BENEMÉRITA UNIVERSIDAD AUTONÓMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

"SISTEMA DE BASE DE DATOS DE NOTAS EN LA NUBE (CLOUD COMPUTING)"

TESIS PROFESIONAL

PARA OBTENER EL TÍTULO DE

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

NORBERTO COTZOMI PALETA

ASESOR:

M.C. MIGUEL RODRIGUEZ HERNANDEZ

Puebla, Pue. **Septiembre 2012**

AGRADECIMIENTOS

A DIOS
Sobre todas las cosas y por todo.

A MI PADRE †
Porque le hubiera gustado tener este documento en sus manos.

A MI MADRE
Por todo su apoyo, enseñanzas, cariño y sobre todo porque sin ella no lo hubiera logrado. Gracias Mamá.

A MI ESPOSA
Por su apoyo incondicional, su amor y por animarme a seguir adelante.

A MIS HIJOS
Porque son la alegría de mi vida y el motor para enfrentar las adversidades.

A MI ASESOR
Por su tiempo, sus enseñanzas y por darme la oportunidad de trabajar a su lado.

A LOS PROFESORES
Que me han formado a lo largo de mi vida. Sin ellos viviría en un mundo de ignorancia.

"El que da, no debe volver a acordarse; pero el que recibe nunca debe olvidar." **(Proverbio hebreo)**

CONTENIDO

INTRODUCCIÓN

El término de Cloud Computing o Computación en la nube es el nuevo rumbo que están tomando las Tecnologías de la Información donde todo es proporcionado a través de servidores remotos y entregados al usuario como un servicio a través de la red.

Aunque el término es relativamente nuevo, el usar el Internet como acceso a servicios se ha venido usando con anterioridad, como es el caso del correo electrónico, donde podemos acceder a el desde cualquier equipo y en cualquier lugar siempre y cuando se cuente con una conexión a Internet y un navegador.

El presente trabajo es un sistema de notas que trabaja en la nube, el cual nos permitirá crear, clasificar, editar, buscar y eliminar notas; que a diferencia del sistema de notas tradicional nos permitirá consultar la información en cualquier momento y en cualquier equipo conexión a Internet.

Debido a que es un sistema en la nube, la aplicación y la base de datos de la misma se encuentran almacenadas en servidores remotos a fin de garantizar la disponibilidad.

CAPITULO 1: MARCO TEÓRICO

1.1 CLOUD COMPUTING

1.1.1 BREVE DESCRIPCIÓN Y DEFINICIÓN DE COMPUTACIÓN EN LA NUBE

La computación en la nube es el siguiente paso en la evolución en el Internet donde las aplicaciones, procesos de negocios, colaboración personal y todo cuanto pueda ser procesado por una computadora son entregados al usuario como un servicio de Internet para ser consultados en cualquier momento y en cualquier lugar [3].

La idea de los servicios en la nube ha revolucionado nuevamente el mundo de la informática, esto solo ha sido posible gracias al avance, madurez y consolidación de la industria del software, Hardware y Telecomunicaciones, y en este mundo cambiante y competitivo es necesario evolucionar para no quedarse rezagado.

La nube en si misma es un conjunto de hardware, redes, medios de almacenamiento, servicios e interfaces que permiten la entrega de los servicios requeridos. Los servicios en la nube incluyen la entrega de software, infraestructura y almacenamiento sobre Internet basado en la demanda del usuario.

Los servicios en la nube es algo que ya se ha venido usando con anterioridad como lo es con los servicios de Google Apps, Amazon EC2 y Microsoft Azure, donde podemos acceder a aplicaciones comunes a partir de un navegador Web; desde consultar nuestro correo hasta el de editar un documento de Word entre otras muchas aplicaciones. En esta nube las computadoras se convierten en terminales ligeras, lo cual tiene sus ventajas y desventajas (ver tabla 1.1).

Ventajas del computo en la nube	Desventajas del computo en la nube
Disponibilidad	Percepción de perdida de privacidad de los datos sensibles
Escalabilidad	Dependencia de terceros para operar
Capacidad de comprar recursos en demanda	Posible dificultad de integración
Reducción de costos	

Tabla 1.1 Ventajas y desventajas del cómputo en la nube.

El cómputo en la nube se puede dividir en tres niveles en función de los servicios que ofrecen los proveedores; desde el nivel más interno hasta el más externo.

1.1.2 INFRAESTRUCTURA COMO UN SERVICIO (IaaS - Infraestructure as a Service)

Proporciona la infraestructura para el procesamiento, almacenamiento, redes y otros elementos sobre los cuales los clientes corren sus sistemas operacionales y aplicaciones.

Las ventajas mas importantes de este tipo de servicio es la de transferir hacia el proveedor problemas relacionados con la administración de equipos de cómputo, además de la reducción de costos al pagar solo lo consumido. Otra ventaja importante es que permite la escalabilidad automática y transparente para el consumidor.

1.1.3 PLATAFORMA COMO UN SERVICIO (PaaS - Plataform as a Service)

Proporciona herramientas y ambientes para desarrollar aplicaciones, en cuyo caso el proveedor se encarga de operar la infraestructura (servidores, redes, medios de almacenamiento y sistemas operacionales) [13].

El proveedor además de resolver problemas de infraestructura de hardware, también se encarga del software. El cliente que hace uso de este tipo de soluciones no necesita instalar, configurar ni dar mantenimiento a sistemas operativos, base de datos y servidores de aplicaciones ya que esto es proporcionado en esta plataforma.

1.1.4 SOFTWARE COMO SERVICIO (SaaS – Software as a Service)

Es un modelo de distribución de software, donde el proveedor se encarga del mantenimiento, operación diaria y soporte del software usado por el cliente; de tal manera que el cliente no tiene por que preocuparse de la configuración, implementación o mantenimiento de las aplicaciones [14].

Se puede describir como aquella aplicación consumida a través de Internet, normalmente a través de un navegador y donde los datos así como la lógica residen en la plataforma del proveedor.

1.2 INGENIERIA DE SOFTWARE

1.2.1 PANORAMA GENERAL DE LA INGENIERÍA DE SOFTWARE

Hoy en día la mayoría de las cosas que usamos habitualmente consta o comprende de algún tipo de sistema informático, los cuales incluyen una computadora y software de control. Áreas tales como el sistema financiero, los sistemas de telecomunicaciones o los sistemas de seguridad y de investigación entre muchos otros dependen de un sistema informatizado. Por lo tanto, producir software costeable es esencial para el funcionamiento de la economía de cualquier país.

La ingeniería de software es una disciplina de la ingeniería cuya meta es el desarrollo costeable de sistemas de software, por lo cual es abstracto e intangible y debido a esta característica la única limitante es nuestra capacidad para producir software [6].

Debido a esta complejidad de producir software se han desarrollado métodos efectivos de especificación, diseño e implementación del software. Estas técnicas reducen el esfuerzo requerido para producir sistemas grandes y complejos.

1.2.2 QUE ES UML

El lenguaje de modelado Unificado (UML) es una familia de notaciones graficas, respaldado por un único meta-modelo, que ayuda a definir y diseñar sistemas de software, particularmente en la construcción de sistemas usando el estilo de orientación a objetos. UML es el lenguaje estándar para visualizar, construir, especificar y documentar artefactos de un sistema usando objetos [12]. Permite modelar cosas conceptuales como son los procesos de negocios y funciones de sistema, además de cosas concretas. Se ha convertido el estándar de facto en la industria, debido a que fue concebido por los autores de los tres métodos mas usados en la orientación a objetos: Grady Booch, Ivar Jacobson y james Rumbaugh. Estos autores fueron contratados por Rational Software Company, para crear una notación unificada en la que basar la construcción de sus herramientas CASE, sin embargo han participado en la creación de UML otras empresas tales como Microsoft, Hewlett-Packard, Oracle, SAP, IBM entre otros; así como un grupo de analistas y desarrolladores.

Antes de la fusión de UML, existía una "guerra de métodos", donde los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas (ver figura 1.2) para formar una herramienta compartida entre todos los ingenieros de software que trabajan en el desarrollo orientado a objetos.

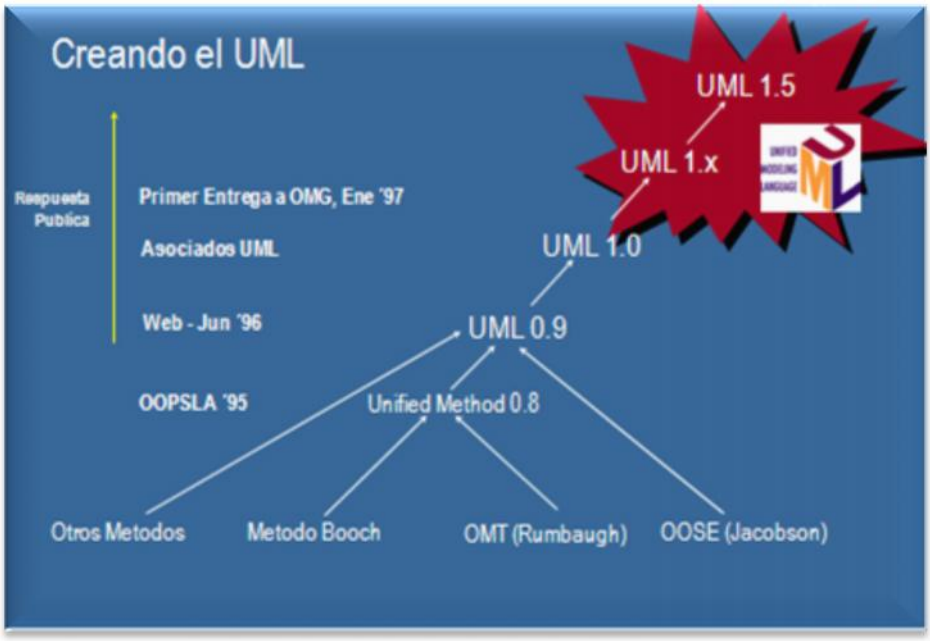

Figura 1.2 Historia de UML

El objetivo principal de la fusión era el posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos en el mercado, para lo cual era necesario definir una notación y semántica común.

Es importante destacar que UML no define un proceso de desarrollo en específico, tan solo se trata de una notación. Es un lenguaje estándar de modelado visual que se usa para especificar,

construir, visualizar y documentar artefactos de un sistema usando objetos [1]. No tiene propietario y esta basado en el común acuerdo de gran parte de la comunidad informática. Pretende abordar los problemas actuales del desarrollo de software, tales como el gran tamaño, distribución, concurrencia, patrones y desarrollo en equipo.

El objetivo final de UML era ser tan simple como fuera posible, pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesitan construir.

UML se debe utilizar para modelar visualmente [12]:

- La iteración del sistema con el mundo externo
- El comportamiento del sistema
- La estructura del sistema
- La arquitectura del sistema
- Los componentes del sistema

1.2.3 MODELOS DE UML

Un modelo representa a un sistema de software desde una perspectiva específica. El modelo capta los aspectos más importantes de lo que estamos modelando, desde cierto punto de vista, y simplifica u omite el resto [1].

La finalidad del modelo es el de captar y enumerar exhaustivamente los requisitos y el dominio de conocimiento, de forma que todos los implicados puedan entenderlos y estar de acuerdo con ellos. En el caso de los modelos de un sistema de software es el de capturar requisitos sobre su dominio de aplicación, la forma en el que los usuarios lo utilizaran, la división en módulos, los patrones y otras cosas. Los implicados que se incluyen son al jefe del proyecto, los analistas, los programadores, los clientes, inversionistas, usuarios finales y los operadores.

Un modelo se puede visualizar desde el punto de vista estático o dinámico (ver figura 1.3). Estas perspectivas nos dan la siguiente clasificación [12]:

Modelo estático o estructural, va a describir los elementos del sistema y sus relaciones con otros elementos y son:

- Diagrama de casos de uso
- Diagrama de clases
- Diagrama de objetos
- Diagrama de despliegue
- Diagrama de componentes

Modelo dinámico o de comportamiento, va a describir el comportamiento del sistema en el tiempo y comprende:

- Diagrama de estados

- Diagrama de actividades
- Diagrama de secuencia
- Diagrama de colaboración

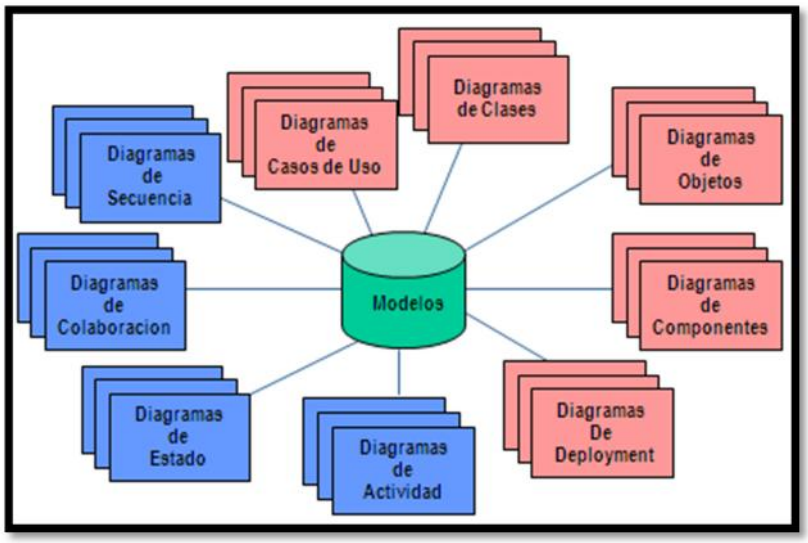

Figura 1.3 Modelos y Diagramas

Para fines de este documento solo se van a revisar los siguientes modelos:

- Modelo de casos de uso
- Modelo conceptual
- Modelo de análisis
    - Diagramas de interacción
        - Diagramas de secuencia
        - Diagramas de colaboración

MODELO DE CASOS DE USO. Organizan los comportamientos del sistema. Un diagrama representa un conjunto de casos de uso, los actores y sus relaciones. Indica como un cliente (actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan. Un caso de uso es una unidad coherente de funcionalidad, expresada como transacción entre los actores y el sistema. El propósito es el de enumerar a los actores y los casos de uso, y demostrar que actores participan en cada caso de uso. Otro de los propósitos es el de comunicar la funcionalidad y comportamiento del sistema al cliente y/o usuario final para que este lo valide.

Los elementos que conforman un caso de uso son:

El actor: Representa cualquier cosa (ser humano, maquina u otro sistema) que interactúa con el sistema [12]. Los actores no son parte del sistema, solo representan los roles dentro del sistema y puede ser un receptor pasivo de información. Cada actor puede participar en uno o más casos de uso. En UML un actor se representa como una persona pequeña con trazos lineales y el nombre debajo (ver figura 1.4).

Figura 1.4 Representación de usuario en UML

Caso de uso: Es una operación o tarea específica que se realiza tras una orden de algún agente externo, que puede ser desde un actor o una invocación desde otro caso de uso. Un caso de uso modela un dialogo entre actores y el sistema; inicia cuando el actor invoca alguna funcionalidad del sistema. El propósito de un caso de uso es definir una pieza de comportamiento coherente, sin revelar la estructura interna del sistema y su representación es mediante un ovalo con el nombre del caso de uso adentro (ver figura 1.5).

Figura 1.5 Representación en UML de un caso de uso

La definición de un caso de uso incluye todo el comportamiento que implica: los flujos principales, los flujos alternos y todas las condiciones excepcionales (ver figura 1.6). Desde el punto de vista del usuario son situaciones excepcionales y desde el punto de los sistemas, son las variaciones adicionales que deben ser descritas y manejadas.

Figura 1.6 Flujo de eventos de casos de uso

Un caso de uso es una descripción lógica de una parte de funcionalidad del sistema. No es una construcción manifiesta en la implementación de un sistema, para ello son útiles las clases que implementan un sistema y es una tarea del diseño encontrar las clases apropiadas de implementación que combinen claramente los roles apropiados para implementar todos los casos de uso.

Relaciones: Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación, esta asociación se representa con una línea con punta de flecha. Esta línea entre el actor y el caso de uso indica que ellos interactúan enviando mensajes uno a otro, donde para cada mensaje enviado se asume una respuesta y la dirección de la flecha indica quien envió el primer mensaje.

Las relaciones de inclusión y extensión se dibujan como flechas de líneas discontiuas con la palabra clave “include” o “extend”. La relación de inclusión apunta al caso de uso a ser incluido; la relación de extensión señala al caso de uso que se extenderá

Un caso de uso también se puede especializar en uno o mas casos de uso hijos, a lo que se le llama generalización de casos de uso

Relación	Función	Notación
Asociación	Línea de comunicación entre un actor y un caso de uso en el que participa	
Extensión	La inserción de comportamiento adicional en un caso de uso base que no tiene conocimiento sobre el	<<extend>>
Generalización de casos de uso	Una relación de un CU general y un CU especifico, que hereda y añade propiedades a aquel	
Inclusión	Inserción de comportamiento adicional en un caso de uso base, que describe explícitamente la inserción	<<include>>

Tabla 1.7 relaciones en los casos de uso

Los casos de uso son muy importantes (como se observa en la tabla 1.7) son la base de todo el sistema y además permite [12]:

- Que los clientes aprueben lo que el sistema debe de hacer
- Los usuarios comprenden mejor el sistema
- Los desarrolladores documentan el comportamiento del sistema
- El líder de proyecto proporciona información para planear los proyectos

Un caso de uso debe especificar un comportamiento deseado, pero no imponer como se llevara a acabo ese comportamiento, esto se realiza utilizando los escenarios; los cuales son una interacción entre el sistema y los actores que puede ser descrito mediante una secuencia de mensajes o dicho

de otra forma es una instancia de un caso de uso. Los escenarios se dividen en primarios y secundarios.

En los primarios o “happy day” contienen el flujo básico y es la forma en la que el sistema debe de funcionar idealmente o la mayoría de las veces.

En los secundarios va a contener los flujos alternos y de excepción

MODELO CONCEPTUAL. Es el primer modelo de clases que se debe de hacer y va a tener el fin de reunir las abstracciones principales (key abstractions) del sistema a construir [12]. Va a explicar cuales son y como se relacionan los conceptos relevantes en la descripción del sistema en base a examinar la descripción del problema y de entrevistas con expertos en el dominio. Este modelo va a incluir un diccionario de modelo y uno o varios diagramas de clases.

Diagrama de clases: Es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema y esta compuesto por las clases y sus relaciones.

Una clase es la unidad básica que encapsula toda la información de un objeto y es a través de ella que se puede modelar el entorno de estudio. Una clase va a identificar un grupo de objetos que tienen

- Propiedades en común (atributos)
- Comportamiento en común (operaciones)
- Relaciones comunes con otros objetos (asociaciones)
- Semántica breve (descripción común)

El nombre de una clase debe ser el sustantivo singular que mejor caracterice la abstracción y deben tomarse directamente del vocabulario del dominio y en caso de usar algún término especial, sigla o apodo debe definirse e incluirse en la documentación del proyecto [12].

Luego de nombrar la clase, debe hacerse una descripción breve y concisa de la clase (work definition), enfocándose principalmente en el propósito de la clase y no en la implementación. El nombre de la clase y la descripción, forman lo que se conoce como un diccionario del modelo.

En UML, una clase es representada por un rectángulo que contiene el nombre de la clase y opcionalmente tres divisiones (ver figura 1.8).

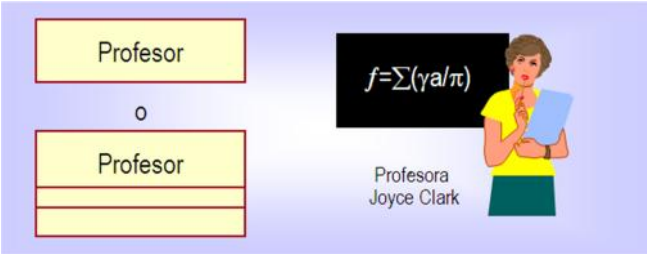

Figura 1.8 Representación de una clase

Donde el primer compartimiento contiene el nombre de la clase, el segundo muestra la estructura o atributos y la tercera muestra el comportamiento o las operaciones. Dependiendo del nivel de detalle que se quiera mostrar en el diagrama de clases se puede suprimir la segunda y tercera sección.

Los diagramas de clases pueden ser desde muy simples hasta muy abstractos, dependiendo de la perspectiva que se requiera y se puede clasificar de la siguiente manera:

La perspectiva conceptual, que es la que se usa en el modelo conceptual y no tiene detalle alguno que solo el nombrar las clases (ver figura 1.9).

La perspectiva de especificación, es usada en el modelo de análisis y tienen un detalle mayor de las clases, están en el punto medio de ser abstractas y concretas, se define que es lo que debe de hacer pero no como hacerlo.

La perspectiva de implementación, es usada para el modelo de diseño y las clases son mas concretas, pues debe de contener la información de cómo construirlas.

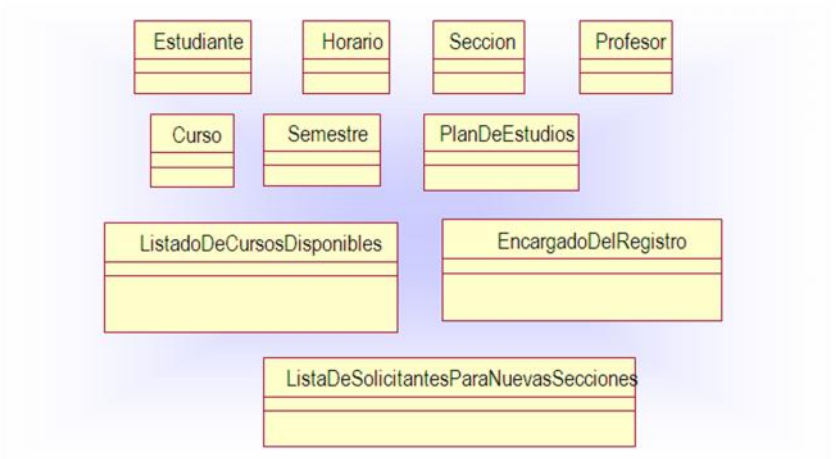

Figura 1.9 Diagrama de clases para el modelo conceptual

MODELO DE ANALISIS. Es el resultado del proceso de análisis y representa la conceptualización del domino del sistema. Para la elaboración se toma como base el modelo de casos de uso y el modelo conceptual; además de la documentación disponible, tales como entrevistas o la propia descripción del problema (ver figura 1.10). Se usan los escenarios de los casos de uso para dirigir, ordenar y ejecutar el proceso [12].

El modelo de análisis puede verse desde dos perspectivas o vistas:

-El modelo estático que representa la estructura del modelo de análisis y va a incluir el diagrama de clases y el diagrama de objetos.

-El modelo dinámico, que va a mostrar las interacciones y responsabilidades que se manejan en el sistema e incluye:

- Los diagramas de interacción
  - Diagrama de secuencia
  - Diagrama de colaboración
- Los diagramas de estado

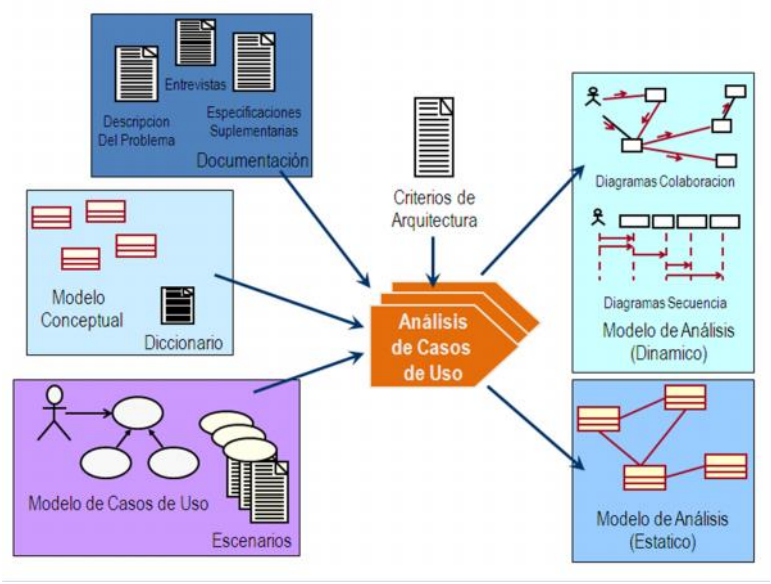

Figura 1.10 Construcción del modelo de análisis

Diagrama de secuencia. Muestra una interacción ordenada según la secuencia temporal de eventos. En particular muestra los objetos participantes en la interacción y los mensajes que intercambian según la secuencia del tiempo [13].

En el eje vertical muestra el tiempo y en el horizontal los objetos y actores participantes en la interacción sin un orden prefijado. Cada objeto o actor tiene una línea vertical y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba hacia abajo.

Se realiza un diagrama de secuencia por cada caso de uso o para una parte de un caso de uso (subcaso de uso)

Diagrama de colaboración. Se muestra una forma alternativa al diagrama de secuencia en el sentido de mostrar un escenario. En este tipo de diagrama se muestran las interacciones entre objetos organizadas entorno a los objetos y los enlaces entre ellos [13].

Los diagramas de secuencia proporcionan una forma de ver el escenario en un orden temporal y los clientes entienden fácilmente este tipo de diagramas, por lo que resultan útiles en las primeras fases de análisis.

1.3 MODELOS DE PROCESO DE SOFTWARE

1.3.1 INTRODUCCIÓN

Los modelos prescriptivos de proceso se propusieron originalmente para ordenar el caos del desarrollo de software y proporcionan un camino a seguir. Estos modelos definen un conjunto distinto de actividades, acciones, tareas, fundamentos y productos de trabajo que se requieren para desarrollar software de alta calidad.

Proporciona estabilidad, control y organización a una actividad que de lo contrario podría volverse caótica.

1.3.2 MODELO LINEAL O (CASCADA)

Es el primer modelo de proceso de desarrollo de software que se publico y se derivo de procesos de ingeniería de sistemas más generales. El modelo en cascada, algunas veces llamado ciclo de vida clásico, sugiere un enfoque sistemático y secuencial hacia el desarrollo del software (ver figura 1.11).

El modelo en cascada es el paradigma más antiguo para la ingeniería de software. Entre los problemas que se encuentran al aplicar el modelo en cascada se encuentran:

- Impide el trabajo en equipo, ya que se debe de esperar a otros para terminar tareas pendientes
- El cliente debe de tener paciencia, ya que una versión que funcione de los programas estará disponible cuando el proyecto este avanzado.
- Con frecuencia es difícil establecer todos los requisitos de manera explicita

Sin embargo como un modelo de proceso útil en situaciones donde los requerimientos están fijos y donde el trabajo se realiza, hasta su conclusión.

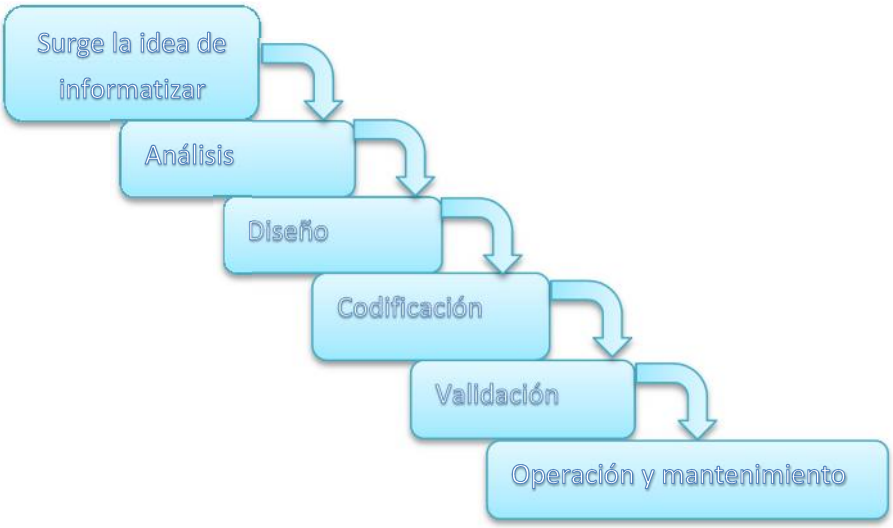

Figura 1.11 Modelo en cascada

1.3.3 MODELO EN ESPIRAL

Es un modelo de proceso de software evolutivo, en donde más que representar el proceso de software como una secuencia de actividades con retrospectiva de una actividad a otra, se representa como una espiral. Cada ciclo en la espiral representa una fase del proceso del software (ver figura 1.12). Así, el ciclo mas interno podría referirse a la viabilidad del sistema, el siguiente ciclo a la definición de requerimientos, el siguiente ciclo al diseño, y así sucesivamente.

Cada ciclo de vida se divide en cuatro sectores:

Definición de objetivos. Para esta fase del proyecto se definen los objetivos específicos. Se identifican las restricciones y los riesgos del proyecto

Evaluación y reducción de riesgos. Se lleva a cabo un análisis detallado para cada uno de los riesgos del proyecto y se definen los pasos para reducir dichos riesgos

Desarrollo y validación. Después de la evaluación de riesgos, se elige un modelo para el desarrollo del sistema.

Planificación. El proyecto se revisa y se toma la decisión de si se debe de continuar con un ciclo posterior de la espiral.

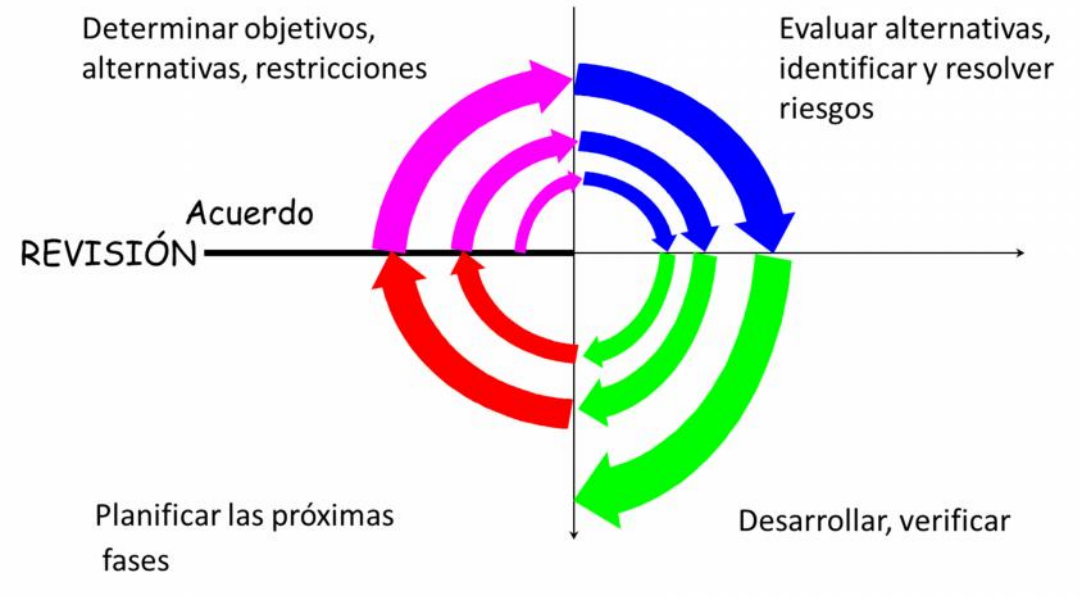

Figura 1.12 Ciclo de vida del modelo en espiral

La diferencia principal entre el modelo en espiral y otros modelos de software es la consideración explicita del riesgo, ya que estos originan problemas en el proyecto.

1.3.4 MODELO DE PROTOTIPOS

Este modelo consiste en crear un prototipo general, comenzando con el diseño y construcción de las partes más importantes de la aplicación, para posteriormente refinarlo y ampliarlo hasta que el prototipo se termine.

Al igual que otras ingenierías, el uso de prototipos es útil para que el cliente observe, confirme y mejore el producto (ver figura 1.13).

El empleo de este modelo es útil para comunicar, discutir las ideas de los diseñadores y las partes responsables. Las etapas del modelo son:

- Investigación preliminar
- Recolección y refinamiento de los requisitos
- Diseño técnico
- Diseño y construcción del prototipo
- Evaluación del prototipo por el cliente
- Renacimiento del prototipo

Una de las ventajas de este modelo es que los clientes pronto saben lo que quieren cuando ven un modelo y lo utilizan.

Este enfoque es útil cuando:

- El cliente no tiene claro lo que quiere
- Al cliente le gustaría ver algo similar para poder hacerse una idea de lo que obtendrá

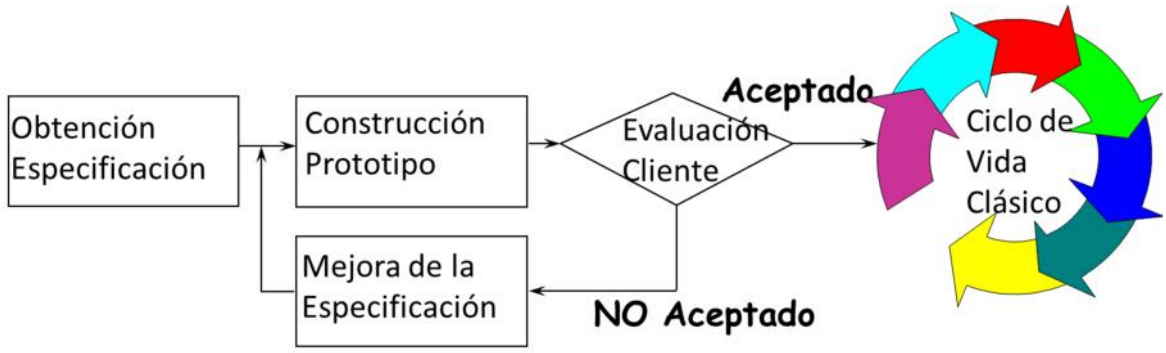

Figura 1.13 Modelo en prototipos

1.3.5 MODELO INCREMENTAL

El modelo incremental combina elementos del modelo en cascada aplicado en forma iterativa, en donde en el primer incremento se muestra un producto esencial (ver figura 1.14). Es decir, se incorporan los requisitos básicos, pero muchas características suplementarias (algunas conocidas, otras no) no se incorporan. Este producto esencial queda en manos del cliente, sometiéndose a una evaluación para posteriormente desarrollar un plan para el incremento siguiente.

El plan afronta la modificación del producto esencial con el fin de satisfacer de una mejor manera las necesidades del cliente y la entrega de características y funcionalidades adicionales. Este proceso se repite después de la entrega de cada incremento mientras no se haya elaborado el producto completo.

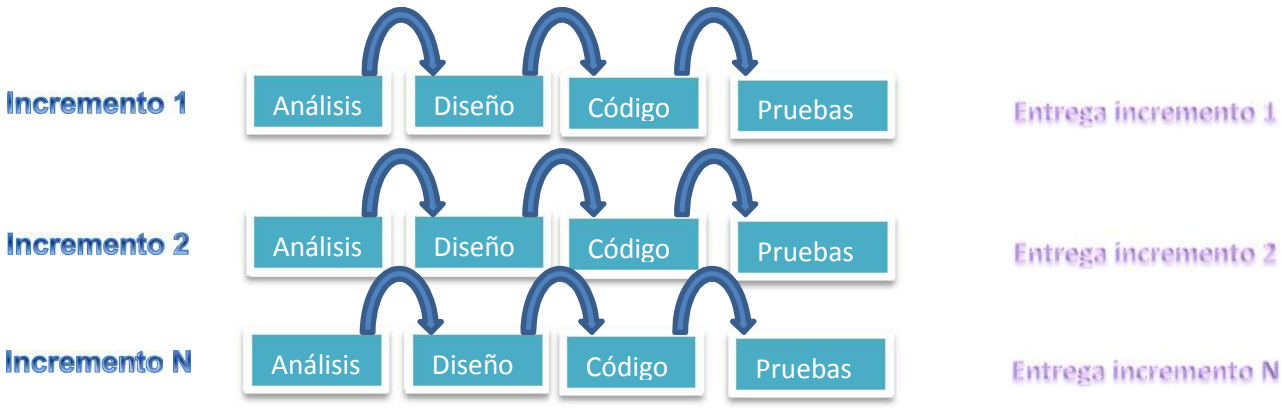

Figura 1.14 Modelo Incremental

1.3.6 PROCESO UNIFICADO DE RATIONAL (RUP)

RUP es un producto comercial desarrollado y comercializado por Rational Software, una compañía de IBM, y es además un proceso de software. El antecedente mas importante se ubica en 1967 con la metodología Ericsson elaborada por Iván Jacobson, la que introdujo el concepto de los Casos de uso.

En los años de 1987 a 1995 Jacobson fundo la compañía Objectory, la que posteriormente en 1995 fue adquirida por Rational Software Corporation. Como producto de esta fusión se desarrolla Rational Objectory Process (ROP) a partir de Objectory 3.8 y del enfoque de Rational.

A partir desde ese entonces se desarrollo e incorporo diversos elementos para expandir ROP y en Junio de 1998 se lanza Rational Unified Process (RUP).

Dentro de las características de RUP se pueden mencionar:

Gestión de requisitos: RUP brinda una guía para encontrar, organizar, documentar y seguir los cambios de los requisitos funcionales y restricciones. Utiliza los casos de uso y los escenarios para representar los requisitos.

Desarrollo de software iterativo: desarrollo mediante hitos bien definidos, en las cuales se repiten las actividades pero con distinto énfasis, según la fase del proyecto.

Desarrollo basado en componentes: Divide el sistema en componentes con interfaces bien definidas, las cuales serán ensambladas para generar el sistema.

Modelado visual: UML es un lenguaje para visualizar, especificar, construir y documentar un sistema. Facilita la gestión de modelos, permitiendo ocultar o exponer detalles cuando sea necesario y en general ayuda a gestionar la complejidad del software.

Verificación continúa de la calidad: Es importante verificar la calidad de los artefactos, especialmente al final de cada iteración. Para los artefactos no ejecutables las revisiones e inspecciones también deben de ser continuas.

Gestión de los cambios: El cambio es un factor de riesgo crítico en los proyectos. Estos cambios pueden ser a lo largo del desarrollo y por mantenimiento, posteriormente a la entrega del producto. Otro gran desafío es cuando se tienen múltiples desarrolladores y quizás trabajando en distintas plataformas. La gestión de cambios y de configuración es la disciplina de RUP encargada de este aspecto y su ausencia conduciría al caos.

En la figura 1.15 se muestra que el proceso puede ser descrito en dos dimensiones o ejes:

El *eje horizontal* que representa el tiempo e indica las características del ciclo de vida del proceso expresado en términos de fases (Inicio, elaboración, construcción y transición), iteraciones e hitos.

El *eje vertical* que representa los aspectos estáticos, donde se describe el proceso en términos de componentes de proceso, disciplinas, flujos de trabajo, actividades, artefactos y roles.

Figura 1.15 Estructura de RUP

RUP se repite a lo largo de una serie de ciclos que constituyen la vida de un producto. Cada ciclo consta de cuatro fases: Inicio, Elaboración, Construcción y Transición, cada una con sus respectivas iteraciones y al final de cada ciclo se tiene un producto para los clientes.

Cada fase se concluye con un hito bien definido, el cual va a ser un punto en el tiempo donde se van a tomar decisiones críticas y alcanzar las metas clave antes de pasar a la siguiente fase, los cuales pueden tener a su vez hitos menores de acuerdo a cada iteración. Los hitos para cada fase se muestran en la figura 1.16.

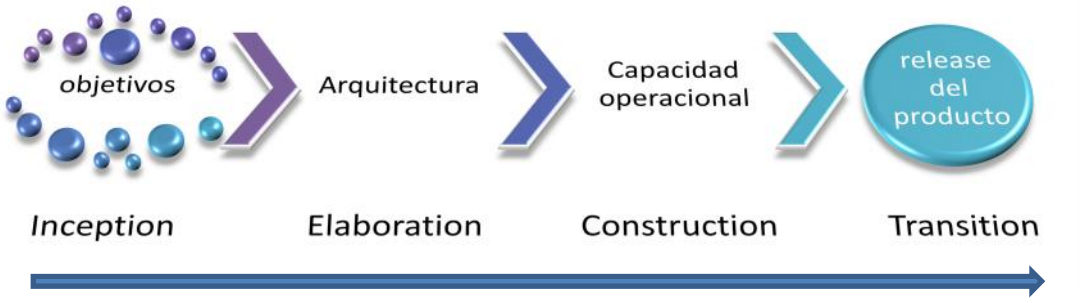

Figura 1.16 Fases e Hitos en RUP

1.4 BASES DE DATOS

1.4.1 INTRODUCCIÓN Y DEFINICIÓN DE BASES DE DATOS

Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. Las bases de datos son ampliamente usadas por empresas u organizaciones para el manejo de la información de los clientes, cuentas, transacciones bancarias hasta una universidad para el manejo de los estudiantes.

La mayoría de la gente interactúa con las bases de datos aunque no este consciente de ello, ya que estas forman parte esencial de la vida cotidiana; por ejemplo cuando se accede a un sitio web de un banco y se consulta un estado de cuenta o los movimientos; la información solicitada se recupera del sistema de base de datos del banco.

Una base de datos es un conjunto de datos persistentes (datos que solo pueden ser removidos por una solicitud explicita) que son utilizados por los sistemas de aplicación de alguna empresa [8]. Se puede considerar a la base de datos como una especie de archivero electrónico moderno, es decir, es una colección de archivos de datos digitalizados. Los usuarios de las bases de datos dependiendo de los privilegios pueden llevar a cabo las siguientes operaciones.

- Agregar nuevos registros a la base de datos
- Insertar datos dentro de los archivos existentes
- Recuperar datos de los archivos existentes
- Modificar datos de los archivos existentes
- Eliminar datos de los archivos existentes
- Eliminar archivos existentes de la base de datos

Un sistema de Base de Datos comprende de cuatro componentes principales [10]:

Datos. En general los datos de un sistema de Base de Datos van a ser Integrados (imaginar la base de datos como la unificación de varios archivos) y Compartidos (las piezas individuales de los datos en la base pueden ser compartidas entre diferentes usuarios)

Hardware. Consta de los sistemas de almacenamiento físico, procesadores de hardware y memoria principal

Software. Entre la capa física (hardware) y los usuarios hay una capa de software conocida como el administrador de la base de datos (DBMS) o el servidor de la base de datos.

Usuarios. Existen tres modos de usuarios los programadores, los usuarios finales y el administrador de la base de datos (ver figura 1.17).

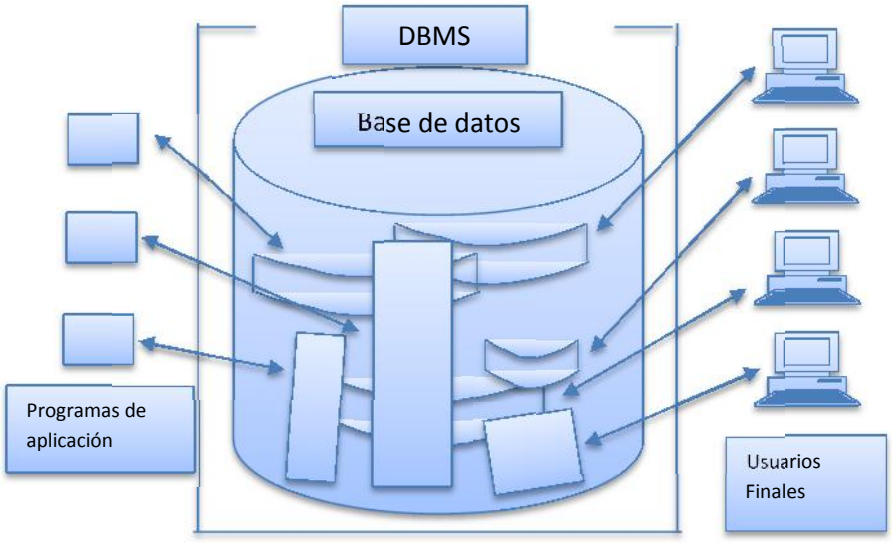

Figura 1.17 Sistema de base de datos

1.4.2 DBMS

El *DBMS* (Database Managment System -Sistema de Administración de la Base de Datos-) Es una colección de programas que permite a los usuarios crear y mantener una base de datos [9]. El DBMS es un software de propósito general que facilita los procesos de definir, construir, manipular y compartir bases de datos entre varios usuarios y aplicaciones.

Definir. Se refiere al especificar el tipo de datos, la estructura y las construcciones para que los datos sean almacenados en la base de datos.

Construir. Es el proceso de almacenar los datos de la base en un medio de almacenamiento controlado por la DBMS.

Manipulación. Incluye funciones de solicitudes a la base de datos tales como solicitudes de datos, actualizar la base de datos y generar reportes de los datos.

Compartir. Permite a los usuarios y programas compartir la información almacenada y el acceso concurrente.

Otras de las funciones importantes del DBMS es el de la protección y el mantenimiento.

Protección. Se refiere a la protección del sistema por el mal funcionamiento del software o hardware, así como también por el acceso sin autorización a la base de datos.

Mantenimiento. Una típica base de datos grande tiene un ciclo de vida de muchos años, asi que una DBMS debe permitir la evolución de la misma.

1.4.3 USUARIOS DE LA BASE DE DATOS

Para una pequeña base de datos, como una lista de direcciones, una persona por lo regular define, construye y administra la base de datos. Sin embargo, muchas personas son requeridas en el diseño, uso y mantenimiento de una gran base de datos [9].

Administradores de la base de datos. En cualquier organización donde muchas personas usan los mismos recursos, es necesario un administrador para revisar y administrar esos recursos. El administrador de la Base de Datos (DBA) es la persona responsable de autorizar el acceso a la base de datos, coordinar y monitorear su uso, y de adquirir el software y hardware necesario. El DBA se tiene que enfrentar a problemas tales como la seguridad o vigilar el rendimiento de la base de datos.

Diseñadores de la base de datos. Los diseñadores son los responsables de identificar los datos a ser almacenados y de elegir las estructuras apropiadas para representar y almacenar los datos. Es responsabilidad de los diseñadores de la base de datos comunicarse con todos los posibles usuarios de la base de datos con el fin de entender sus necesidades, y tener un diseño que cumpla con todos los requerimientos

Usuarios finales. Son las personas quienes ingresan a la base de datos para consultar, actualizar y generar reportes. Aunque estos son los principales usos, hay categorías de usuarios:

- *Usuarios casuales.* Ocasionalmente ingresa a la base de datos, necesitando diferente información y típicamente son administradores de mediano o alto nivel.
- *Usuarios ingenuos*. Su principal trabajo principalmente es el de consultar y actualizar la base de datos usando tipos de consultas estándar, las cuales son cuidadosamente programadas y probadas.
- *Usuarios sofisticados.* Incluye a los ingenieros, científicos, analistas de negocios, y otros quienes se familiaricen con el uso del DBMS.
- *Usuarios solitarios.* Mantienen bases de datos personales usando programas que proporcionan un manejo fácil a base de menús y gráficos.

Analistas de sistemas e ingenieros de software. Determinan las necesidades de los usuarios finales y las implementan en un programa. Ellos prueban, depuran, documentan y mantienen esas implementaciones.

1.5 MODELO DE BASES DE DATOS

1.5.1 MODELO DE DATOS

El modelo de datos es una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de consistencia. Ejemplos de modelo de datos son el

modelo Entidad-Relación o el modelo relacional, los cuales se van a explicar posteriormente. Los diferentes modelos de datos que se han propuesto se clasifican en

- *Modelos lógicos basados en objetos.* Se usan para describir datos en el nivel conceptual y de visión. Ejemplos son el modelo Entidad-Relación y el modelo orientado a objetos.
- *Modelos lógicos basados en registros*. Se usan para describir el nivel conceptual y el nivel físico. Ejemplos son el modelo de red, jerárquico y relacional.
- *Modelos físicos*. Se usan para describir datos en el nivel mas bajo. Hay pocos modelos físicos de datos en uso.

1.5.2 MODELO ENTIDAD- RELACIÓN (E-R)

El modelo Entidad-Relación (E-R) esta basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades así como de las relaciones entre estos objetos.

Una entidad es un objeto que existe y que se distingue de otros objetos de acuerdo a sus características llamadas atributos. Estas entidades pueden ser concretas como una persona o abstractas como una fecha.

Las entidades se describen en una base de datos mediante un conjunto de atributos, los cuales van a describir las propiedades que posee cada miembro de un conjunto de entidades. Por ejemplo los atributos matricula, nombre-estudiante y cursos pueden describir a la entidad estudiante. Los atributos se pueden caracterizar por:

- *Atributos simples*. Que no se dividen en subpartes
- *Atributos compuestos*. Que se pueden dividir en otros atributos
- *Atributos monovalorados*. Consta de un solo valor para una entidad concreta.
- *Atributos multivalorados*. Tiene un conjunto de valores para una entidad específica.
- *Atributos derivados*. Su valor se puede obtener de otros atributos o entidades.
- *Atributos clave*. Es un atributo cuyo valor es distinto para cada entidad individual y sirven para identificar de manera única a cada entidad.

Un atributo toma el valor nulo cuando una entidad no tiene valor para un atributo, lo que puede indicar "no aplicable" o que no existe, que el valor es desconocido o es un valor perdido.

Una relación es una asociación entre diferentes entidades y el conjunto de todas las entidades y relaciones del mismo tipo se denominan respectivamente conjunto de entidades y conjunto de relaciones. Una relación binaria es donde se tienen dos conjuntos de relaciones y al número de entidades que participan en un conjunto de relaciones se le conoce como grado.

La cardinalidad va a expresar el número de entidades a las que otra entidad puede estar asociada vía un conjunto de relaciones. Para un conjunto de relaciones binarias R entre los conjuntos de entidades A y B, la correspondencia de la cardinalidad puede ser:

- *Uno a uno*. Donde una entidad de A se asocia a lo sumo con una entidad de B y viceversa.

- *Uno a varios*. Una entidad A se asocia con varias de B, pero una entidad B solo se puede asociar con una entidad de A.

- *Varios a uno*. Una entidad A se puede asociar con una entidad B, pero una entidad B se puede asociar con varias entidades A.

- *Varios a varios*. Cualquier entidad A se puede asociar con cualquier número de entidades B y viceversa.

Una clave permite identificar un conjunto de atributos suficiente para distinguir las entidades entre si y su designación denota una restricción. Existen claves minimales, a las que se les llama claves candidatas y una clave primaria es una clave candidata que es elegida para representar la entidad.

La estructura lógica general de una base de datos se puede expresar gráficamente (ver figura 1.18) mediante un diagrama E-R que consta de los siguientes componentes:

- *Rectángulos*. Que representan a las entidades
- *Elipses*. Que representan a los atributos
- *Rombos*. Que representan a las relaciones entre conjuntos de entidades
- *Líneas*. Que unen los atributos con los conjuntos de entidades y los conjuntos de entidades con las relaciones
- *Elipses dobles*. Atributos multivalorados
- *Elipses discontinuas*. Atributos derivados
- *Líneas dobles*. Participación total de una entidad en un conjunto de relaciones
- *Rectángulos dobles*. Conjunto de entidades débiles
- La clave primaria se *subraya*
- *Línea dirigida* denota uno
- *Línea no dirigida* denota varios

Cada componente se etiqueta con la entidad o relación que representa.

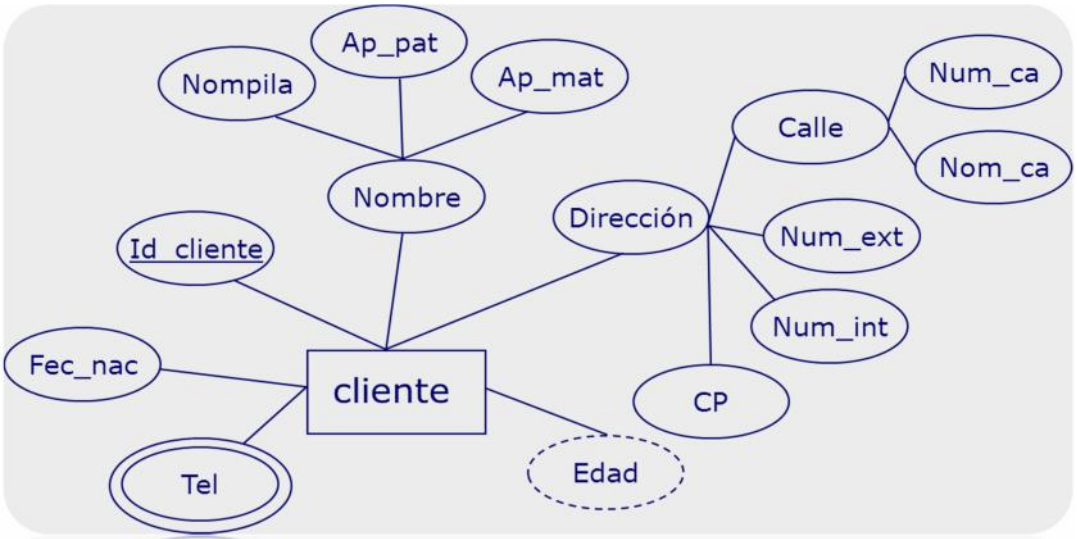

Figura 1.18 Ejemplo de Modelo Entidad-Relación

1.5.3 MODELO RELACIONAL

El modelo relacional es uno de los modelos más usados debido a que utiliza tablas bidimensionales para representar la lógica de sus datos y sus relaciones. Dentro de los objetivos de este modelo son:

- *Independencia física*. La forma de almacenar los datos, no debe de influir en su manipulación lógica.
- *Independencia lógica*. Las aplicaciones que utilizan las bases de datos no deben ser modificadas por que se modifiquen elementos de la base de datos.
- *Flexibilidad*. La base de datos ofrece fácilmente distintas vistas en función de los usuarios y aplicaciones.
- *Uniformidad*. Las estructuras lógicas siempre tienen una única forma conceptual (las tablas).
- *Sencillez*.

Las bases de datos relacionales se basan en el uso de tablas (llamadas también relaciones). Las tablas se representan como una estructura rectangular formada por filas y por columnas. Cada columna almacena información sobre una propiedad determinada de la tabla (Atributo) como el nombre, apellidos, dirección, etc. Cada fila posee una ocurrencia o ejemplar de la instancia o relación representada por la tabla (tuplas) (ver figura 1.19).

Atributo1	Atributo2	……	Atributo n	
Valor 1,1	Valor 1,2	…….	Valor 1,n	Tupla 1
Valor 2,1	Valor 2,2	…….	Valor 2,n	Tupla 2
…….	……	…….	……	…..
Valor m,1	Valor m,2	…….	Valor m,n	Tupla m

Figura 1.19 Relación donde se muestran los atributos y las tuplas

En el caso de las bases de datos relacionales deben cumplir:

- No puede haber dos tuplas iguales.
- El orden de las tuplas no importa.
- El orden de los atributos no importa.
- Cada atributo solo puede tomar un valor en el dominio en el que se encuentra.

Ya que en una relación no hay tuplas repetidas, estas se pueden distinguir unas de otras, es decir se pueden identificar de un modo único. La forma de identificarlas es mediante los valores de sus atributos.

1.5.4 LLAVES PRIMARIAS Y FORÁNEAS

Una *superclave* es un atributo o conjunto de atributos que identifican de manera única a las tuplas de una relación.

Una *clave candidata* es el atributo o conjunto de atributos de la relación R que satisfacen:

- *Unicidad*: No hay dos tuplas en la relación R con el mismo valor de K
- *Irreducibilidad*: Ningún subconjunto de K tiene la propiedad de unicidad, es decir no se pueden eliminar componentes de K sin destruir la unicidad.

La *clave primaria* de una relación es aquella clave candidata que se escoge para identificar sus tuplas de un modo único. Ya que en una relación no hay tuplas repetidas, siempre va a existir una clave candidata, y por lo tanto siempre habrá una clave primaria. En el peor de los casos la clave primaria estará formada por todos los atributos de la relación.

Las claves candidatas que no son escogidas como claves primarias son denominadas *claves alternativas*.

Una *clave foránea* o *ajena* es un atributo o conjunto de atributos de una relación cuyos valores coinciden con los valores de la clave primaria de alguna otra relación (puede ser la misma); las claves foráneas representan relaciones entre datos.

1.6 NORMALIZACIÓN

1.6.1 DEFINICIÓN DE NORMALIZACIÓN

Una vez obtenido el esquema relacional resultante del modelo E-R que representaba la base de datos, normalmente tendremos una buena base de datos. Pero otras veces, debido a fallas en el diseño o a problemas indetectables en la fase del diseño, tendremos un esquema que puede producir una base de datos que tenga estos problemas:

- *Redundancia*. Son los datos que se repiten continua e innecesariamente por las tablas de las bases de datos.
- *Ambigüedades*. Son los datos que no identifica suficientemente el registro al que representan.
- *Anomalías en operaciones de modificación de datos*. El echo de que al insertar un solo elemento haya que repetir tuplas en una tabla para variar unos pocos datos o que al eliminar un elemento suponga eliminar varias tuplas.

El principio fundamental es que las tablas deben de referirse a objetos o situaciones muy concretas, pero en ocasiones conceptualmente es difícil obtener ese problema y la solución suele ser dividir la tabla en otras tablas mas adecuadas.

La normalización es el proceso de organizar bases de datos, donde se incluye la creación de tablas y las relaciones entre ellas. Estas relaciones se establecen mediante reglas con la finalidad de proteger los datos así como para hacer más flexible la eliminación de la redundancia y las dependencias incoherentes de las tablas recién creadas o de las existentes en la base de datos.

Básicamente son reglas progresivas para normalizar una base de datos y son cinco. A cada una de ellas se le denomina "*forma normal*".

1.6.2 FORMAS NORMALES

Una *dependencia funcional* es una conexión entre uno o más atributos. Las dependencias funcionales del sistema se escriben utilizando una flecha, de la siguiente manera:

FechaDeNacimiento $\longrightarrow$ Edad

En este caso a FechaDeNacimiento se le conoce como determinante. Se puede leer de dos formas FechaDeNacimiento determina a Edad o Edad es funcionalmente dependiente de FechaDeNacimiento.

Existen 3 propiedades de las dependencias funcionales o axiomas de Armstrong:

Dependencia funcional Reflexiva. Si X esta incluido en X, entonces X $\longrightarrow$ X, a partir de cualquier atributo o conjunto de atributos siempre puede deducirse el mismo, por ejemplo, si en una clave de identificación están incluidos el nombre o la dirección de una persona, entonces en esa clave podemos determinar el nombre o la dirección.

Dependencia funcional aumentativa. Si en una clave de identificación, se determina el nombre de una persona, entonces la clave mas la dirección también se determina el nombre o su dirección.

x->Y entonces xz->yz

Clave->Nombre

Clave. Dirección->nombre. Dirección

Dependencia funcional transitiva. Si X, Y, Z son tres atributos de la misma entidad. Si Y depende funcionalmente de X y Z de Y, pero X no depende funcionalmente de Y, se dice entonces que Z depende transitivamente de X, es decir:

X->Y->Z entonces X->Z

FechaNacimiento->Edad

Edad->Conducir

FechaNacimiento->Edad->Conducir

Entonces tenemos que FechaNacimiento determina a Edad y la Edad determina si se es apto para conducir, por lo que indirectamente podemos deducir a través de FechaNacimiento si se puede Conducir.

PRIMERA FORMA NORMAL (1FN)

Una tabla esta en primera forma normal si todos los atributos son atómicos, es decir que los elementos de un dominio son indivisibles o mínimos. Esto significa que:

- La tabla contiene una clave primaria única.
- La clave primaria no contiene atributos nulos.
- No debe existir variación en el número de columnas.
- Debe existir una independencia del orden tanto de las filas como de las columnas.
- Una tabla no puede tener múltiples valores en cada columna.

Esta forma normal elimina los valores repetidos dentro de una BD.

SEGUNDA FORMA NORMAL (2FN)

Una relación esta en 2FN si esta en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal, es decir que no existen dependencias parciales (todos los atributos que no son clave principal deben depender únicamente de la clave principal).

La 2FN se aplica a las relaciones que tienen claves primarias compuestas por dos o más atributos. Si una relación esta en 1FN y su clave primaria es simple (con un solo atributo). Entonces también esta en 2FN.

Para pasar una relación de 1FN a 2FN hay que eliminar las dependencias parciales de la clave primaria. Para ello, se eliminan los atributos que son funcionalmente dependientes y se ponen en una nueva relación con una copia de su determinante (los atributos de la clave primaria de los que dependen).

TERCERA FORMA NORMAL (3FN)

Una relación esta en tercera forma normal si, y solo si, esta en 2FN y, además, cada a tributo no primo depende transitivamente de la clave primaria. La dependencia x->z existe, si existen las dependencias x->y, y->z, siendo x, y atributos o conjuntos de atributos de una misma relación.

Aunque las relaciones en 2FN tienen menos redundancias que las relaciones en 1FN, todavía pueden sufrir anomalías frente a las actualizaciones. Para pasar una relación de 2FN a 3FN hay que eliminar las dependencias transitivas. Para ello, se eliminan los atributos que dependen transitivamente y se ponen en una nueva relación con una copia de su determinante (el atributo o atributos no clave de los que dependen).

1.7 ARQUITECTURA CLIENTE – SERVIDOR

1.7.1 INTRODUCCIÓN

La tecnología Cliente-Servidor, es un modelo que permite la distribución de la información de una forma ágil y eficaz, que con el uso de las tecnologías de la información, se ha desarrollado enormemente.

Un elemento básico e imprescindible en el modelo Cliente-Servidor, es el uso de las bases de datos, por lo que es necesario el conocer una arquitectura que en este momento es una de las más importantes para enviar y recibir información. También es una herramienta potente para guardar los datos en una base de datos como un servidor.

La arquitectura Cliente-Servidor agrupa conjuntos de elementos que efectúan procesos distribuidos y computo cooperativo.

En esta arquitectura la computadora de los usuarios (clientes), produce una demanda de información a cualquiera de las computadoras que proporcionan la información, conocidas como servidores que responden a las demandas del cliente.

Los clientes y servidores realizan funciones distintas a través de una red local o una red amplia o mundial como Internet (ver tabla 1.20).

FUNCIONES DEL CLIENTE	FUNCIONES DEL SERVIDOR
Mantener y procesar el dialogo con el usuario	Acceso, almacenamiento y organización de datos
Manejo de pantallas	Actualización de datos almacenados
Menús e interpretación de comandos	Administración de recursos compartidos
Entradas de datos y validación	Ejecución de toda lógica para procesar una transacción
Procesamiento de ayudas	Procesamiento común de elementos del servidor (datos, capacidad de CPU, almacenamiento en disco, capacidad de impresión, manejo de memoria y comunicación

Tabla 1.20 Funciones del cliente y el servidor

1.7.2 PHP

PHP es el heredero de un producto anterior llamado PHP/F1, creado por Rasmus Lerdorf en 1995, inició como un simple conjunto de scripts de Perl para controlar accesos a su trabajo online. A ese conjunto de scripts los llamó 'Personal Home Page Tools' y según se requería más funcionalidad Rasmus fue escribiendo una implementación en C, capaz de comunicarse con bases de datos.

PHP permite embeber pequeños fragmentos de código dentro de la pagina HTML y realizar acciones de una forma fácil y eficaz sin tener que generar programas programados íntegramente en un lenguaje distinto al HTML; además PHP ofrece un sin fin de funciones para la explotación de bases de datos de una manera llana y sin complicaciones.

1.7.3 MySQL

MySQL es un DBMS relacional rápido y fácil de usar, usado en bases de datos de muchos sitios de Internet. Desde un principio, la velocidad fue el atributo más importante para los desarrolladores y aunque MySQL tiene menos características que sus competidores comerciales, tiene todas las características necesarias para la gran mayoría de desarrolladores de bases de datos. Es más fácil de instalar y usar que sus competidores comerciales, esto a demás de la diferencia en el precio que favorece mucho a MySQL.

Se ofrecen dos tipos de licencias, la libre o Licencia Publica General (GPL) sin ningún costo, la cual se puede usar si tiene una base de datos en un sitio web e incluso si es esta ganando dinero con su sitio y la licencia comercial para aquellos que prefieren una alternativa a la GPL, la cual se debe de usar si un desarrollador desea usar MySQL como parte de un nuevo producto de software y desea vender el nuevo producto en vez de ponerlo a disposición bajo GPL.

1.7.4 HTML

HTML es un lenguaje utilizado para desarrollar páginas y documentos web. A diferencia de los lenguajes convencionales, HTML utiliza una serie de etiquetas especiales intercaladas en un documento de texto sin formato. Dichas etiquetas serán posteriormente interpretadas por los exploradores encargados de visualizar la pagina o el documento web con el fin de establecer el formato.

Dentro del lenguaje HTML una parte muy usada son los formularios, los cuales permiten crear interfaces graficas de usuario dentro de una página web. Los componentes de estas interfaces, denominados también controles, serán cajas de texto, cajas para clave de acceso, botones de pulsación, botones de opción, casillas de verificación, menús, tablas, listas desplegables, etc.

CAPITULO 2: ANÁLISIS Y DISEÑO (ORIENTADO A OBJETOS)

2.1. INTRODUCCIÓN

En esta sección se va a presentar el diseño del prototipo del sistema, así como las bases de datos relacionales que se desarrollaron. Se realiza una presentación con la forma en como funciona el sistema para posteriormente presentar el diseño en UML y la descripción general de los módulos, que hacen que funcione el sistema, concluyendo con el bosquejo de la base de datos del sistema.

2.2 PLANTEAMIENTO DEL PROBLEMA

Hoy en día es común que una persona de la ciudad cuente con una computadora personal en casa, otra en la oficina, una portátil, un teléfono inteligente, además de tener acceso a diferentes computadoras publicas a través de los conocidos cafés Internet. Este entramado complica la sincronización de aplicaciones y datos entre los diferentes dispositivos, ocasionando problemas de redundancia y la inexactitud de la información en cada uno de los dispositivos

Aprovechando los avances de infraestructura en las telecomunicaciones, los nuevos modos de programación, el Internet y los nuevos modelos en su uso, se han creado nuevas soluciones a estos problemas.

La computación en la nube es una nueva tendencia que le permitirá al usuario tener disponible sus programas o archivos sin tener que almacenarlos en su disco duro, ya que estos se almacenarían en un servidor de un centro de datos, en cualquier parte del mundo; lo que le permitirá acceder a este desde cualquier computadora o dispositivo y desde cualquier sitio donde se tenga acceso a Internet. Todo esto sin que los usuarios tengan que tener ningún conocimiento acerca de la estructura que opera para ofrecer los servicios.

Un ejemplo de estos servicios son las páginas de correo electrónico, Google, FaceBook, Wikipedia o la Banca en línea, ya que son aplicaciones web las cuales accedemos como servicios.

En cualquier momento anotar algo como una idea o un recordatorio es importante y hoy en día esto generalmente se realiza mediante los llamados post-tick o simplemente mediante alguna hoja de papel; pero existe el inconveniente de que se pueden llegan a traspapelar, mojarse o simplemente perderse por alguna razón. Esto además de que no existe un orden y de que no siempre se puede andar cargando con todo el paquete de notas, lo que lo hace poco funcional este sistema. Por otro lado en las computadoras existe software que permite almacenar notas y clasificarlas, pero existe el inconveniente de que la información solo es accesible siempre y cuando se cuente con la computadora que las almacena y un daño físico o lógico en el disco duro puede ocasionar que se pierda nuestra información.

Por lo que se requiere de una base de datos que permita a los usuarios generar notas en la nube, para tener acceso a ellas en cualquier momento. Cuando se genere una nota, esta deberá de contener la fecha de creación, titulo, contenido, clasificación y en su caso, según lo requiera el usuario, puede tener algún archivo adjunto, donde además del archivo en cuestión se deberá de almacenar el tamaño del archivo, el tipo de archivo y el nombre del archivo.

El sistema deberá de manejar las operaciones de creación, eliminación, modificación y listado, además de contar con una opción de contacto con el administrador.

El acceso al sistema deberá ser controlado por un nombre de usuario y una contraseña, por lo que para cada usuario le tiene que corresponder una clave, la cual debe ser asignada con un registro previo, donde se tiene que solicitar el nombre completo y correo electrónico, además de registrar la fecha en que se dio de alta al usuario.

2.3. OBJETIVO GENERAL

Diseñar y elaborar un sistema de Bases de datos en la nube para tomar apuntes de varios temas y almacenarlos en una base de datos. La idea es que desde cualquier sitio y desde cualquier dispositivo que cuente con un navegador y conexión a Internet se puedan tomar notas, almacenarlas y consultarlas en cualquier momento.

2.4 REQUERIMIENTOS DEL SISTEMA

Se debe realizar un sistema de base de datos que permita almacenar en la nube notas o apuntes de varios temas y almacenarlos para su posterior consulta. La idea es que desde cualquier parte y con cualquier dispositivo se pueda realizar la toma de notas, consultar, eliminar, editar y generar reportes de las notas.

2.4.1 REQUERIMIENTOS FUNCIONALES

Se ha adoptado la Web como el entorno donde se alojara el sistema y donde el usuario interactuara con el mismo, esto debido a las ventajas que ofrece de accesibilidad y portabilidad.

La interfaz Web es el punto de acceso a la herramienta o subsistemas asociados. El sistema cuenta con las siguientes interfaces:

- Registro. Esta interfaz permite que una persona se pueda registrar para que pueda tener posterior acceso al sistema.
- Ingreso. Donde el usuario puede ingresar su usuario y contraseña para poder ingresar al sistema.
- Crear nota. Aquí el usuario podrá ingresar los datos necesarios para crear su nota, asi mismo en esta parte se podrá guardar la nota si el usuario lo desea.
- Modificar nota. El usuario podrá modificar una nota guardada anteriormente.
- Eliminar nota. Se podrá eliminar una nota de forma permanente.
- Buscar nota. Interfaz para realiza la búsquedas de notas almacenadas anteriormente.
- Contacto. Sección de contacto con el administrador.

2.4.2 REQUERIMIENTOS NO FUNCIONALES

- Debido a que es un sistema Web es necesario contar con un navegador y una conexión a internet para poder visualizar el sistema.
- Es necesario contar con una cuenta de correo electrónico para poder completar la sección de registro.

- En la sección de crear nota se pueden adjuntar archivos de un máximo de 100Kb.

2.5 ALCANCES

El presente proyecto solo se limitara a las funciones básicas del sistema tales como:

- Creación, almacenamiento y clasificación de una nota en una base de datos.
- Edición de una nota almacenada.
- Eliminar una nota almacenada.
- Adjuntar un archivo a una nueva nota.
- Sección de contacto para soporte.

Por lo que alguna otra función adicional estará fuera del alcance de este documento.

2.6 METODOLOGÍA

La metodología empleada para el presente proyecto es el de cascada, debido a que se requiere de una metodología que nos permita mantener un orden en la realización de las diferentes etapas del proyecto. Por otra parte también es un modelo apropiado debido a que no se requiere un trabajo en equipo.

2.7 ANÁLISIS DEL SISTEMA

2.7.1 GLOSARIO DE TÉRMINOS

- *NOMBRE: CLAVE*
  Definición de trabajo: Elemento del sistema que nos va a permitir manejar los usuarios y contraseñas de los usuarios.
- *NOMBRE: NOTA*
  Definición de trabajo: Elemento donde se va a almacenar la información que escriba el usuario e información de identificación de la misma.
- *NOMBRE: USUARIO*
  Definición de trabajo: Elemento donde se van a almacenar los datos personales de los usuarios al momento de realizar el registro.
- *NOMBRE: ARCHIVO*
  Definición de trabajo: Elemento opcional que se va a poder adjuntar a la nota y que va a consistir en archivos de diferentes tipos.

2.7.2 DIAGRAMA DE CASOS DE USO

Los diagramas de casos de uso nos van a permitir capturar la información de como trabaja el sistema o de como se desea que trabaje.

De acuerdo al planteamiento del problema el sistema se cuenta con los siguientes actores:

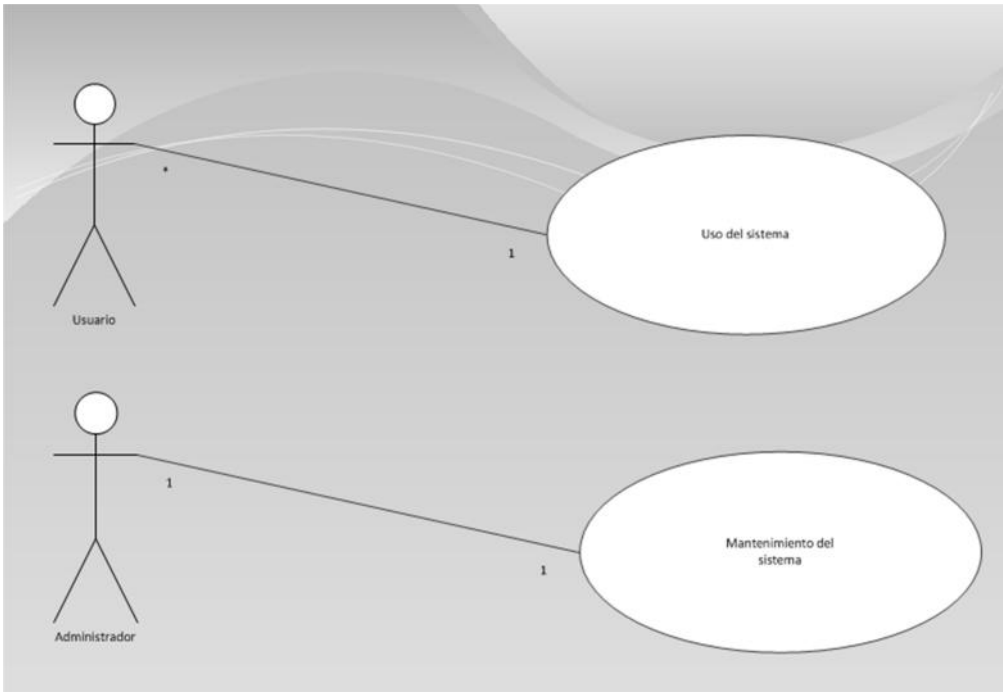

Figura 2.1 Modelo casos de uso de los actores

El actor usuario va a ser la persona que va a hacer uso del sistema y cuya función va a ser el de crear, visualizar, eliminar o editar una nota.

El actor administrador va a ser la persona que va a llevar el control de la base de datos

Para el caso de nuestro sistema solo se va a estudiar los casos de uso relacionados de la interacción del usuario con el uso del sistema.

De acuerdo al planteamiento del problema se encuentran los siguientes casos de uso que nos indican todas las funciones elementales de una base de datos, así como de otras tales como el registro.

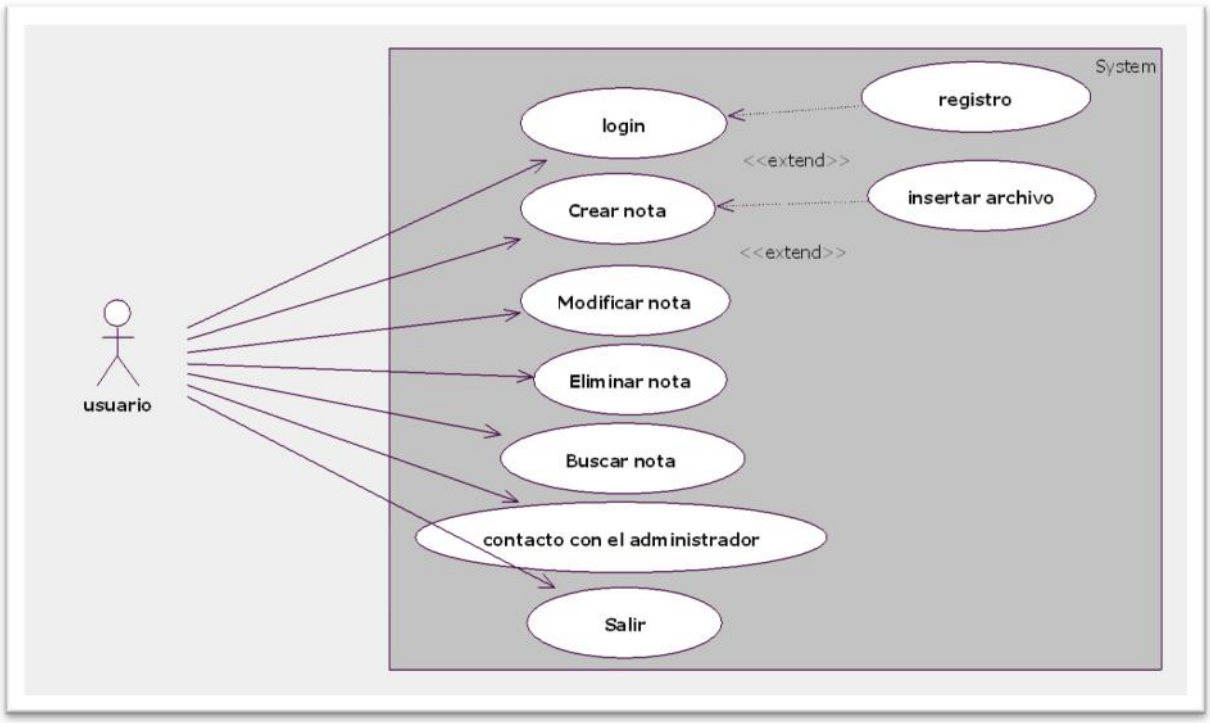

Figura 2.2 Diagrama general de casos de uso

Es esta figura se muestra un panorama general del funcionamiento de nuestro sistema, lo que nos permitirá tener una idea clara de las funciones que se llevan a cabo.

2.7.3 ESPECIFICACIÓN DE CASOS DE USO

En esta sección se muestra una descripción breve de cada uno de los casos de uso (esto se observa en las tablas 2.3 a 2.11), mostrando la secuencia de los procesos que se llevan a cabo en cada uno de los casos de uso, así de como de las excepciones que se pueden llegar a presentar

C1:REGISTRO		
Descripción	Recaba datos personales del usuario	
Precondición	Ninguna	
Secuencia	1	El usuario selecciona la opción de REGISTRO
	2	El sistema muestra una pantalla con un formulario
	3	El usuario completa el formulario, llena el campo del CAPTCHA y selecciona la opción de REGISTRAR
	4	El sistema valida el registro y guarda la información
Pos condición	Usuario con claves de Usuario/contraseña	
Excepciones	E1	La clave usuario no esta disponible
	E2	El usuario no completa todos los campos requeridos
	E3	Correo electrónico registrado anteriormente
	E4	Las contraseña y confirmación de contraseña no coincide
	E5	CAPTCHA incorrecto

Tabla 2.3 Especificación caso de uso Registro

C2: LOGIN		
Descripción	Se validan los datos de usuario/contraseña del usuario	
Precondición	El usuario debe de haber realizado con anterioridad el C1: REGISTRO	
Secuencia	1	El usuario ingresa sus datos de Usuario/Contraseña al sistema y selecciona INGRESAR
	2	El sistema valida los datos ingresados, permitiendo o denegado el acceso al sistema
Pos condición	Usuario con acceso al sistema	
Excepciones	E1	Datos Usuario / contraseña incorrectos
	E2	Usuario no ingresa datos y selecciona INGRESAR

Tabla 2.4 Especificación caso de uso Login

C3: CREAR NOTA		
Descripción	Crea una nota nueva en la Base de Datos	
Precondición		
Secuencia	1	El usuario elige la opción de CREAR
	2	El sistema mostrara una pantalla para escribir la nota deseada
	3	El usuario agrega la nota deseada y termina seleccionando la opción de GUARDAR
	4	El sistema clasificara y dará de alta la nota en la Base de Datos
Pos condición	Una nueva nota en la Base de Datos	
Excepciones	1	El usuario seleccionara la opción de INSERTAR ARCHIVO
	2	El usuario no ingresa nota y selecciona GUARDAR

Tabla 2.5 Especificación caso de uso Crear Nota

C4:INSERTAR ARCHIVO		
Descripción	Adjunta un archivo a la nota	
Precondición		
Secuencia	1	El sistema muestra una pantalla para elegir el archivo
	2	El usuario elige el archivo y selecciona ADJUNTAR
Pos condición	Archivo adjunto a la nota	
Excepciones	1	Se rebaso la capacidad máxima para adjuntar

Tabla 2.6 Especificación caso de uso Insertar Archivo

C5: MODIFICAR NOTA		
Descripción	Cambia el contenido de una nota	
Precondición	Una nota ya existente	
Secuencia	1	El usuario elige la opción de MODIFICAR
	2	El sistema muestra la pantalla donde selecciona el archivo a editar
	3	El usuario selecciona la nota y elige EDITAR
	4	El sistema muestra en pantalla la nota a editar
	5	El usuario modifica la nota y selecciona la opción de GRABAR
	6	El sistema guarda los cambios en la Base de Datos
Pos condición	Nota modificada	
Excepciones	1	El usuario seleccionara la opción de ADJUNTAR ARCHIVO

Tabla 2.7 Especificación caso de uso Modificar nota

C6: ELIMINAR NOTA		
Descripción	Elimina una nota ya existente	
Precondición	Una nota ya existente	
Secuencia	1	El usuario elige la opción de eliminar una nota
	2	El sistema muestra un listado de las notas alamcenadas
	3	El usuario elige la nota y selecciona ELIMINAR
	4	El sistema muestra una pantalla de confirmación
	5	El usuario confirma la eliminación
	6	El sistema elimina la nota de la Base de Datos
Pos condición	Nota eliminada	
Excepciones	1	No hay notas en la Base de datos
	2	El usuario elige NO cuando se le solicita confirmar la eliminación

Tabla 2.8 Especificación caso de uso Eliminar Nota

C7: BUSCAR NOTA		
Descripción	Muestra un listado de las notas guardadas	
Precondición	Notas existentes en la base de datos	
Secuencia	1	El usuario elige BUSCAR
	2	El sistema muestra un listado de las notas guardadas
excepciones	1	No hay notas en la Base de datos

Tabla 2.9 Especificación caso de uso Buscar Nota

C8: CONTACTO		
Descripción	Se envían mensajes por correo electrónico al administrador	
Precondición		
Secuencia	1	El usuario elige CONTACTO
	2	El sistema muestra un formulario
	3	El usuario llena el formulario y da clic en ENVIAR
excepciones		

Tabla 2.10 Especificación caso de uso Contacto

C9: SALIR		
Descripción	Permite la salida segura del sistema	
Precondición	Contar con sesión iniciada	
Secuencia	1	El usuario elige SALIR
	2	El sistema termina la sesión y cierra la base de datos
excepciones		

Tabla 2.11 Especificación caso de uso Salir

2.7.4 ESCENARIOS

A continuación se detallan los escenarios (ver tablas 2.12 a 2.20) que van a ser como un ejemplo de funcionamiento del sistema, lo que nos permitirá obtener mas detalles, tales como los tipos de datos a usar, además de que muestran como va a ser la interacción del usuario con el sistema.

C1: REGISTRO
ESCENARIO PRINCIPAL
• El usuario Fabiola ingresa a la página del sistema, donde selecciona la opción de registro para obtener sus claves de acceso. • El sistema le muestra un formulario, donde le solicita NOMBRE, APELLIDOS, NOMBRE DE USUARIO, CONTRASEÑA, CONFIRMACION DE CONTRASEÑA, CORREO ELECTRÓNICO e ingresar IMAGEN CAPTCHA. • Fabiola ingresa en el campo NOMBRE: Fabiola, APELLIDO PATERNO: Alvarado, APELLIDO MATERNO: Quecholac, CORREO ELECTRONICO: hola1_23@hotmail.com, NOMBRE DE USUARIO: falvarado, CONTRASEÑA: ******, ingresa la imagen CAPTCHA y selecciona la opción de ENVIAR. • El sistema le envía una pantalla donde le confirma el registro
ESCENARIO SECUNDARIO
• El usuario Fabiola ingresa a la página del sistema, donde selecciona la opción de registro para obtener sus claves de acceso. • El sistema le muestra un formulario, donde le solicita NOMBRE, APELLIDOS, NOMBRE DE USUARIO, CONTRASEÑA, CONFIRMACION DE CONTRASEÑA, CORREO ELECTRÓNICO e ingresar IMAGEN CAPTCHA. • Fabiola ingresa en el campo NOMBRE: Fabiola, APELLIDO PATERNO: Alvarado, APELLIDO MATERNO: Quecholac, CORREO ELECTRONICO: hola1_23@hotmail.com, NOMBRE DE USUARIO: falvarado, CONTRASEÑA: ******, ingresa la imagen CAPTCHA y selecciona la opción de ENVIAR. • El sistema le envía una pantalla donde le indica “Nombre de usuario ocupado” y la regresa a la pantalla del formulario • Brenda introduce en NOMBRE DE USUARIO: falvarado1 y CONTRASEÑA: ******, ingresa la imagen CAPTCHA y selecciona la opción de ENVIAR. • El sistema le envía una pantalla de confirmación de registro.

Tabla 2.12 Escenarios del caso de uso Registro

C2: LOGIN
ESCENARIO PRINCIPAL
• Brenda ingresa a la página del sistema, ingresa su NOMBRE DE USUARIO: falvarado y CONTRASEÑA: ****** y selecciona la opción de Enviar. • El sistema verifica los datos introducidos y permite el ingreso al sistema.
ESCENARIO SECUNDARIO
• Brenda ingresa a la página del sistema y selecciona la opción de Enviar • El sistema muestra pantalla de campos vacíos y le solicita introducir nombre de usuario y contraseña

Tabla 2.13 Escenarios del caso de uso Login

C3: CREAR NOTA
ESCENARIO PRINCIPAL
• El usuario Fabiola selecciona la opción de CREAR • El sistema le muestra una pantalla donde se le solicita ingresar el titulo, contenido y clasificación de la nota • Fabiola agrega al TITULO “mi primer nota”, en CONTENIDO “Hola Mundo” selecciona la clasificación ➢ IDEAS ESPONTANEAS ➢ RECORDATORIOS ➢ NOTAS PARA AGENDA ➢ IDEAS PARA TURISTEAR ➢ TAREAS ➢ CUMPLEAÑOS ➢ SIN CLASIFICAR Y selecciona GRABAR. • El sistema almacena la nota y le confirma al usuario el almacenamiento.
ESCENARIO SECUNDARIO
• El usuario Fabiola selecciona la opción de CREAR • El sistema le muestra una pantalla donde se le solicita ingresar el titulo, contenido y clasificación de la nota • Fabiola agrega al TITULO “mi primer nota”, en CONTENIDO “Hola Mundo” selecciona la clasificación ➢ IDEAS ESPONTANEAS ➢ RECORDATORIOS ➢ NOTAS PARA AGENDA ➢ IDEAS PARA TURISTEAR ➢ TAREAS ➢ CUMPLEAÑOS ➢ SIN CLASIFICAR Y selecciona la opción de INSERTAR ARCHIVO • El sistema ejecuta el C8: INSERTAR ARCHIVO • Fabiola selecciona la opción de GRABAR • El sistema almacena la nota y le confirma al usuario el almacenamiento.

Tabla 2.14 Escenarios del caso de uso Crear Nota

C4: INSERTAR ARCHIVO
ESCENARIO PRINCIPAL
• El sistema le muestra al usuario la pantalla donde podrá seleccionar el archivo a adjuntar • Fabiola selecciona un archivo denominado prueba.txt y selecciona Abrir • El sistema relaciona el archivo prueba.txt a la nota y regresa al C3: CREAR NOTA
ESCENARIO SECUNDARIO
• El sistema le muestra al usuario la pantalla donde podrá seleccionar el archivo a adjuntar • Fabiola adjunta un archivo denominado prueba2.ppt de 600KB y le da clic en abrir • El sistema le muestra un mensaje de capacidad máxima rebasada.

Tabla 2.15 Escenarios del caso de uso Insertar Archivo

C5: MODIFICAR NOTA
ESCENARIO PRINCIPAL
• El sistema le muestra al usuario un listado de las notas almacenadas • Fabiola selecciona la nota “mi primer nota” • El sistema carga la nota con el titulo y contenido mostrándose en pantalla • Fabiola cambia el contenido de “Hola mundo” por “Hola mundo maravilloso” y da clic en Grabar • El sistema le muestra un mensaje donde le indica de los cambios guardados

Tabla 2.16 Escenario del caso de uso Modificar Nota

C6: ELIMINAR NOTA
ESCENARIO PRINCIPAL
• El sistema muestra una lista de las notas almacenadas • Fabiola da clic en eliminar en la nota deseada • El sistema muestra un mensaje de confirmación • Fabiola da clic en continuar • El sistema elimina la nota

Tabla 2.17 Escenario del caso de uso Eliminar Nota

C7: BUSCAR NOTA
ESCENARIO PRINCIPAL
• Fabiola da clic en la opción de Buscar • El sistema le muestra en pantalla un listado con las notas almacenadas

Tabla 2.18 Escenario del caso de uso Buscar nota

C8: CONTACTO
ESCENARIO PRINCIPAL
• Fabiola da clic en la opción de Contacto • El sistema le muestra una pantalla donde tiene que ingresar su nombre, correo y el contenido del mensaje a enviar • Fabiola ingresa en nombre: Fabiola, correo: hola1_23@hotmail.com y en contenido: mensaje de prueba y da clic en la opción de Enviar • El sistema envía el mensaje

Tabla 2.19 Escenario del caso de uso Contacto

C9: SALIR
ESCENARIO PRINCIPAL
• Fabiola selecciona la opción de salir • El sistema cierra la sesión y regresa a la pantalla de Login

Tabla 2.20 Escenario del caso de uso Salir

2.7.5 DIAGRAMA DE CLASES

En la figura 2.12 se muestra el diagrama de clases, donde nos muestra las clases involucradas en el sistema de notas, así como las operaciones que realizan.

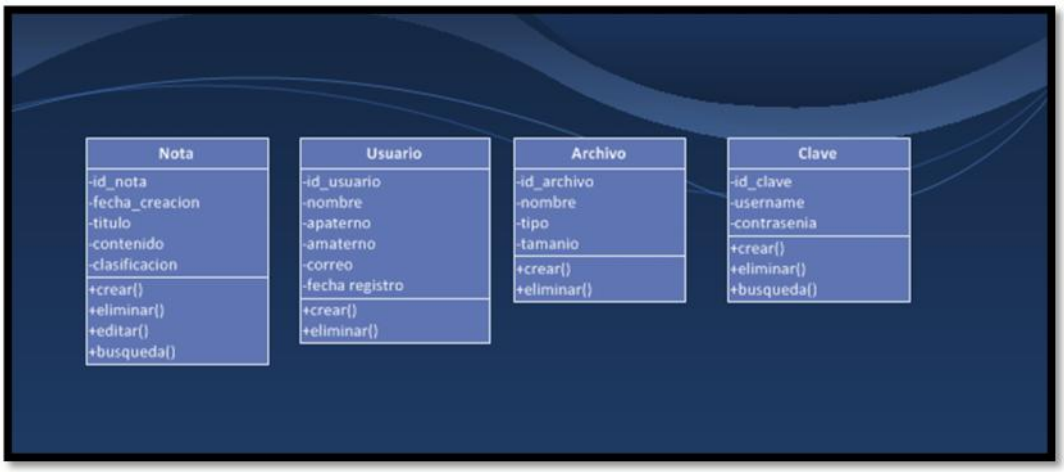

Figura 2.21 Diagrama de clases del sistema

2.8 DISEÑO DEL SISTEMA

2.8.1 DIAGRAMA DE CLASES DE DISEÑO

Para el caso de uso REGISTRO (ver figura 2.22) se va a mostrar un formulario, el cual se va a llenar y posteriormente validado por la clase de control ValidaFormulario, para que no existan nombres de usuario o contraseñas duplicadas, además de otras validaciones y posteriormente almacenados los datos en las entidades Usuario y Clave.

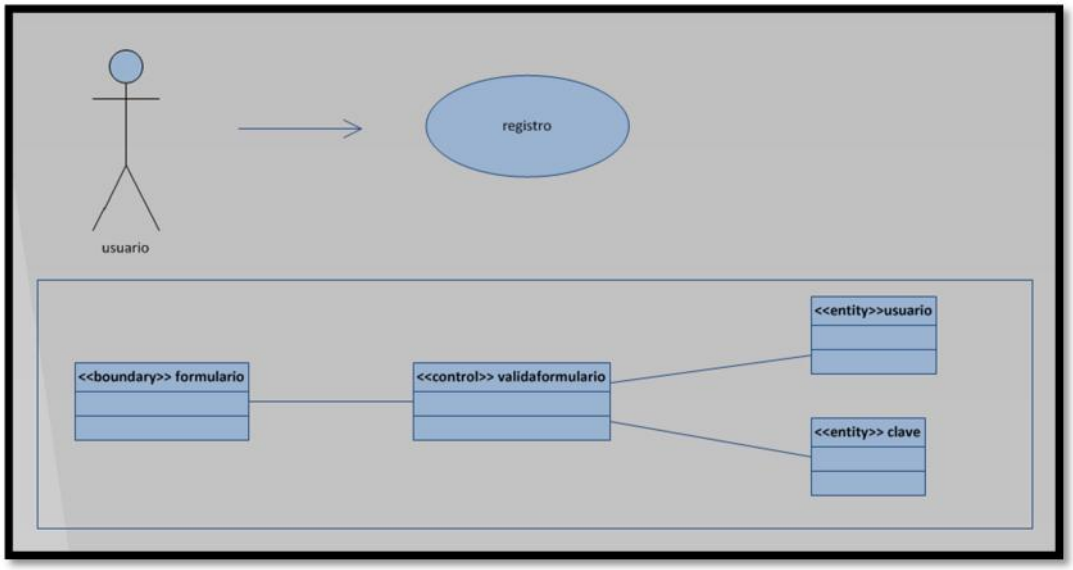

Figura 2.22 Diagrama de clases esteoripadas del caso de uso Registro

Para el caso de uso Login se van a involucrar tres clases (ver figura 2.23), una boundary que nos va a permitir captar las claves del usuario, otra de control que nos va a permitir validar las claves y una de entidad.

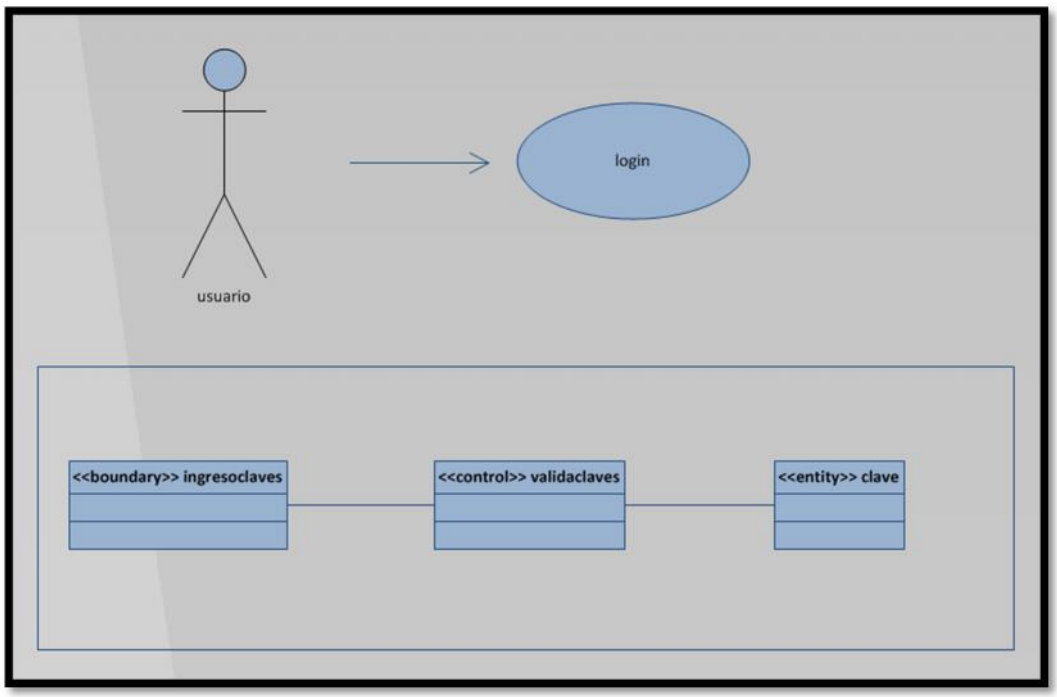

Figura 2.23 Diagrama de clases esteoripadas del caso de uso Login

En este diagrama correspondiente al caso de uso Crear Nota (ver figura 2.24), involucra a la entidad <<boundary >> IngresarNota que va a ser un formulario que permitirá la captación de la

nota, la entidad <<control>> Almacenamiento que va a ser la clase que nos va a permitir el almacenamiento de la nota.

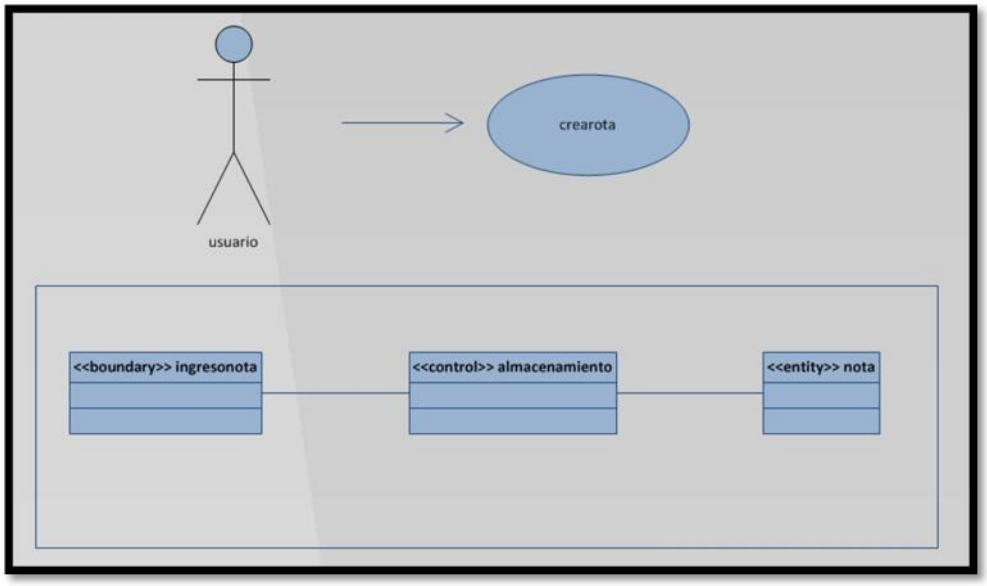

Figura 2.24 Diagrama de clases esteoripadas del caso de uso Crear Nota

En el diagrama para el caso de uso InsertarArchivo (ver figura 2.25), las clases involucradas son un <<boundary>> que le va a permitir al usuario seleccionar un archivo, una clase <<control>> que va a relacionar la nota con el archivo adjuntado y la clase entidad que va a contener el archivo en cuestión.

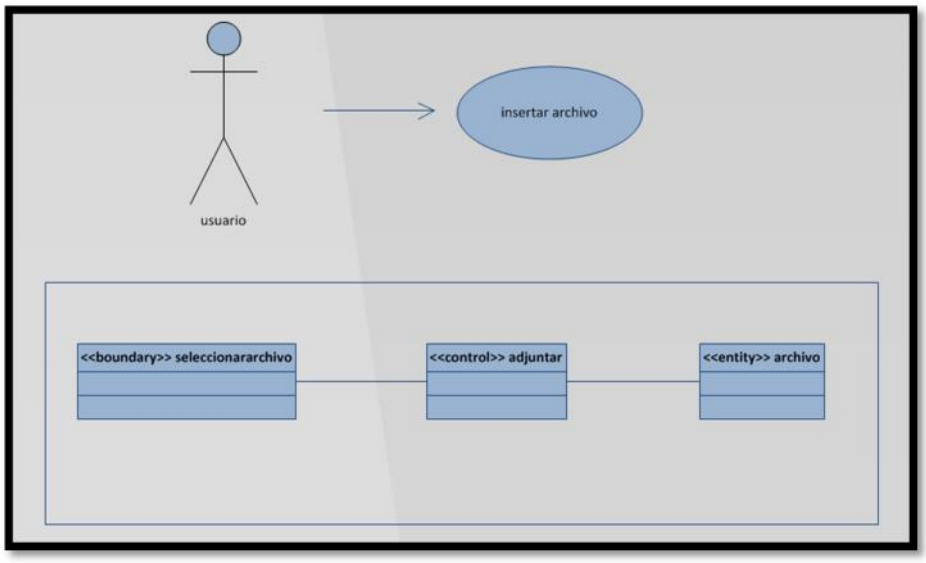

Figura 2.25 Diagrama de clases esteoripadas para insertar archivo

En el diagrama de editar nota (ver figura 2.26) se muestran las clases que participan en este caso de uso, donde participan dos entidades <<boundary>>, uno para seleccionar la nota a editar y otro donde se edita la nota en cuestión, así mismo dos clases <<control>> uno que nos permitirá cargar la nota seleccionada y otro que nos permitirá guardar los cambios después de la edición y por ultimo se usara una clase <<entity>> que será de la clase nota.

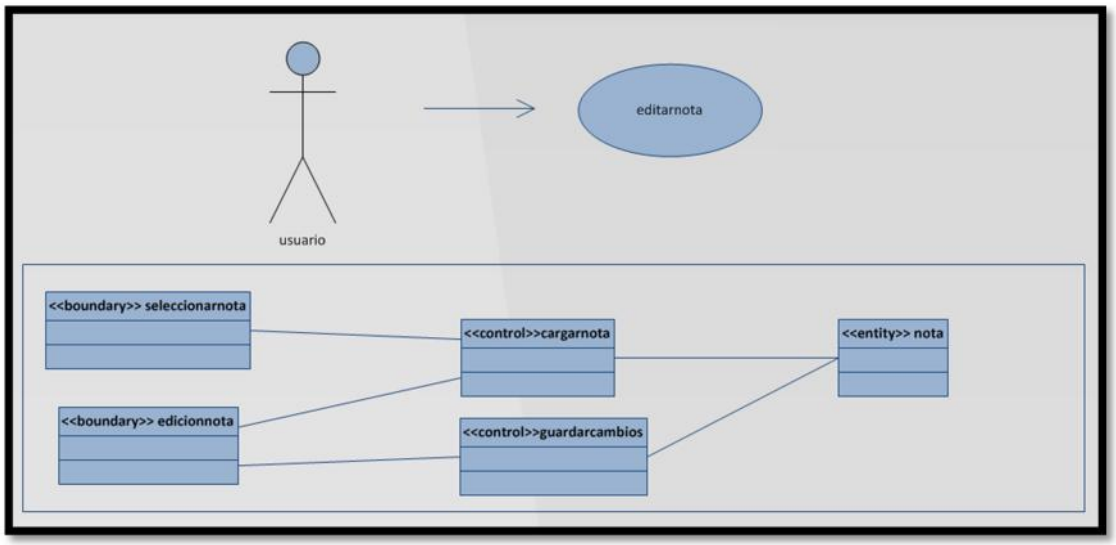

Figura 2.26 Diagrama de clases esteoripadas del caso de uso Editar Nota

En el diagrama de clases de eliminar nota (ver figura 2.27), se observan las clases involucradas; las cuales son seleccionarNota, que es una <<boundary>>, la cual nos permitirá seleccionar una nota para eliminarla, una clase <<control>> que nos permitirá eliminar la nota de la base de datos y una clase <<entity>> Nota, que será la nota en cuestion.

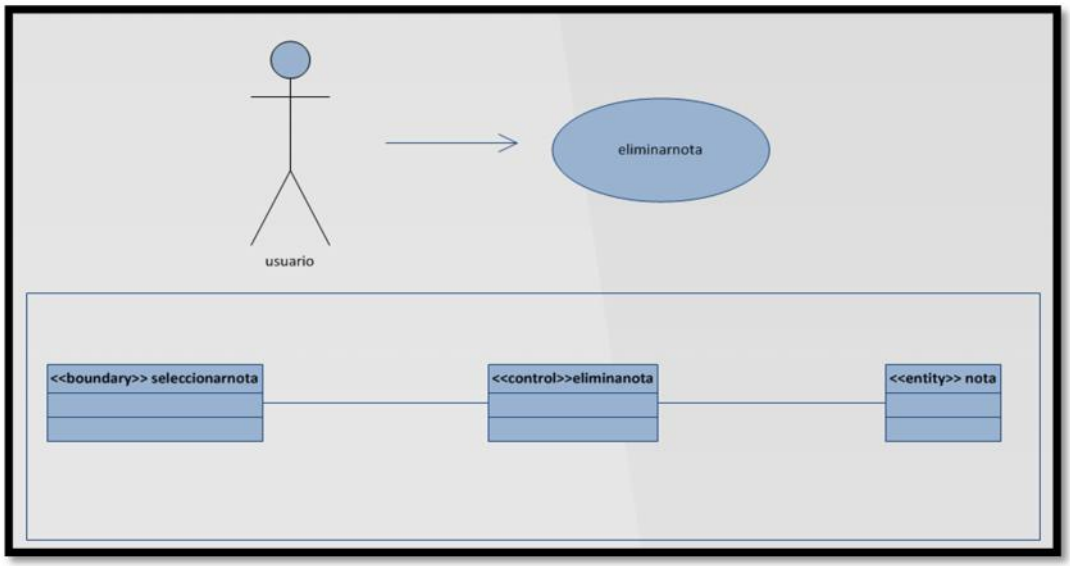

Figura 2.27 Diagrama de clases esteoripadas del caso de uso Eliminar Nota

En el diagrama de la figura 2.28, se muestran las clases involucradas para realizar la búsqueda de notas, las cuales será una <<boundary>> que será seleccionarbuscar, la que nos permitirá seleccionar la búsqueda, una clase <<control>> que nos realizara la búsqueda y una clases <<entity>> que será la entidad nota.

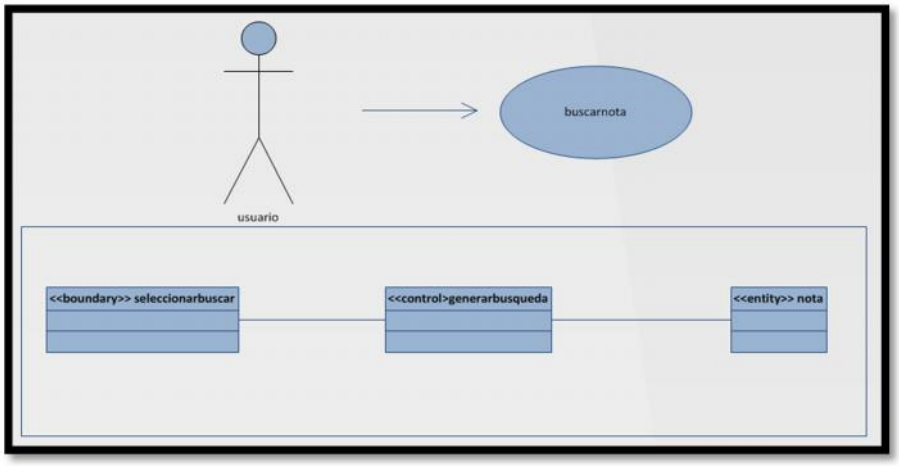

Figura 2.28 Diagrama de clases esteoripadas del caso de uso Buscar

Para el caso de uso de contacto, se muestra el diagrama de clases (ver figura 2.29), donde las clases involucradas son un <<boundary>> que será el formulario a llenar por parte del usuario, una clase <<control>> que será la clase encargada del formulario a correo del administrador y una clase <<entity>> usuario, debido a que de esta entidad se extraerán los datos del usuario como es su correo y nombre.

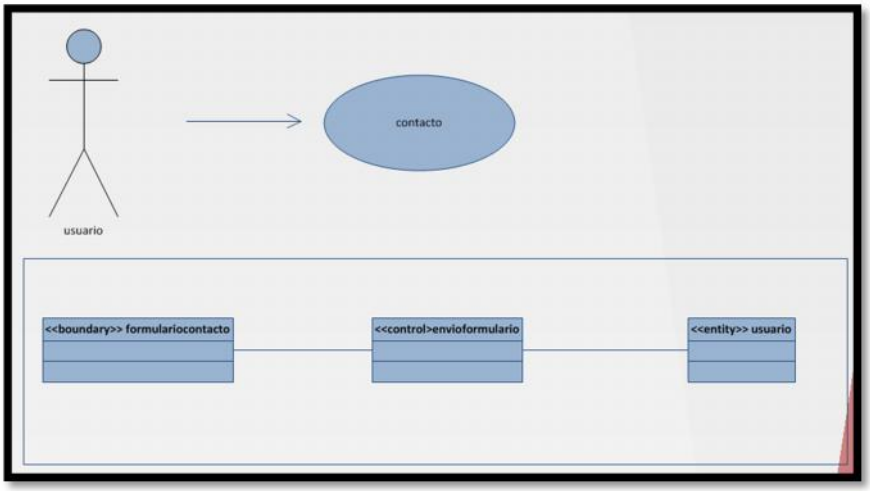

Figura 2.29 Diagrama de clases esteoripadas para el caso de uso contacto

En la figura 2.30 se tiene el diagrama correspondiente al caso de uso salir, donde se tienen tres clases; una <<boundary>> que nos permitirá seleccionar la opción de salir y una clase <<control>> que cierra la sesión, liberando las claves.

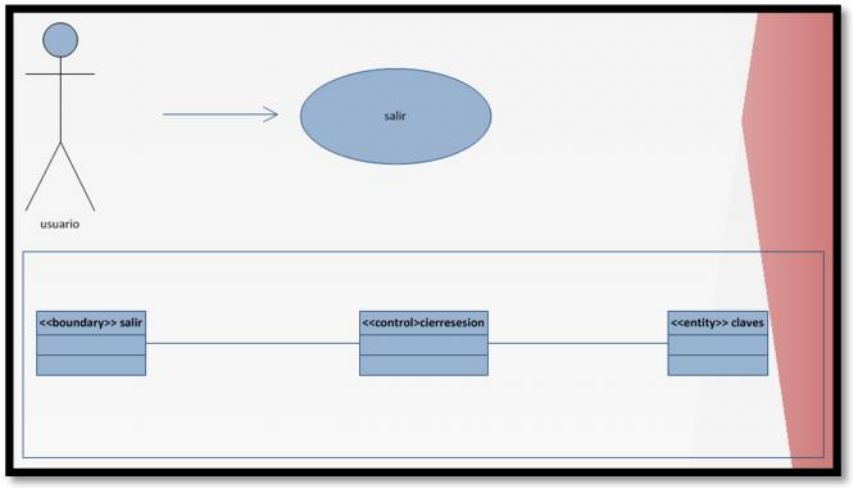

Figura 2.30 Diagrama de clases esteoripadas para el caso de uso salir

2.8.2 DIAGRAMAS DE SECUENCIA

En los siguientes diagramas de secuencia, se muestran los objetos, su línea de vida y la secuencia en que se ejecutaran.

En el diagrama de secuencia del proceso de registro (ver figura 2.31), se observa que se llena primero el formulario y posteriormente se validan los datos del registro antes de guardar la información del usuario.

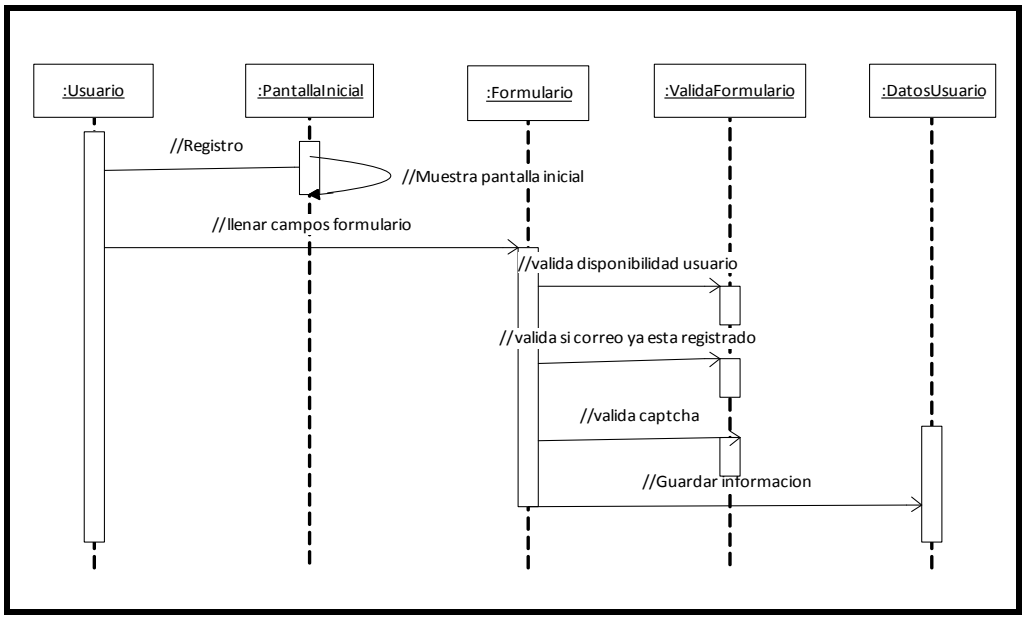

Figura 2.31 Diagrama de Secuencia del caso de uso Registro

En el diagrama de secuencia del proceso de Login (ver figura 2.32), se observa el proceso de ingreso al sistema, donde se ingresan claves de usuario y contraseña; se validan las claves con las almacenadas en el sistema, para posteriormente permitir el acceso al sistema y abrir la sesion.

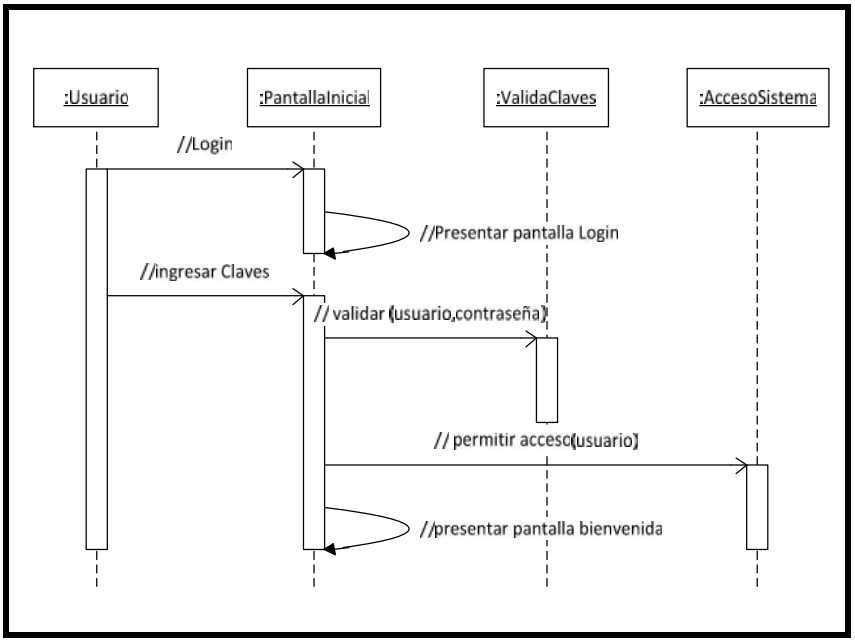

Figura 2.32 Diagrama de Secuencia del caso de uso Login

El diagrama de secuencia del caso de uso de Crear una Nota (ver figura 2.33), se observa que después de seleccionar la opción respectiva, se va a mostrar una ventana donde se van a poder ingresar la nueva nota, así como la opción de agregar un archivo, para posteriormente guardar la nota.

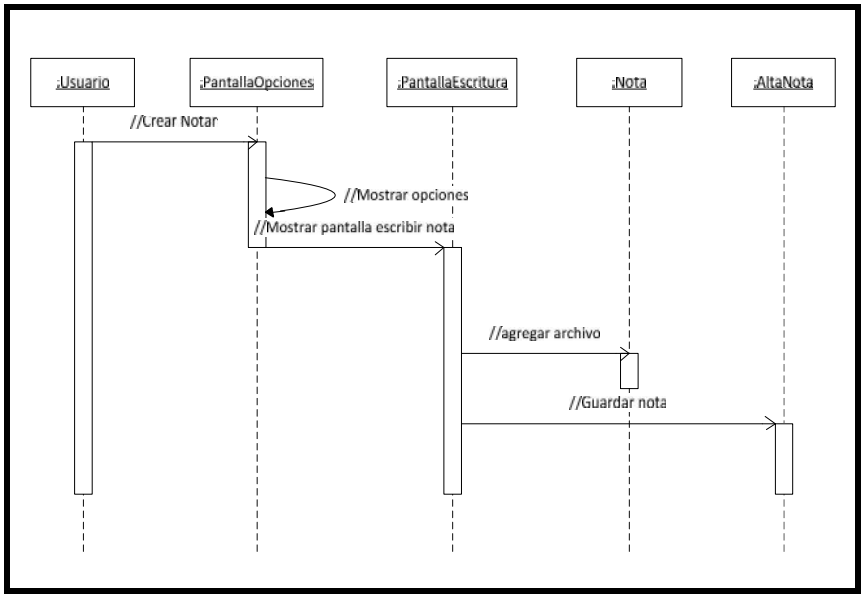

Figura 2.33 Diagrama de Secuencia del caso de uso Crear Nota

Para el caso de uso de Editar Nota, en su diagrama de secuencia (ver figura 2.34) se puede observar que se selecciona primero la nota a editar, posteriormente se modifica la nota y por ultimo se guardan los cambios.

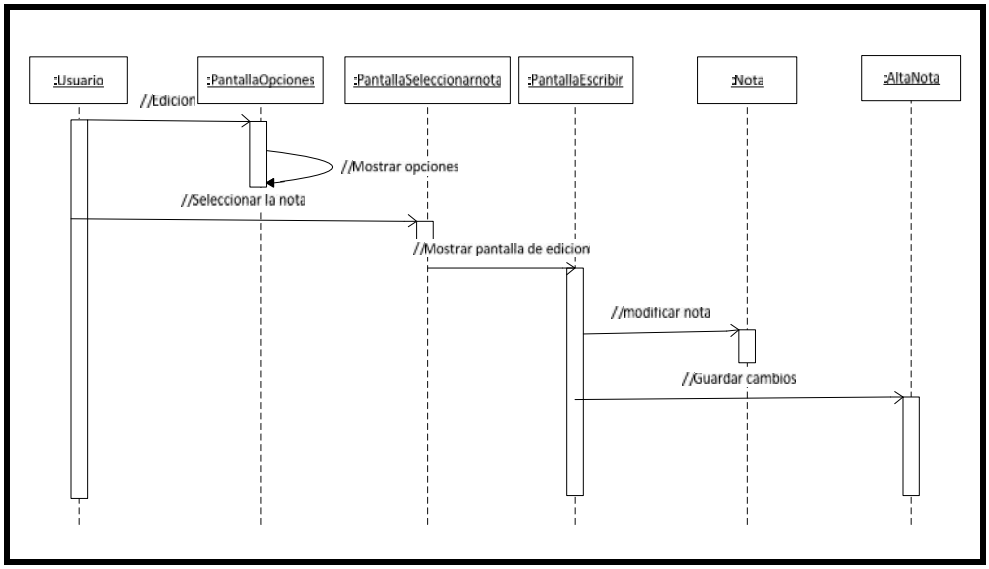

Figura 2.34 Diagrama de Secuencia del caso de uso Editar Nota

En el diagrama de secuencia para eliminar una nota (ver figura 2.35). Se elige la opción deseada, posteriormente se muestra una lista de las notas almacenadas; después el usuario elije la nota a eliminar y el usuario confirma la eliminación, para que sea eliminada la nota de la base de datos.

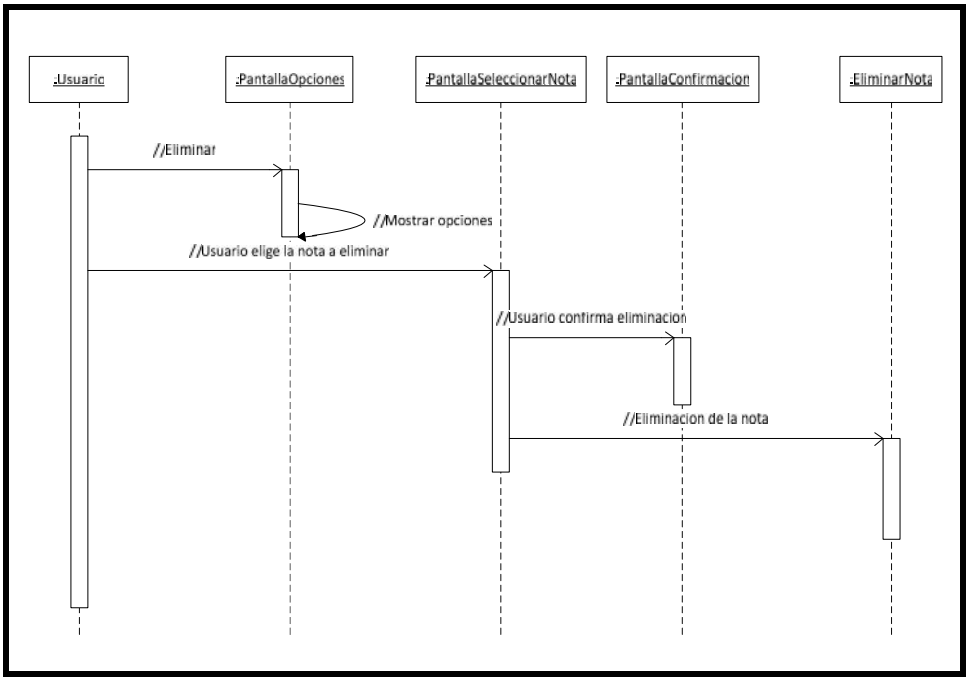

Figura 2.35 Diagrama de Secuencia del caso de uso Eliminar Nota

El diagrama de secuencia para búsqueda (ver figura 2.36). Se elije la opción deseada, se realiza la búsqueda y muestra un listado de las notas almacenadas.

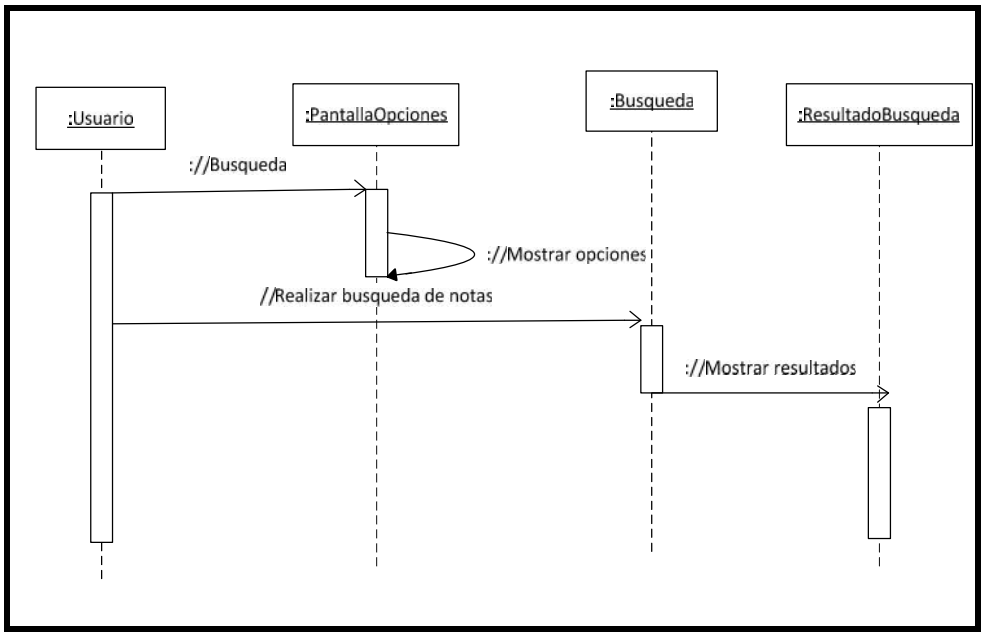

Figura 2.36 Diagrama de Secuencia del caso de uso Búsqueda de nota

El diagrama de secuencia del caso de uso contacto se muestra en la figura 2.37, donde el usuario selecciona la opción de contacto, posteriormente se muestra el formulario de contacto, el cual es llenado por el usuario y después es enviado al correo del administrador.

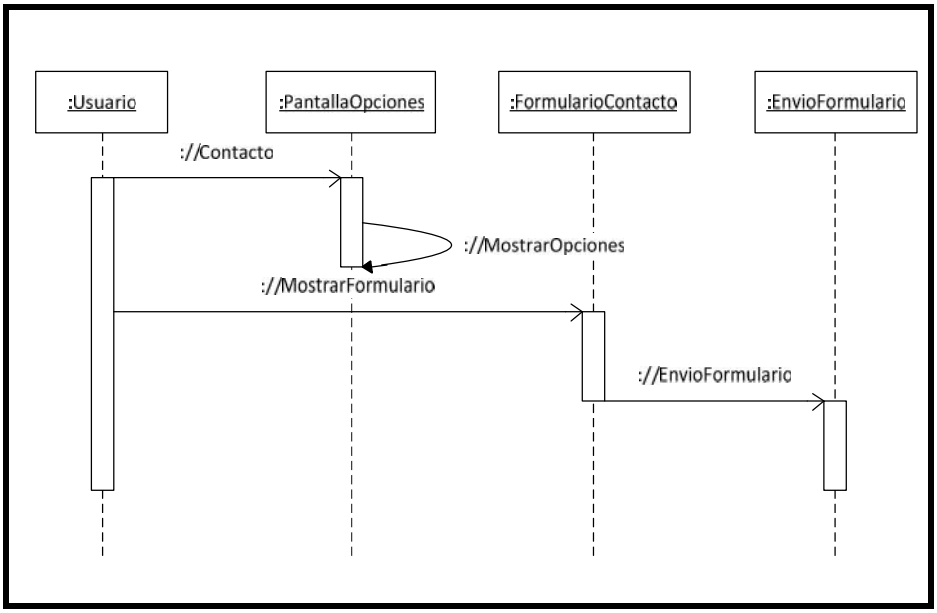

Figura 2.37 Diagrama de secuencia del caso de uso Contacto

Para el diagrama de secuencia del caso de uso Salir (ver figura 2.38), simplemente el usuario elige la opción de salir, se cierra la sesión y se liberan las claves.

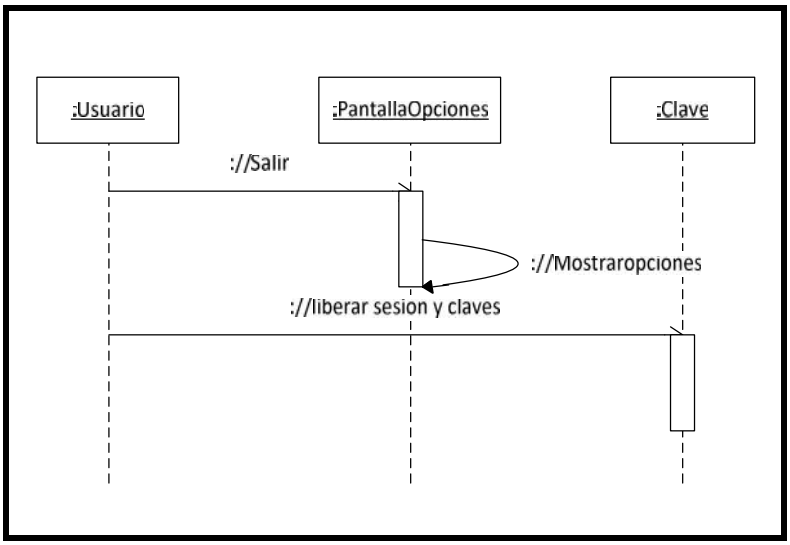

Figura 2.38 Diagrama de secuencia del caso de uso Salir

2.8.3 DIAGRAMAS DE COLABORACIÓN

En los siguientes diagramas (ver figuras de la 2.39 a 2.46), se muestra los mensajes intercambiados en los diferentes objetos de nuestro sistema. Estos diagramas solo nos va a dar una visión diferente a la que se tiene en los diagramas de secuencia vistos en el capitulo anterior.

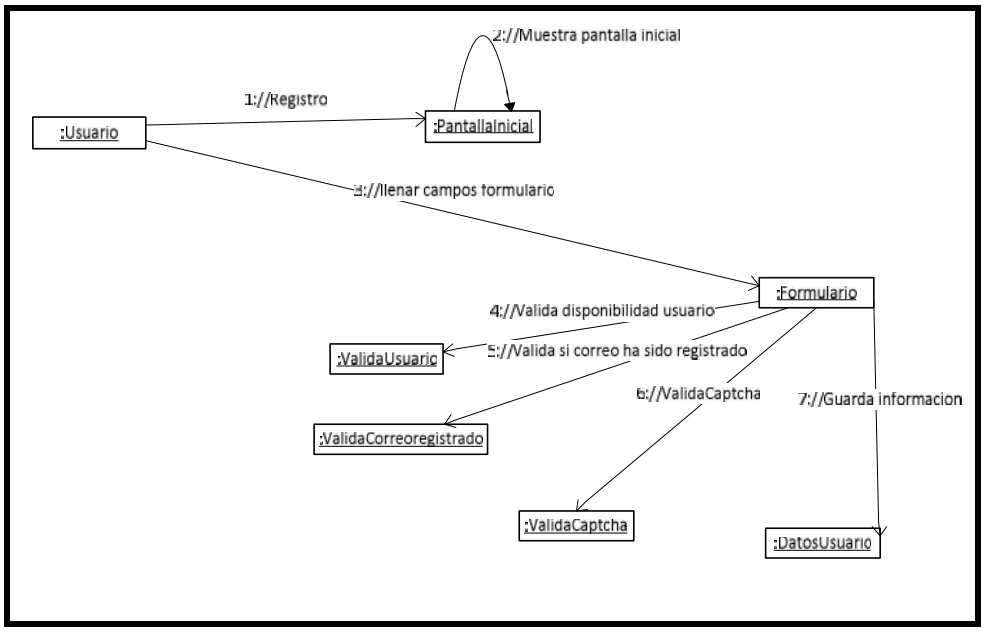

Figura 2.39 Diagrama de colaboración del caso de uso Registro

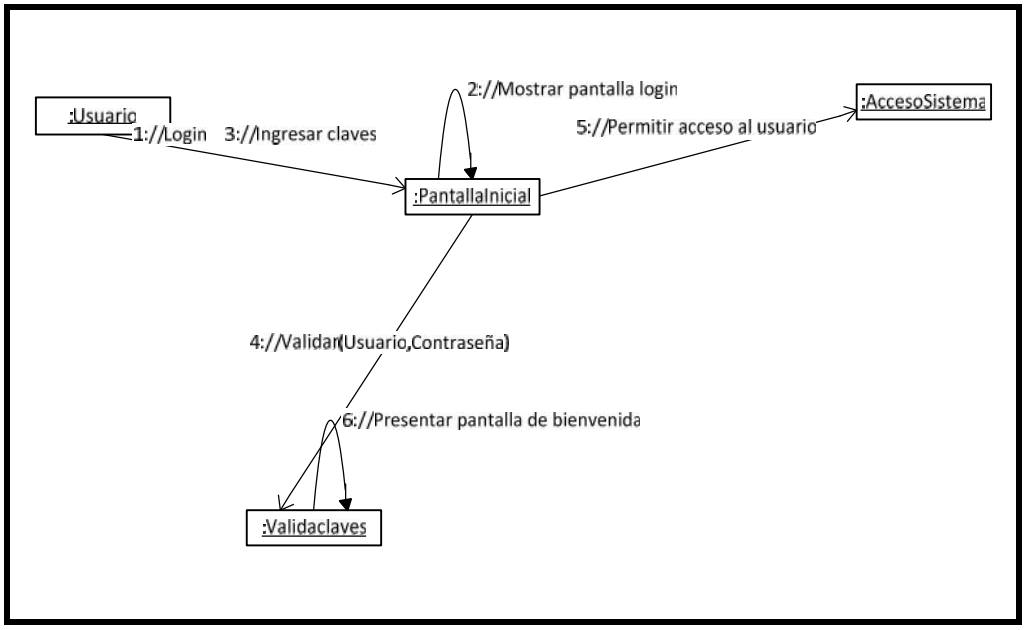

Figura 2.40 Diagrama de colaboración del caso de uso Login

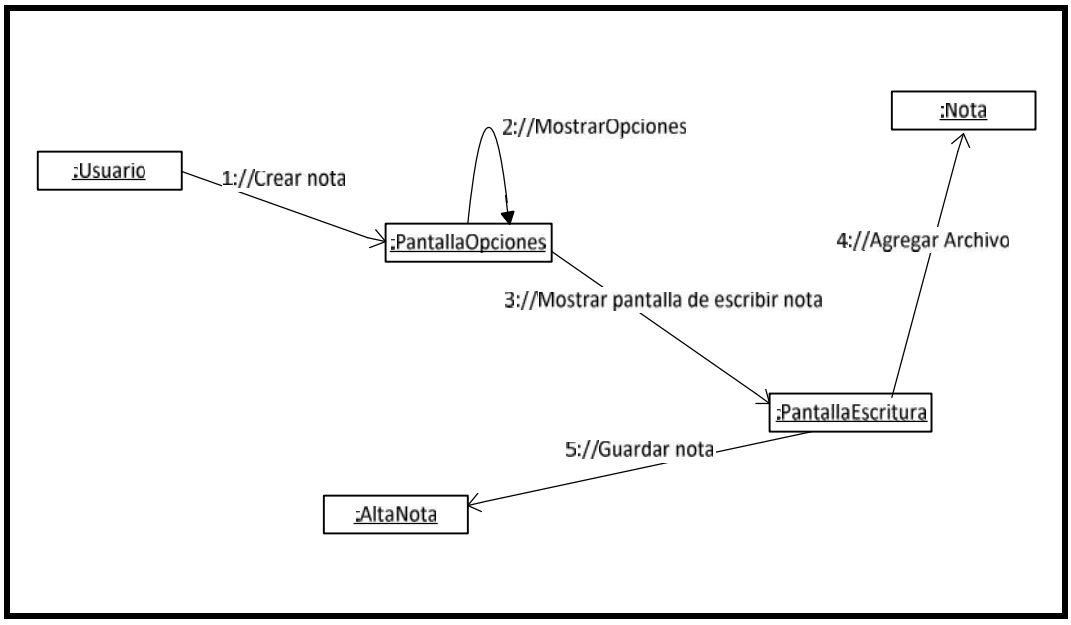

Figura 2.41 Diagrama de colaboración del caso de uso Crear Nota

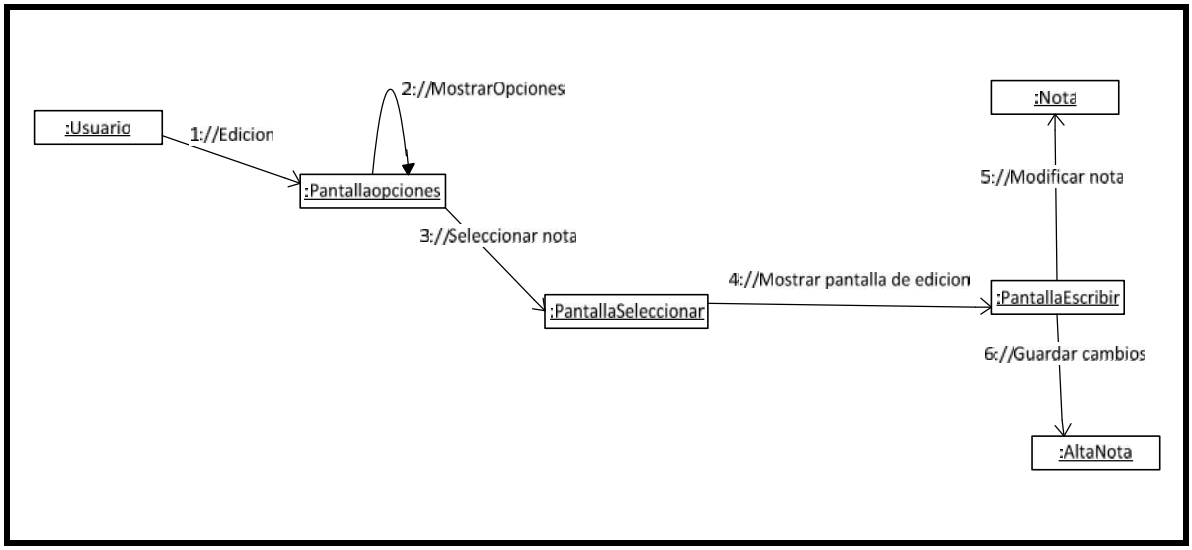

Figura 2.42 Diagrama de colaboración del caso de uso Editar Nota

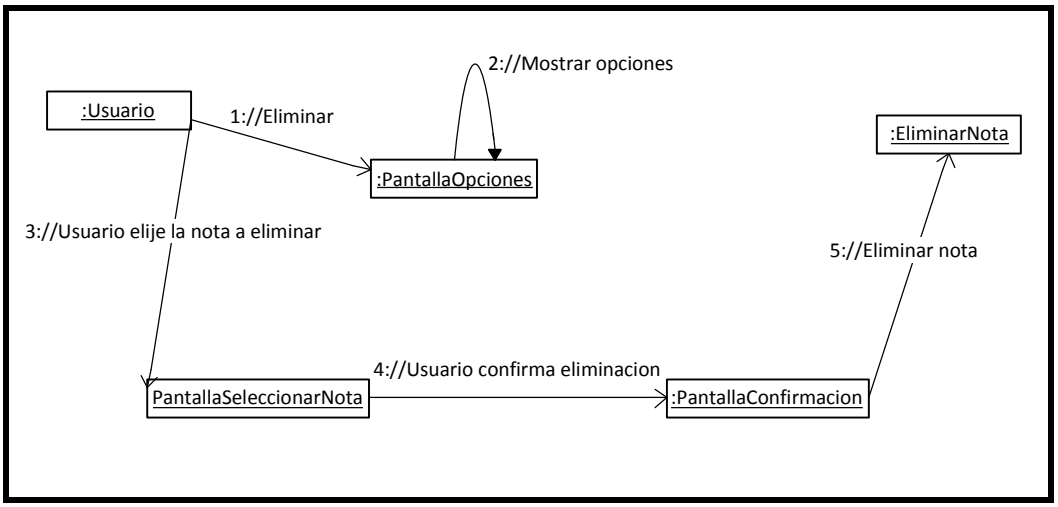

Figura 2.43 Diagrama de colaboración del caso de uso Eliminar

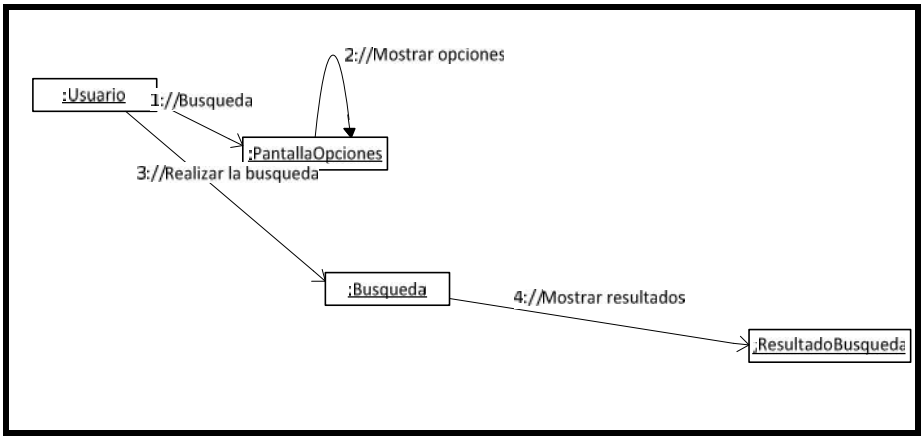

Figura 2.44 Diagrama de colaboración del caso de uso Búsqueda

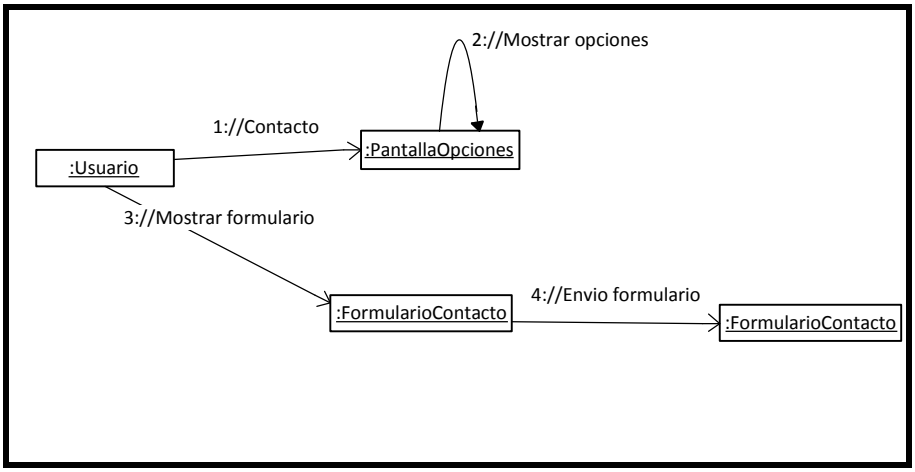

Figura 2.45 Diagrama de colaboración del caso de uso Contacto

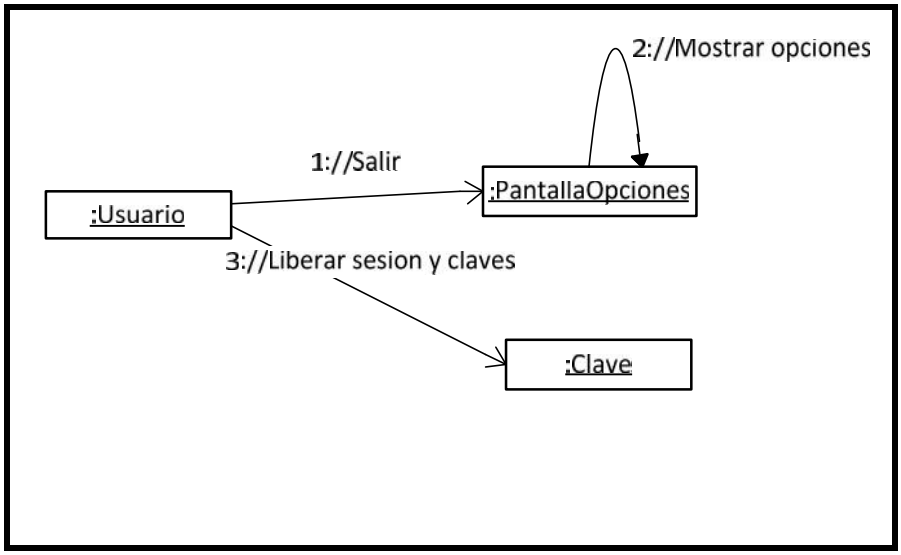

Figura 2.46 Diagrama de colaboración del caso de uso Salir

CAPITULO 3: DISEÑO DE LA BASE DE DATOS

3.1. DISEÑO CONCEPTUAL DE LA BASE DE DATOS

3.1.1. MODELO ENTIDAD – RELACIÓN

De acuerdo al planteamiento del problema se tienen claramente las entidades Usuario y Nota. De la parte del enunciado que indica que "*el sistema deberá ser controlado por un usuario y una contraseña*", pueden ser atributos de una entidad llamada *Clave*; así como también indica que las notas se les pueden asignar un archivo, por lo cual este también puede ser parte de un atributo llamado *Archivo*.

Las relaciones encontradas son:

- *Generar* entre Usuario y Nota
- *Corresponde* entre Usuario y Clave
- *Tiene* entre Nota y Archivo

Ahora iremos analizando los supuestos semánticos del planteamiento del problema:

- Los usuarios pueden generar ninguna o muchas notas
- A cada usuario le corresponde solamente una clave
- Una nota puede contener algún archivo adjunto

Ahora analicemos de los supuestos no dados en el planteamiento del problema:

- Una *nota* puede ser generada por un usuario
- Una *clave* le corresponde solo un usuario
- Un *archivo* esta contenida dentro de una nota

De lo anterior se puede tener el siguiente modelo E-R (ver figura 3.1) con las entidades correspondientes y sus relaciones, así como de sus cardinalidades.

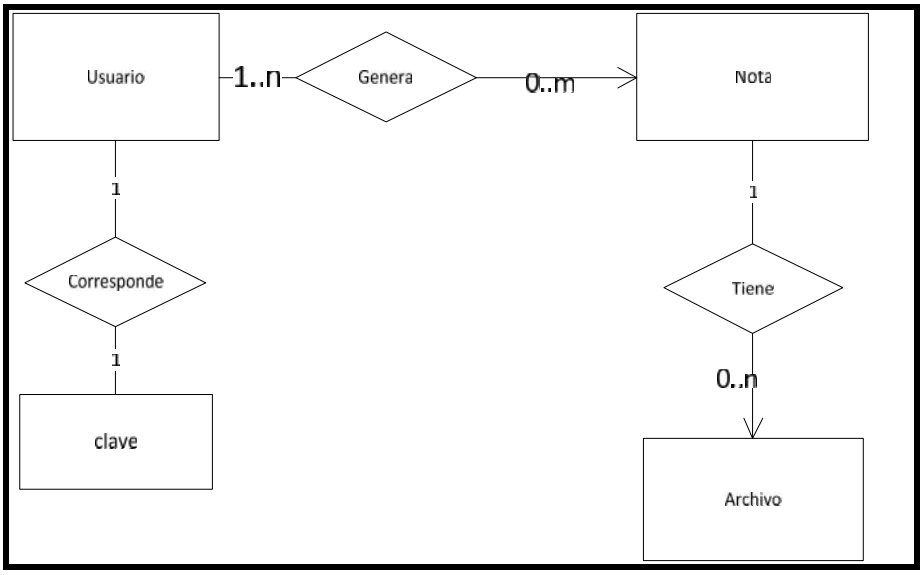

Figura 3.1 Diagrama E.R del sistema de notas, mostrando solo las entidades, relaciones y cardinalidades.

3.1.2 DESCRIPCIÓN DE ENTIDADES Y RELACIONES

Como se menciono anteriormente las entidades encontradas de acuerdo al planteamiento del problema son *Usuario, Nota, Clave* y *Archivo*.

Además de las relaciones, entidades y cardinalidad, cada entidad tiene atributos.

La entidad Nota cuenta con los tributos:

- Nombre
- Apellido Paterno
- Apellido Materno
- Fecha de registro

La entidad Nota lleva los atributos:

- Fecha de creación
- Titulo
- Contenido
- Clasificación

La entidad Clave lleva los atributos:

- Contraseña
- Nombre de usuario

La entidad Archivo tiene los atributos:

- Tamaño
- Nombre del archivo
- Tipo de archivo

Todos los atributos son simples, ya que no se dividen en subpartes, además de ser monovalorados, debido a que para cada atributo le corresponde un solo valor para la entidad concreta y por ultimo no contiene atributos derivados porque no obtienen sus valores de otras entidades.

Con fines prácticos a cada entidad se les va a asignar una clave, la cual fungirá también como clave primaria que tendrá la función de representar a la entidad. Con los datos obtenidos el modelo ER se muestra en la figura 3.2

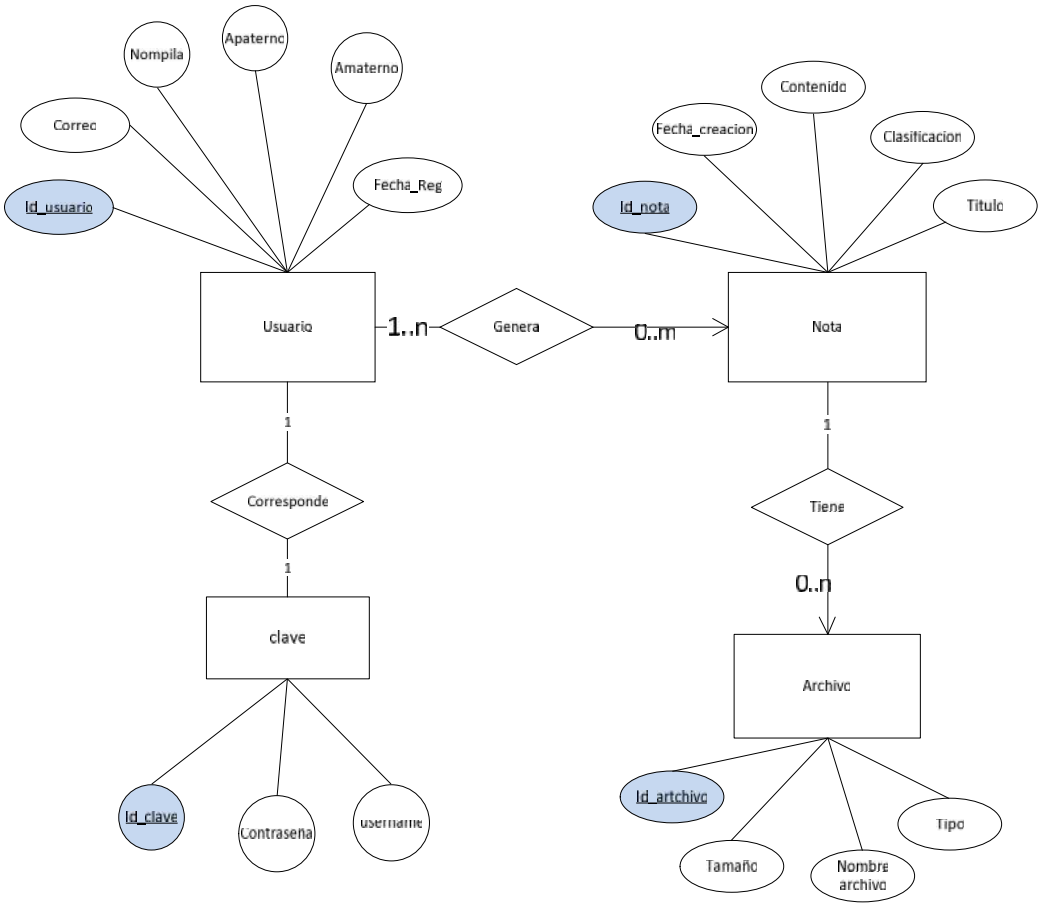

Figura 3.2 Diagrama E-R completo del sistema de notas

3.1.3 DISEÑO LÓGICO

Un esquema, va a ser la descripción lógica de la base de datos, proporcionando los nombres de las entidades y sus atributos especificando las relaciones que existen entre ellos.

Del modelo ER obtenido anteriormente se van a obtener los esquemas, los cuales van a ser un primer acercamiento a nuestro modelo relacional. A continuación Iremos revisando las reglas de transformación [11] que también nos proporcionaran al final un modelo relacional ya normalizado hasta la 3FN:

1.-Por cada tipo de entidad fuerte E del esquema ER se crea una relación R que contenga todos los atributos simples y no multivaluados de E. Además, dado que el modelo relacional no admite los valores no atómicos, R contendrá también solo los atributos simples que formen parte de cada atributo compuesto (no multivaluados) de E. Como Clave primaria de R se escogerá el atributo o atributos simples que formen parte de la clave primaria de E. Los atributos derivados se ignoran.

En nuestro caso, aplicando la regla que acabamos de ver, resultan los siguientes esquemas, a los que les vamos a dar el mismo nombre de cada entidad y vamos a agregar los atributos simples. Por ultimo en nuestro diagrama ER no existen atributos compuestos y como clave primaria se elige a la que aparece como clave primaria en el diagrama.

Esquema_usuario= (id_usuario, nompila, apaterno, amaterno, correo, fecha_reg)

Esquema_clave= (id_clave, username, contraseña)

Esquema_nota= (id_nota, fecha_creacion, contenido, clasificación, titulo)

Esquema_archivo= (id_archivo, tamaño, nombre_archivo, tipo)

En cuanto a la clave primaria se agrega y se denota como el primer atributo y se subraya. Por ultimo no existen atributos derivados en nuestro modelo ER.

2.- Por cada tipo de entidad débil E del esquema ER se crea una relación R que contenga todos los atributos simples y no multivaluados de E. Además, dado que el modelo relacional no admite valores no atómicos, R contendrá también solo los atributos simples que formen parte de cada atributo compuesto (no multivaluados) de E. Como clave primaria de R se escogerá el atributo o atributos simples que formen parte del discriminante de E, además de la clave primaria del tipo de entidad fuerte E' del que dependa E. Los atributos derivados se ignoran.

Para nuestro caso en particular de nuestro modelo ER, no se cuenta con entidades débiles, por lo que esta segunda regla no aplica.

3.- Por cada tipo de relación (de grado 2) R del esquema ER, de cardinalidad 1:1, se identifican a las relaciones S y T del esquema relacional que representan a los tipos de entidades participantes. Se escoge una de las dos relaciones (por ejemplo S) y se incluye como clave foránea de S la clave primaria de T. además, se incluyen en S todos los atributos (no multivaluados) del tipo de relación R, incluidos aquellos que conformen un atributo compuesto. Los atributos derivados se ignoran.

Nota: Para escoger la relación S en la que incluir los atributos, es mejor pensar en una que corresponda a un tipo de entidad con **participación total** en el tipo de relación a representar

En nuestro modelo ER se tiene un caso de cardinalidad 1:1, se trata de *Corresponde* que se encuentra entre *Usuario* y *Clave*. *Clave* representa una representación total, mientras que *Usuario* representa participación parcial, por lo que elegimos a Clave para incluir en ella a los atributos que permitan representar a la relación y la clave primaria de la entidad Clave pasa a ser llave foranea.

Clave (id_clave, contraseña, username, id_usuario)

4.- Por cada tipo de relación (de grado 2) R del esquema ER, de cardinalidad 1:N, se identifica a la relación S que representa al tipo de entidad participante del lado N, y a la relación T que representa al tipo de entidad participante del lado 1. Se incluye como clave foránea de S la clave primaria de T. Se incluyen también en S los atributos (no multivaluados) del tipo de relación, incluidos aquellos que conformen un atributo compuesto. Los atributos derivados se ignoran.

Para nuestro proyecto en particular tenemos un tipo en particular con cardinalidad 1:N y es:

Tiene que se encuentra entre Nota y Archivo.

En este tipo de relación Tiene; Archivo se encuentra del lado N y la parte de Nota del lado 1, por lo tanto, utilizamos la relación asociada, Archivo, de modo que cada tupla que represente a un archivo incluirá el numero de la nota que representa.

Archivo (id_archivo, tamaño, nombre_archivo, tipo, id_nota)

5.- Por cada tipo de relación (de grado 2) R del esquema ER, de cardinalidad M:N, se crea una nueva relación S que tendrá como atributos de clave foránea los atributos que formen la clave primaria de los dos tipos de entidad participantes en R. Además, S incluirá los atributos simples (no multivaluados) del tipo de relación, incluidos aquellos que conformen un atributo compuesto. La clave primaria de S estará formado por los atributos de clave primaria de los tipos de entidad participantes en el tipo de relación. Los atributos derivados se ignoran.

Para nuestra transformación tenemos a la relación Genera que se encuentra entre Usuario y Nota. De acuerdo a la regla vamos a crear una nueva relación, donde se incluyen como claves las claves primarias de las entidades relacionadas

Genera (id_usuario, id_nota)

En esta relación, las tuplas nos permitirán vincular a cada usuario con cada nota y viceversa, usando los identificadores de ambos.

6.- Por cada atributo multivaluado corresponde a un tipo de entidad o tipo de relacion R del esquema ER, se crea una nueva relacion S que tendrá como atributos de clave foránea los atributos de clave primaria de R. Ademas, S incluirá el atributo multivaluado; si el atributo es compuesto, se

incluirán los atributos simples que lo integren. La clave primaria de S será la misma que la de R, unida al atributo multivaluado.

Para nuestro modelo en cuestión no existen atributos multivaluados por lo que esta regla no aplica.

Como resultado nuestros esquemas quedan como se muestra en la figura 3.3.

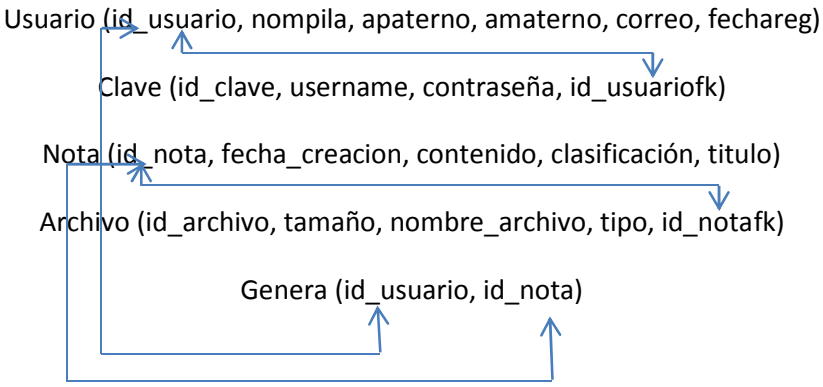

Figura 3.3 Esquemas del sistema de notas

En la tablas de la 3.4 a 3.8 se van a presentar las instancias de los esquemas anteriores, con el fin de observar los datos en un momento dado.

Id_usuario	Nompila	Apaterno	amaterno	correo	Fechareg
1	Fabiola	Alvarado	Quecholac	Hola@hotmail.com	2012-09-12
2	Beto	Cotzomi	Paleta	bcp@hotmail.com	2012-09-12
3	Usario	Prueba	Prueba	prueba@hotmail.com	2012-09-12

Tabla 3.4 Instancia de la tabla Usuario

Id_clave	Username	Contraseña	Id_usuariofk
1	Faby	*********	1
2	Ncotzomi	*********	2
3	Usrprueba	*********	3

Tabla 3.5 Instancia de la tabla Clave

Id_nota	Titulo	Fecha_creación	Contenido	Clasificación
1	Primer nota	2012-09-12	Mi primer nota	Ideas espontaneas
2	Segunda nota	2012-09-12	Nota de prueba	Sin clasificar
3	Hola	2012-09-12	Hola mundo	Sin clasificar

Tabla 3.6 Instancia de la tabla Nota

Id_archivo	Tamaño	Nombre_archivo	Tipo	Id_notafk
1	28343	Nota.jpg	Image	1
2	34563	Sol.jpg	Image	1
3	94672	Saludo.bmp	Image	2

Tabla 3.7 Instancia de la Tabla Archivo

Id_usuario	Id_nota
2	1
3	2
2	3

Tabla 3.8 Instancia de la tabla Genera

El modelo relacional obtenido de los esquemas de la figura 3.3 es el que se muestra en la figura 3.9.

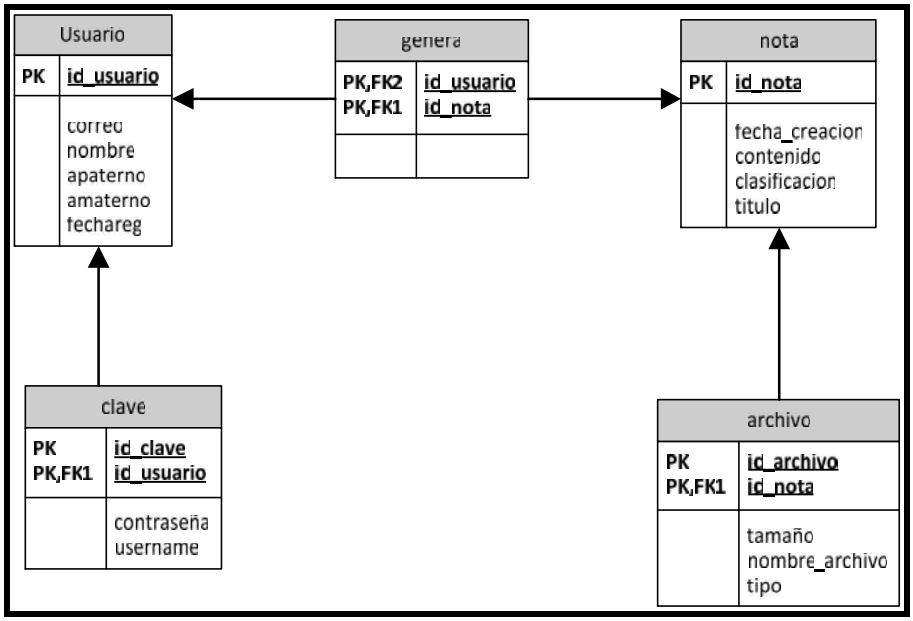

Figura 3.9 Modelo relacional del sistema de notas

3.1.4 NORMALIZACIÓN

Una vez creadas las tablas hay que verificarlas y revisar si aun se pueden reducir u optimizar de alguna manera.

La normalización es uno de los principales retos para obtener una estructura estable y lógica. La normalización nos va a permitir evitar anomalías de almacenamiento y para que el modelo lógico pueda modificarse sin ningún problema.

Esta normalización nos va a permitir disminuir reducir los riesgos inherentes a la redundancia de la información y la inconsistencia de los datos. En general lo que vamos a realizar en el siguiente proceso de normalización de nuestra base de datos, es la búsqueda de la mejor estructuración a través de las siguientes formas normales:

Forma Normal1

"*Todos los atributos son atómicos, es decir que los elementos de un dominio son indivisibles o mínimos*."

De acuerdo a la definición que se dio en el marco teórico; observamos que todas las tablas contienen una clave primaria única y además esta no contiene valores nulos, debido a que las claves son autoincrementables. No existe variación en el numero de columnas y a cada columna le corresponde un solo valor atómico.

Por lo tanto todas las tablas se encuentran en primera forma normal.

Forma Normal2

"*Una relación esta en 2FN si esta en 1FN y si los atributos no forman parte de ninguna clave dependen de forma completa de la clave principal, es decir que no existen dependencias parciales (todos los atributos que no son clave principal deben depender únicamente de la clave principal*)"

Ahora procedemos a aplicar la segunda forma normal, es decir, tenemos que eliminar cualquier columna no clave que no dependa de la clave primaria de la tabla.

Todas las tablas se encuentran en la segunda forma normal. Cualquier valor único de cada clave primaria determina un solo valor para cada columna.

Forma Normal3

Una tabla esta normalizada en esta forma si todas las columnas no clave son funcionalmente dependientes por completo de la clave primaria y no hay dependencias transitivas.

Todas las tablas se encuentran en Tercera Forma Normal, debido que al revisar las tablas no se encuentran dependencias transitivas entre columnas de una misma relación con la clave primaria.

CAPITULO 4: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

4.1 IMPLEMENTACIÓN

El objetivo de esta sección es el diseño del prototipo del sistema así como la base de datos a partir del modelo lógico de la sección anterior. En primer lugar se crean las tablas de la base de datos en Mysql a través del software WAMP SERVER y posteriormente con la ayuda de Dreamwaver se creara el código PHP de nuestro sistema y a través de una plantilla le daremos una interfaz agradable para el usuario. Una vez que se encuentre realizado nuestro sistema este se montara en un servidor Web para que sea consultado a través de Internet.

4.2 IMPLEMENTACIÓN DE LA INTERFAZ

La implementación de la interfaz se realiza a través de una hoja de estilo bajada previamente desde internet y se va amoldando a las necesidades del programa con ayuda de Dreamwaver, así como también se va insertando código PHP y el enlace con los comandos de MySQL .

4.2.1 PÁGINA DE INICIO

En la página de principal de nuestro sistema se tienen limitadas funciones, ya que para tener acceso al sistema es necesario el registrarse previamente y así poder obtener las claves de acceso.

En esta página (ver figura 4.1) solo se van a tener las opciones de Registro y de Contacto. Esta pagina y sus opciones son las únicas que se pueden ingresar directamente tecleando desde la URL del navegador, ya que las demás opciones del sistema manejan el uso de sesiones y para ingresar a ellas se requiere que sea iniciada la sesión.

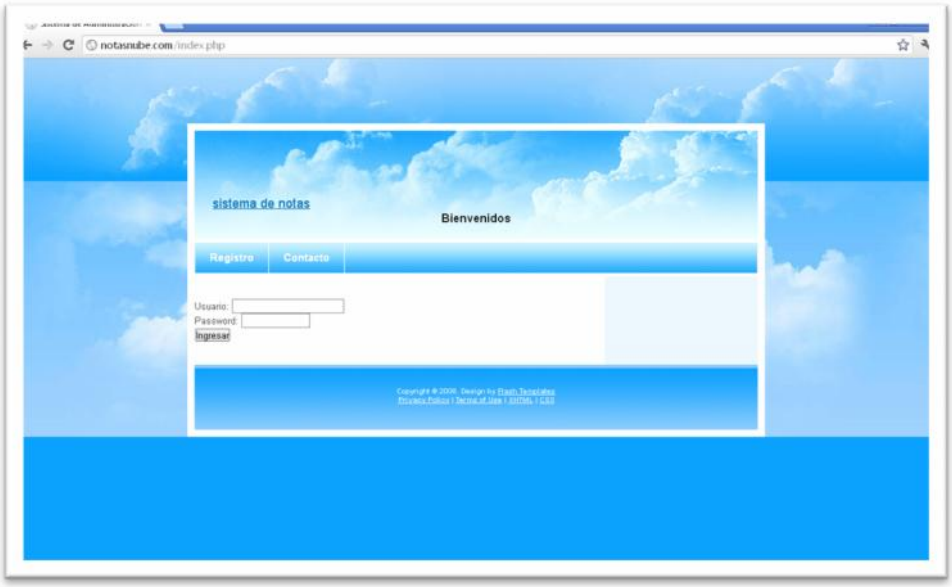

Figura 4.1 Página de inicio del sistema de notas

En la sección de registro (ver figura 4.2) es en la parte donde se pueden obtener las claves para poder accesar al sistema. En esta parte solo se les van a solicitar los datos necesarios para llenar

las tablas de Usuario y Clave, además se cuenta con un Captcha que nos va a permitir librarnos de los robots y nos generen muchos registros.

Figura 4.2 Sección de registro del sistema

La sección de contacto (ver figura 4.3) es otra opción a la que se puede ingresar desde esta página inicial. Básicamente esta sección esta pensada para que si algún usuario tiene algún problema con el sistema, este pueda enviar un mensaje al administrador.

Figura 4.3 Sección de contacto con el administrador

4.2.2 INTERFACES DE USUARIO

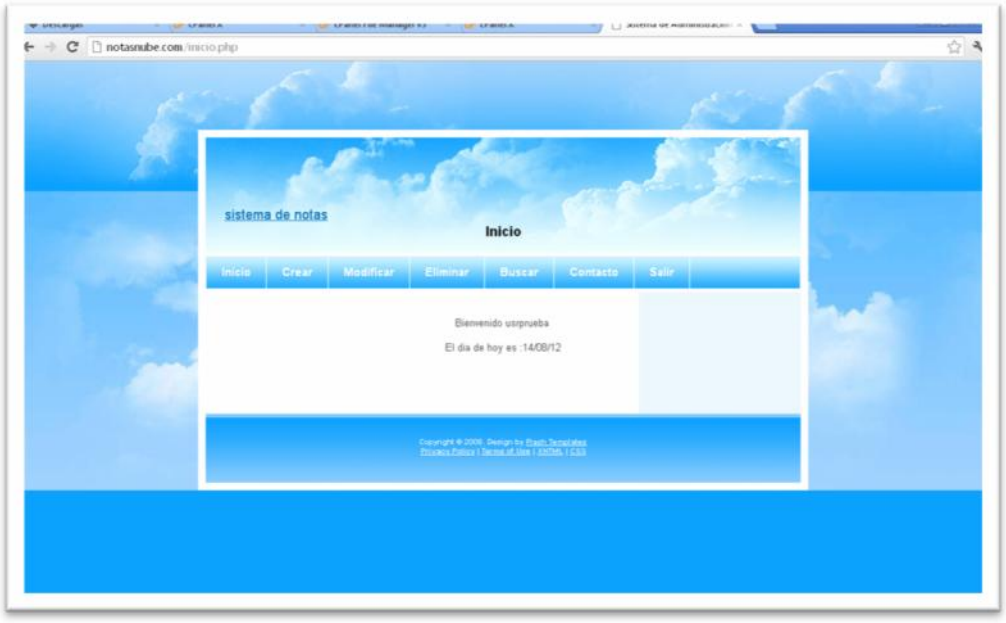

Figura 4.4 Página de Bienvenida

Una vez iniciada la sección el usuario podrá ver una página de bienvenida (ver figura 4.4) y en ese momento va a poder navegar en las diferentes opciones para poder crear y administrar sus notas.

CREAR. En la sección de crear nota el usuario va a poder crear una nota, donde se van a solicitar los datos que son necesarios para llenar la tabla de nota y en su caso si así lo requiere el usuario los datos para llenar la tabla de archivo. Los datos que se solicitan para crear una nota son el titulo, contenido de la nota, la clasificación y la fecha de creación se ingresara automáticamente al dar de alta la nota.

Si el usuario desea agregar un archivo a la nota, en la pagina de crear nota existe la opción de Seleccionar archivo, se presiona este y nos envía a otra pagina donde se nos solicitara seleccionar el archivo deseado y una vez seleccionado se grabara en un espacio de nuestro sistema el archivo adjunto, así mismo se extraerán los datos necesarios solicitados para llenar la tabla de archivo.

MODIFICAR. Para el caso de modificar una nota se muestra primero un listado de las notas creadas por el usuario y así este podrá seleccionar la nota a editar.

Una vez seleccionada la nota a editar se abre una nueva página donde se cargan los datos de dicha nota para ser modificados u una vez hechos los cambios se da clic en el botón de grabar para guardar los cambios.

ELIMINAR. En esta sección de nuestro sistema, se podrá eliminar alguna nota en particular del listado que se le muestra al usuario, solo con presionar en el valor de eliminar del atributo de borrar y en automático se eliminara la nota y en su caso también el archivo adjunto.

BUSCAR. Aquí se mostrara un listado de las notas almacenadas del usuario que haya iniciado sesión. Entre los datos que se muestran son el titulo de la nota, la fecha de creación, el contenido, la clasificación y el archivo adjunto.

CONTACTO. Aquí se podrá contactar con el administrador del sistema vía correo electrónico. Para poder enviar el mensaje se llena un formulario con el nombre del usuario, su correo y un espacio donde puede indicar su pregunta.

SALIR. Nos va a permitir la salida del sistema y liberación de la sesión.

4.2.3 INTERFAZ DEL ADMINISTRADOR

Debido a que nuestro sistema se encuentra alojada en Internet, su interfaz de administrador también se va a encontrar en Internet.

Para accesar a la interfaz del administrador (ver figura 4.5), en lo particular, se tiene que ingresar a la siguiente dirección: notasnube.com/cpanel; donde nos van a solicitar un nombre de usuario y una contraseña y se va a presionar el botón de acceder.

La dirección tecleada para la interfaz del administrador va a depender del hosting del sitio donde se encuentre alojada nuestra aplicación.

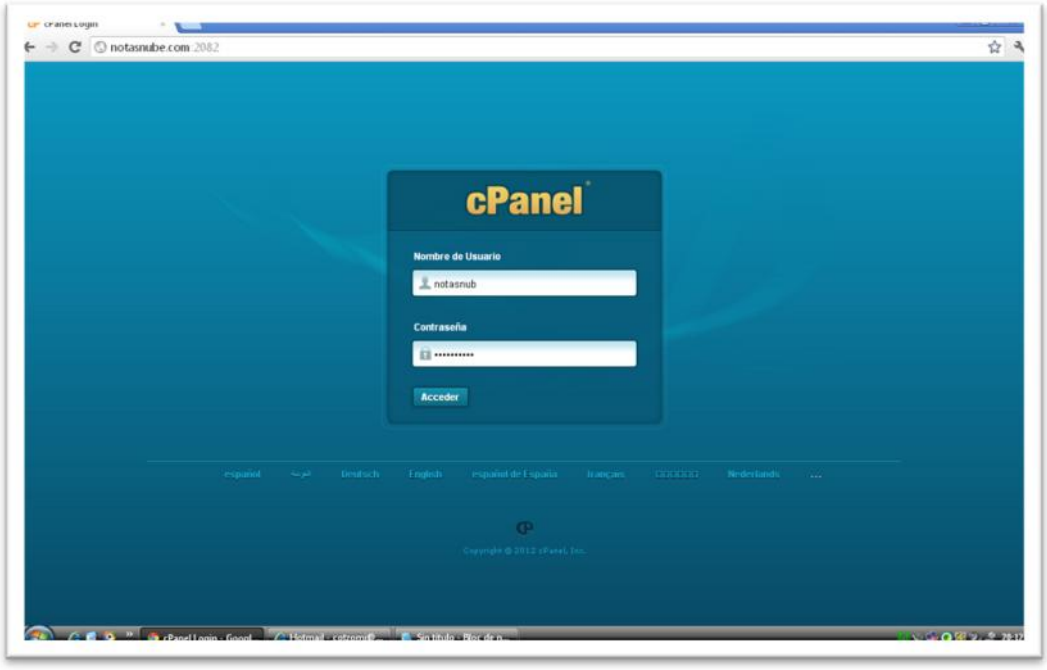

Figura 4.5 Interfaz de acceso del administrador

Una vez ingresadas las claves correspondientes, se va a poder ingresar a la sección HOME de nuestra interfaz de administrador (ver figura 4.6), donde se pude tener acceso a diferentes opciones y servicios, que esto dependerá del sitio donde se encuentre alojado nuestro sistema, aunque en lo general nos vamos a interesar en la parte de base de datos y la sección de archivos.

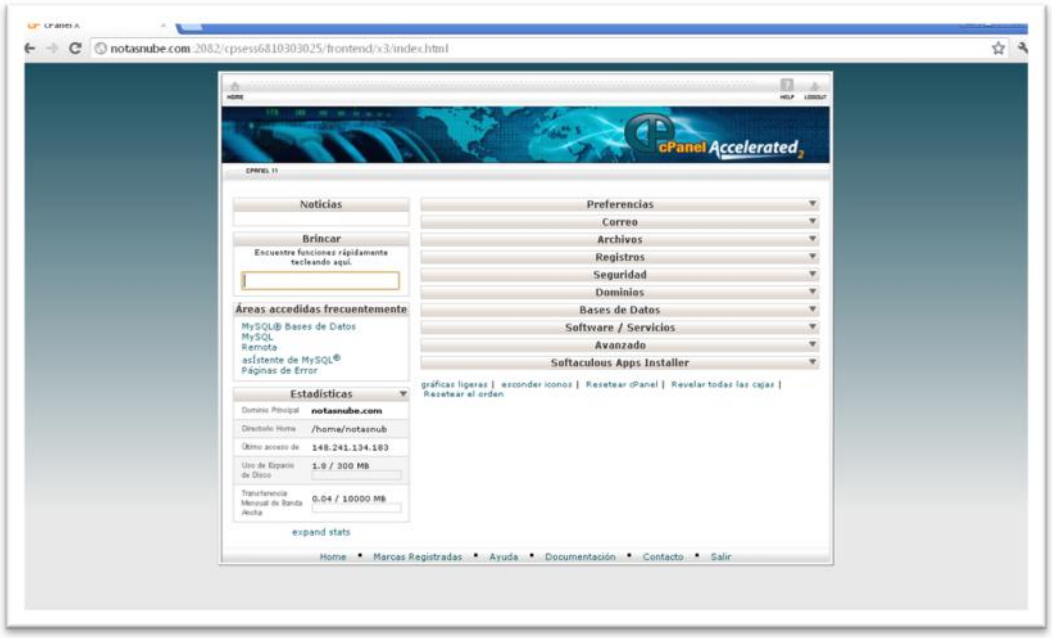

Figura 4.6 Panel principal de la interfaz del administrador

En la sección de base de datos se cargo la base de datos a través de PHPMyADMIN (ver figura 4.7), de donde se podrá cargar y/o modificar la base de datos.

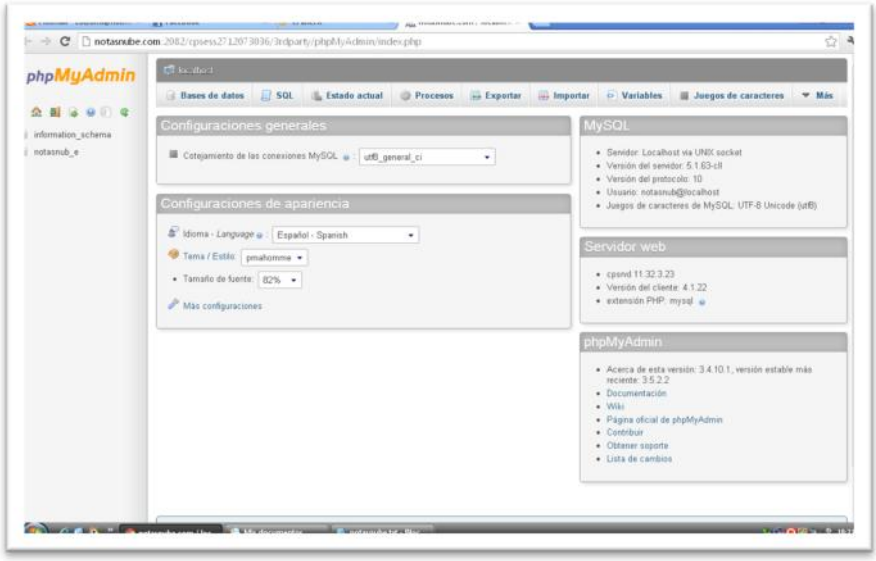

Figura 4.7 Interfaz de PHPmyAdmin en la pagina del administrador

Y por ultimo también nos es de utilidad la sección del administrador de Archivos (ver figura 4.8), donde nos mostrara unas secciones de archivos y elegiremos Public HTML. Este administrador de archivos nos va a permitir cargar los archivos necesarios para la pagina.

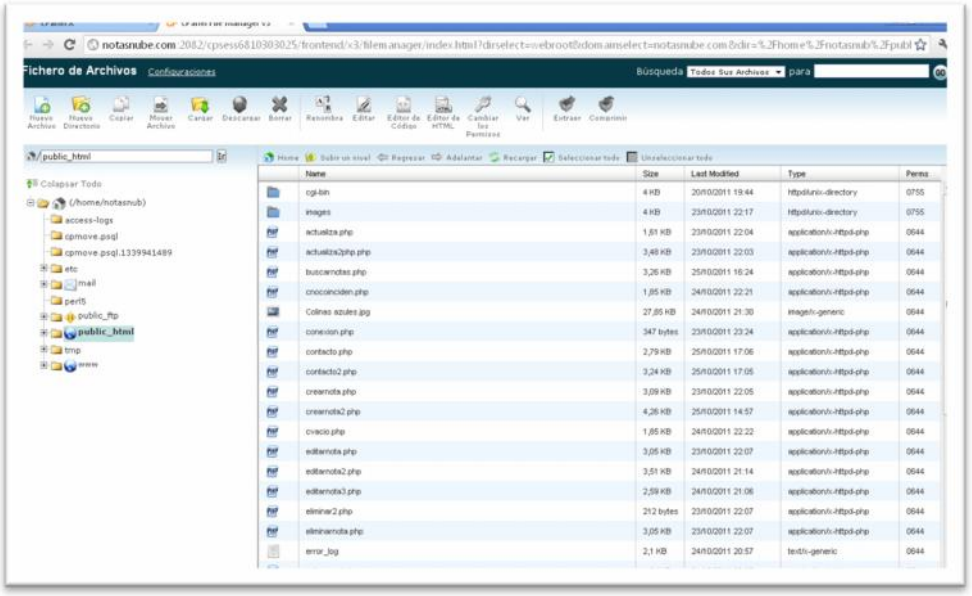

Figura 4.8 Sección de administración de archivos en la interfaz del administrador

4.3 SEGURIDAD

Aparte de la implementación de la base de datos y del mismo sistema, este cuenta con mecanismos de seguridad que nos ayudaran a proteger la información tanto de la base de datos y hasta el mismo código, para evitar ser victimas de algún tipo de ataque.

Claves de acceso. Para poder hacer uso del sistema, es necesario que el usuario se registre proporcionando unos datos personales y claves para el acceso. Es importante este procedimiento para que el usuario mantenga confidencialidad en sus mensajes.

Manejo de sesiones. El sistema cuenta con el manejo de sesiones lo que evitara que solo se teclee en la barra del navegador la dirección de alguna de las secciones del programa y se pueda tener acceso a la misma, sino que es necesario mediante el acceso mediante las claves y se pueda iniciar una sesión y cuando el usuario seleccione la opción de salir esta sesión sea terminada.

Encriptamiento de la contraseña. En este sistema además de no visualizar en la parte de contraseña, lo que se ingresa; también se encripta la misma con MD5 y aunque no es de los métodos mas robustos si no permitirá que la información no se envié en texto plano y de esta manera no sea victima de un ataque.

Contraseña de administrador. La base de datos esta protegida por una contraseña de administrador, lo que nos permitirá proteger a la base de datos de posibles ataques ya sea tanto a la información como al código de la misma base de datos.

Uso de Captcha. Se implementa el uso del mecanismo de seguridad Captcha, el cual evitara los ataques por parte de robots; evitando así la creación de cuentas fantasma.

4.4 PRUEBAS

En este capitulo se presenta como funciona el programa de notas en la nube, donde para nuestro caso como el sistema esta montado en un servidor web, debemos de acceder al programa mediante un navegador y teclear la dirección notasnube.com y nos aparecerá una página como la mostrada en la figura 4.9.

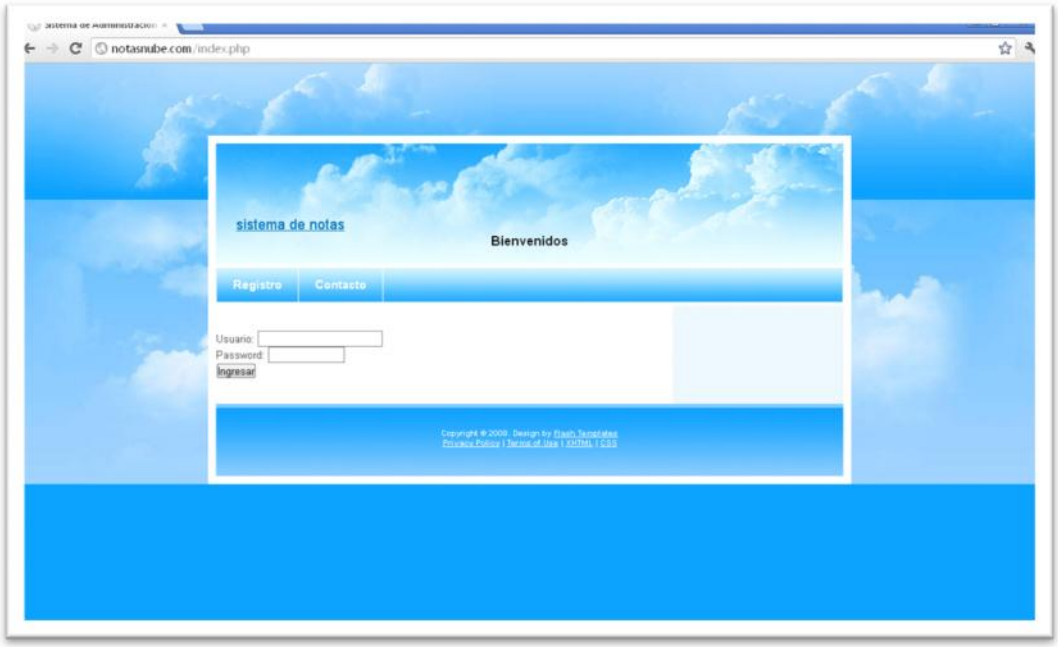

Figura 4.9 Pantalla principal de acceso

Una vez en el sitio para poder ingresar al sistema es necesario, como se menciono anteriormente, registrarse. Para registrarse es necesario presionar en la opción de Registro y nos mostrara la pantalla (ver figura 4.10). En esta sección podremos ingresar los datos solicitados para crear una cuenta (que para el caso particular será de prueba) y por ultimo se escribirá lo que aparece en el captcha, para darle clic en el botón de registrar.

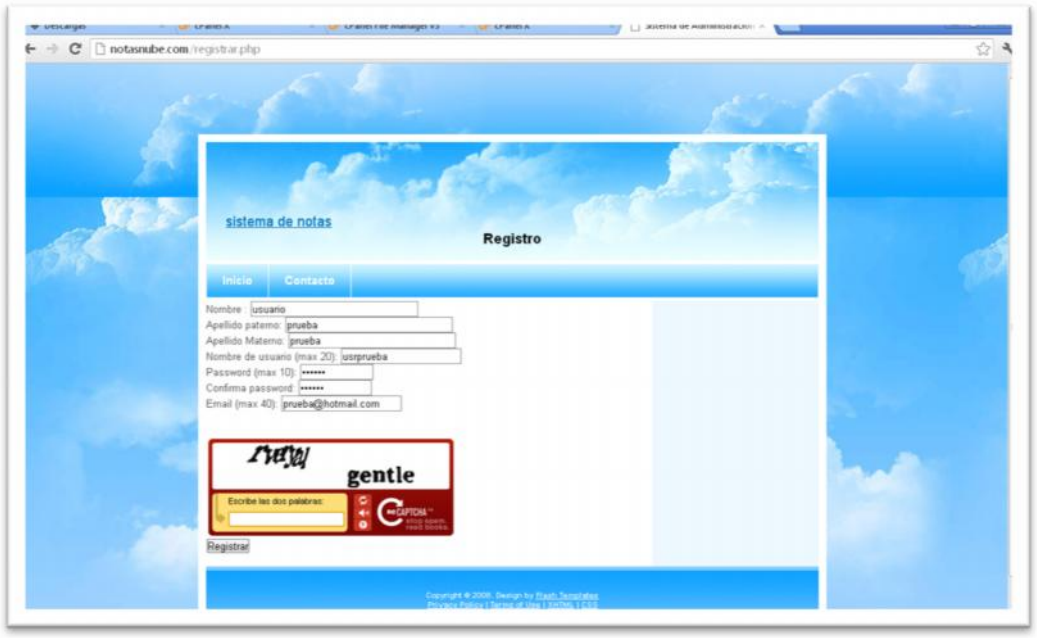

Figura 4.10 Sección de registro

Y nos tendrá que aparecer un mensaje de registro exitoso (ver figura 4.11). En caso de ingresar un correo o un nombre de usuario, que ha sido dado de alta anteriormente, nos aparecerá que un mensaje de error y nos regresara a la pantalla de registro.

Figura 4.11 Mensaje de registro exitoso

En la figuras 4.12 y 4.13 se muestran como en la base de datos de nuestro sistema ya se encuentra dado de alta el usuario con los datos que ingreso en el formulario. Cabe recalcar que los datos

personales del usuario se guardan en la tabla de Usuario y tanto el usuario como la contraseña en la tabla Clave.

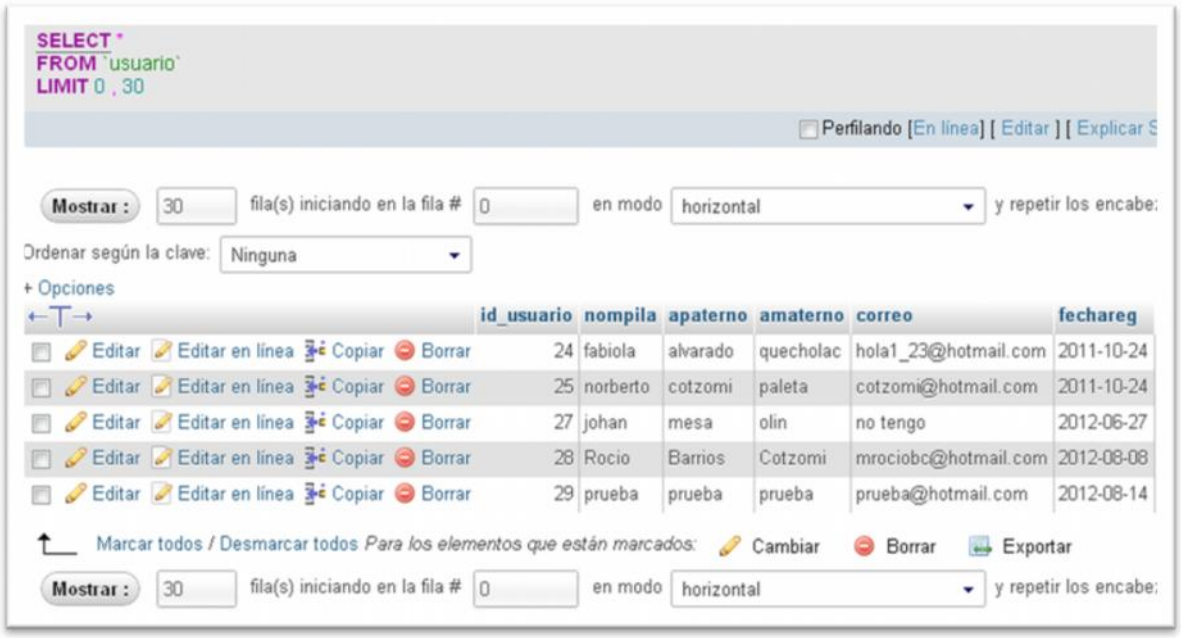

Figura 4.12 Datos almacenados en la tabla Usuario

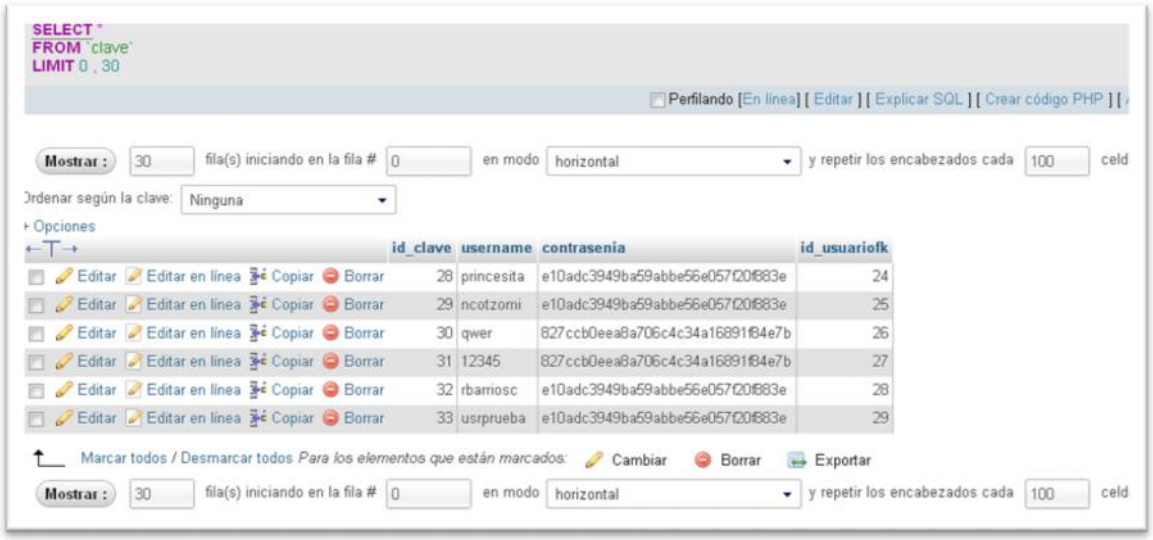

Figura 4.13 Datos almacenados en la tabla Clave

Una vez que tengamos listas nuestras claves nos vamos a la página principal e ingresaremos nuestras claves, desde donde podremos navegar entre las diferentes opciones para la creación y manejo de nuestras notas (ver figura 4.14).

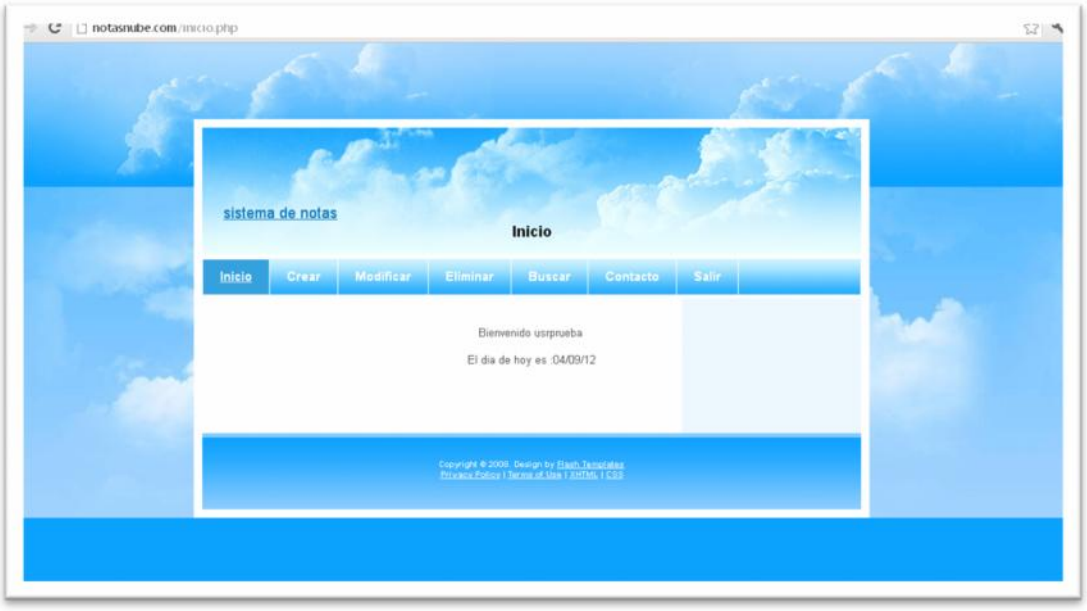

Figura 4.14 Página inicial del sistema de notas

Para crear una nota, es necesario seleccionar la opción de Crear y se nos solicitara que se ingresen los datos necesarios para dar de alta una nota (ver figura 4.15), como son el titulo, contenido, seleccionar la clasificación y en su caso un archivo adjunto.

Figura 4.15 Sección de crear notas

Para adjuntar un archivo (ver figura 4.16), se abre una ventana para seleccionar el archivo a adjuntar, aclarando que por situaciones de espacio y solo con fines de prueba, se restringió el tamaño del archivo a máximo 100kb.

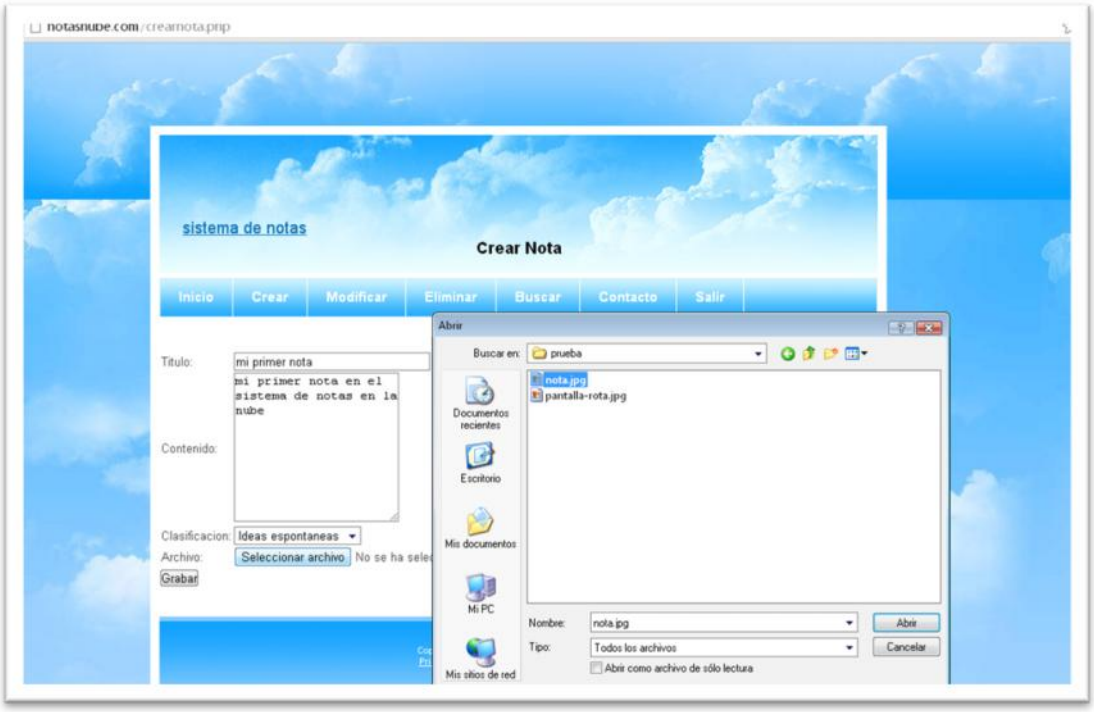

Figura 4.16 Selección de archivo adjunto

Una vez creada la nota se notificara la creación de la misma con un mensaje de confirmación de la creación de la nota (ver figura 4.17).

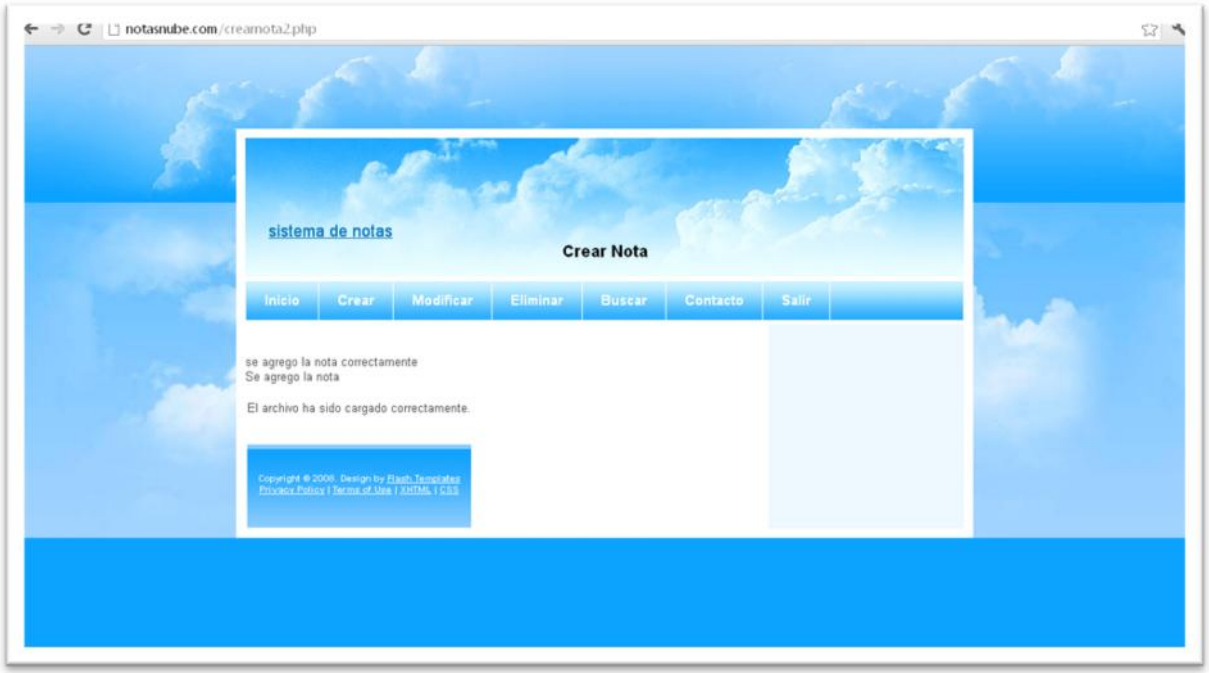

Figura 4.17 Notificación de creación de la nota

En la figura 4.18 se muestra como han sido dados de alta en la base de datos la nueva nota, así también, se muestra en la figura como ha sido también cargado el archivo adjunto a la nota en la tabla *Archivo* de la figura 4.19.

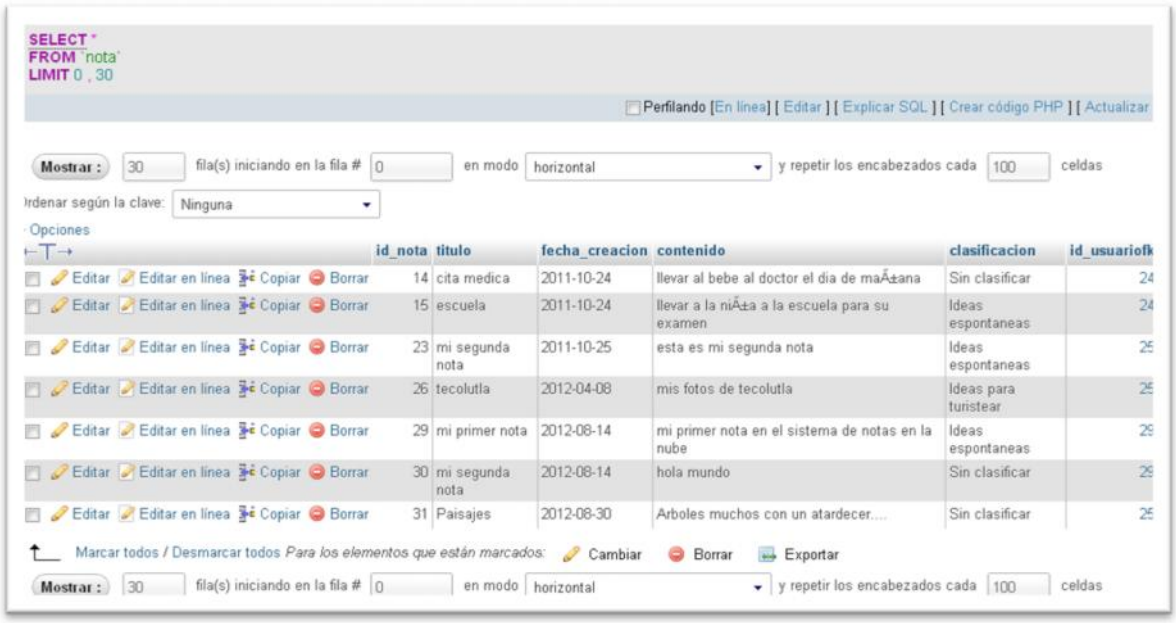

Figura 4.18 Datos almacenados en la tabla de Nota

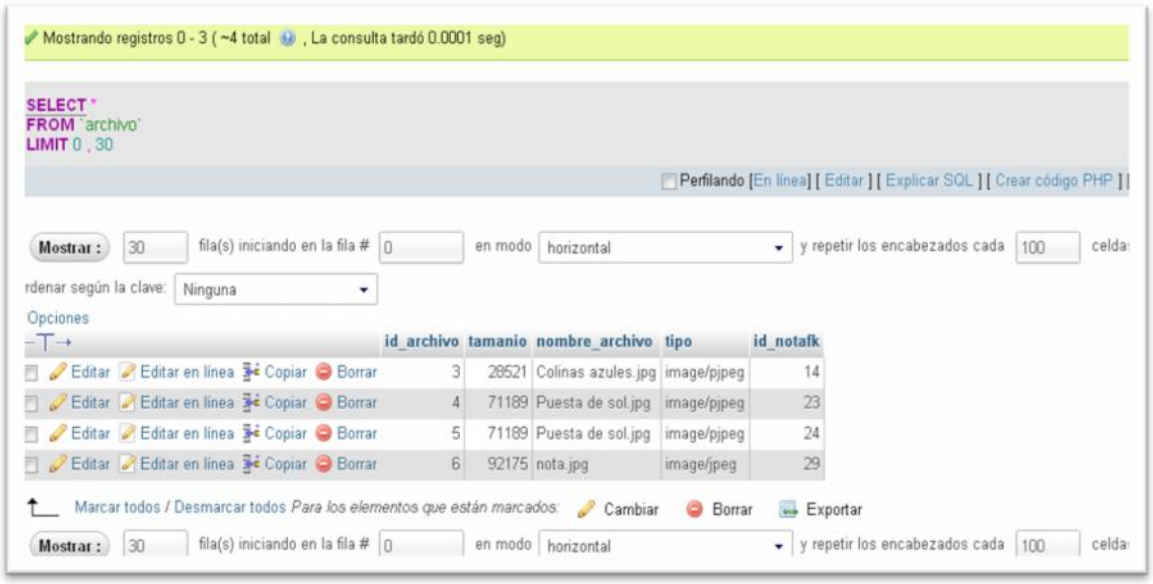

Figura 4.19 Datos almacenados en la tabla Archivo

Para modificar una nota basta con irnos a la sección de Modificar y dar clic en Editar de la nota deseada (ver figura 4.20), para mostrarnos la pantalla de edición con los datos de la nota a editar y listos para se modificados (ver figura 4.21).

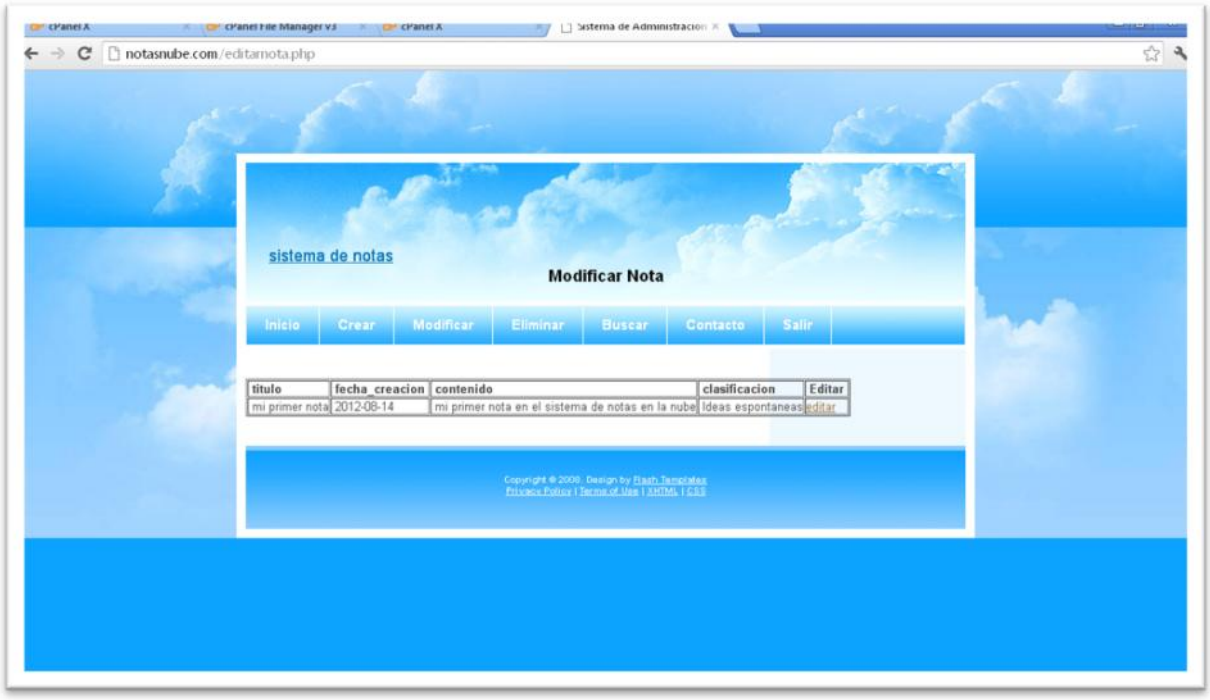

Figura 4.20 Selección de una nota para editar

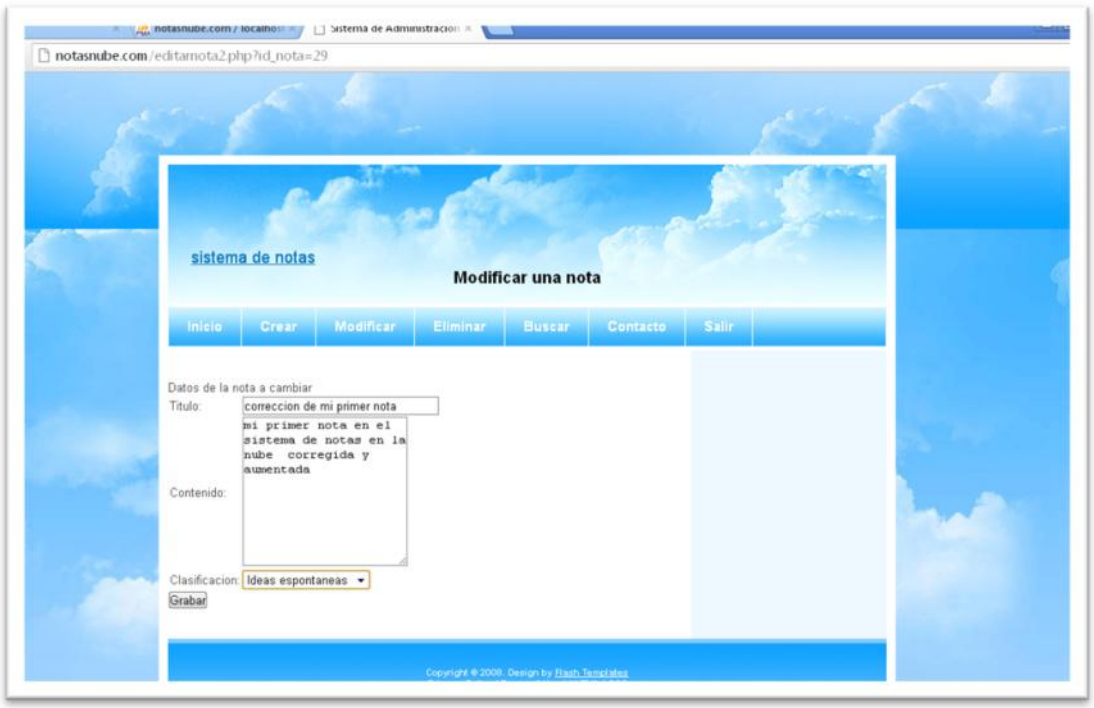

Figura 4.21 Sección de modificación de una nota

En la figura 4.22 se muestran la tabla *Nota* con los cambios hechos a la nota que corresponde al id_nota 29.

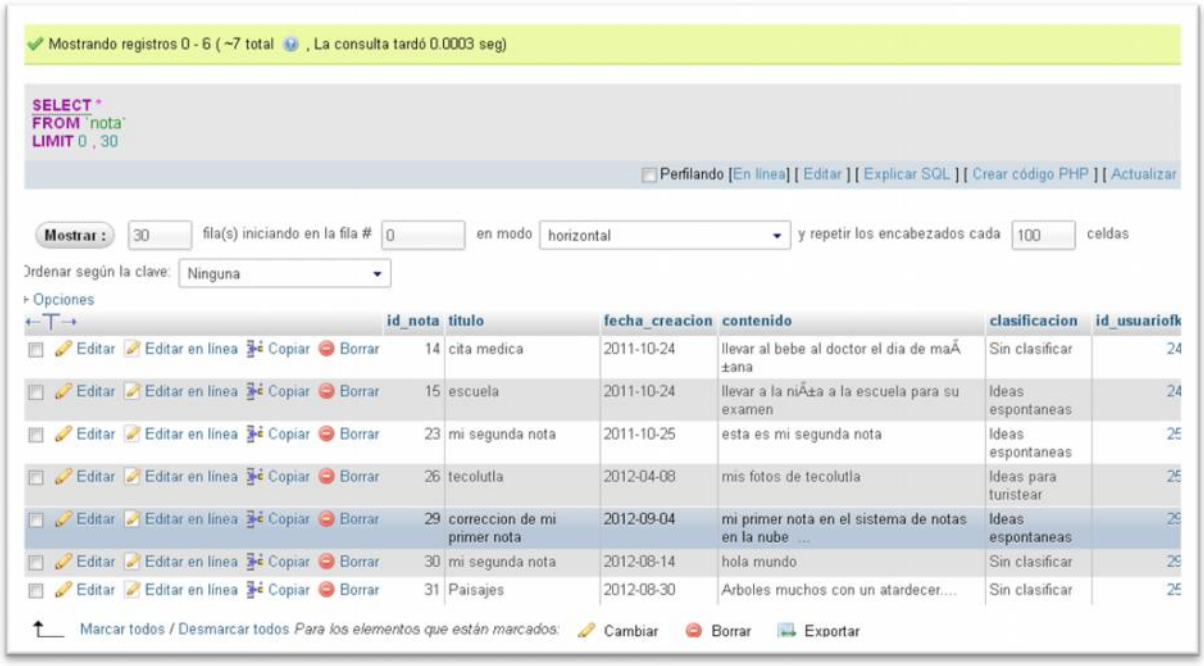

Figura 4.22 Cambios reflejados en la base de datos

En caso de solicitar un listado (ver figura 4.23) de las notas creadas por el usuario, nos tendríamos que ir a la parte de Buscar y nos mostrara un listado de las notas almacenadas.

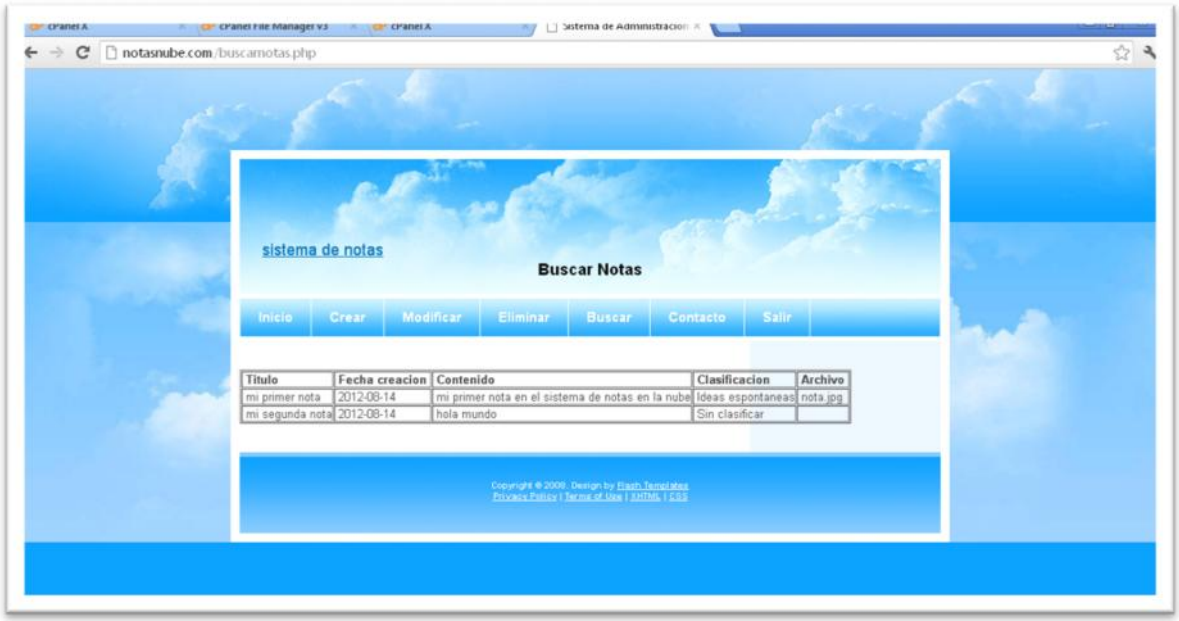

Figura 4.23 Sección de búsqueda de notas

Para eliminar una nota almacenada, es necesario irnos a la opción de Eliminar, donde se nos expondrá el listado de las notas almacenadas, cada una con una opción de eliminar (ver figura 4.24). Se elige la opción de eliminar a la nota deseada y la nota se eliminara de la base de datos.

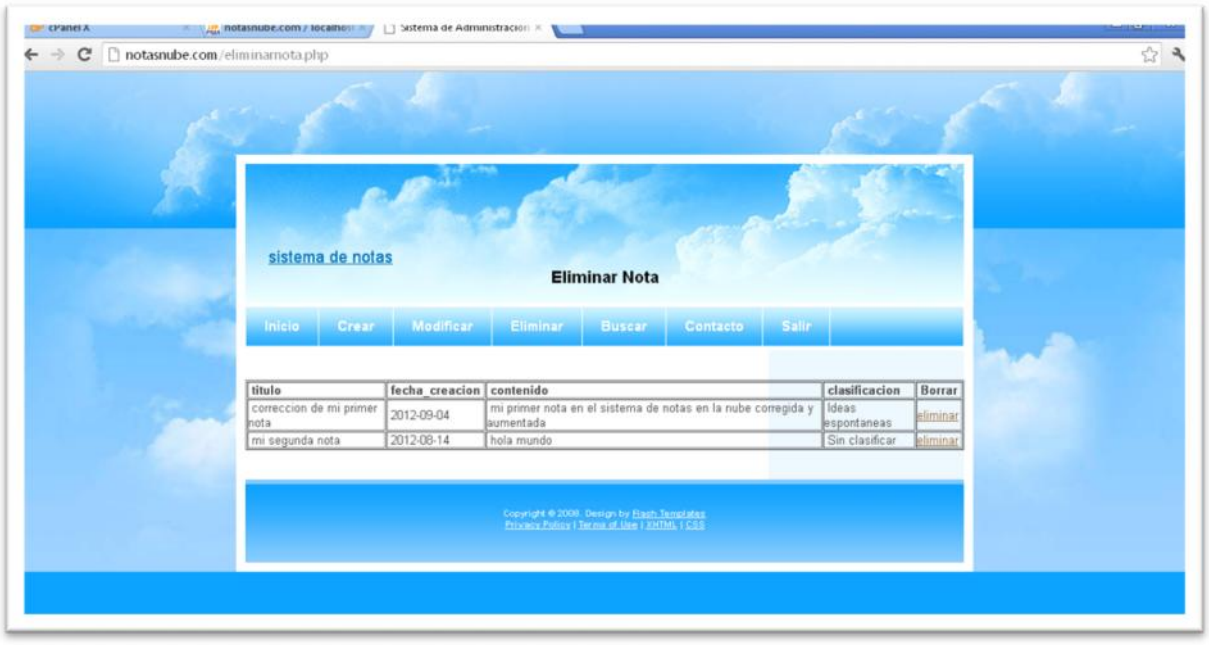

Figura 4.24 Selección de eliminación de una nota

Para el caso particular se creo una nueva nota llamada segunda nota y es la nota que se va a eliminar. En la figura 4.25 se muestra fue eliminada dicha nota.

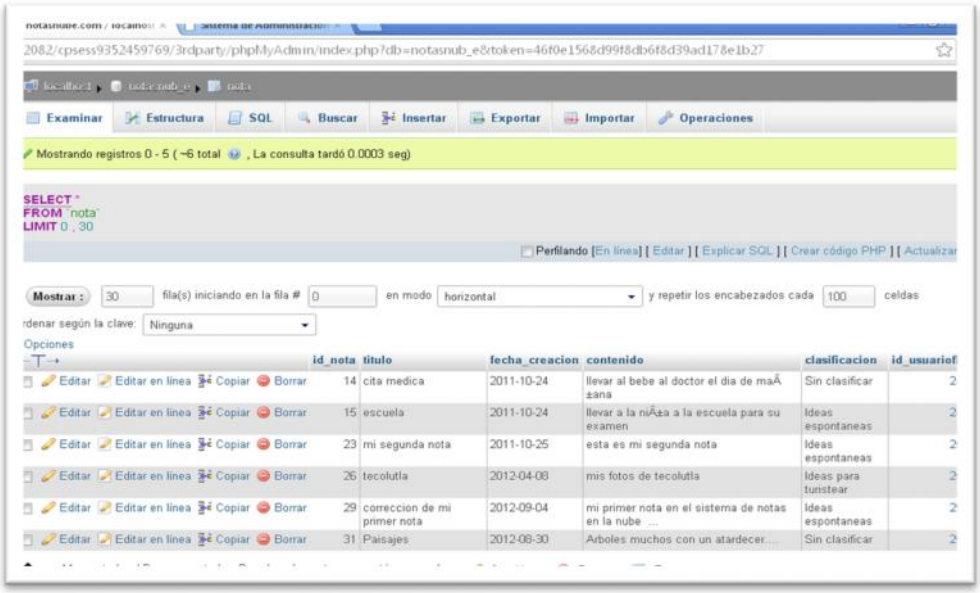

Figura 4.25 Cambios realizados en la tabla Nota después de la eliminación

Dentro del este sistema también es posible el enviar un mensaje al administrador vía correo electrónico a través de la sección de Contacto (ver figura 4.26).

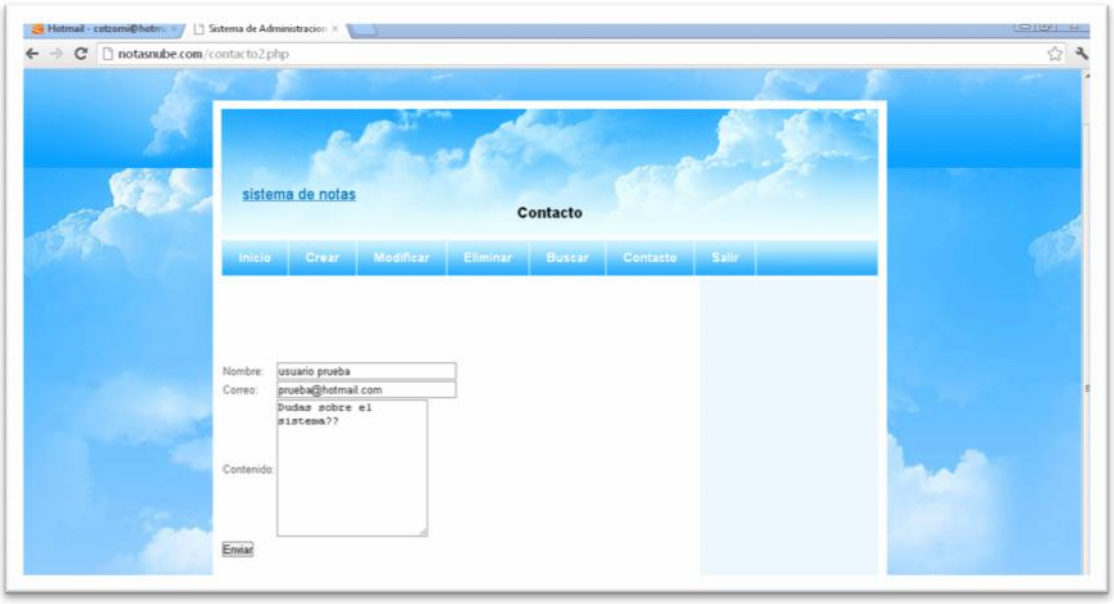

Figura 4.26 Sección de contacto

Desde el correo electrónico del administrador se observa como fue recibido el mensaje (ver figura 4.27).

Figura 4.27 Mensaje recibido en el correo del administrador

CONCLUSIONES

Por ultimo se presentan las conclusiones de las aportaciones hechas por el sistema, aplicación de los conocimientos adquiridos y las ventajas de haber utilizado técnicas de análisis y diseño, tanto del sistema como de la base de datos.

Con el sistema de Notas en la Nube, se mostraron las ventajas de esta nueva visión de la computación, con algo tan sencillo pero tan necesario como lo es una nota, que de nada sirve al tenerla almacenada en un equipo, siendo que se podría consultar en cualquier equipo conectado a Internet y en cualquier momento.

Fue de vital importancia aplicar las técnicas de ingeniería de software, con el fin de hacer más fácil y rápida la construcción del sistema. Además otra de las ventajas es la de tener un modelo que seguir a la hora de la implementación y para su posterior mantenimiento.

Una breve conclusión sobre cada uno de los capítulos se presenta a continuación:

Capitulo 1. En el marco teórico se presentaron las principales teorías acerca del computo en la nube, de los modelos de proceso de software, modelado de base de datos, normalización y una breve descripción de PHP, MySQL y HTML; esto con el fin de presentarle al lector un marco teórico de los temas relacionados con el desarrollo del sistema.

Capitulo 2. En este capitulo nos permitió realizar el análisis del problema para posteriormente realizar el diseño del sistema con la ayuda de los casos de uso y de los diagramas presentados. Este análisis y diseño nos permitió ver al sistema desde diferentes puntos de vista y en distintos momentos, permitiendo un mejor entendimiento del sistema para el futuro mantenimiento.

Capitulo 3. Este capitulo corresponde al diseño de la base de datos y aquí se realizo el análisis, diseño y normalización de la base de datos. Esto nos permitió entender la base de datos desde un nivel alto con las entidades y relaciones, para posteriormente pasarlas al modelo relacional para poder usar las entidades y relaciones en un modelo de tablas. Por ultimo se normalizo con el fin de evitar problemas de redundancia, ambigüedades y anomalías al eliminar o insertar datos.

Capitulo 4. En este capitulo se basa precisamente en la implementación de los modelos obtenidos en los capítulos anteriores, añadiéndole características de seguridad que ayudo al sistema a mantener la integridad y confidencialidad de la información, que para un sistema en Internet es primordial.

TRABAJOS A FUTURO

El presente trabajo es un programa sencillo, el cual no permitió ejemplificar la utilidad de un sistema que trabaja en la nube, pero que tiene muchas áreas de oportunidad, tales como:

- Implementar alarmas en cada nota para notificar de un suceso en un día u hora en particular.
- Programar para que una alarma se notifique ya sea por mensaje de texto, correo o una ventana emergente, de acuerdo al dispositivo en el que se encuentre el usuario.
- Crear una interfaz más amigable con el usuario
- Permitir mediante un sistema calendarizado, como una agenda para crear notificaciones y notas.
- Que se permita adjuntar archivos de mayor tamaño.
- Mejorar el sistema de búsqueda de notas, para que se muestren los resultados mediante criterios mas sofisticados.
- Desarrollar un enlace del sistema con las redes sociales, para que se puedan publicar, si así lo requiere el usuario, sus notas.

BIBLIOGRAFÍA

[1] El lenguaje unificado de modelado. Manual de referencia.
James Rumbaugh, Ivar Jacobson, Grady Booch.
Pearson education S. A. Madrid 2000
ISBN: 84-7829-037-0

[2] Revista Linux+
Junio 2009
ISSN 1732-7121

[3] Cloud Computing For Dummies
Judith Hurwitz, Robin Bloor, Marcia Kaufman, Fern Halper
Wiley Publishing, Inc
ISBN: 978-0-470-48470-8

[4] Revista Electrónica Ingeniería Primero, Universidad Rafael Landivar
No. 19, Octubre 2010, paginas 40-44
ISSN: 2076-3166

[5] Revista Electrónica. Seguridad - Defensa Digital, UNAM
No. 8, 13 de Mayo de 2011
Url:http://revista.seguridad.unam.mx/numero-08/c%c3%b3mputo-en-nube-ventajas-ydesventajas

[6] Ingeniería de Software 7ª. Ed.
Ian Sommerville
Pearson Educación, S. A. Madrid 2005
ISBN: 84-7829-074-5

[7] Ingeniería del Software. Un Enfoque Practico 6ª. Ed.
Roger s. Pressman
Mc. Graw Hill Interamericana
ISBN: 970-10-5473-3

[8] Introducción a los Sistemas de Bases de Datos 7ª. Ed.
C. J. Date
Pearson Education, México, 2001
ISBN: 968-444-419-2

[9] Fundamental of Database Systems 4th. Ed.
Ramez elmasri, Shamkant B. Navathe
Pearson Addison Wesley
ISBN: 0-321-12226-7

[10] Base de Datos –Apuntes- BUAP
M. C. Beatriz Beltrán Martínez
Verano 2011

[11] Transformación ER – Relacional Para el Diseño de Base de Datos Relacionales
Juan Ramón López

[12] Presentaciones del Modulo 1, Análisis y Diseño de base de datos. BUAP.
M.C. Miguel Rodríguez Hernández
Verano 2011

[13] Notas del Modulo1, Análisis de Sistemas de Base de Datos
M.C. Miguel Rodríguez Hernández
Verano 2011

[14] http://www.racsa.co.cr/servicios/empresariales/computo_en_la_nube/faq.html