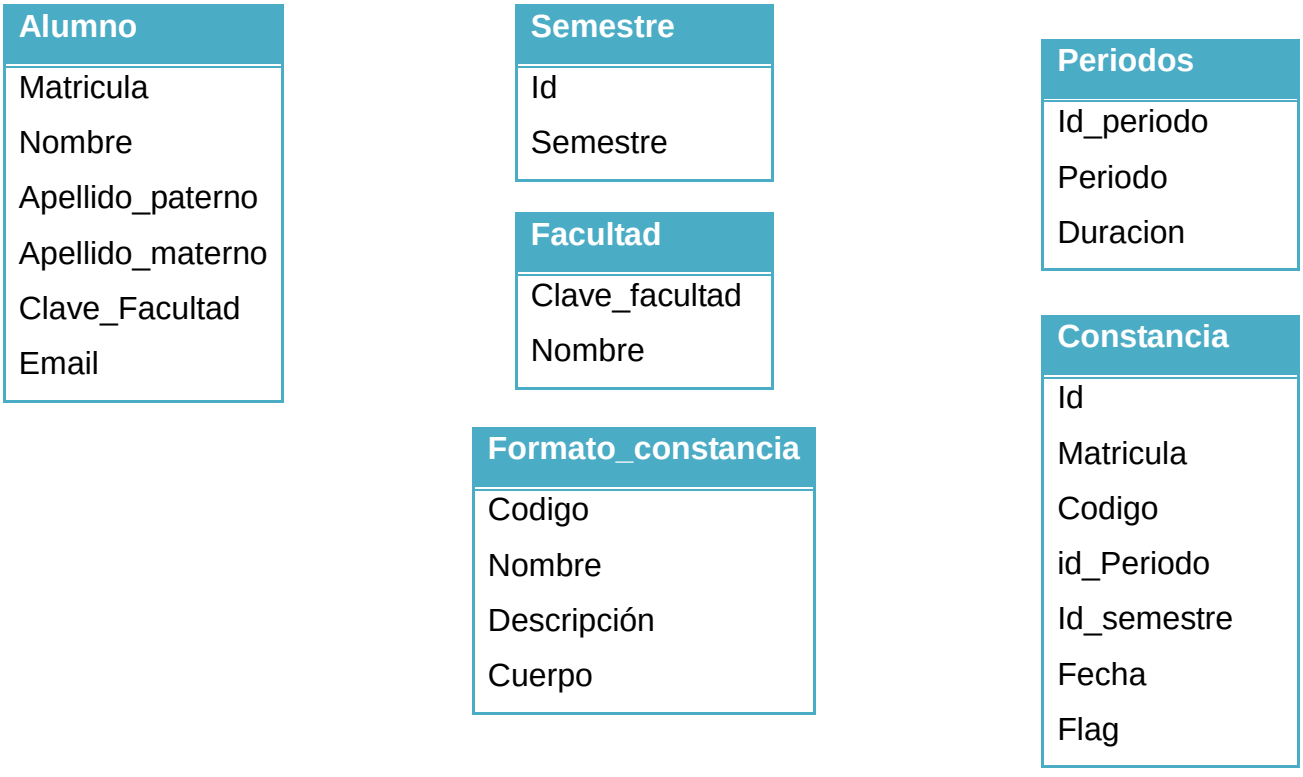

Figura 3.4 Tercera forma normal

3.4 Diseño físico

El diseño físico es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos.

Para llevar a cabo esta etapa, se debe haber decidido cuál es el SGBD que se va a utilizar, ya que el esquema físico se adapta a él. Entre el diseño físico y el diseño lógico hay una realimentación, ya que algunas de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico.

En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior. Concretamente, en el modelo relacional, esto consiste en:

- Obtener un conjunto de relaciones (tablas) y las restricciones que se deben cumplir sobre ellas.
- Determinar las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir unas prestaciones óptimas.
- Diseñar el modelo de seguridad del sistema.

Para llevar a cabo esta etapa, se decidió utilizar MySQL como el SGBD, ya que el esquema físico se adapta a él.

En la siguiente figura 3.5 se muestra la construcción física de la base de datos:

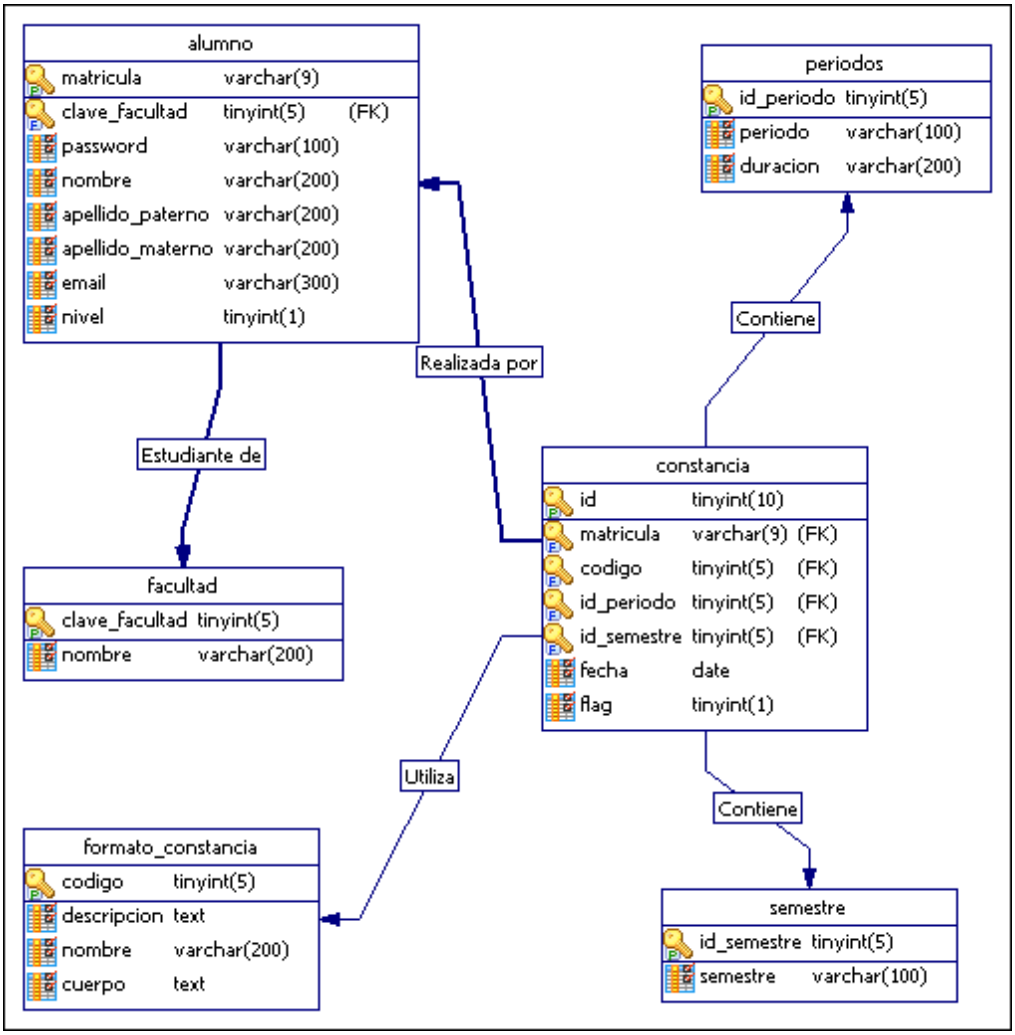

Figura 3.5 Construcción física de la base de datos

3.5 Diccionario de datos.

Un diccionario de datos contiene las características lógicas de los datos que se van a utilizar en el sistema que estamos programando, incluyendo nombre, descripción, alias, contenido y organización.

3.6 Identificación de entidades.

Las entidades identificadas después de haber realizado el análisis de requerimientos del sistema, son las siguientes:

Entidad	Descripción
Alumno	Contiene la información de los alumnos
Facultad	Contiene la información de las facultades que acepta el sistema
Formato_constancia	Contiene el nombre, descripción y formato utilizado para las constancias.
Periodo	Contiene la información de los periodos con los que se pueden realizar documentos.
Semestre	Contiene los nombres de los semestres.
Constancia	Contiene toda la información necesaria para realizar una constancia.

Figura 3.6 Tabla de entidades

3.7 Identificación de relaciones.

Una vez definidas las entidades, se procede a identificar las relaciones existentes entre ellas junto con la cardinalidad con la que participa cada entidad en cada una de las relaciones, las cuales se muestran a continuación:

Relaciones	Relación	Cardinalidad
Alumno – Constancia	Solicita	Muchos a muchos
Alumno – Facultad	Estudia en	Muchos a uno
Constancia – Semestre	Contiene	Muchos a uno
Constancia – Periodo	Contiene	Muchos a uno
Constancia – Formato_constancia	Contiene	Muchos a uno

Figura 3.7 Tabla de relaciones

3.8 Identificación de claves primarias.

Cada entidad posee al menos un identificador primario ó clave primaria (PK). Se trata de encontrar todos los identificadores de cada una de las entidades, los cuales pueden ser simples o compuestos. De cada entidad hemos escogido uno de los identificadores como clave primaria en la fase del diseño lógico.

Entidad: Alumno

Atributo	Identificador
matricula	PK

Entidad: Facultad

Atributo	Identificador
Clave_facultad	PK

Entidad: Periodos

Atributo	Identificador
Id_periodo	PK

Entidad: Semestre

Atributo	Identificador
Id	PK

Entidad: Formato_Constancia

Atributo	Identificador
Codigo	PK

Entidad: Constancia

Atributo	Identificador
id	PK

Figura 3.8 Tablas de claves primarias

Capítulo IV

Implementación del sistema

4.1 Implementación de la base de datos en MYSQL

Tabla: alumno

Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
matricula	varchar(9)	latin1_swedish_ci		No		
password	varchar(100)	latin1_swedish_ci		No		
clave_facultad	tinyint(5)		UNSIGNED	No		
nombre	varchar(200)	latin1_swedish_ci		No		
apellido_paterno	varchar(200)	latin1_swedish_ci		No		
apellido_materno	varchar(200)	latin1_swedish_ci		No		
email	varchar(300)	latin1_swedish_ci		No		
nivel	tinyint(1)		UNSIGNED	No		

Figura 4.1 Tabla de alumnos

Tabla: facultad

Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
clave_facultad	tinyint(5)		UNSIGNED	No		auto_increment
nombre	varchar(200)	latin1_swedish_ci		No		

Figura 4.2 Tabla de facultad

Tabla: periodos

Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
id_periodo	tinyint(5)		UNSIGNED	No		auto_increment
periodo	varchar(100)	latin1_swedish_ci		No		
duracion	varchar(200)	latin1_swedish_ci		No		

Figura 4.3 Tabla de periodos

Tabla: semestre

Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
id	tinyint(5)		UNSIGNED	No		auto_increment
semestre	varchar(100)	latin1_swedish_ci		No		

Figura 4.4 Tabla de semestres

Tabla: formato_constancia

Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
codigo	tinyint(5)		UNSIGNED	No		auto_increment
descripcion	text	latin1_swedish_ci		No		
nombre	varchar(200)	latin1_swedish_ci		No		
cuerpo	text	latin1_swedish_ci		No		

Figura 4.5 Tabla de formatos de constancias

Tabla: constancia

Campo	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Extra
matricula	varchar(9)	latin1_swedish_ci		No		
codigo	tinyint(5)			No		
id_periodo	tinyint(5)		UNSIGNED	No		
fecha	varchar(200)	latin1_swedish_ci		No		
id_semestre	tinyint(5)		UNSIGNED	No		
id	tinyint(10)		UNSIGNED	No		auto_increment
flag	tinyint(1)			No		

Figura 4.6 Tabla de constancias

4.2 Interfaz del sistema

El diseño de la interfaz está realizado con varias herramientas para el diseño web tales como CSS, FLASH y JavaScript. Las funciones principales del sistema para un usuario son las siguientes:

- Crear una cuenta de usuario
- Control de acceso
- Solicitar una constancia
- Historial de constancias realizadas
- Datos personales
- Enviar sugerencias o dudas al administrador

Figura 4.7 Sistema generador de constancias.

Figura 4.8 Registro de usuarios

Una vez que el usuario ha registrado correctamente sus datos, se enviara un correo electrónico al usuario indicándole sus datos de acceso, posteriormente será dirigido a la página de acceso, en donde deberá colocar su matrícula y su password para acceder al sistema.

Figura 4.8 Control de acceso

Si el alumno ingreso algún dato incorrecto, se desplegara una pantalla en donde muestra el error que se encontró al tratar de acceder al sistema.

Figura 4.9 Error en el control de acceso.

Un vez que el usuario acceso correctamente al sistema, se muestran los formatos de constancias disponibles a realizar, en el cual el alumno seleccionara el formato necesario.

Figura 4.10 Formatos de constancia

Si el alumno, selecciona alguno de los formatos permitidos, se procede a solicitar los datos necesarios para terminar y emitir el documento.

Figura 4.11 Solicitar constancia.

Cuando el alumno, termina de solicitar la constancia, se mostrará la siguiente figura, en la cual se encuentran los datos de la constancia emitida, así como también la opción de ver el documento en formato PDF.

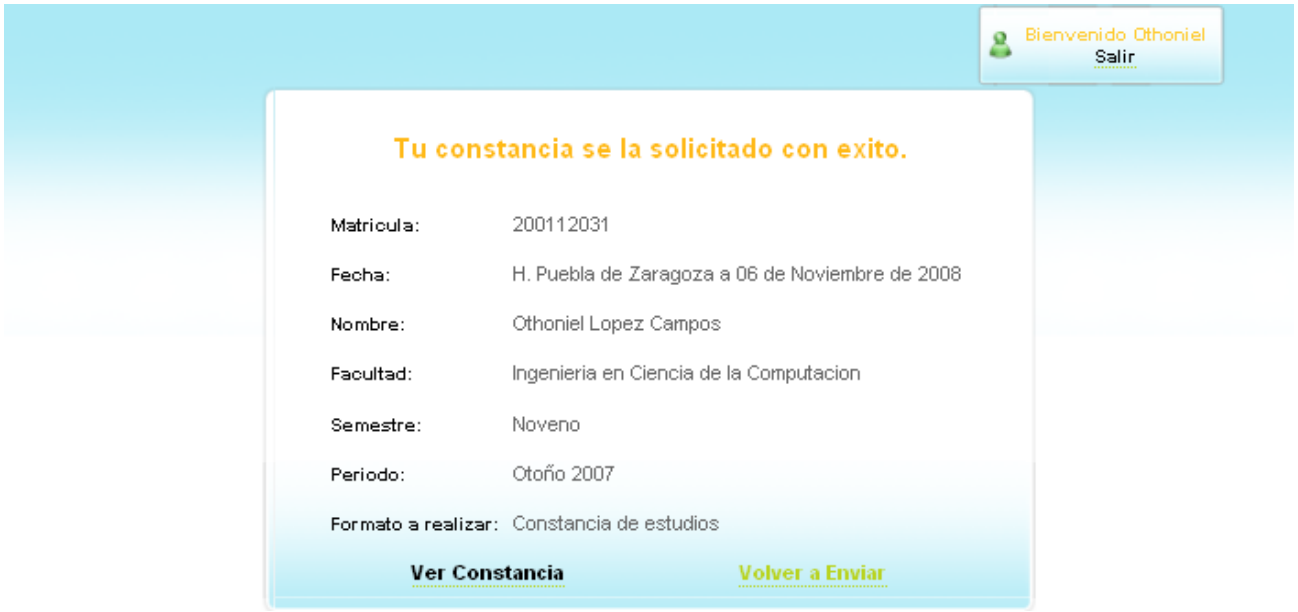

Figura 4.12 Constancia realizada

Después de solicitar algún documento, el sistema contiene una sección en la que el alumno puede ver un historial de todas las constancias solicitadas, de esta forma, volver a solicitarlas.

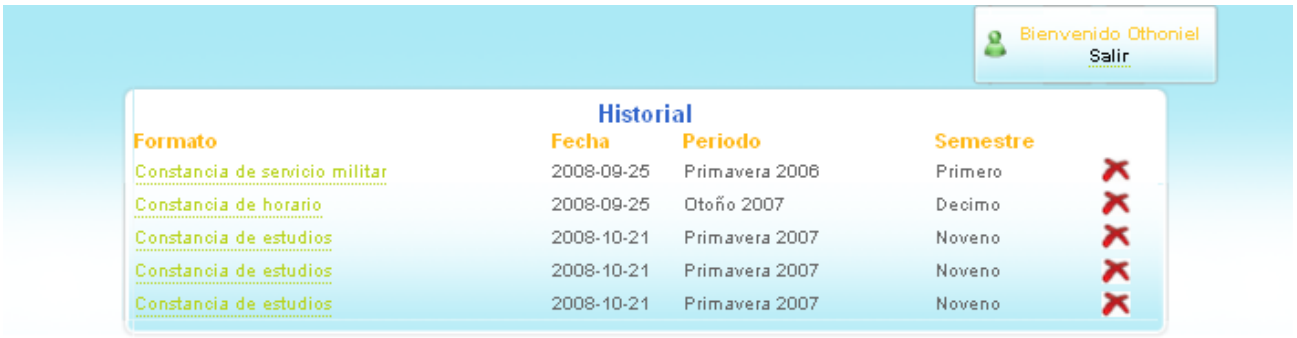

Figura 4.13 Historial de constancias

Dentro del sistema, el alumno puede modificar sus datos personales, tales como nombre, apellidos, correo electrónico, facultad en la que estudia, y password.

Figura 4.15 Perfil de usuario

Para modificar la información en el perfil de usuario, basta con dar click sobre el campo que se desea modificar y se desplegara un campo de texto para modificar los datos.

Figura 4.16 Edición de datos en perfil.

Dentro del sistema, puedes enviar mensajes de sugerencias o críticas a la administración desde la sección de contacto.

Figura 4.17 Sección de contacto

El sistema cuenta con una sección para el administrador, en la que se administran las constancias solicitadas, los usuarios registrados, los formatos permitidos, y otros datos que contiene el sistema.

En la siguiente figura se muestran las constancias solicitadas por el alumno, en esta pantalla el administrador puede enviar correos electrónicos al usuario que solicito la constancia, así como también borrar la constancia de la base de datos, imprimirla o borrarla. Una vez que la constancia fue revisada, el sistema enviara un correo electrónico al alumno, indicándole que su constancia ha sido revisada y terminada correctamente, por lo que puede pasar a recogerla a las oficinas de la facultad. Las constancias, que no han sido revisadas, se muestran al inicio, indicando que están en espera.

Figura 4.18 Constancias en el panel de control.

El administrador puede ver los usuarios registrados, así como también darlos de baja.

Figura 1.19 Administración de alumnos.

También puede modificar la tabla de facultades, agregando nuevas facultades de las cuales se puede solicitar constancias, o borrar facultades de la base de datos.

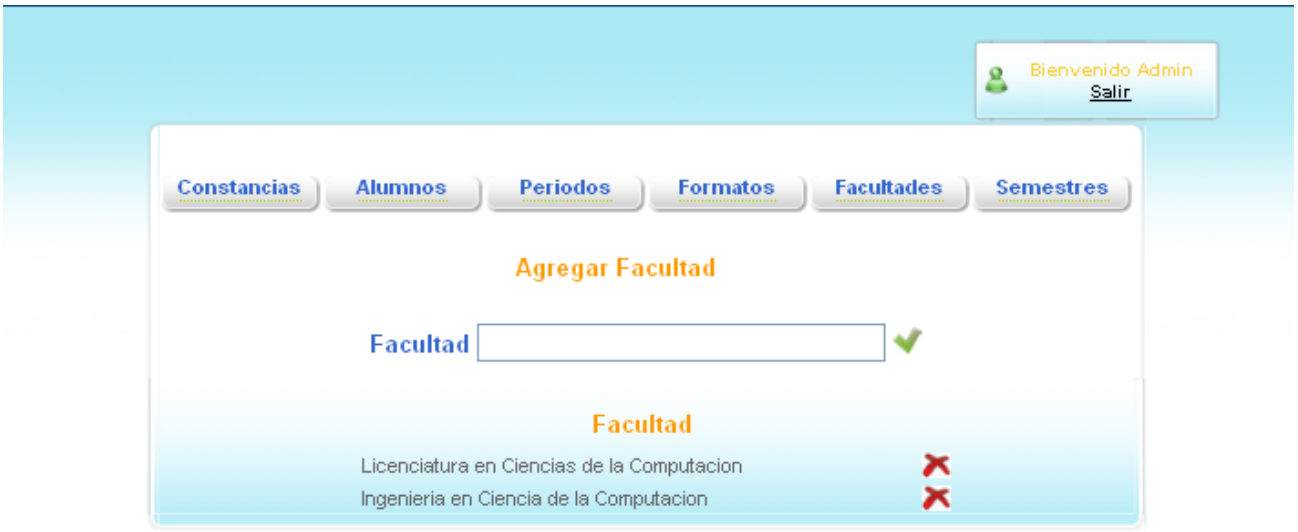

Figura 4.20 Administración de facultades

El administrador puede modificar los semestres permitidos en las constancias.

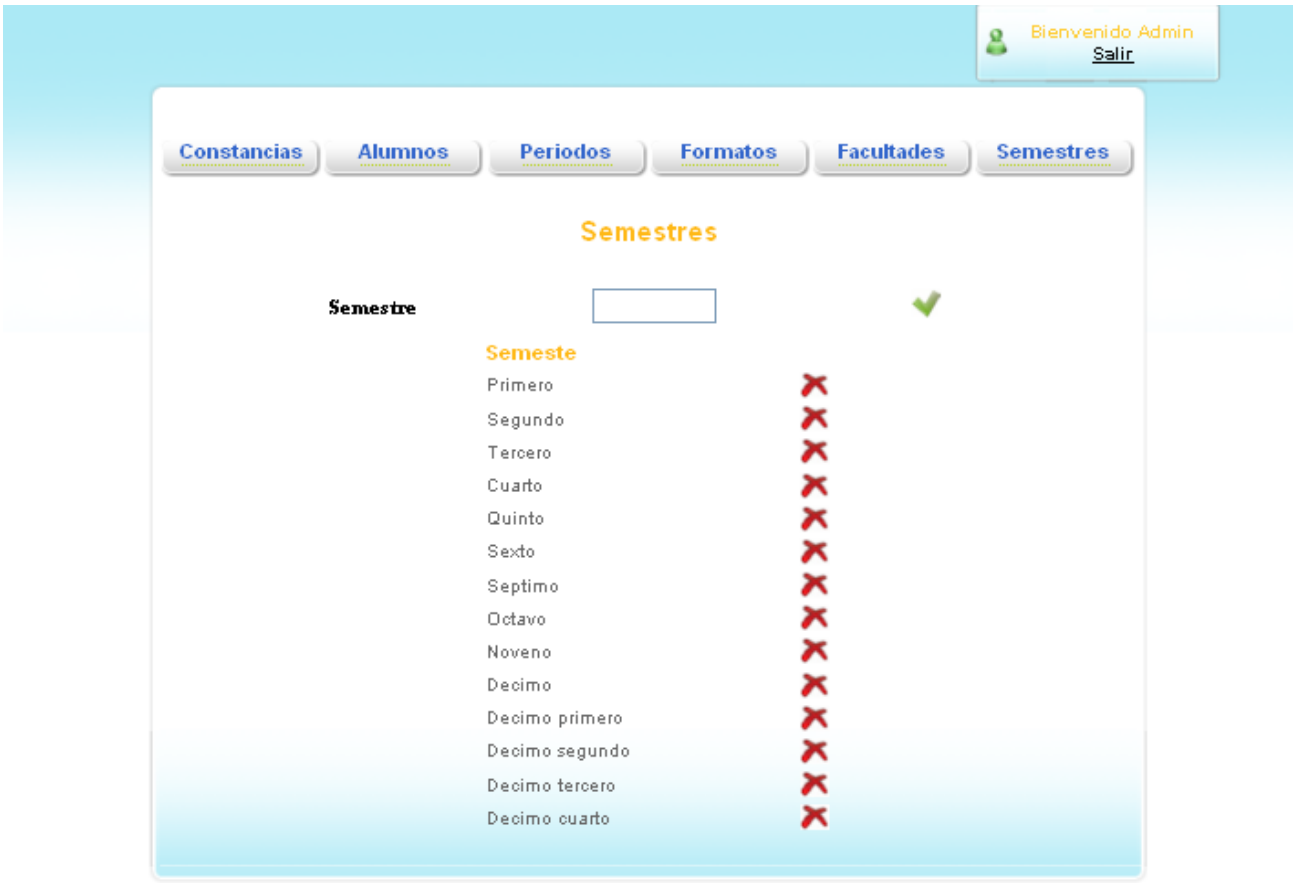

Figura 4.21 Administración de semestres

El panel de administración, cuenta con una sección para poder administrar los formatos de constancias permitidos por el sistema.

Figura 4.22 Administración de formatos de constancias.

Si el administrador desea realizar un nuevo formato, se desplegara la pantalla mostrada en la siguiente figura, en la cual se solicitan el nombre y descripción de la constancia.

Figura 4.23 Realizar nuevo formato.

Cuando el administrador completa los campos solicitados anteriormente, procede a dar click en el botón de siguiente, se mostrara la siguiente figura. En la cual se colocaran los datos necesarios en la constancia. El sistema utilizara algunas palabras restringidas como comandos para sustituirlos por los datos del alumno, por lo que el administrador deberá colocarlos de la forma en que se muestra en el sistema para que funcionen correctamente.

El editor utilizado en el sistema, cuenta con una opción por medio de la cual el administrador puede importar un documento hecho en Microsoft Word y convertirlo en un formato permitido por el sistema

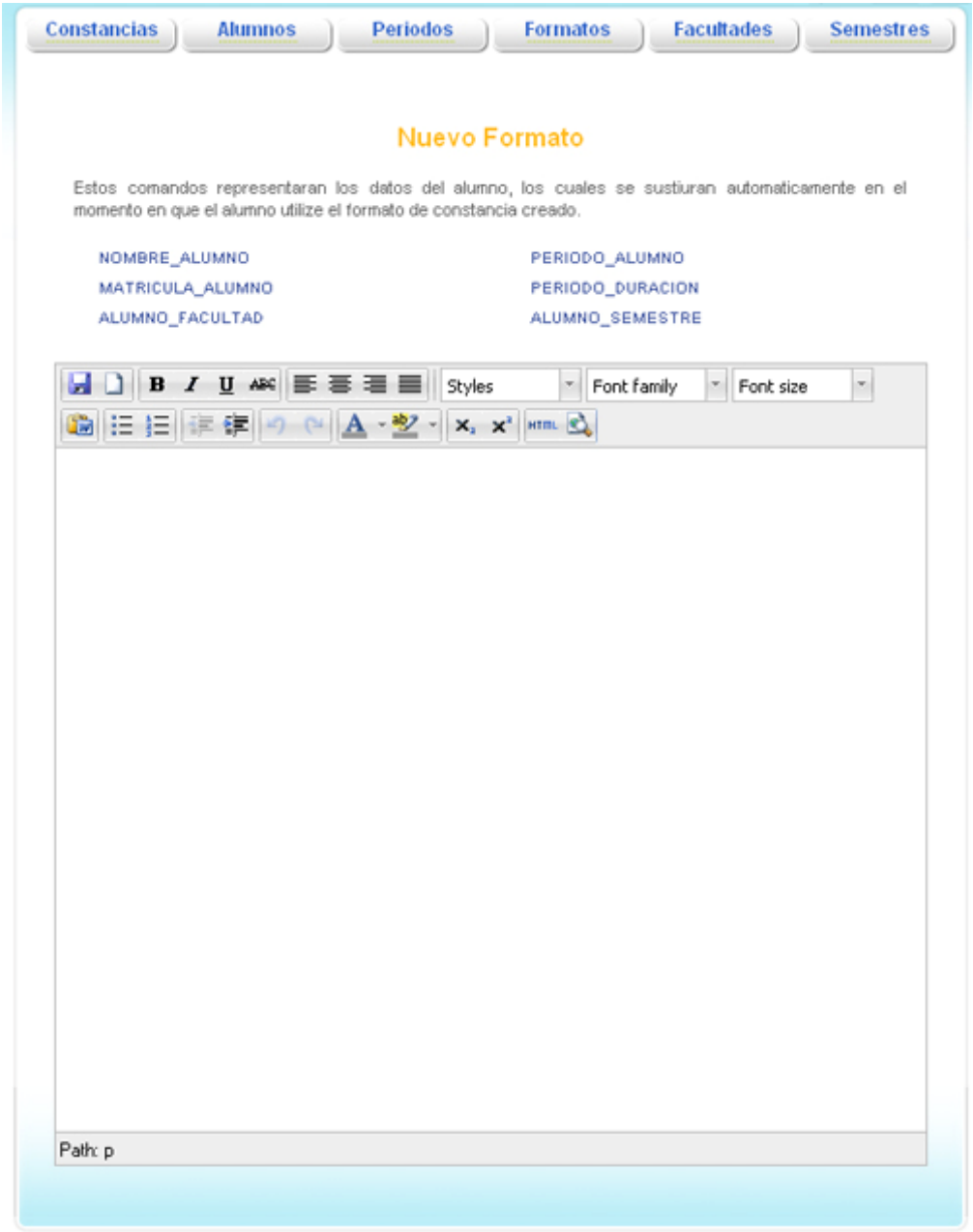

Figura 4.24 Nuevo formato

El administrador también puede editar los formatos existentes, para modificar el nombre o la descripción, basta con dar click sobre el dato que se requiere modificar para que se vuelva editable.

Cuando el administrador termine de modificar la constancia, debe dar click sobre el icono de guardar para que el formato se modifique.

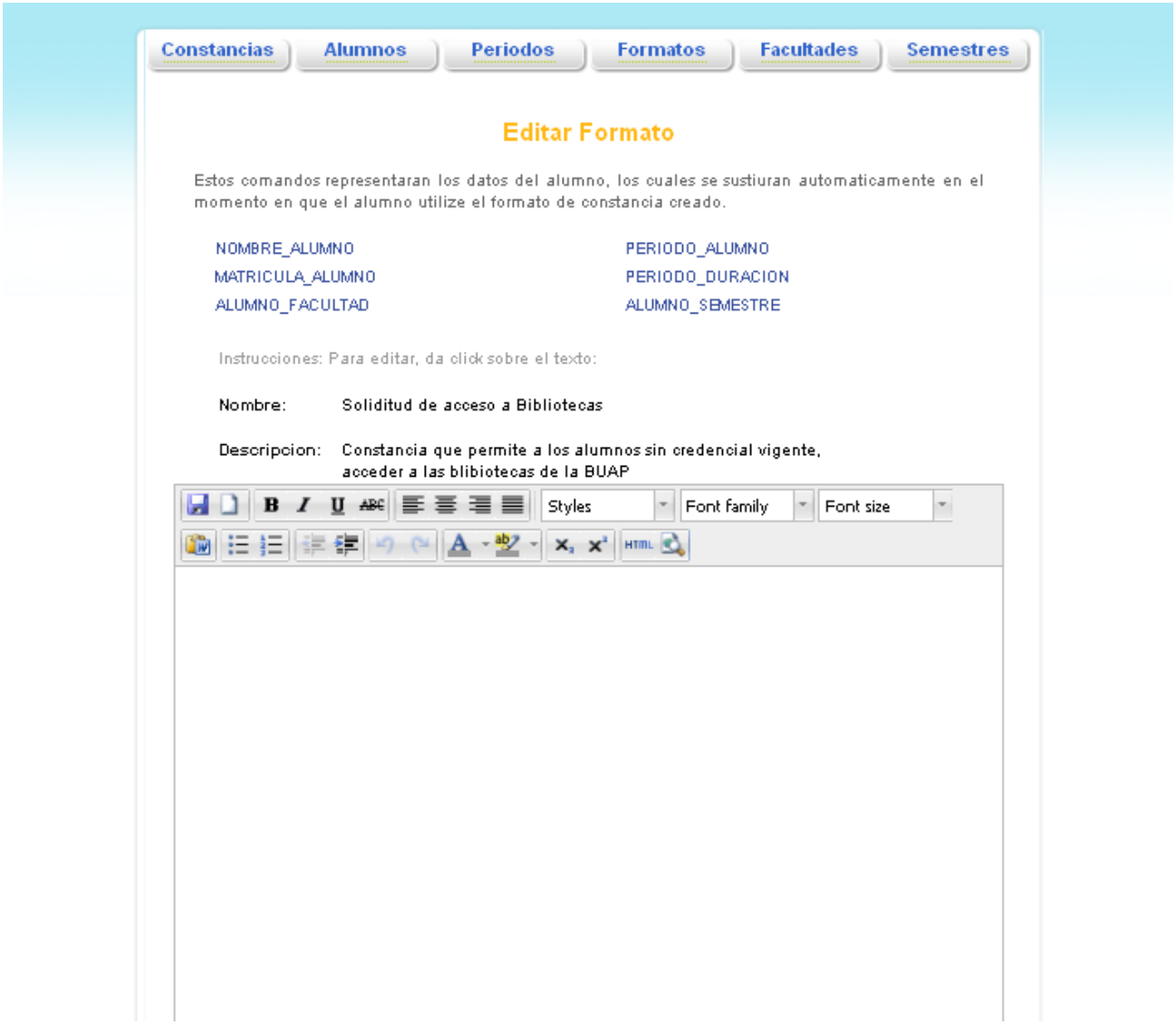

Figura 4.25 Editar Formato.

Conclusiones

El análisis, diseño e implementación del sistema cumplieron el objetivo general, ya que ahora tenemos un sistema capaz de generar constancias escolares, el cual facilita la solicitud de las mismas, tanto para el alumno como para el administrador.

Se decidió utilizar PHP, MySQL debido a la experiencia que se tenía trabajando con estos lenguajes.

Mediante este sistema, se pretende generar un gran beneficio tanto económico como de ahorro de tiempo al realizar las constancias emitidas, y debido al uso de AJAX, se espera que sea más fácil el uso del sistema, así como también se espera que sea más amigable para el usuario y/o administrador.

Limitaciones

Una limitación del sistema, es que si en algún momento se requiere un campo adicional en alguna constancia que contenga datos específicos del alumno, el cual no se encuentre entre los definidos actualmente, no lo puede agregar un administrador, por lo que debe ser modificado por el programador.

Perspectivas

Que el sistema sea capaz de crear nuevas palabras restringidas de modo dinámico, de tal forma que se solucione la limitación actual.

Actualmente la tecnología lleva al uso del internet mediante dispositivos móviles, tales como el celular. Por lo que seria interesante realizar un sitio compatible con los dispositivos móviles para que se puedan realizar constancias desde cualquier sitio con acceso a internet.

Que el sistema sea capaz de generar reportes gráficos y estadísticas para el administrador los cuales harían más amigable el sistema, y permitiría evaluar el porcentaje de rendimiento del sistema.

Bibliografía

[SOM05] "Ingenieria del Software" (Sommerville05) by Ian Sommerville, Maria Isabel Alfonso Galipienso, and Antonio Botia Martinez (Paperback - Jan 2005)

[ABR07] Fundamentos de Bases de Datos by Abraham Silberschatz (Feb 2007)

[PRE01] "Software Engineering: A Practitioner's Approach" by Roger S Pressman and Roger Pressman (Hardcover - April 2, 2001)

[CLI08] Cliente-Servidor
http://es.wikipedia.org/wiki/Cliente-servidor

[PHP08] PHP: Hypertext Preprocessor. 2008
http://php.net/
http://www.php.net/manual/es/

[MYS08] Manual de referencia de MySQL 5.0 - MySQL AB
http://dev.mysql.com/doc/refman/5.0/es/index.html

[AJA07] Introducción a AJAX, Javier Eguíluz Pérez

[SIE07] Ajax En J2ee
MARTIN SIERRA, A. J. (Editorial Ra-ma)

[JAM08] Flash 8. Imagen, Animación E Interactividad
Mohler, James L. (Ed. Anaya Multimedia)