Benemérita

Universidad Autónoma De Puebla

Facultad de Ciencias de la Computación

“SISTEMA DE CONTROL DE PROYECTOS“

TESIS PROFESIONAL

Para obtener el título de:

LICENCIADA EN CIENCIAS DE LA COMPUTACIÓN

Presenta:

HILDA QUECHOL TEOPILA

Asesor:

DRA. DARNES VILARIÑO AYALA

Puebla, Púe. SEPTIEMBRE 2009

INDICE DE CONTENIDO

INTRODUCCIÓN

En la Facultad se desarrollan proyectos de investigación que pueden ser internos, proyectos VIEP o proyectos CONACYT en los cuales trabajan profesores, que pueden tener a su cargo, uno o más proyectos y pueden contar con la participación de estudiantes. Sin embargo la información pertinente para ser registros, consultas y mantenimiento generalmente se encuentra dispersa y solo la conoce el responsable del proyecto, por ello se plantea desarrollar un Sistema que automatice las actividades asociadas a dichos proyectos.

Por lo anteriormente planteado el Objetivo General de este trabajo consiste en desarrollar un sistema de Control de Proyectos para minimizar el tiempo de registro y consultas.

Los objetivos específicos son los siguientes

- Análisis de la problemática con respecto a la asignación y consultas de proyectos escolares.
- Elaborar un estudio sobre la teoría de Base de Datos para el Sistema.
- Diseño del Sistema con base en el Modelo Entidad-Relación.
- Desarrollo e implementación del Sistema en Mysql y Php.
- Redacción de la tesis.

Al elaborar el Sistema de Control de proyectos se podrán adquirir beneficios importantes como lo son:

- Control de la información de los profesores y alumnos que participan en algún tipo de proyecto de investigación.
- Manejo confiable de datos de los profesores y alumnos.
- Consultas eficaces.

El contenido presentado de la tesis es el siguiente:

El CAPITULO 1, presenta los conceptos de Ingeniería de Software y Bases de Datos que servirán para el desarrollo y elaboración del Sistema.

El CAPITULO 2, presenta el análisis y diseño del Sistema, desarrollo de los diagramas de flujos de datos, el diccionario de datos y el diagrama funcional del sistema. Se describe la información que se almacenó en la base de datos. Se discute el modelo entidad –relación y la normalización de la base en aras de garantizar la consistencia e integridad de la misma.

En el CAPITULO 3 se discute la implementación desarrollada del sistema, por último se plantean las Conclusiones del trabajo desarrollado.

CAPÍTULO 1
MARCO TEORICO

1.1. INGENIERIA DE SOFTWARE

La ingeniería de Software aparece a fines de los años sesenta. El término ingeniería de software abarca al grupo de método, técnicas y herramientas que se utilizan en el desarrollo de la programación, que es la actividad principal al momento de crear un software.

La ingeniería de Software se define como una disciplina de la ingeniería en donde se resuelven problemas relacionados con el desarrollo de sistemas de software el cual permite obtener software confiable, rentable, que trabaje con máquinas reales y que tenga funcionamiento eficiente para satisfacer las necesidades del usuario. Cabe mencionar que se preocupa de la fiabilidad y el rendimiento, tratando de reducir costos y complejidad.

También se define como la disciplina tecnológica preocupada de la producción sistemática y del mantenimiento de los productos de software, que son desarrollados y modificados en tiempo y dentro de un presupuesto definido. Las actividades de la ingeniería de Software son: analizar, diseñar, construir e implementar sistema de información.

La ingeniería de software, por lo tanto, incluye el análisis previo de la situación, el diseño del proyecto, el desarrollo del software, las pruebas necesarias para confirmar su correcto funcionamiento y la implementación del sistema. Cabe destacar que el proceso de desarrollo de software implica lo que se conoce como ciclo de la vida del software, que está formada por cuatro etapas: concepción, elaboración, construcción y transición.

1.1.1 Ciclo de vida del software

El término ciclo de vida del software describe el desarrollo del software, desde la fase inicial hasta la fase final. Su propósito es definir las distintas fases intermedias que se requieren para validar el desarrollo de la aplicación, es decir, para garantizar que el software cumpla los requerimientos para la aplicación y verificación de los procedimientos de desarrollo: se asegura que los métodos utilizados son apropiados.

El ciclo de vida permite que los errores se detecten lo antes posible y por lo tanto, permite a los desarrolladores concentrarse en la calidad del software, en los plazos de implementación y en los costos asociados

El ciclo de vida básico de un software consta de los siguientes procedimientos:

- Definición de objeto: definir el resultado del proyecto y su papel en la estrategia global.
- Análisis de los requerimientos y su viabilidad: recopilar, examinar y formular los requerimientos del cliente y examinar cualquier restricción que se pueda aplicar.
- Diseño general: requisitos generales de la arquitectura de la aplicación.
- Diseño en detalle: definición precisa de cada subconjunto de la aplicación.
- Programación: es la implementación de un lenguaje de programación para crear las funciones definidas durante la etapa de diseño.

- Prueba de unidad: prueba individual de cada subconjunto de la aplicación para garantizar que se implementaron de acuerdo con las especificaciones.
- Documentación: sirve para documentar información necesaria para los usuarios del software y para desarrollos futuros.
- Implementación.
- Mantenimiento: para todos los procedimientos correctivos y las actualizaciones secundarias del software (mantenimiento continuo).

1.1.2 Modelos de ciclo de vida

Los modelos de ciclo de vida se han actualizado para reflejar las etapas de desarrollo involucradas y la documentación requerida, de manera que cada etapa se valida antes de continuar con la siguiente etapa. Un modelo de ciclo de vida se define como una vista de las actividades que ocurren durante el desarrollo del software, intenta determinar el orden de las etapas involucradas y los criterios de transición asociados entre estas etapas.

Existen varios modelos del ciclo de vida, pero según la definición del proyecto, el modelo del ciclo de vida que se seleccione, influye en las decisiones de planificación y en el éxito que pueda tener el proyecto. Este puede orientar al proyecto y ayudar a asegurar que cada paso se acerque más al objetivo.

Dependiendo del modelo del ciclo de vida que se seleccione, se podría aumentar la velocidad del desarrollo, la calidad, el control y el seguimiento del proyecto, minimizar riesgos y gastos. La selección errónea del modelo de ciclo de vida puede ocasionar que sea lento, innecesario y redundante el trabajo.

1.1.2.1.1 Modelo en cascada

El modelo de ciclo de vida en casaca o también conocido como ciclo de vida básico o modelo lineal se comenzó a diseñar en 1966 y se terminó alrededor de 1970. Se define como una secuencia de fases en la que al final de cada una de ellas se reúne la documentación, para garantizar que cumpla las especificaciones y los requerimientos de pasar a la fase siguiente.

El modelo de cascada sirve como bloque de construcción para los demás modelos de ciclo de vida. El modelo de cascada dice que el desarrollo del software puede ser a través de una secuencia simple de fases. Donde cada fase tiene un conjunto de metas bien definidas y las actividades dentro de una fase construyen a la satisfacción de metas de esa fase, o posiblemente a una secuencia de metas de la fase.

Ejemplo de ciclo de vida en cascada se muestra en la fig.1

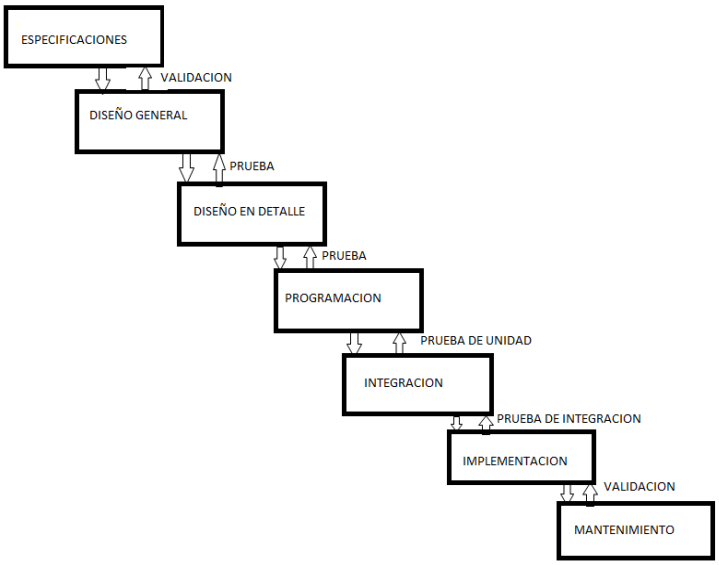

FIGURA 1. MODELO DE CASCADA

La flecha de avance muestra el flujo normal. Las flechas hacia atrás muestran la retroalimentación.

1.1.2.1.2 Modelo V

El modelo de ciclo de vida V proviene del principio que establece que los procedimientos utilizados para probar si la aplicación cumple las especificaciones ya deben haberse creado en la fase de diseño

1.1.2.1.3 Modelo de Desarrollo Incremental

Los riesgos asociados con el desarrollo de sistemas largos y complejos son enormes, una de las formas de reducir los riesgos es construir sólo una parte del sistema, reservados otros aspectos para niveles posteriores.

El desarrollo incremental es el proceso de construcción siempre incrementando subconjuntos de requerimientos para el sistema completo. El desarrollo incremental no demanda una forma específica de observar el desarrollo de algún otro incremento. El modelo de desarrollo incremental provee algunos beneficios significativos para los proyectos:

- Construir un sistema pequeño es siempre menos riesgoso que construir un sistema grande.
- Al ir desarrollando parte de las funcionalidades, es más fácil determinar si los requerimientos planeados para los niveles subsiguientes son correctos.
- Si hay algún error, sólo la última interacción necesita ser descartada.
- Reduce el tiempo de desarrollo de un sistema en el caso que incremente el sistema, decrecen las probabilidades que esos requerimientos de usuarios puedan cambiar durante el desarrollo.

- Si un error importante se realizó, el incremento previo puede ser usado.
- Los errores de desarrollo realizados en un incremento, pueden ser arreglados antes del comienzo del próximo incremento.

1.1.2.1.4 Modelo Espiral

El modelo espiral es un modelo del ciclo de meta-vida. En este modelo, el esfuerzo de desarrollo es iterativo. El modelo espiral ha sido creado para cubrir las mejores características tanto del ciclo de vida clásico, como de la creación de prototipos, añadiendo al mismo tiempo un nuevo elemento: el análisis de riesgo.

El modelo espiral define cuatro actividades principales:

I. Planificación: determinación de objetivos, alternativas y restricciones.
II. Análisis de riesgo: análisis de alternativas e identificación/resolución de riesgos.
III. Ingeniería: desarrollo del producto del “siguiente nivel “.
IV. Evaluación del cliente valorización de los resultados de la ingeniería.

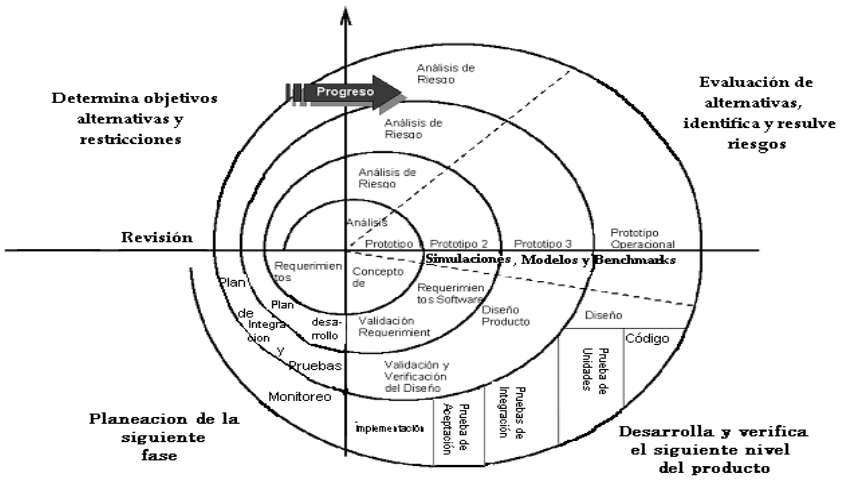

Figura 2. Modelo de Ciclo de Vida de Espiral

Con cada interacción alrededor de la espiral (comenzando en el centro y siguiendo hacia el exterior), se construyen sucesivas versiones del software, cada vez más complejas y, al final, al propio sistema operacional.

1.1.2.1.5 Modelo de Prototipo

El modelo de prototipo se utiliza cuando los requerimientos del sistema no son muy claros, o no se identifican en forma detallada, los requerimientos de entrada, salida y funciones.

El modelo de prototipo puede tomar algunas de las siguientes formas:

- Una demostración (porciones de código que realizan algunas funciones).
- Un escenario (simulación del uso del sistema).
- Una versión o (aplicación liberada que puede usarse bajo condiciones preliminares añadiendo, cambiando o quitando funciones existentes y creándole su documentación).

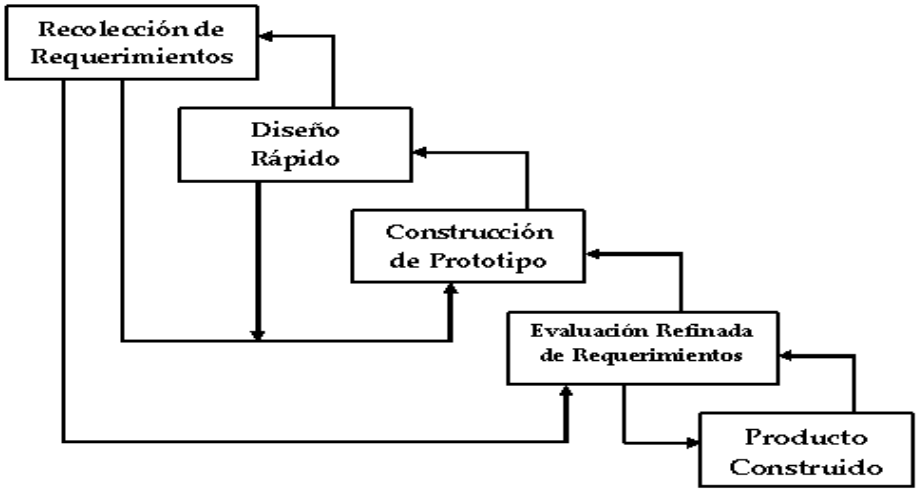

Figura 3. Modelo del Ciclo de Vida de Prototipo.

1.2 BASE DE DATOS

Base de Datos es un conjunto exhaustivo, no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquinas accesibles en tiempo real, y compatibles con usuarios concurrentes con necesidad de información diferente y no predicable en tiempo. Las Bases de datos surgen a mediados de los años setenta, Codd propuso el modelo relacional, este modelo es el que ha marcado la línea de investigación.

Un sistema de base de datos debe de tener implementados componentes de seguridad que avalen la integridad de la información, a pesar de caídas del sistema o intentos de accesos no autorizados. Un objetivo principal de un sistema de base de datos es proporcionar a los usuarios finales una visión de los datos, esto se logra escondiendo ciertos detalles de cómo se almacena y mantienen los datos.

1.2.1 Ventajas de las Bases de Datos

- Independencia de datos y tratamientos: cambio en datos no implica cambio en programas y viceversa esto significa menor costo de mantenimiento.
- Coherencia de resultados: reduce redundancia (acciones lógicamente únicas, se evita inconsistencia).
- Mejora en la disponibilidad de datos: no hay dueño de datos (no igual a ser públicos, ni aplicaciones ni usuarios), guardamos descripciones (idea de catálogos).
- Cumplimiento de ciertas normas: restricciones de seguridad (acceso a usuarios a datos, operaciones sobre datos).
- Más eficiente gestión de almacenamiento.

1.2.2 La arquitectura de una Base de Datos.

Los SBD pueden ser estudiados desde 3 niveles distintos:

1. Nivel Físico.

   Es el nivel real de los datos almacenados. Es decir cómo se almacena los datos, ya sea un registro. Este nivel es usado por muy pocas personas que deben estar cualificadas para ello. Este nivel lleva una representación de datos, que es lo que denominamos esquema físico.

2. Nivel Conceptual.

Es el correspondiente a una visión de base de datos desde el punto de vista del mundo real. Es decir se trata con la entidad u objetos representados, sin importarnos como está representado o almacenados. Este nivel lleva asociado el Esquema Conceptual.

3. Nivel Visión.

   Son parte del esquema conceptual. El nivel conceptual presenta toda la base de datos, mientras que los usuarios por lo general solo tienen acceso a pequeñas parcelas de esta. El nivel visión es el encargado de dividir estas parcelas.

1.2.3. Tipos de Bases de Datos.

Actualmente las Bases de datos ocupan el recurso de Internet para información bibliográfica. Una ventaja de este recurso radica en la máxima difusión que pueden adquirir a través de la red, en conseguir un fácil acceso a la dirección a través de la web, de centros académicos o de investigación esto produce una consulta o búsqueda más agradable para el usuario. Sin embargo hay diferentes tipos de bases de datos que son:

- MySql: es una base de datos con licencia GPL basada en un servidor. Se caracteriza por su rapidez. No es recomendable usar para grandes volúmenes de datos.

- Access: es una base de datos desarrollada por Microsoft. Esta base de datos, debe ser bajo el programa Access, el cual crea un archivo .mdb con la estructura ya explicada.

- PostgreSQL y Oracle: son sistemas de base de datos poderosos. Administra muy bien grandes cantidades de datos y suelen ser utilizados en intranets y sistemas de gran calibre.

- Microsoft SQL Server: es una base de datos más potente que Access desarrollada por Microsoft. Se utiliza para manejar grandes volúmenes de información.

1.2.4 Sistemas de Administración de la Base de Datos (DBMS)

Es un conjunto de programas que maneja todo el acceso a la Base de Datos es decir:

1. Un usuario solicita acceso empleando algún sublenguaje de datos.
2. El DBMS interpreta la solicitud y la analiza.
3. El DBMS inspecciona en orden el esquema externo de ese usuario, la correspondencia externa conceptual asociada, el esquema conceptual la correspondencia conceptual interna y la estructura de almacenamiento.
4. El DBMS ejecuta las operaciones necesarias sobre la base de datos almacenados.

1.2.4.1 Funciones Especificas del DBMS

Un DBMS debe incluir por lo menos las siguientes funciones:

1. **Definición de Datos.-** Debe ser capaz de aceptar definiciones de datos (esquema externo, conceptual, interno y todas las correspondencias asociadas en versión fuerte y convertirla en una versión de objeto apropiada). Debe incluir componentes de procesadores de lenguajes para cada uno de los diversos lenguajes de definición de datos, además debe

tener las definiciones en DBL (Lenguaje de Definición de Datos) para poder interpretar y resolver las solicitudes.

2. **Manipulación de datos**.- Se manipulan las peticiones del usuario para recuperar, actualizar los datos existentes en la base de datos, o agregar nuevos datos a la base de datos, esto es que el DBMS debe incluir un lenguaje de manipulación de datos (DML), esto es que las peticiones pueden ser planeadas o no planeadas.
3. **Seguridad e Integridad de Datos.-** El DBMS debe de supervisar las solicitudes de los usuarios y rechazar los intentos de violar las necesidades de seguridad e integridad de los datos definidos para el DBA.

4. **Recuperación y Concurrencia de los Datos.**-El DBMS debe cuidar el cumplimiento de ciertos controles de recuperación y concurrencia. El administrador de transacciones actúa en caso de que el DBMS no funcione.

5. **Diccionario de datos**.- El diccionario contiene datos sobre los datos. En particular, todos los esquemas fuentes y objetos estarán almacenados en el diccionario.

6. **Desempeño**.- Puede decirse en síntesis que la función global del DBMS es proporcionar una interfaz de usuario al sistema de base de datos.

7. **Administrador de comunicaciones de datos.-** Las solicitudes de un usuario final son dirigidas a la Base de Datos donde son transmitidas en forma de mensajes de comunicación o a la inversa, las respuestas al usuario son tomados como mensajes del mismo tipo, todas las transmisiones se efectúan bajo el control de otros grupos de programas llamados administrador de comunicación de datos, el cual debe trabajar en forma

conjunta con el DBMS para poder satisfacer los requerimientos de los usuarios.

1.2.5 Bases de Datos Relaciónales

Una base de datos relacional se define como una base de datos en donde todos los datos evidentes al usuario están organizados como tablas de valores, y todas las operaciones de la base de datos operan las tablas. Las tablas se organizan agrupando datos relacionados con el mismo tema y contienen columnas o atributos y filas o tuplas de información. Después cuando las bases de datos son solicitadas, el motor de las bases de datos las relaciona entre sí.

Las Bases de Datos relacionales son vistas por los usuarios como una colección de relaciones normalizadas de diversos grados que varían con el tiempo.

1.2.5.1 Enfoque Relacional

El modelo relacional es la forma de ver los datos, es decir una receta para representar los datos mediante tablas y para manipular su representación. El modelo relacional se ocupa de tres puntos importantes de los datos:

1. Estructura.
2. Integridad
3. Manipulación.

1.2.5.1.1 Estructura del Enfoque Relacional

Los componentes de la estructura son básicamente: Relación, Tupla, Cardinalidad, Atributo, Grado, Dominio y Clave Primaria.

Una **relación** corresponde a lo que se conoce como tabla o entidad; una **tupla** corresponde a una fila de esta tabla, un **atributo** corresponde a una columna de la misma tupla, el número de las tuplas se denomina **cardinalidad,** el número de los atributos se llama **grado,** la **clave primaria** es un identificador único para la tabla es decir, una columna o combinación de una columna con la siguiente propiedad, nunca existen dos filas de la tabla con el mismo valor de la columna o combinación de una columna. El **Dominio** es una combinación de valores de los cuales uno o más atributos obtienen sus valores reales.

La estructura de un enfoque relacional es en si donde las asociaciones entre tuplas se representan únicamente por valores de datos en las columnas sacadas de un dominio común.

Termino Relacional	Termino Informal
Relación	Tupla
Tupla	Fila o Registro
Cardinalidad	No. de Filas
Atributo	Campo o Columna
Grado	No. de Columna
Clave Primaria	Identificador único
Dominio	Valores legales

Figura 4. Terminología de la estructura de datos relacional.

Las relaciones están compuestas por dos partes una cabecera y un cuerpo, se define la relación sobre un conjunto de dominios. La cabecera está formada por un conjunto fijo de atributos. El cuerpo está formado por un conjunto de tuplas el cual varía con el tiempo, cada tupla está formada por un conjunto de pares

(atributo, valor). Por mencionar algunos de los diferentes tipos de relaciones que pueden existir en un sistema relacional tenemos:

- **Relaciones base o reales.-** Corresponde al concepto tabla es decir una relación autónoma cuya importancia está dada por el diseñador para un uso específico dentro de una aplicación.
- **Relaciones virtuales o de vistas.-** Una vista es una relación derivada con nombre representada dentro del sistema exclusivamente mediante su definición en término de otras relaciones, no posee datos almacenados propios, separados y distinguibles a diferencia de las relaciones bases, en si una VISTA.
- **Relaciones instantáneas (SnapShop).-** Es también una relación derivada con nombre como una vista, pero a diferencia de esta última las instantáneas son reales, no virtuales, esto es están representadas no sólo por su definición, en término de otras relaciones con nombre, sino también por sus propios datos almacenados.

Un **campo** es un grupo de caracteres relacionados entre si, que se tratan como una unidad, como sería un elemento de información en un registro.

Un **registro** es un conjunto de elementos relacionados entre si, que se tratan como una unidad. Están formados por campos. Dentro de una base de datos un registro contendrá la información completa de una entidad.

Cada **columna** (campo) de una tabla debe tener asignado un nombre, un tipo de datos, una longitud, una intercalación y un estado de nulidad. Se pueden colocar las columnas en cualquier orden en la definición de tabla. Cada columna debe tener un nombre único dentro de la tabla.

La nulidad de una columna se refiere en si a que se requiere o no de una entrada para esa columna. Para que una columna indique que el valor es desconocido se especifica NULL. Si se desea insistir que cada fila tenga una entrada en esa columna, se especifica NOT NULL.

1.2.5.1.2 Reglas Generales de Integridad

La mayor parte de las Bases de Datos están sujetas a un gran número de reglas de integridad. La primera regla general de integridad se relaciona con las claves primarias y la segunda con las claves ajenas.

Claves Primarias.- Es posible tener una relación con más de un identificador único, se dirá que la relación tiene varias claves candidatas. Las condiciones que debería tener la clave candidata es si el atributo K de la relación R es una clave candidata de R si y sólo si satisface las siguientes dos propiedades independientes (unicidad y minimicidad).

- **Unicidad.-** En cualquier momento no existen dos tuplas en R con el mismo valor de K.
- **Minimicidad.-** Si k es compuesto no será posible eliminar ningún componente de K sin destruir la propiedad de Unicidad.

Toda relación tiene por lo menos una clave candidata, al razonamiento para elegir la clave primaria cuando existen varias claves candidatas, queda del alcance del modelo relacional en la práctica la elección es más sencilla.

Regla de Integridad de Entidades: Ningún componente de la llave primaria de una relación base puede aceptar valores nulos.

Claves Ajenas.- Es un atributo de una relación R2 cuyos valores deben concordar con las llaves primarias de alguna relación R1, es decir un valor de la clave ajena representa una referencia a la tupla donde se encuentra el valor correspondiente de la clave primaria.

Regla de Integridad Referencial: La base de datos no debe contener valores de llave ajena sin concordancia.

1.2.5.1.3 Manipulación del Enfoque Relacional

La manipulación es la tercera y última parte del enfoque relacional, la parte manipulativa se divide a su vez en dos partes:

1. Un conjunto de operadores, como los de reunión, que forman un conjunto lo que se conoce como álgebra relacional.
2. Una operación de asignación que asigna el valor de alguna expresión arbitraria del álgebra a una relación nombrada.

1.2.5.1.3.1 Álgebra Relacional

El álgebra relacional se define como un lenguaje formal con una serie de operadores que trabajan sobre una o varias relaciones para obtener otra relación resultado, sin que cambien las relaciones originales. Tanto los operándos como los resultados son relaciones, por lo que la salida de una operación puede ser la entrada de otra operación.

El álgebra relacional consiste en conjunto de operadores de alto nivel que operan sobre relaciones, cada uno de estos operadores se divide en dos grupos que son:

1. Involucra las operaciones tradicionales de un conjunto tales como:

    a. **Unión.-** Construye una relación formada por todas las tuplas que aparecen en cualquiera de las dos relaciones involucradas.
    b. **Intersección.-** Construye una relación formada por aquellas tuplas que aparezcan en las dos relaciones involucradas.
    c. **Producto Cartesiano.-** A partir de 2 relaciones especificadas construye una relación que contiene todas las posibles combinaciones de tuplas, una de cada una de las relaciones tales que las dos tuplas participantes en una combinación dada, satisfagan alguna condición especifica. Esta es muy parecida a la restricción pero depende de los elementos que intervienen.
    d. **Diferencia.**- Construye una relación formada por todas las tuplas de la primera relación que no aparezcan en la segunda relación.

2. El segundo involucra las operaciones relacionales tales como restricción, proyección, división y reuniones.

    a. **Restricción.-** Extrae las tuplas especificadas en una relación dada, es decir restringe la relación a tuplas que satisfagan una condición.
    b. **Proyección.-** Extrae los atributos especificados de una relación dada.
    c. **Reunión.-** A partir de dos relaciones especificadas construye una relación que contiene todas las posibles combinaciones de tuplas, una de cada una de las relaciones tales que las dos tuplas participantes es una combinación dada satisfagan alguna condición especificada. Esta es muy parecida a la restricción pero depende de los elementos que intervienen.

1.2.6 Normalización de Bases de Datos.

El proceso de normalización consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo **entidad-relación** al **modelo relacional.** Las bases de datos relacionales se normalizan para:

- **Evitar la redundancia de los datos.**
- **Evitar problemas de actualización de los datos en las tablas.**
- **Proteger la integridad de los datos.**

En el modelo relacional es frecuente llamar tablas a una relación, aunque para que una tabla sea considerada como una relación tiene que cumplir con algunas restricciones:

- Cada columna debe tener su nombre único.
- No puede haber dos filas iguales. No se permiten los duplicados.
- Todos los datos en una columna deben ser del mismo tipo.

1.2.6.1 Formas Normales.

Las formas normales son aplicadas a las tablas de una base de datos. Esto quiere decir que una base de datos está en la forma normal N, es decir que todas sus tablas tienen la forma normal N. Existen varias formas normales, pero por lo general la mayoría de los programadores sólo llegan a las tres primeras formas normales.

1.2.6.1.1 Primera Forma Normal (1FN)

Una tabla está en Primera Forma Normal sólo si

- Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son indivisibles, mínimos.
- La tabla contiene una clave primaria.
- La tabla no contiene atributos nulos.
- Si no posee ciclos repetitivos.

Una columna no puede tener múltiples valores, si cada valor de X le pertenece un valor de Y, entonces a cada valor de Y le pertenece un valor de X. De esta forma eliminamos los valores repetitivos dentro de una base de datos.

1.2.6.1.2 Segunda Forma Normal (2FN)

Para tener la 2FN primero debe de estar en 1FN y también si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave primaria. Es decir que no existen dependencias parciales.

La Segunda Forma Normal está basada en el concepto de dependencia completa funcional. Una dependencia funcional $x \longrightarrow y$ es completamente funcional si al eliminar los atributos A de X significa que la dependencia no es mantenida, esto es que A Є X, (X – {A}) -x-> Y. Una dependencia funcional $x \longrightarrow y$ es una dependencia parcial si hay algunos atributos $A \in X$que pueden ser removidos de X y la dependencia todavía se mantiene.

1.2.6.1.3 Tercera Forma Normal (3FN)

No se puede realizar si todas las condiciones de la 2FN no son satisfechas y consiste en eliminar aquellos datos que no sean claves y que puedan derivarse de una combinación de otros datos y tampoco son claves en ninguna relación, esto es que no debe existir transitividad entre sus atributos.

1.2.6.1.4 Forma Normal de Boyce-Codd (FNBC)

La tabla se encuentra en BCNF si cada determinante, atributo que determina completamente a otro, es clave candidata.

1.2.6.1.5 Cuarta Forma Normal (4FN)

Una tabla se encuentra en 4FN si y sólo si, para cada una de sus dependencias múltiples no funcionales X->->Y, siendo X una súper-clave que, X es o una clave candidata o un conjunto de claves primarias.

1.2.6.1.6 Quinta Forma Normal (5FN)

Una tabla se encuentra en 5FN si:

- La tabla está en 4FN
- No existen relaciones de dependencias no triviales que no siguen los criterios de las claves. Una tabla que se encuentra en la 4FN se dice que está en la 5FN si, y sólo si, cada relación de dependencia se encuentra definida por las claves candidatas.

1.3 HERRAMIENTAS DE DESARROLLO

Las herramientas de desarrollo surgieron para intentar dar solución a los problemas inherentes a los proyectos de generación de aplicación informática: plazos y presupuestos incumplidos, insatisfacciones del usuario, baja calidad de desarrollos.

El principal objetivo del las herramientas de desarrollo son las siguientes:

- Ayuda de la herramienta: Es una ayuda incorporada al programa, brindando información sobre el uso de los componentes de la propia herramienta, fácil acceso y como utilidades de búsqueda de temas o palabras claves.
- Diccionario de datos: Es la descripción lógica para el usuario. También reúnen los datos almacenados en una base de datos como descripción, significado, estructura consideración de seguridad.
- Ingeniería de Software. Es el tratamiento sistemático de todas las fases del ciclo de vida del software, abordando el desarrollo del sistema de información.
- Ingeniería Directa. Es el proceso de producción del código de una aplicación a partir de sus especificaciones.
- Ingeniería inversa: Conjunto de tareas destinadas a obtener las especificaciones de un sistema de información. Es una actividad típica del mantenimiento de aplicaciones, cuando no existen las especificaciones de diseño de la aplicación a mantener.
- Metodología de planificación y desarrollo de aplicaciones. Es el conjunto de métodos que basados en unos principios, se integran en el marco del ciclo de vida de los sistemas. La metodología debe recoger las tareas a realizar, los responsables de cada una de ellas y los productos a obtener en el desarrollo de un sistema de información. También puede incluir o hacer referencia a las técnicas a emplear en cada momento.
- Reingeniería de Sistemas. Es la modificación de los componentes de una aplicación, sin cambiar sus funcionalidades, por ejemplo: la mejora de la codificación de un programa. A veces también se emplea este término para referirse conjuntamente a la ingeniería directa e inversa.

1.3.1 Herramientas CASE

Es un conjunto de métodos, utilidades y técnicas que facilitan la automatización de ciclos de vida del desarrollo de sistemas de información, completamente o en algunas de sus fases. Este puede ser generalmente aplicado a cualquier sistema o colección de herramientas, que ayuda a automatizar el proceso de diseño y desarrollo de software. Compiladores, editores estructurados, sistemas de control de código fuente, y herramientas de modelado son todas, estrictamente hablando herramientas CASE. Ellas impiden a los programadores tratar tan directamente con el hardware y les permiten trabajar en un alto nivel de abstracción en la definición de un sistema de software que entonces será construido

Algunos de los componentes de las herramientas CASE permiten:

- Confeccionar la definición de requerimientos de los usuarios.
- Mejorar el diseño de los sistemas.
- Mejorar la eficiencia en la programación (por su generación automática de códigos).
- Otorgar a la administración un mejor soporte en la documentación.

Para ello, y sin importar la arquitectura de la herramienta CASE, en general tales herramientas deben abarcar las siguientes propiedades:

Tener una interfaz gráfica y textual, que le permita al usuario manejar los objetos de diseño.

- Contar con un Diccionario de Datos, a fin de rastrear y controlar los objetos diseñados.
- Disponer de un conjunto de herramientas que permitan: validar las reglas del diseño y analizar la lógica del diseño.

Existen diferentes tipos de herramientas case, eso habitualmente se clasifican basadas en las fases de desarrollo que cubren.

- *Upper CASE*: Herramientas que ayudan en las fases de planificación, análisis de requisitos y estrategia del desarrollo, usando diagramas UML.
- *Middle CASE*: Herramientas para automatizar tareas en el análisis y diseño de la aplicación.
- *Lower CASE*: herramientas que semiautomatizan la generación de código, crean programas de detección de errores, soportan la depuración de programas y pruebas. Además automatizan la documentación completa de la aplicación.

Por funcionalidad podríamos diferenciar algunas como:

- Herramientas de generación semiautomática de código.
- Editores UML.
- Herramientas de Refactorización de código.
- Herramientas de mantenimiento como los sistemas de control de versiones.

CAPÍTULO 2
ANÁLISIS
Y
DISEÑO DEL
SISTEMA

2.1. ANALISIS Y PLANTEAMIENTO DEL PROBLEMA

Al llevar un control de asignación de proyectos manualmente nos encontramos con algunos problemas, como es no tener un control eficiente y actualizado de la información que se maneja de los alumnos, de los profesores y de los proyectos que se están desarrollando y han sido registrados. Recuperar los datos de manera clara y precisa se vuelve un serio problema.

En el presente trabajo se desarrolla un sistema que permite automatizar la información asociada a los proyectos que se encuentran registrados por parte de la facultad, datos de los alumnos y profesores que participan en los mismos, se desea conocer cuantos proyectos tiene a su cargo cada profesor, una descripción de cada proyecto, periodo de validez, entre otros datos.

Esta base de datos consta de cuatro tablas importantes que son las siguientes: la de alumnos, ésta tendrá como atributos la matricula, nombre del alumno, y cuatrimestre en que se encuentra el mismo. La tabla de profesores tiene como atributos la clave del profesor, nombre del profesor, especialidad y antigüedad en la facultad, la tabla de proyectos tiene como atributo clave del proyecto, título del proyecto, descripción del proyecto y duración del mismo. Por último tenemos la tabla asociar que almacena la clave del profesor, clave del proyecto y fecha de asignación de éste.

Para trabajar en la base de datos se tiene que realizar algunas operaciones que permiten hacer cambios como lo son: las altas, bajas, modificación y consultas. En la operación de Altas permitirá agregar alumnos, profesores, proyectos y registrar en qué proyecto trabaja cada profesor, como a su vez también se podrá hacer modificación de algunos datos de la tabla, que se requieran actualizar. También se podrá eliminar de la tabla los datos que uno quiera. El sistema permite que se realicen consultas y se puedan extraer las tuplas de la tabla elegida.

2.2 Análisis de Factibilidad

Para el diseño, desarrollo, pruebas y documentación se requiere del siguiente equipo:

Procesador Pentium IV
Disco duro de 40gb
512Mb en RAM
Monitor de 17¨

El software que se determinó emplear para el desarrollar el sistema es:

Sistema Operativo Windows
Mysql y php
Aplicación WEB

Una vez analizado y definido correctamente los requerimientos se ha propuesto ocupar el modelo de Cascada para el desarrollo del sistema, los datos serán almacenados en una base de datos relacional.

2.3 Desarrollo del Diagrama de Flujo de Datos

El diagrama de flujo de datos es utilizado como una forma de presentar gráficamente los detalles algorítmicos de un proceso multifactorial. El DFD nos muestra la interacción entre el sistema y la entidad externa.

Al tener todos los requerimientos del sistema y analizado el sistema el segundo paso es elaborar un Diagrama de Flujo de Datos que ayudará a plasmar correctamente los requisitos, para que sean aprovechados con eficacia todos los recursos y se realice un procedimiento de datos correcto y legible.

A continuación se presenta el bosquejo del Diagrama de Flujo de datos comenzando por la parte del Alumno y muestra el flujo de los datos.

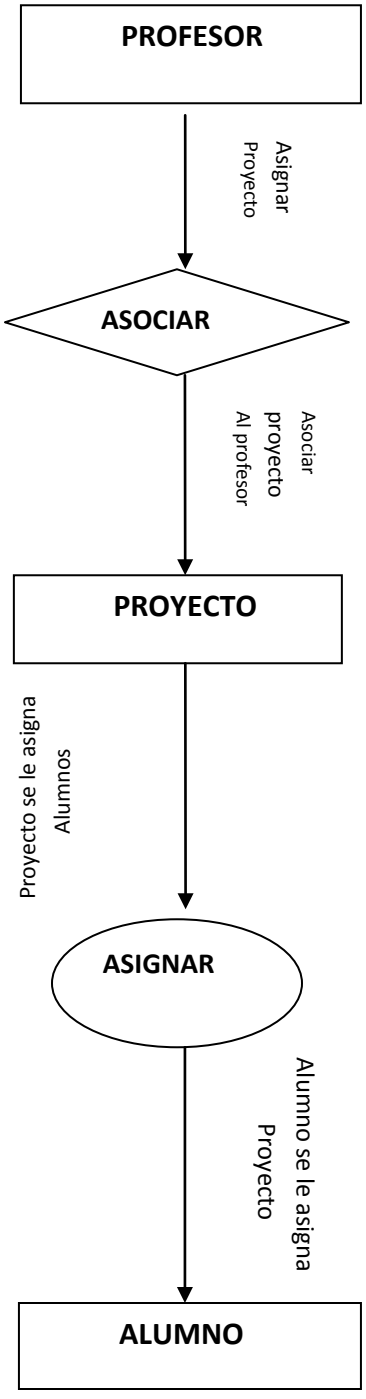

Fig. 4 DFD Parcial

A continuación se muestra un segundo DFD, el cual se propone algo más claro, ya que va controlando las entradas y salidas, especificando todas las funciones o procesos que se elaboran en el sistema más ampliamente y nos ayude a la correcta elaboración.

En este DFD se presentan las entradas y las salidas en donde se definen los procesos como, verificar si existe el profesor, el proyecto y el alumno.

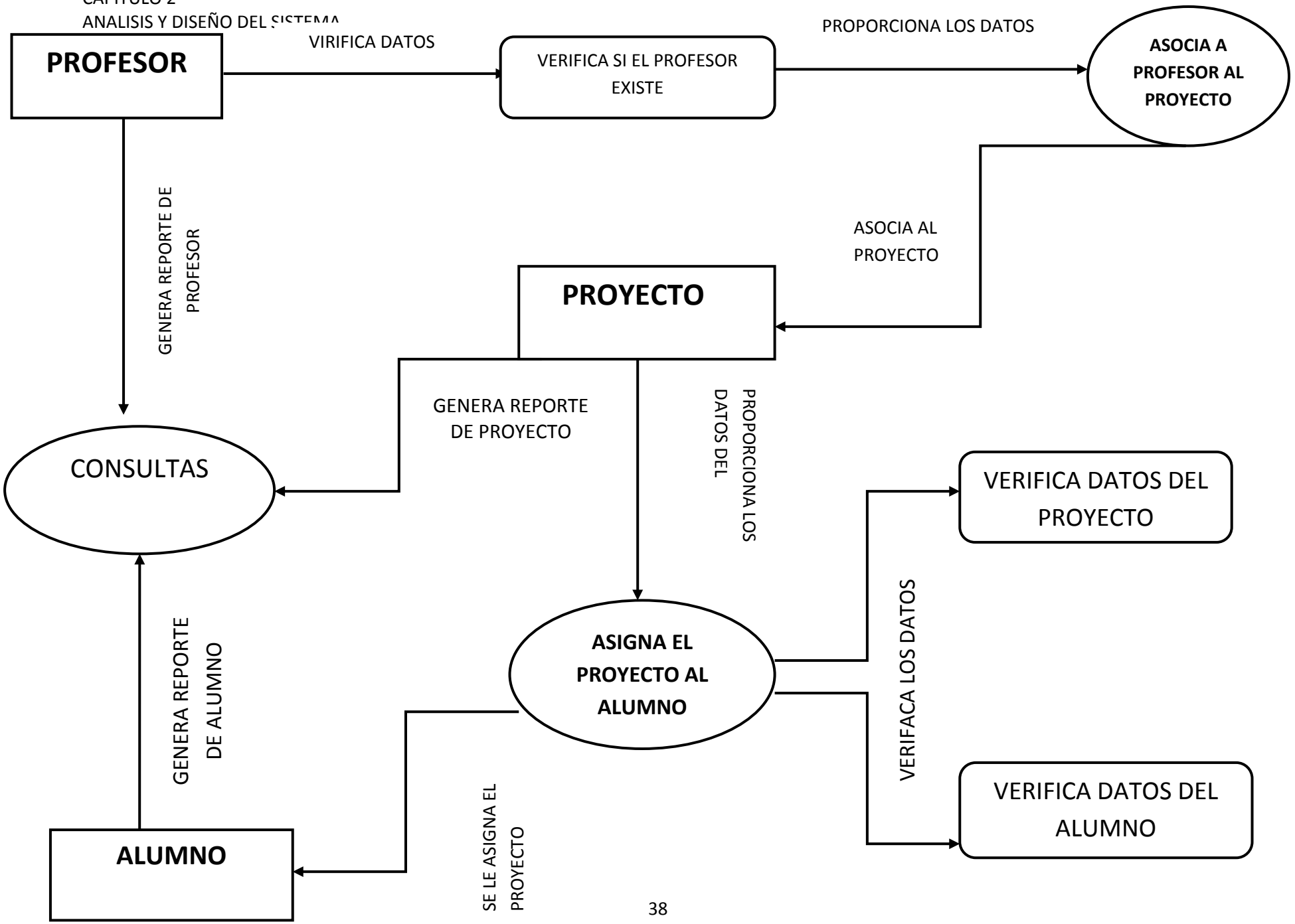

Figura 5. DFD Final

2.4 Diccionario de Datos

El diccionario de datos es el segundo componente del análisis del flujo de datos, el cual proporciona información adicional sobre el sistema. Cada entrada en el diccionario de datos consiste en un conjunto de detalles que describen los datos utilizados o producidos en el sistema. Además de describir qué es lo que hace cada función y de qué se necesita para llevar a cabo la misma.

Función:	**Verificar datos del Profesor**
Descripción:	Verifica si existe el profesor
Entradas:	Clave del profesor
Fuente (Origen):	Profesor
Salidas:	Profesor correcto
Destinación:	Asocia
Requerimientos:	Estar dado de alta en la tabla de profesor

Función:	**Verificar Datos Alumno/Proyecto**
Descripción:	Comprueba que los datos del alumno/proyecto sean correctos o esté en el listado
Entradas:	Matrícula/ Claveproy
Fuente (Origen):	Alumno, Proyecto
Salidas:	Datos del alumno/proyecto correctos
Destinación:	Asignar

Requerimientos:	Estar dado de alta en la Tabla de Alumnos o Proyecto

Función:	**Registrar la asociación del proyecto**
Descripción:	Guarda los datos del profesor y proyecto que han sido asignados.
Entradas:	Clave y Claveproy
Fuente (Origen):	Profesor y Proyecto
Salidas:	Asignar proyecto al profesor
Destinación:	Registrar la asignatura de proyecto
Requerimientos:	Ninguna

DISEÑO DEL SISTEMA

2.5 Diagrama Funcional

El diagrama Funcional o Modelo funcional es un bosquejo que representa un conjunto real con cierto grado de precisión y en la forma más completa posible, proporciona una descripción concisa de la organización entera y se divide típicamente entre grandes funciones, y entonces se revisa cada una de estas grandes funciones para pasar a una descripción más detallada que cubra el alcance de la organización

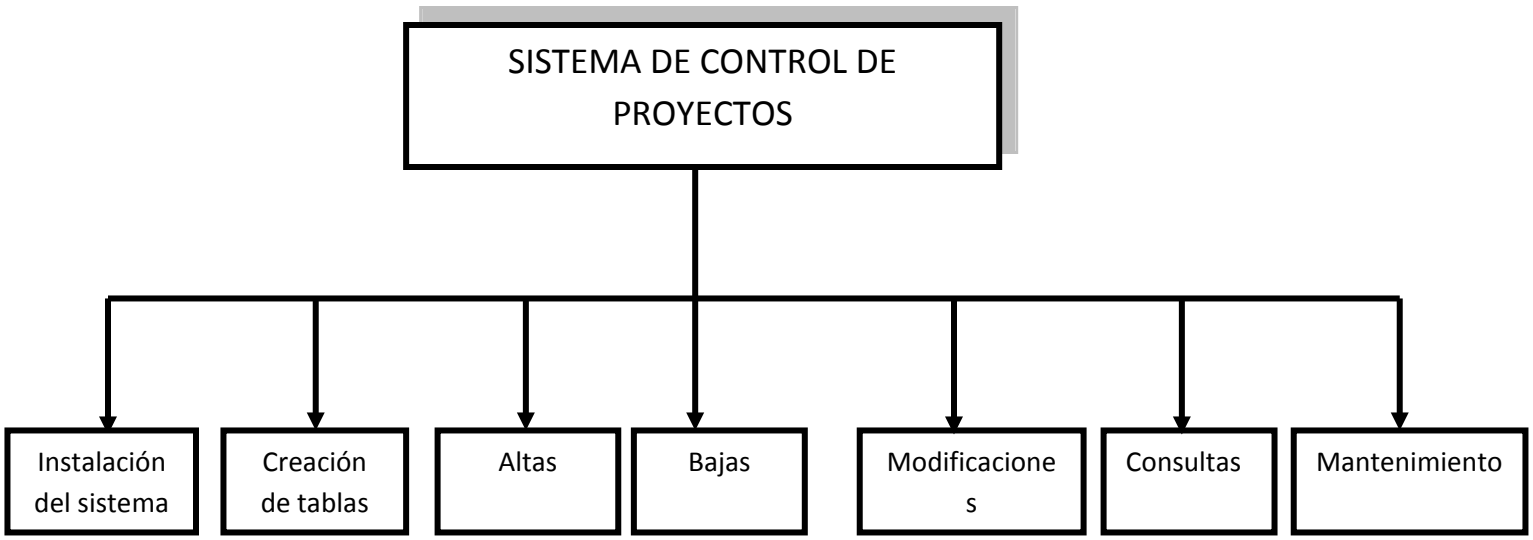

Figura 6. Modelo Funcional del Sistema de Control de Proyectos

En las siguientes figuras se presenta una descomposición funcional de elementos para el sistema del Control de Proyectos.

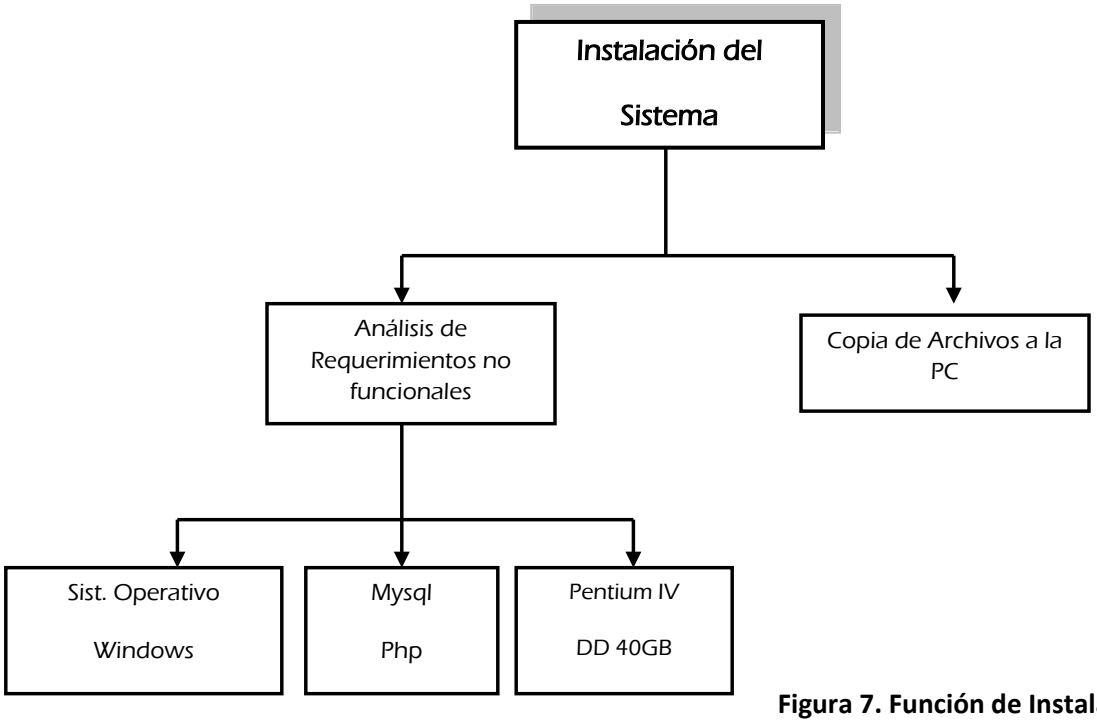

Figura 7. Función de Instalación del Sistema

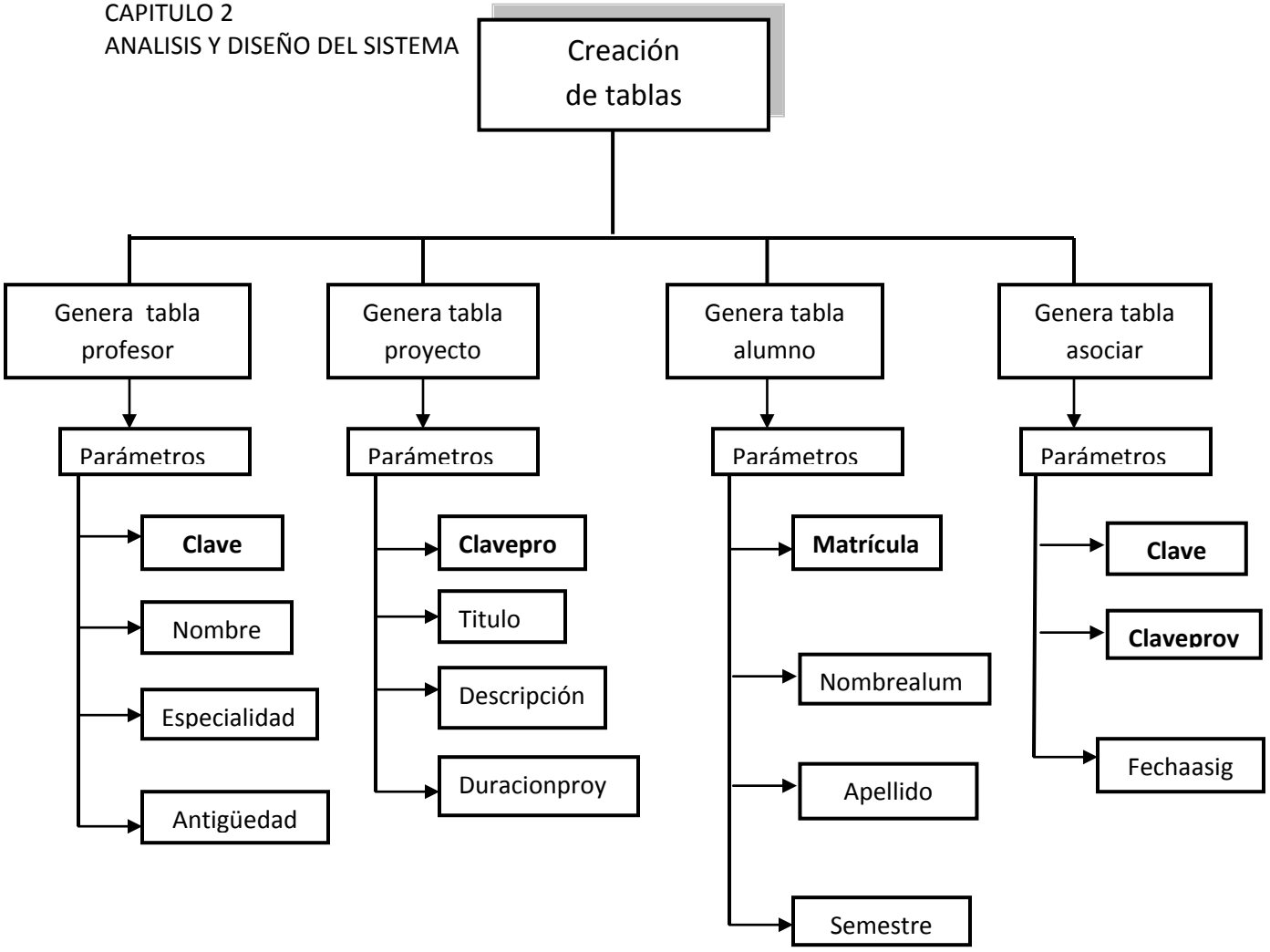

Figura 8. Creación de tablas para el sistema

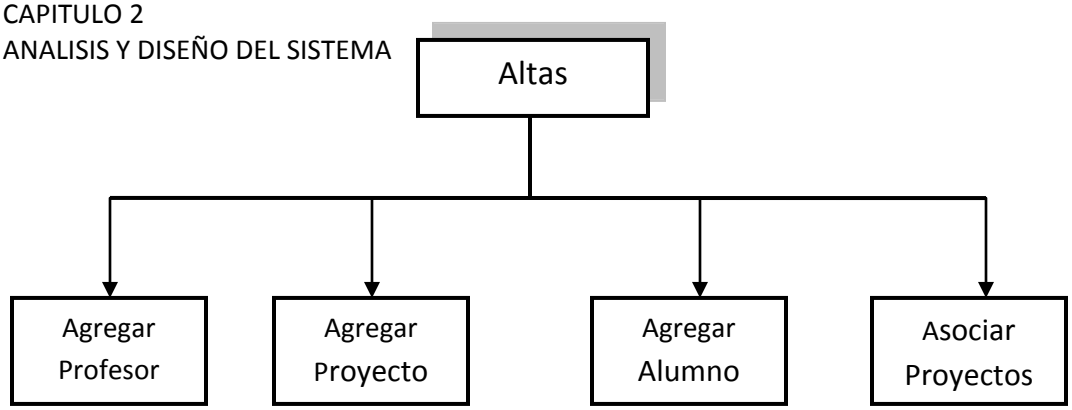

Figura 9. Función de Altas del Sistema

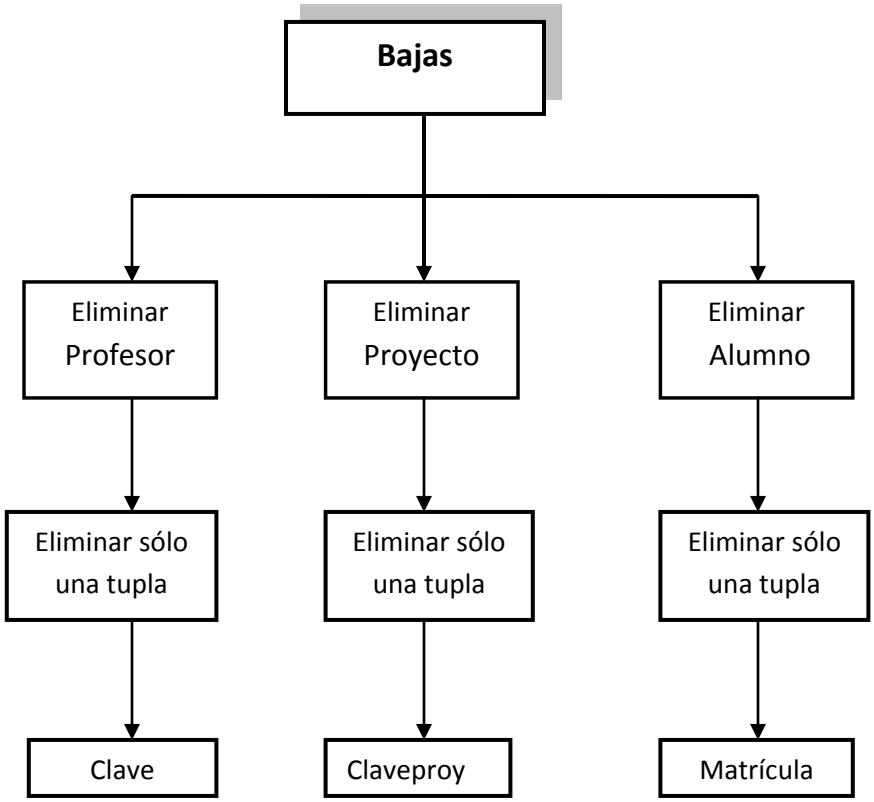

Figura 10. Función de Bajas del Sistema

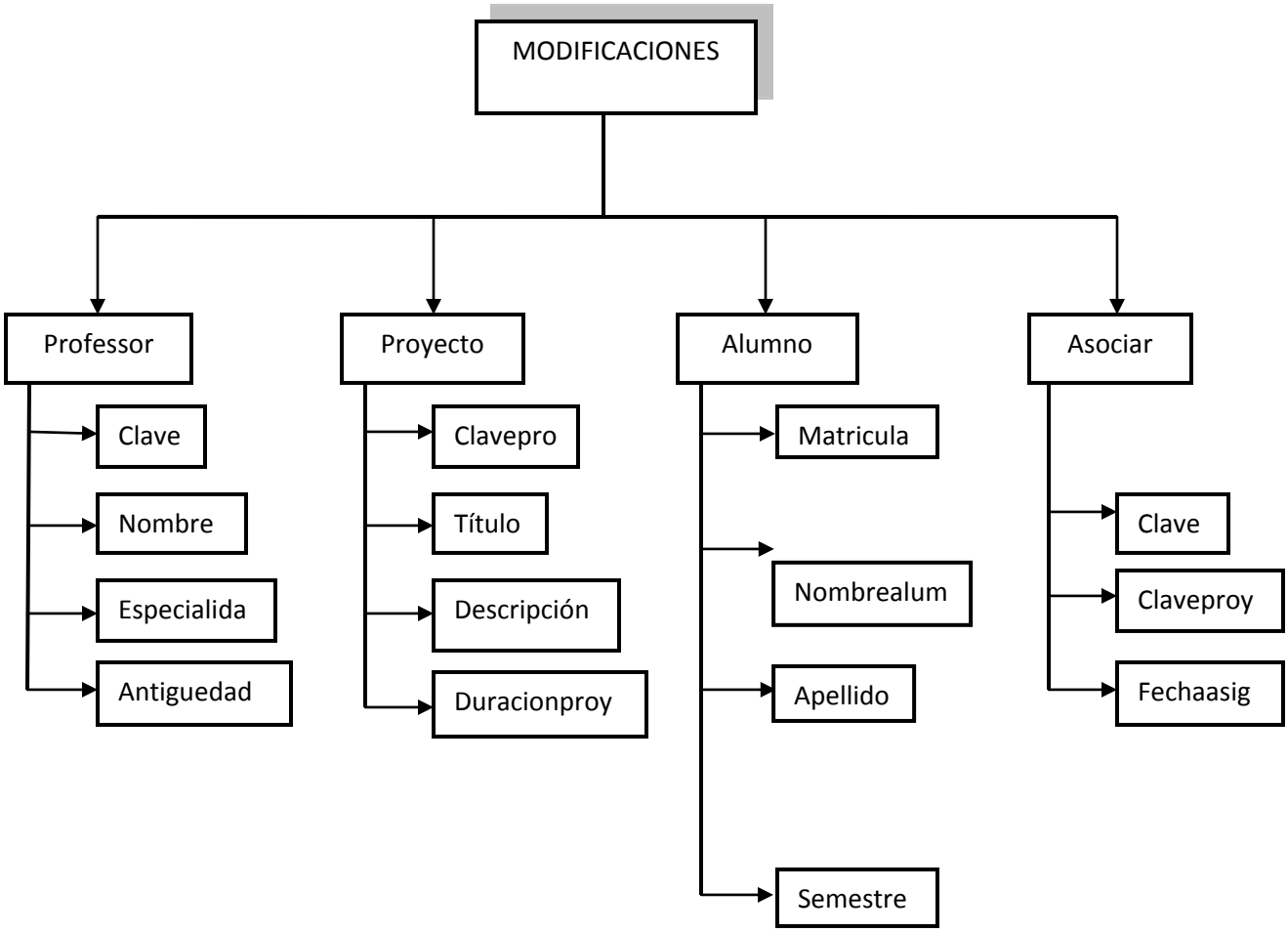

Figura 11. Función de Modificación del Sistema

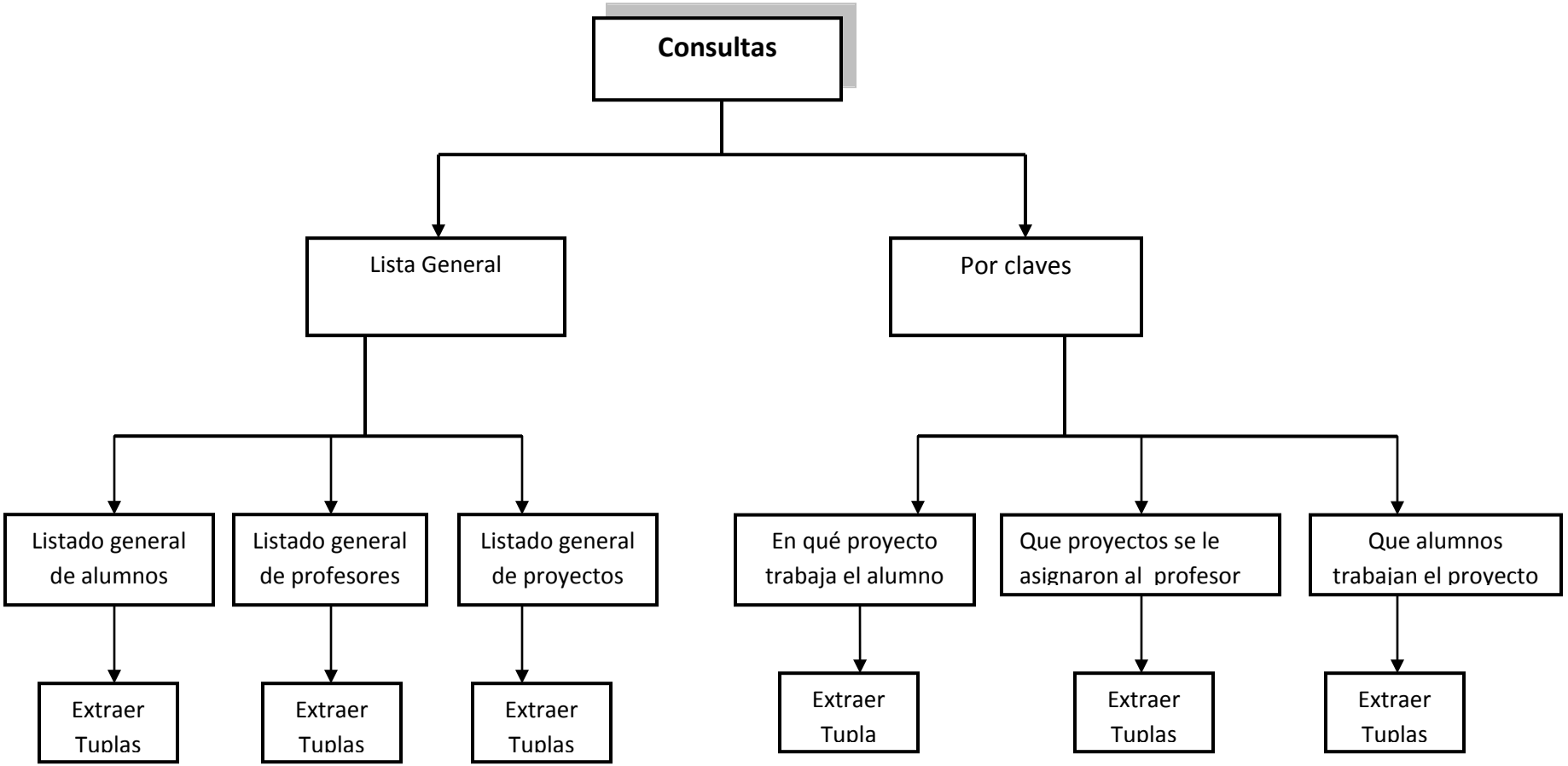

Figura 12. Función de Consultas del Sistema

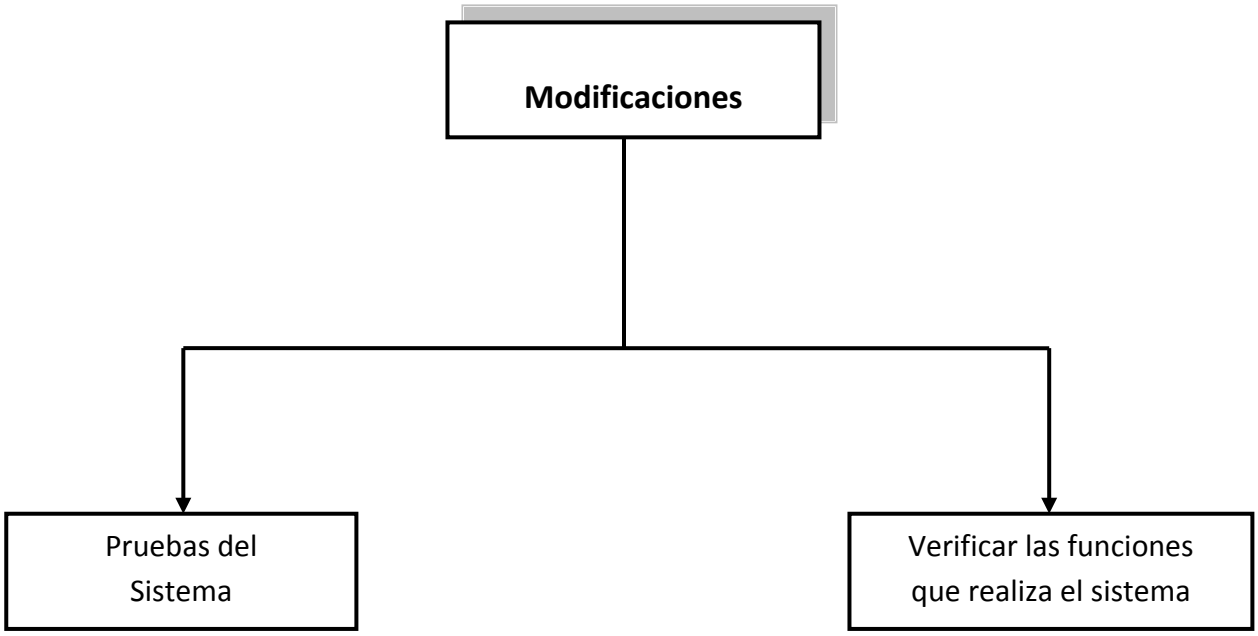

Figura 13. Función de Mantenimiento del Sistema

2.5.1 Narrativa de las funciones

En el sistema de control de proyectos se encuentra representado por funciones que se presentan a continuación:

1. Creación de Tablas

En este proceso el Administrador se generan las tablas de Profesor, Proyecto y Alumnos con los parámetros o atributos que se asignaron como se muestra a continuación.

- Profesor
  a. Clave
  b. Nombre
  c. Especialidad
  d. Antigüedad

- Proyectos
    a. Claveproy
    b. Título
    c. Descripción
    d. Duración de proyecto

- Alumno
    a. Matrícula
    b. Nombrealum
    c. Apellido
    d. Semestre

- Asociar
    a. Clave (clave del maestro)
    b. Claveproy (clave del proyecto)
    c. Fecha de duración

2. Altas

En este proceso el sistema se encarga de agregar datos de Profesores, Proyecto y de Alumnos.

3. Bajas

En el proceso de Bajas el sistema permite eliminar lo siguiente:

- Eliminar Tuplas de Profesores
    a. Eliminar un conjunto de Tuplas

- Eliminar Tuplas de Proyectos

a. Eliminar un conjunto de tuplas

- Eliminar Tuplas de Alumnos

a. Eliminar un conjunto de tuplas

4. Modificaciones

La operación de Modificación se puede hacer de la siguiente manera:

- Modificaciones de las Tuplas de la Tabla de Alumno, mediante la Matrícula que es la llave primaria.
- Modificaciones de las Tuplas, de la Tabla de Proyecto mediante Claveproy que es la llave primaria.
- Modificaciones de las Tuplas de la Tabla de Profesores mediante Clave que es la llave primaria.

5. Consultas

En este proceso se permite hacer consultas sobre las tablas de Profesores, Proyectos y Alumnos, extrayendo las tuplas que se seleccionen, así como también se hacen consultas de una combinación de ambas tablas dependiendo de lo que se desee saber de las tablas, estas consultas se hacen mediante las llaves primarias, por ejemplo si se busca por alumnos se introduce la matricula de él, si es por proyecto se introduce la clave del proyecto y si es por profesor es la clave del profesor.

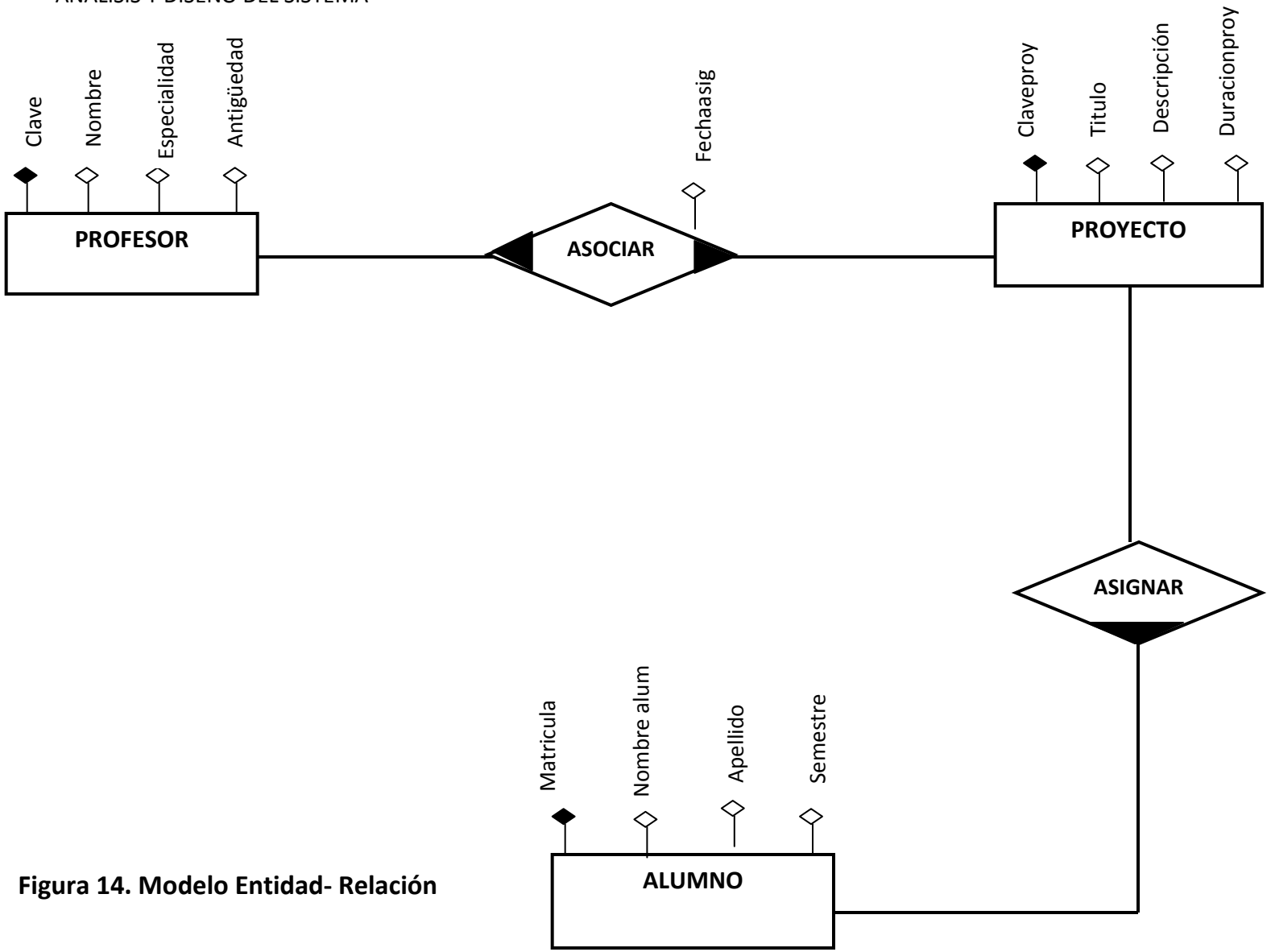

Figura 14. Modelo Entidad- Relación

2.6 Modelo Relacional

Tomando como base el Modelo Entidad-Relación discutido anteriormente, se definen las entidades con sus atributos y como quedarían cada una de las tablas:

Tabla Profesor

Profesor (**Clave**, Nombre, Especialidad, Antigüedad)

Por lo tanto: Llave primaria: Clave

Los atributos de la tabla Profesor son: Nombre, Especialidad, Antigüedad.

El grado de la relación es de 4.

Tabla Asociar

Asociar (**Clave, Claveproy**, Fechaasig)

Por lo tanto: Llave primaria foránea: Clave + Claveproy

El atributo de la Tabla Asociar es: Fechaasig.

El grado de la tabla de asociar es de 3.

Tabla Proyecto

Proyecto (**Claveproy,** Titulo, Descripción, Duracionproy)

Por lo tanto: Llave Primaria: Claveproy

Los atributos de la tabla de Ejemplares son: Estado.

El grado de la relación es de 4.

Tabla Alumno

Alumno (**Matricula**, Claveproy, Nombrealum, Apellido, Semestre)

Por lo tanto: Llave primaria: Matricula

Llave Foránea: Claveproy

Los atributos de la tabla de Alumno es: Nombrealum, Apellido, Semestre.

El grado de la tabla de Alumnos es de 5.

2.6.1 Normalización

Después de la elaboración del modelo relacional se presenta lo siguiente:

Profesor (**Clave**, Nombre, Especialidad, Antigüedad)

Asociar (**Clave**, **Claveproy**, fechaasig)

Proyecto (**Claveproy**, Titulo, Descripción, Duracionproy)

Alumnos (**Matricula**, **Claveproy**, Nombrealum, Apellido, Semestre)

A continuación se analizarán cada una de las tablas para ver si se requiere necesariamente de la normalización.

- **La tabla Profesor:**

  Profesor (**Clave**, Nombre, Especialidad, Antigüedad)

Se observa que esta tabla está en Primera Forma Normal, debido a que cuenta con atributo atómico, la tabla cuenta con clave primaria la cual es Clave y no contiene atributos nulos, está en Segunda Forma Normal ya que no cuenta con

dependencias parciales y tercera forma normal debido a que cumple con la primera, segunda forma normal y no existe transitividad entre sus atributos.

- **La tabla Proyecto:**

  Proyecto (**Claveproy**, Titulo, Descripción, Duracionproy)

  Esta tabla se encuentra en primera, segunda y tercera forma normal.

- **La tabla Alumno:**

  Alumnos (**Matricula**, **Claveproy**, Nombrealum, Apellido, Semestre)

- **La tabla Asociar:**

  Asociar (**Clave, Claveproy,** , Fechaasig)

Como puede apreciarse ambas tablas se encuentran en primera, segunda y tercera forma normal.

CAPÍTULO 3
IMPLEMENTACIÓN DEL SISTEMA

3.1 Implementación de la Base de Datos

La implementación de la base de datos se realizó en MySQL, en particular se empleó PHPMyadmin, la base de datos se denomina, controlproyectos, como se muestra en la fig. 15.

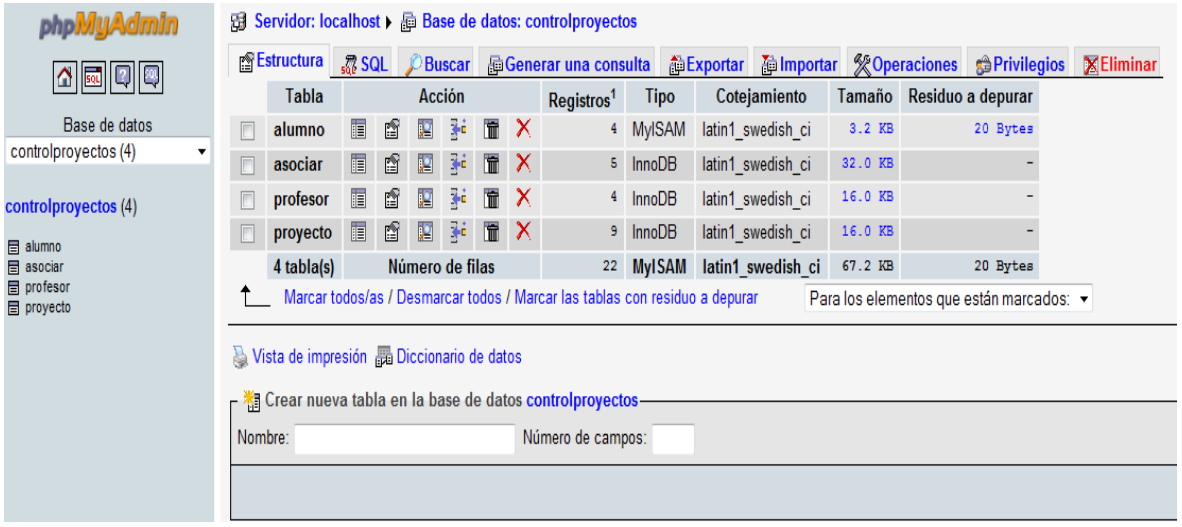

Figura 15. Crear la Base de Dato Controlproyectos

En la figura 16 se muestra la tabla Alumno, la cual presenta sus campos correspondientes, donde el campo matrícula es la llave primaria.

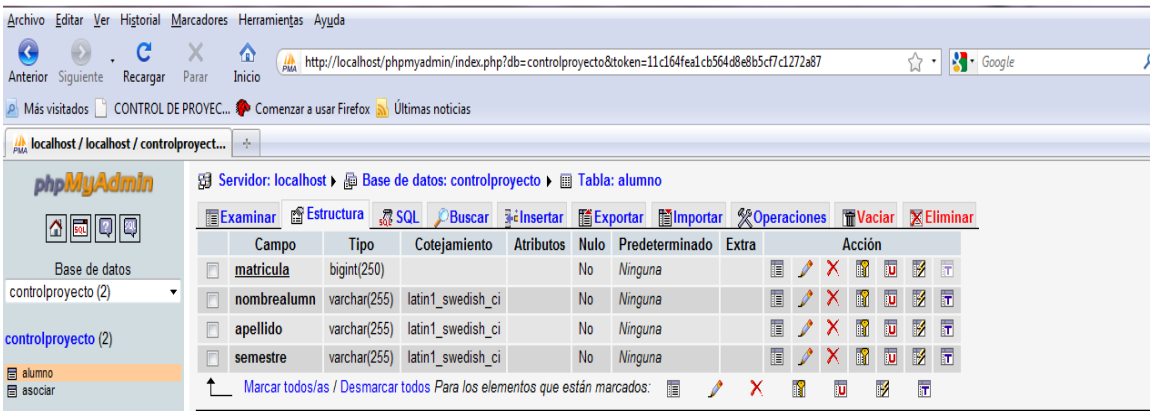

Figura 16. Tabla Alumno

En la siguiente figura 17 se muestra la tabla asociar, la llave primaria es la compuesta entre la clave del profesor y la clave del proyecto.

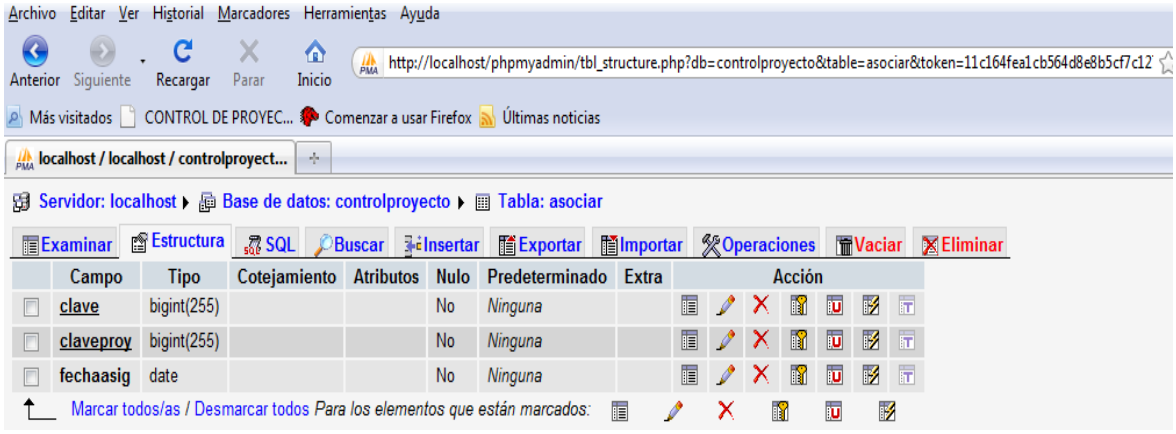

Figura 17. Tabla Asociar

En la tabla proyecto. La clave primaria es claveproy (la clave del proyecto).

De la misma manera se desarrolló la tabla proyecto y alumno.

3.2 Implementación de las Interfaces

El sistema consta de una ventana principal, en la cual se presenta un menú a través de un link, que permite el acceso a las distintas opciones que se muestran en diferentes ventanas.

ALTAS

- Altas de Alumnos
- Alta de Profesores
- Altas de Proyectos
- Asociar

BAJAS

- Borrar Alumno
- Borrar Profesor
- Borrar Proyecto

CONSULTAS

- Lista de Alumno
- Lista de Profesor
- Lista de Proyecto

BUSQUEDA

- Por Profesor
- Por Alumno
- Por Proyecto

3.2.1 Interfaz del Menú Principal

La ventana del menú principal contiene varios *link*, donde cada *link* hará referencia a una ventana en particular, para realizar alguna acción con el sistema.

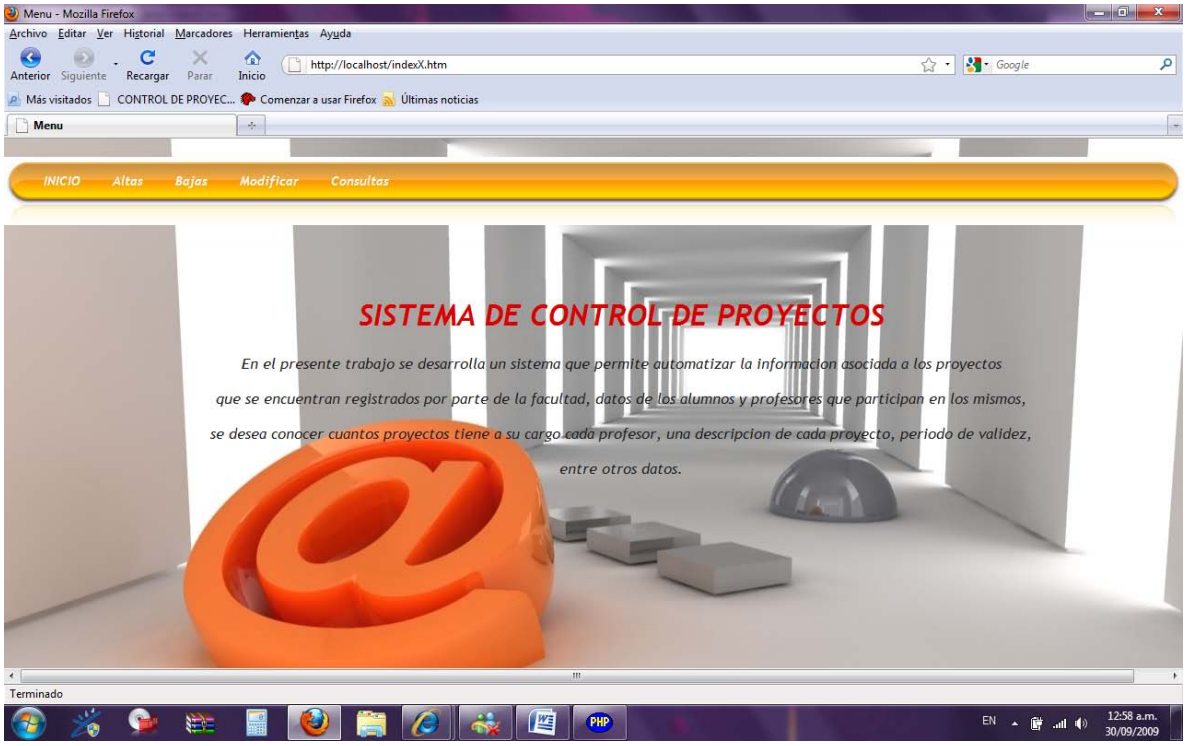

Figura 20. Interfaz del Menú Principal

3.2.1.1 Interfaz de Altas

Cuando se presiona el botón de altas ubicado en la interfaz del menú principal, se puede acceder a varias opciones para realizar, éstas son: dar de Alta a: Alumnos, Profesores, Proyectos y Asignar Proyecto. Este interfaz permite agregar datos a nuestra Base de Datos dependiendo la operación a realizar. Lo podemos observar en la figura 21.

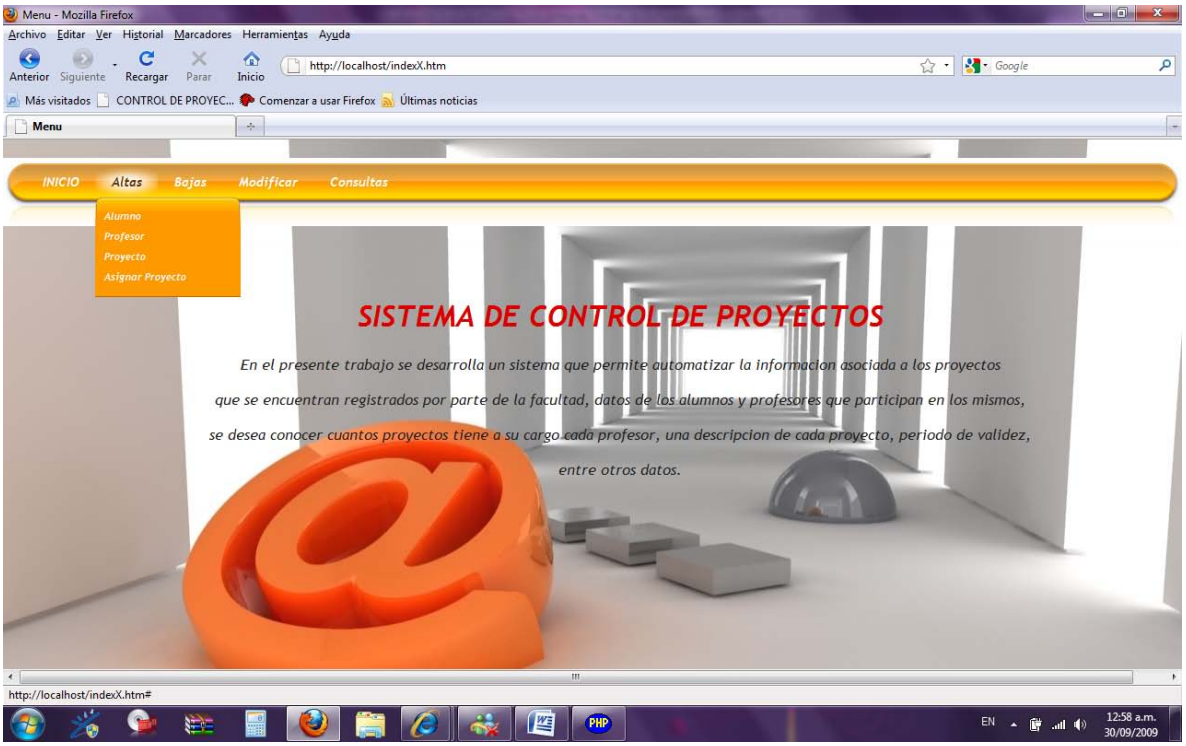

Figura 21. Despliegue de submenús

Una vez seleccionada la opción deseada del submenú, se muestra una pantalla (ver fig. 22), donde se proporcionan los datos, y se muestra un botón, que permite el envío de información a la Base de Datos en Mysql.

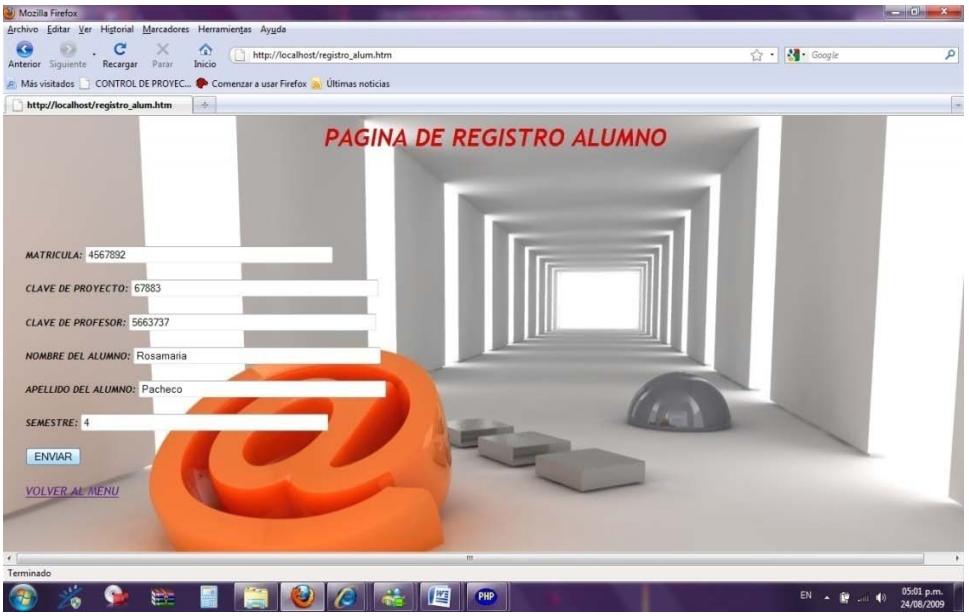

Figura 22. Interfaz de Altas (Alumno)

Las etiquetas que se usaron:

En la parte de inserción de datos se utiliza una forma para mostrar un formulario y se envían los datos por medio de un post.

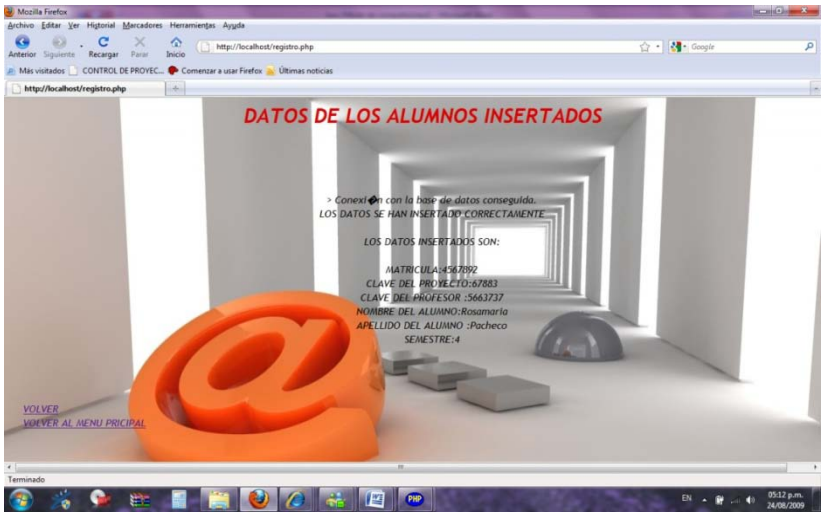

Figura 23. Datos Almacenados

En esa misma aplicación (ver fig. 23) se muestran dos botones que son:

Regresa a la aplicación del formulario para insertar datos.

Nos regresa a la primera interfaz, donde nos muestra la hoja de inicio.

3.2.1.2 Interfaz de Bajas

El usuario podrá acceder a la interfaz de bajas, una vez que haya presionado el botón de Bajas de la interfaz del menú principal. Ahí nos encontraremos con un submenú que nos despliega opciones para las baja de Alumno, Profesor y Proyecto. Se muestra en la siguiente figura 24.

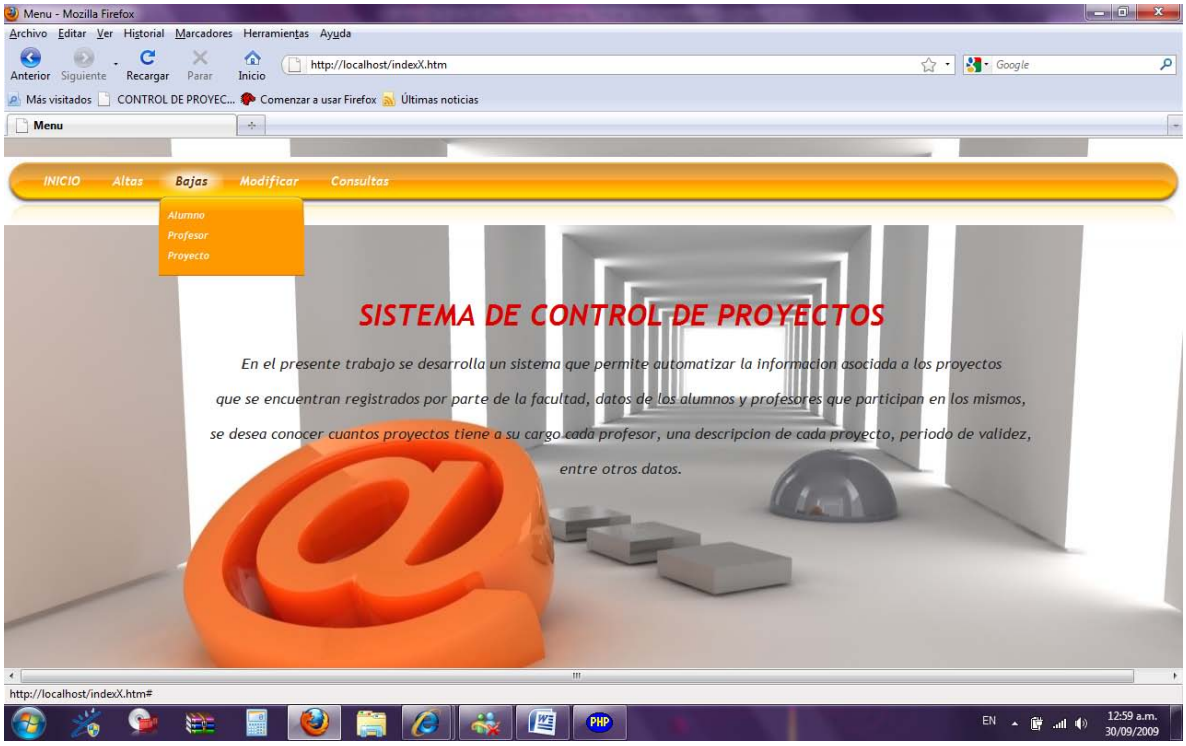

Figura 24. Interfaz de Bajas

En esta parte de Bajas se podrán hacer por medio de la Matricula del Alumno, Clave del Profesor y Clave del Proyecto dependiendo el caso. Una vez proporcionado el identificador para realizar la baja nos mostrará una nueva ventana, donde nos informa que se realizó la baja exitosamente. Además cuenta también con un botón, donde podemos regresar al menú principal o a la aplicación anterior.

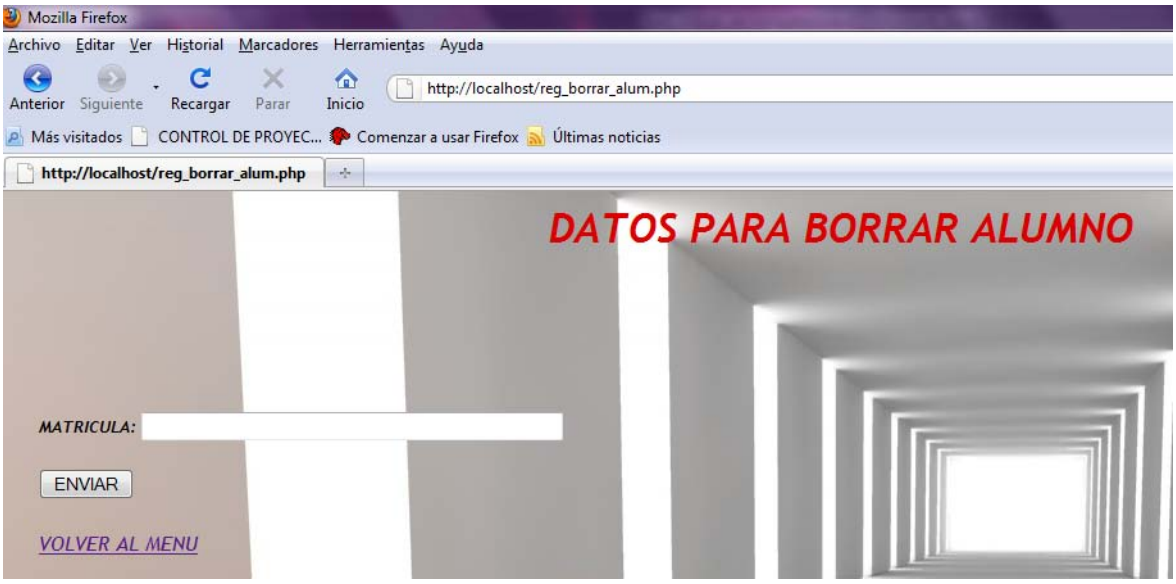

Figura 25. Interfaz de Bajas

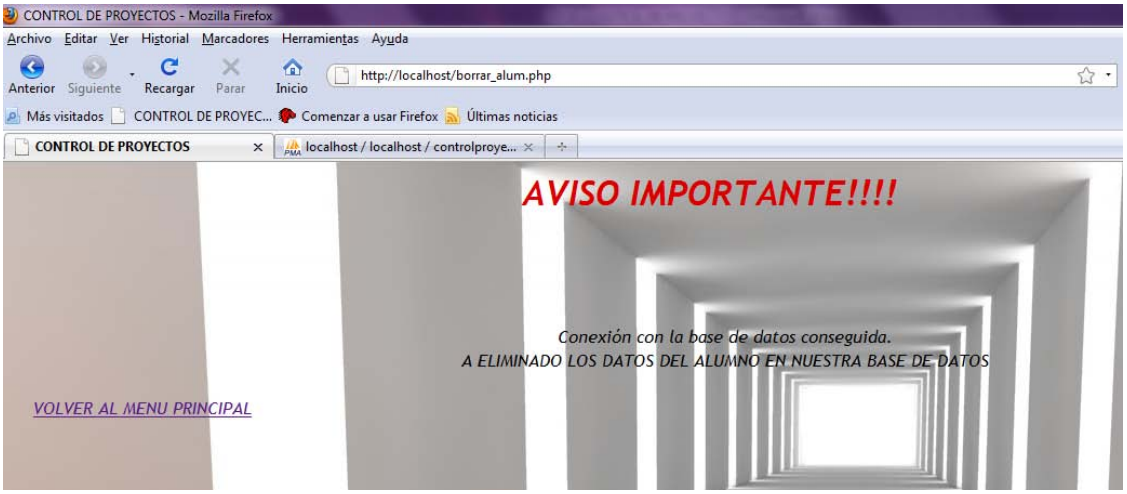

Figura 26. Interfaz de Bajas

En la interfaz de bajas se usaron las funciones post, delete y mysql_query. El identificador se envía por medio de post, se busca en la tabla correspondiente y se pide que borre el dato, actualizando la base de datos.

3.2.1.3 Interfaz para Modificar

El usuario podrá tener acceso a esta interfaz por medio del botón de Modificar, ubicado en el menú principal. Esto nos despliega un submenú donde podemos acceder a Alumno, Profesor y Proyecto. Para modificar ingresaremos nuevamente los datos como si estuviéramos dando de Alta, con la diferencia que es obligatorio tener el mismo número de identificador, el cual puede ser la matrícula del alumno, clave del profesor o la clave del proyecto, dependiendo el caso, en los demás campos se hacen las modificaciones que uno necesite.

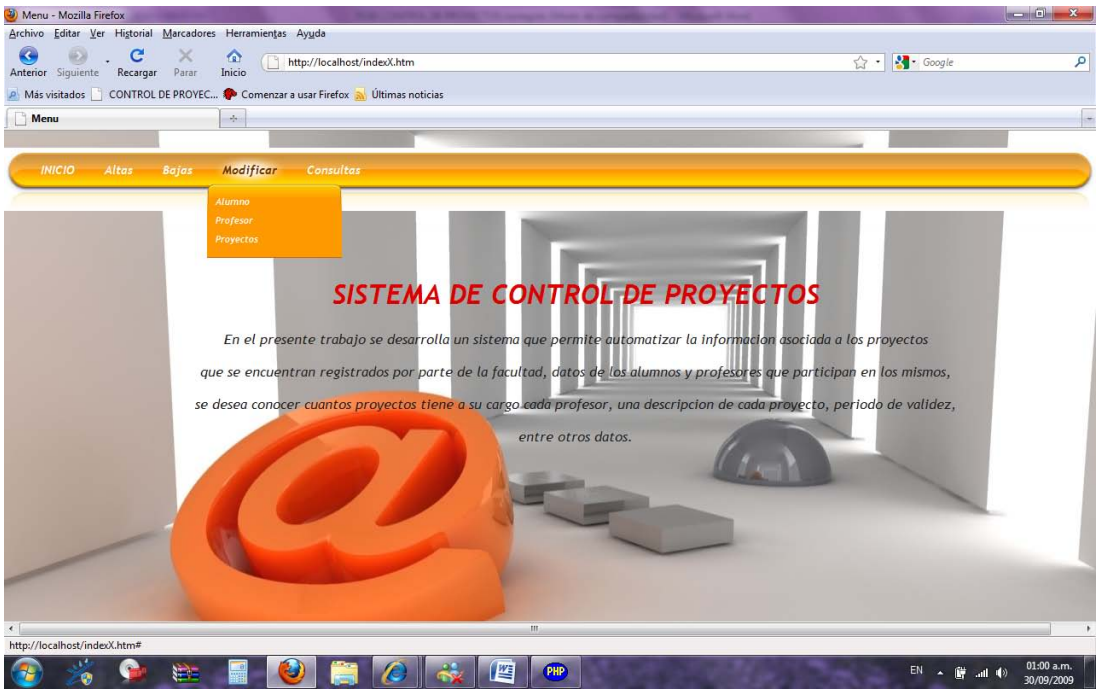

Figura 27. Interfaz de Modificar

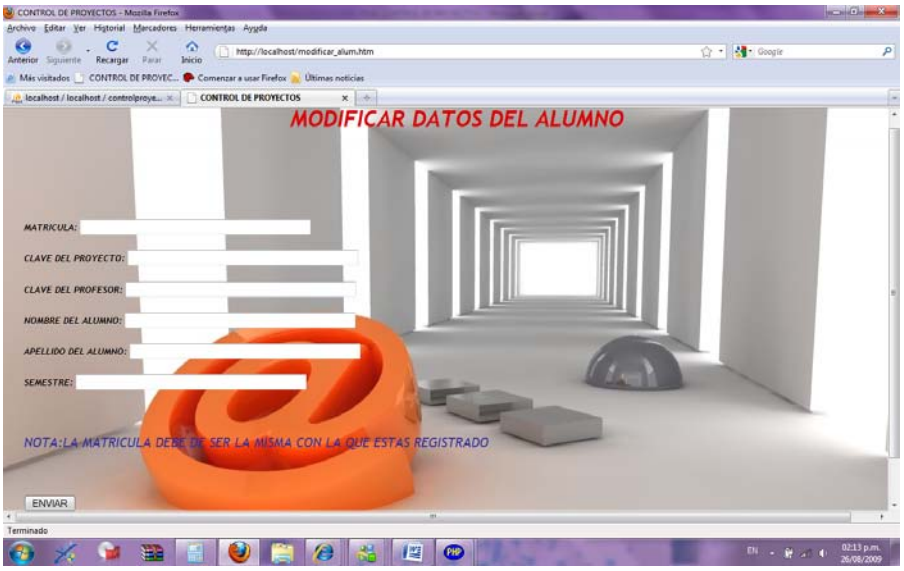

Figura 28. Interfaz de Modificar (alumno)

Una vez llenado todos los campos se envían a la base de datos por medio de un post, y nuestra base de datos actualiza dichos campos con un update y nos envía un mensaje que los datos han sido modificados. En esta misma aplicación nos encontramos con un botón que nos envía al menú principal (ver fig. 28).

3.2.1.4 Interfaz de Consultas

El usuario tiene acceso a esta interfaz de consulta por medio del menú principal en donde nos despliega un submenú en la cual tenemos dos categorías que son: consultas generales y listas independientes.

Las consultas generales se refieren a que despliega una lista completa de cada tabla de la base de datos. Las tablas pueden ser Alumno, Profesor y Proyecto, en esta misma interfase nos encontramos con un botón, el cual nos regresa al menú principal, como se muestra en las siguientes figuras 29 y 30.

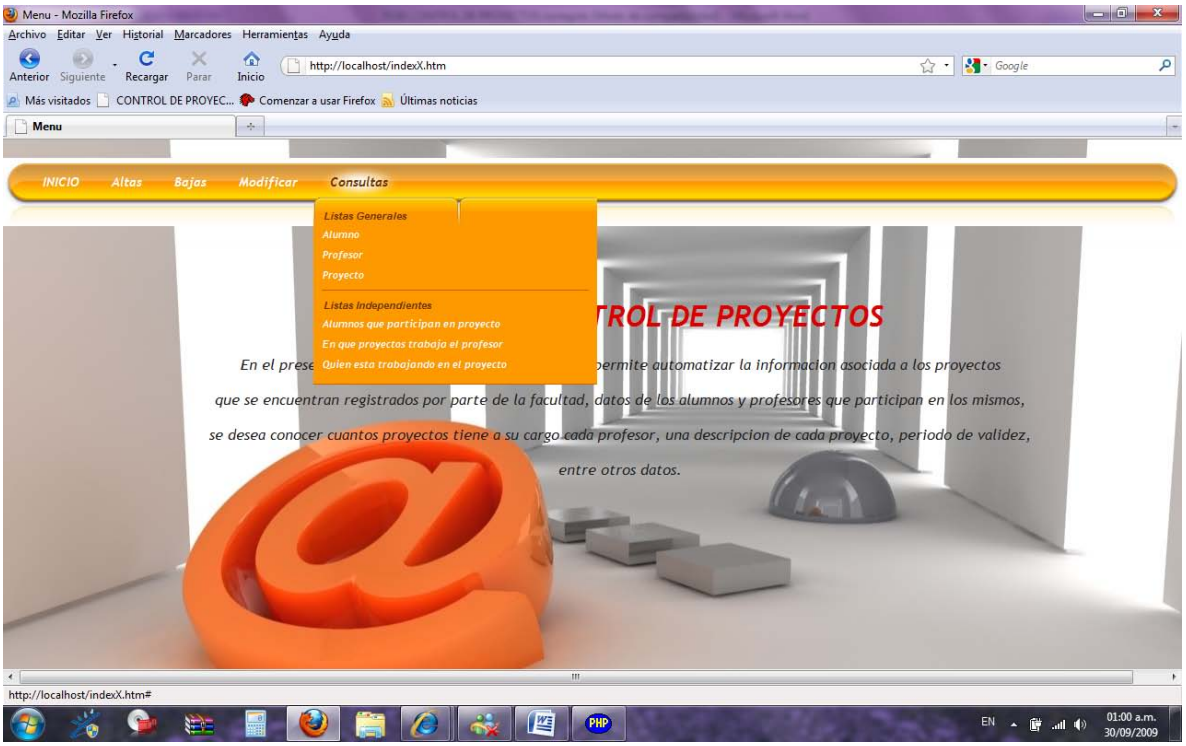

Figura 29. Interfaz de Consultas

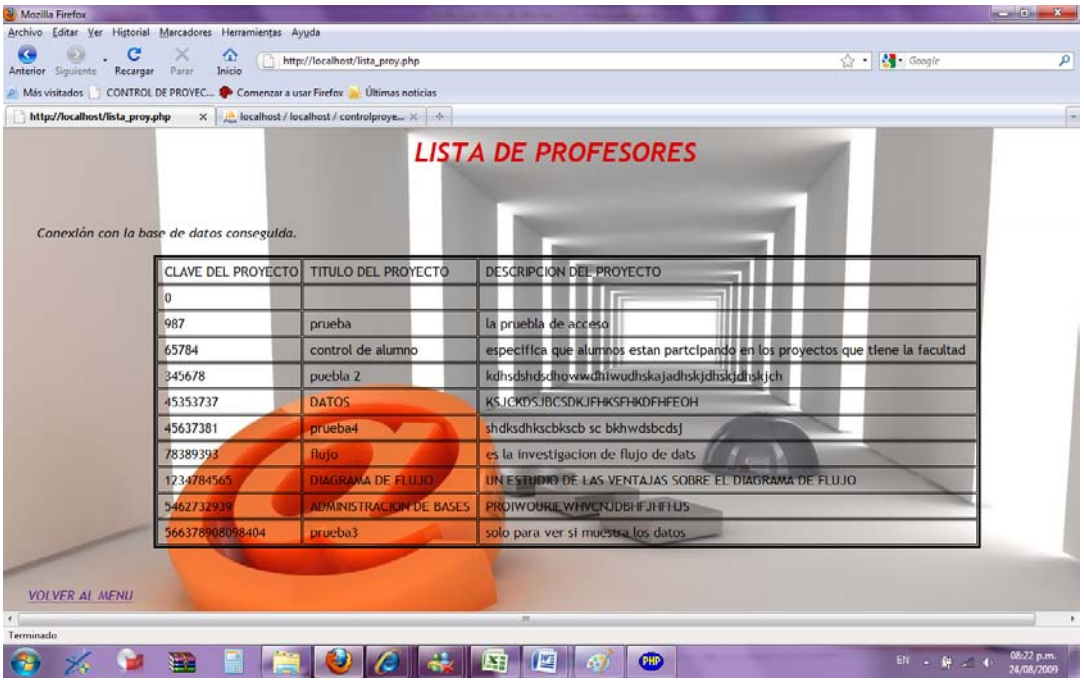

Figura 30. Interfaz de Lista Generales

Para acceder a las consultas de lista independiente, se realiza por medio de la interfaz del menú principal y nos proporciona tres tipos de listas específicas, según lo que se necesita saber, las opciones son las siguientes: por medio del Alumno se inserta la matrícula y arroja una lista en donde se tiene una información más concreta, que es tener el nombre del alumno completo, en que proyecto colabora, la descripción de dicho proyecto y la fecha en que comenzó a trabajar en dicho proyecto. En esa misma aplicación también nos encontramos con un botón que nos regresa al menú principal, como en el siguiente ejemplo, fig. 31.

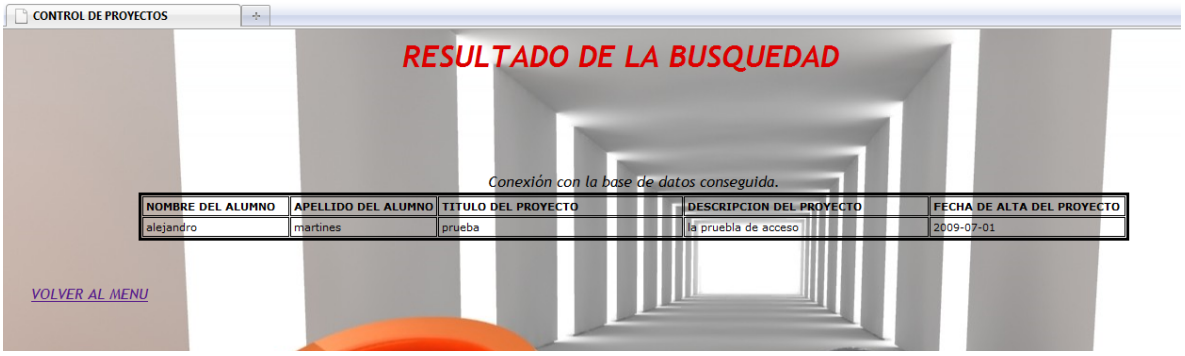

Figura 31. Interfaz de Consulta por Alumno

La segunda opción que encontramos en esa interfaz es la consulta por profesor, ahí proporcionamos la clave del Profesor en donde realizará una consulta y extraerá de la base de datos la siguiente información: la clave del proyecto o proyectos en el que colabora, el nombre del proyecto o proyectos, el nombre del profesor y la fecha o fechas en las que le fueron asignados dichos proyectos. También encontraremos en esa misma interfaz un botón que nos regrese a la aplicación del menú principal, como se muestran en las siguientes figuras (ver fig. 32 y 33).

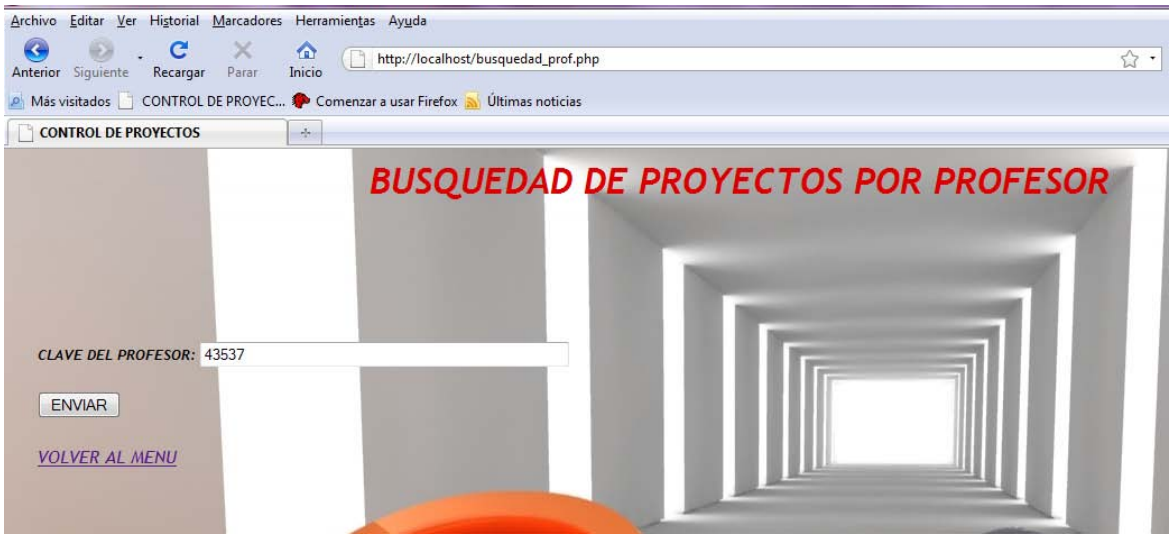

Figura 32. Interfaz de Consultas por Profesor

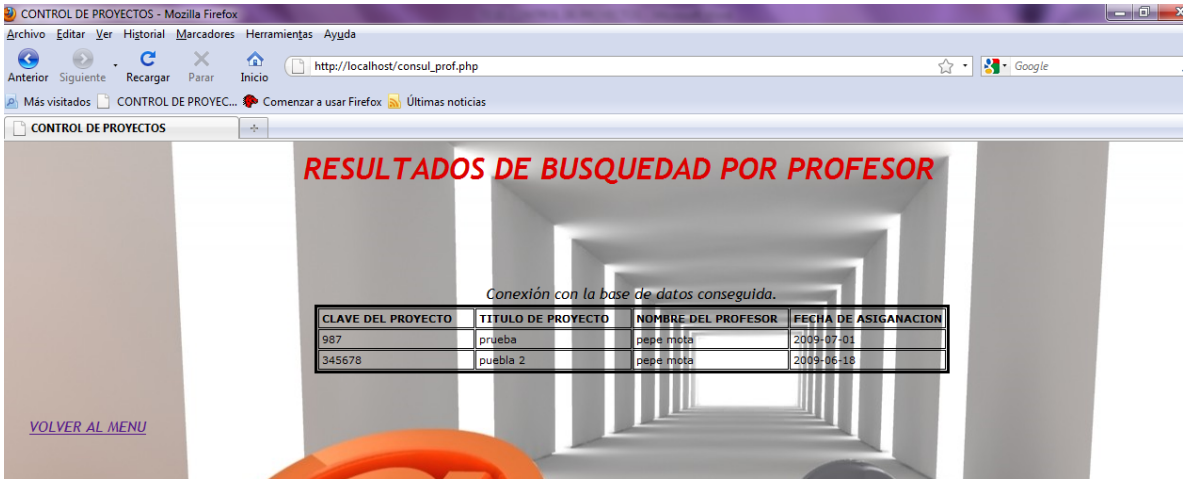

Figura 33. Interfaz de Resultado de la consulta (Profesor)

Por último se tiene la interfaz de consultas por proyecto, en donde se inserta la clave del proyecto que se desea buscar o consultar.

Figura 34. Interfaz de Consultas por Proyecto

CONCLUSIONES

El presente trabajo se cumple con el objetivo general y con los requisitos planteados en la tesis, la automatización de la información de los proyectos que se desarrollan en la Facultad por parte de alumnos y profesores.

Se analizó, desarrolló y diseñó el sistema de acuerdo a los objetivos específicos, planteando soluciones y estrategias para elaborar y crear un Sistema eficaz y manejable, que le permita al usuario manipular la información.

Es importante mencionar que el sistema se encuentra en etapa de Implantación.

BIBLIOGRAFIA

1. Diseño de Bases de Datos
   Wierderhold
   Editorial McGraw-Hill
2. Diseño y Administración de Bases de Datos
   Hansen G.W. & Hansen J.V.
   Prentice Hall
3. Ingeniería de Software
   Farley, Richard
   Editorial McGraw-Hill, 1985
4. Php y MySQL
   Jose Lopez Quijado
   Editorial Alfaomega Grupo Editor, 2008
5. Fundamentos de Bases de Datos
   Korth, Silberschatz
   Editorial McGraw-Hill
6. Ingeniería de Software un Enfoque Práctico
   Roger S. Pressman
   Editorial McGraw-Hill
7. Introducción a las Bases de Datos
   C.J. Date
   Editorial Adison Wesley Iberoamericana
8. Introducción al Diseño de Base de Datos Relacionales
   Jackson
   Editorial Anaya Multimedia.
9. Notas de Ingeniería de Software
   Lic. Jorge Jiménez González, Marzo 2005
10. Notas de Introducción a las Bases de datos
    Mtra. Beatriz Bernabé Loranca, Marzo 2005
11. Notas de Análisis y Diseño de Base de Datos
    M.C. Jesús García Fernández, Mayo 2005.
12. Notas de Base de Datos Locales
    Dra. Darnes Vilariño Ayala, Junio-Julio 2005.
13. Base de Datos
    http://www.monografias.com/trabajos11/basda/basda.shtml
14. Normalización de bases de datos
    http://es.wikipedia.org/wiki/Normalizaci%C3%B3n_de_bases_de_datos
15. Herramientas para el Desarrollo de Sistemas de Información
    http://www.alipso.com/monografias/desarrollo_de_sistemas_de_informacion/
16. Metodología case
    www.ingenieria.uady.mx/Jpech/SIC2005/.../MetodologiaCASE.doc
17. URL Ingeniería de Software
    http://www.monografias.com/trabajos4/cicdevida/cicdevida.shtml
18. URL Modelo de Espiral
    http://www.itlp.edu.mx/publica/tutoriales/analisis/24.htm
19. URL Diagrama de Flujo de Datos
    http://www.gestiopolis.com/recursos/documentos/fulldocs/ger/diagflu.htm