BENEMERITA UNIVERSIDAD AUTONOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

"SISTEMA ADMINISTRATIVO PARA LA PLANTA FISICA DE LA BUAP"

TESIS PROFESIONAL

PARA OBTENER EL TÍTULO DE

LICENCIADA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

GABRIELA TEHUITZIL COYOTL

ASESORA:

M.C. MELIZA CONTRERAS GONZALEZ

ENERO 2013

Agradecimientos

La presente tesis la dedico:

A **mis padres** por ser mi gran apoyo y base para iniciar una nueva etapa en mi vida, por su apoyo incondicional y esfuerzo para terminar un logro profesional.

Porque compartieron conmigo mis desvelos y desesperación.

A **mi hermana** por acompañarme en las buenas y malas, por hacer de cada día único, especial y divertido.

A **mi hermano** que a pesar de la distancia siempre tengo su apoyo por.

A **mis hermanos** por compartir conmigo mis alegrías.

A **mis amigas** que siempre tuvieron una palabra de aliento para seguir adelante y no caer en el camino, por todos aquellos momentos de diversión y de trabajo.

A **mis profesores** por su tiempo dedicado y enseñanza.

A **mi asesora** de tesis por ser una gran maestra, por sus consejos y sobre todo porque en ella encontré a una amiga.

A **la persona** actualmente está conmigo, por involucrarse en mis alegrías y tristezas, por saber ser mi amigo en los momentos que más lo necesito, por todo ese apoyo que siempre me brinda para seguir adelante, simplemente por ser parte ya de mi vida.

INDICE

Introducción

La Planta Física de la BUAP es un sistema que está diseñado para facilitar tanto al alumno como al público en general a tener una mejor ubicación de cada una de las instalaciones de la BUAP.

Este sistema permitirá a un Administrador ingresar la imagen de un nuevo edifico, salón, así como algunas características para su mejor ubicación.

También le permite a un Secretario Administrativo hacer actualizaciones de información referente a la Facultad, Edificio o Salón.

Esto con el objetivo de presentarle al usuario una forma flexible de la localización y ubicación de cierto inmueble.

1. CAPITULO: Introducción a la Ingeniería de Software

1.1. Ingeniería de Software

La **Ingeniería del Software** es una disciplina que ofrece métodos y técnicas para desarrollar y mantener software de calidad, resuelven problemas de todo tipo, aporta herramientas y procedimientos sobre los que se apoya la ingeniería de software [7].

1. Mejorar la calidad de los productos de software
2. Aumentar la productividad y trabajo de los ingenieros del software.
3. Facilitar el control del proceso de desarrollo de software.
4. Suministrar a los desarrolladores las bases para construir software de alta calidad en una forma eficiente.
5. Definir una disciplina que garantice la producción y el mantenimiento de los productos software desarrollados en el plazo fijado y dentro del costo estimado.

1.2. RUP

El Proceso Unificado de Rational (Rational Unified Process) habitualmente resumido como RUP, es un modelo de software que permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Aunque con el inconveniente de generar mayor complejidad en los controles de administración del mismo, junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Su meta principal es asegurar la producción de software de alta calidad que cumpla con las necesidades de los usuarios, con una planeación y presupuesto predecible.

1.2.1. Características Esenciales

Principales características:

1. Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
2. Pretende implementar las mejores prácticas en Ingeniería de Software.
3. Desarrollo iterativo.
4. Administración de requisitos.
5. Uso de arquitectura basada en componentes.
6. Control de cambios.
7. Modelado visual del software (UML).
8. Verificación de la calidad del software.

1.2.2. Ciclo de Vida RUP

El ciclo de vida RUP es una implementación del desarrollo en espiral que organiza las tareas en fases e iteraciones.

Fase de Inicio: Esta fase tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificar los riesgos asociados al proyecto, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones posteriores.

Fase de elaboración: En esta fase se seleccionan los casos de uso que permiten definir la arquitectura base del sistema, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar.

Fase de Desarrollo: En esta fase es completar la funcionalidad del sistema, para ello se deben clarificar los requisitos pendientes, administrar los cambios de acuerdo a las

evaluaciones realizadas por los usuarios y se realizan las mejoras para el proyecto.

Fase de Cierre: Esta fase asegura que el software esté disponible para los usuarios finales, ajusta los errores y defectos encontrados en las pruebas, capacita a los usuarios y provee el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto.

1.2.3. Que es UML

Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir.

Se puede aplicar en el desarrollo de software gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

1.2.4. Diagramas de Casos de Uso

El modelo de casos de uso describe la funcionalidad propuesta del nuevo sistema. Un caso de uso representa una unidad discreta de interacción entre un usuario (humano o máquina) y el sistema. Cada caso de uso tiene una descripción que muestra la funcionalidad que se construirá en el sistema propuesto puede "incluir" la funcionalidad de otro caso de uso o "extender" a otro caso de uso con su propio comportamiento.

Un actor es un usuario del sistema incluye usuarios humanos y otros sistemas computarizados. Un actor usa un caso de uso para desempeñar alguna porción de trabajo que es de valor para el negocio.

Diagramas de clase

Un diagrama de Clases representa las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Se utilizan para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de convencimiento.

Un diagrama de clases está compuesto por los siguientes elementos: Clase: atributos, métodos y visibilidad. Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

Diagrama de estado

Los diagramas de estado muestran el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación en respuesta a eventos junto con sus respuestas y acciones. Normalmente contienen estados y transiciones. Como los estados y las transiciones incluyen, a su vez, eventos, acciones y actividades.

Diagrama de Secuencia

Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican con qué otros objetos y qué mensajes disparan esas comunicaciones. Los mensajes se muestran como flechas.

Diagrama de actividades

Un diagrama de actividades representa los flujos de trabajo paso a paso de negocio y operacionales de los componentes en un sistema. Un Diagrama de Actividades muestra el flujo de control general.

Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables, o paquetes. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

Diagrama de distribución

Los diagramas de distribución consisten de nodos los cuales representan algún dispositivo de hardware como servidores, estaciones de trabajo, PCs, sensores, teléfonos entre otros y muestran la configuración de los nodos, procesos, componentes y objetos que residen en ellos en tiempo de ejecución. Las conexiones entre los nodos corresponden a las conexiones de red entre ellos, representadas como estereotipos.

1.3. Modelos de Proceso de Software

El modelo de proceso de software se define como "Una representación simplificada de un proceso de software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de procesos del software es una abstracción de un proceso real."

Los modelos genéricos no son descripciones definitivas de procesos de software; sin embargo, son abstracciones útiles que pueden ser utilizadas para explicar diferentes enfoques del desarrollo de software.

1.3.1. Modelo Lineal o Cascada

También llamado "Ciclo de vida básico" o "Modelo de cascada" tiene su origen en el "Modelo de cascada" ingeniado por Winston Royce, sugiere un enfoque sistemático o más bien secuencial del desarrollo de software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento y contempla las siguientes actividades:

1. **Análisis de los requerimientos del software**: es la fase en la cual se reúnen todos los requisitos que debe cumplir el software. En esta etapa es fundamental la presencia del cliente que documenta y repasa dichos requisitos.
2. **Diseño:** es una etapa dirigida hacia la estructura de datos, la arquitectura del software, las representaciones de la interfaz y el detalle procedimental (algoritmo).
3. **Generación del código:** es la etapa en la cual se traduce el diseño para que sea comprensible por la máquina.
4. **Pruebas:** esta etapa se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado y en la detección de errores.
5. **Mantenimiento:** debido a que el programa puede tener errores, puede no ser del completo agrado del cliente o puede necesitar eventualmente acoplarse a los cambios en su entorno, entonces se realizan cambios sobre la base de uno ya existente.

1.3.2. Modelo Espiral

El modelo espiral para la ingeniería de software ha sido desarrollado para cubrir las mejores características tanto del ciclo de vida clásico, como de la creación de prototipos, añadiendo al mismo tiempo un nuevo elemento: el análisis de riesgo. El modelo representado mediante la espiral define cuatro actividades principales:

1. **Planificación:** determinación de objetivos, alternativas y restricciones.
2. **Análisis de riesgo:** análisis de alternativas e identificación/resolución de riesgos.
3. **Ingeniería:** desarrollo del producto del "siguiente nivel".
4. **Evaluación del cliente:** Valorización de los resultados de la ingeniería.

En la primera vuelta del espiral se definen los objetivos, las alternativas y las restricciones, se analizan e identifican los riesgos. Si el análisis de riesgo indica que hay una incertidumbre en los requisitos, se puede usar la creación de prototipos en el cuadrante de ingeniería para dar asistencia tanto al encargado de desarrollo como al cliente.

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

Utiliza un enfoque evolutivo para la ingeniería de software, permitiendo al desarrollador y al cliente entender y reaccionar a los riesgos en cada nivel evolutivo. Utiliza la creación de prototipos como un mecanismo de reducción de riesgo y permite a quien lo desarrolla aplicar el enfoque de creación de prototipos en cualquier etapa de la evolución de prototipos.

1.3.3. Modelo de Prototipos

Este modelo arranca con el establecimiento de los requerimientos del sistema, se definen los objetivos del sistema y los requisitos conocidos con base en las áreas de mayor prioridad e importancia para el sistema.

Un prototipo es una representación o modelo del producto de programación que a diferencia de un modelo de simulación, incorpora componentes del producto real.

Al igual que el ciclo de vida clásico, la construcción de prototipos puede ser problemática por las siguientes razones:

1. El cliente ve funcionando lo que parece ser una primera versión del software. Cuando se le informa al cliente de que el producto debe ser reconstruido, se vuelve loco y solicita se aplique "cuantas mejoras" sean necesarias para hacer del prototipo un producto final que funcione.

2. El técnico de desarrollo, frecuentemente impone ciertos compromisos de implementación con el fin de obtener un prototipo que funciones rápidamente. Puede que utilice un sistema operativo inapropiado, o un lenguaje de programación equívoco.

1.3.4. Proceso Unificado de Desarrollo del Software

El Proceso Unificado es un proceso de software genérico que puede ser utilizado para una gran cantidad de tipos de sistemas de software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de competencia y diferentes tamaños de proyectos.

Su meta es asegurar la producción de software de muy alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible.

El Proceso Unificado tiene dos dimensiones:

1. Un eje horizontal que representa el tiempo y muestra los aspectos del ciclo de vida del proceso a lo largo de su desenvolvimiento (dinámico).
2. Un eje vertical que representa las disciplinas, las cuales agrupan actividades de una manera lógica de acuerdo a su naturaleza (estático).

El Proceso Unificado se basa en componentes, lo que significa que el sistema en construcción está hecho de componentes de software interconectados por medio de interfaces bien definidas.

Los aspectos distintivos del Proceso Unificado están capturados en tres conceptos clave: dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Esto es lo que hace único al Proceso Unificado.

1.3.5. Modelo Incremental

El modelo incremental es el proceso de construcción siempre incrementando subconjuntos de requerimientos del sistema. Típicamente, un documento de requerimientos es escrito al capturar todos los requerimientos para el sistema completo.

Note que el desarrollo incremental es 100% compatible con el modelo cascada. El modelo de desarrollo incremental provee algunos beneficios significativos para los proyectos:

1. Construir un sistema pequeño es siempre menos riesgoso que construir un sistema grande.
2. Al ir desarrollando parte de las funcionalidades, es más fácil determinar si los requerimientos planeados para los niveles subsiguientes son correctos.
3. Si un error importante es realizado, sólo la última iteración necesita ser descartada.
4. Reduciendo el tiempo de desarrollo de un sistema decrecen las probabilidades que esos requerimientos de usuarios puedan cambiar durante el desarrollo.
5. Si un error importante es realizado, el incremento previo puede ser usado.

2. CAPITULO: Base de Datos

2.1. Introducción

Base de Datos es un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquina accesibles en tiempo real y compatibles con usuarios concurrentes con necesidad de información diferente.

2.1.1. Definición de Bases de Datos

Una **base de datos** es una entidad en la cual se pueden almacenar datos de manera estructurada, con la menor redundancia posible.

Una base de datos proporciona a los usuarios el acceso a datos, que pueden visualizar, ingresar o actualizar, con los derechos de acceso que se les hayan otorgado.

Una base de datos puede ser local, es decir que puede utilizarla sólo un usuario en un equipo, o puede ser distribuida, es decir que la información se almacena en equipos remotos y se puede acceder a ella a través de una red. La principal ventaja de utilizar bases de datos es que múltiples usuarios pueden acceder a ellas al mismo tiempo.

2.1.2. DBMS

Un sistema de administración de bases de datos DBMS (Database Management System) es un sistema basado en computador (software) que maneja una base de datos o una colección de bases de datos o archivos. La persona que administra un DBMS es conocida como el DBA (Database Administrator).

Está compuesto por:

1. **DDL:** Lenguaje de Definición de Datos.
2. **DML:** Lenguaje de Manipulación de Datos.
3. **SQL:** Lenguaje de Consulta.

2.1.3. Usos y Funciones de un DBMS

Los sistemas de administración de bases de datos son usados para permitir a los usuarios acceder y manipular la base de datos proveyendo métodos para construir sistemas de procesamiento de datos para aplicaciones que requieran acceso a los datos.

Algunas de las funciones de un DBMS son:

1. Definición de la base de datos: como la información que va a ser almacenada y organizada.
2. Creación de la base de datos: almacenamiento de datos en una base de datos definida.
3. Recuperación de los datos: consultas y reportes.
4. Actualización de los datos: cambiar los contenidos de la base de datos.
5. Programación de aplicaciones de para el desarrollo de software.
6. Control de la integridad de la base de datos.
7. Monitoreo del comportamiento de la base de datos.

El DBMS puede dividirse en tres subsistemas:

1. El sistema de administración de archivos para almacenar información en un medio físico.
2. El DBMS interno para ubicar la información en orden.
3. El DBMS externo representa la interfaz del usuario.

2.1.4. Usuarios de la Base de Datos

Podemos definir a los usuarios como toda persona que tenga todo tipo de contacto con el sistema de base de datos desde que este se diseña, elabora, termina y se usa.

Los usuarios que accedan una base de datos pueden clasificarse como:

Programadores de aplicaciones: Los profesionales en computación que interactúan con el sistema por medio de llamadas en DML (Lenguaje de Manipulación de Datos), las cuales están incorporadas en un programa escrito en un lenguaje de programación (Por ejemplo, COBOL, PL/I, Pascal, C, etc.).

Usuarios sofisticados: Los usuarios sofisticados interactúan con el sistema sin escribir programas en cambio escriben sus preguntas en un lenguaje de consultas de base de datos.

Usuarios especializados: Algunos usuarios sofisticados escriben aplicaciones de base de datos especializadas que no encajan en el marco tradicional de procesamiento de datos.

Usuarios ingenuos: Los usuarios no sofisticados interactúan con el sistema invocando a uno de los programas de aplicación permanentes que se han escrito anteriormente en el sistema de base de datos, podemos mencionar al usuario ingenuo como el usuario final que utiliza el sistema de base de datos sin saber nada del diseño interno del mismo.

2.2. Diseño de Bases Datos

En esta sección se describen con más detalle los objetivos de cada una de las etapas del diseño de bases de datos: diseño conceptual, diseño lógico y diseño físico. La metodología a seguir en cada una de estas etapas se describe en los capítulos que siguen a éste.

2.2.1. Diseño Conceptual

En esta etapa se debe construir un esquema de la información que se usa en la empresa, independientemente de cualquier consideración física. A este esquema se le denomina esquema conceptual. Al construir el esquema, los diseñadores descubren la semántica (significado) de los datos de la empresa: encuentran entidades, atributos y relaciones.

El esquema conceptual se puede utilizar para que el diseñador transmita a la empresa lo que ha entendido sobre la información que ésta maneja. Para ello, ambas partes deben estar familiarizadas con la notación utilizada en el esquema.

La más popular es la notación del modelo entidad-relación, que se describirá en el capítulo dedicado al diseño conceptual.

El diseño conceptual es completamente independiente de los aspectos de implementación, como puede ser el SGBD que se vaya a usar, los programas de aplicación, los lenguajes de programación, el hardware disponible o cualquier otra consideración física.
El esquema conceptual es una fuente de información para el diseño lógico de la base de datos.

2.2.2. Diseño Lógico

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

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

El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos.

El **diseño conceptual** y **el diseño lógico** son etapas clave para conseguir un sistema que funcione correctamente. Si el esquema no es una representación fiel de la empresa, será difícil si no imposible, definir todas las vistas de usuario (esquemas externos) o mantener la integridad de la base de datos.

2.2.3. Diseño Físico

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

Para llevar a cabo esta etapa, se debe haber decidido cuál es el SGBD que se va a utilizar, ya que el esquema físico se adapta a él.

Entre el diseño físico y el diseño lógico hay una realimentación, ya que algunas de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico.

En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico y consiste en:

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

2.3. Diseño de Aplicaciones

En esta sección se examinan los dos aspectos del diseño de las aplicaciones: el diseño de las transacciones y el diseño de las interfaces de usuario.

2.3.1. Diseño de Transacciones

Una transacción es un conjunto de acciones llevadas a cabo por un usuario o un programa de aplicación, que acceden o cambian el contenido de la base de datos.

Las transacciones representan eventos del mundo real, como registrar un inmueble para ponerlo en alquiler, concertar una visita con un cliente a un inmueble, dar de alta un nuevo empleado o registrar un nuevo cliente.

Una transacción puede estar compuesta por varias operaciones, como la transferencia de dinero de una cuenta bancaria a otra. Sin embargo, desde el punto de vista del usuario, estas operaciones conforman una sola tarea. Desde el punto de vista del SGBD, una transacción lleva a la base de datos de un estado consistente a otro estado consistente.

El SGBD garantiza la consistencia de la base de datos incluso si se produce algún fallo y también garantiza que una vez que se ha finalizado una transacción, los cambios realizados por ésta quedan permanentemente en la base de datos, no se pueden perder ni deshacer (a menos que se realice otra transacción que compense el efecto de la primera).

El objetivo del diseño de las transacciones es definir y documentar las características de alto nivel de las transacciones que requiere el sistema. Esta tarea se debe llevar a cabo al principio del proceso de diseño para garantizar que el esquema lógico es capaz de soportar todas las transacciones necesarias.

Las características que se deben recoger de cada transacción son las siguientes:

1. Datos que utiliza la transacción.
2. Características funcionales de la transacción.
3. Salida de la transacción.
4. Importancia para los usuarios.
5. Frecuencia de utilización.

Hay tres tipos de transacciones:

1. En las transacciones **de recuperación** se accede a los datos para visualizarlos en la pantalla a modo de informe.
2. En las transacciones **de actualización** se insertan, borran o actualizan datos de la base de datos.
3. En las transacciones **mixtas** se mezclan operaciones de recuperación de datos y de actualización.

2.3.2. Diseño de Interfaces de Usuario

Antes de implementar los formularios y los informes hay que diseñar su aspecto y tener en cuenta las siguientes recomendaciones:

1. Utilizar títulos que sean significativos, que identifiquen sin ambigüedad el propósito del informe o formulario.
2. Dar instrucciones breves y fáciles de comprender.
3. Agrupar y secuenciar los campos de forma lógica.
4. Hacer que el aspecto del informe o formulario sea atractivo a la vista.
5. Utilizar nombres familiares para etiquetar los campos.
6. Utilizar terminología y abreviaturas consistentes.
7. Hacer un uso razonable y consistente de los colores.
8. Dejar un espacio visible para los datos de entrada y delimitarlos.
9. Permitir un uso sencillo y adecuado del cursor.
10. Permitir la corrección carácter a carácter y de campos completos.
11. Dar mensajes de error para los valores "ilegales''.
12. Marcar los campos que sean opcionales.

13. Dar mensajes a nivel de campo para explicar su significado.
14. Dar una señal que indique cuándo el informe o formulario está completo.

2.4. Metodología de Diseño de Bases Datos

El diseño de una base de datos es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en sub-problemas y se resuelve cada uno de estos sub-problemas independientemente, utilizando técnicas específicas. Así, el diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico.

Un modelo de datos es una serie de conceptos que puede utilizarse para describir un conjunto de datos y las operaciones para manipularlos. Los modelos conceptuales se utilizan para representar la realidad a un alto nivel de abstracción.

En el diseño de bases de datos se usan primero los modelos conceptuales para lograr una descripción de alto nivel de la realidad y luego se transforma el esquema conceptual. El motivo de realizar estas dos etapas es la dificultad de abstraer la estructura de una base de datos que presente cierta complejidad.

Los modelos conceptuales deben ser buenas herramientas para representar la realidad, por lo que deben poseer las siguientes cualidades:

1. **Expresividad:** deben tener suficientes conceptos para expresar perfectamente la realidad.
2. Simplicidad: deben ser simples para que los esquemas sean fáciles de entender.

3. **Minimalidad:** cada concepto debe tener un significado distinto.
4. **Formalidad:** todos los conceptos deben tener una interpretación única, precisa y bien definida.

En general, un modelo no es capaz de expresar todas las propiedades de una realidad determinada, por lo que hay que añadir aserciones que complementen el esquema.

2.5. El Modelo Entidad-Relación

El modelo **entidad-relación** es el modelo conceptual más utilizado para el diseño conceptual de bases de datos, fue introducido por Peter Chen en 1976. Está formado por un conjunto de conceptos que permiten describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas.

Originalmente, el modelo entidad-relación sólo incluía los conceptos de entidad, relación y atributo. Más tarde, se añadieron otros conceptos, como los atributos compuestos y las jerarquías de generalización, en lo que se ha denominado modelo entidad-relación extendido.

2.5.1. Entidad

Cualquier tipo de objeto o concepto sobre el que se recoge información: cosa, persona, concepto abstracto o suceso. Por ejemplo: coches, casas, empleados, clientes, empresas, oficios, diseños de productos, conciertos, excursiones, etc.

Las entidades se representan gráficamente mediante rectángulos y su nombre aparece en el interior. Un nombre de entidad sólo puede aparecer una vez en el esquema conceptual.

Hay dos tipos de entidades: fuertes y débiles.

1. Una entidad débil es una entidad cuya existencia depende de la existencia de otra entidad.
2. Una entidad fuerte es una entidad que no es débil.

2.5.2. Relación

Es una correspondencia o asociación entre dos o más entidades. Cada relación tiene un nombre que describe su función. Las relaciones se representan gráficamente mediante rombos y su nombre aparece en el interior.

El número de participantes en una relación es lo que se denomina grado de la relación. Por lo tanto, una relación en la que participan dos entidades es una relación binaria; si son tres las entidades participantes, la relación es ternaria; etc.

La participación de una entidad en una relación es obligatoria (total) si la existencia de cada una de sus ocurrencias requiere la existencia de, al menos, una ocurrencia de la otra entidad participante. Si no, la participación es opcional (parcial). Las reglas que definen la cardinalidad de las relaciones son las reglas de negocio.

2.5.3. Atributo

Es una característica de interés o un hecho sobre una entidad o sobre una relación. Los atributos representan las propiedades básicas de las entidades y de las relaciones.

Cada atributo tiene un conjunto de valores asociados denominado dominio. El dominio define todos los valores posibles que puede tomar un atributo. Puede haber varios atributos definidos sobre un mismo dominio los cuales pueden ser simples o compuestos.

Un **atributo simple** es un atributo que tiene un solo componente, que no se puede dividir en partes más pequeñas que tengan un significado propio. Un **atributo compuesto** es un atributo con varios componentes, cada uno con un significado por sí mismo.

Un grupo de atributos se representa mediante un atributo compuesto cuando tienen afinidad en cuanto a su significado, o en cuanto a su uso. Un atributo compuesto se representa gráficamente mediante un óvalo.

Los atributos también pueden clasificarse en monovalentes o polivalentes. Un **atributo monovalente** es aquel que tiene un solo valor para cada ocurrencia de la entidad o relación a la que pertenece. Un **atributo polivalente** es aquel que tiene varios valores para cada ocurrencia de la entidad o relación a la que pertenece.

A estos atributos también se les denomina multivaluados, y pueden tener un número máximo y un número mínimo de valores. La cardinalidad de un atributo indica el número mínimo y el número máximo de valores que puede tomar para cada ocurrencia de la entidad o relación a la que pertenece. El valor por omisión es (1, 1).

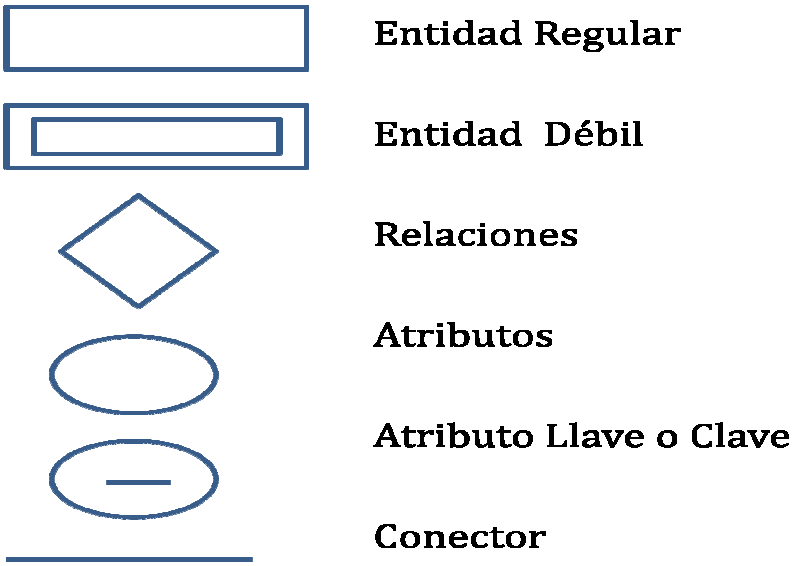

Figura 2.5.3 Representación gráfica de entidad, relación y atributo.

2.5.4. Identificador

Un identificador de una entidad es un atributo o conjunto de atributos que determina de modo único cada ocurrencia de esa entidad. Un identificador de una entidad debe cumplir dos condiciones:

1. No pueden existir dos ocurrencias de la entidad con el mismo valor del identificador.
2. Si se omite cualquier atributo del identificador, la condición anterior deja de cumplirse.

2.5.5. Jerarquía de Generalización

Una entidad E es una generalización de un grupo de entidades E□, E^2,.....E^n, si cada ocurrencia de cada una de esas entidades es también una ocurrencia de E. Todas las propiedades de la entidad genérica E son heredadas por las sub-entidades. Cada jerarquía es total o parcial, y exclusiva o superpuesta.

Una jerarquía es total si cada ocurrencia de la entidad genérica corresponde al menos con una ocurrencia de alguna sub-entidad.

Es parcial si existe alguna ocurrencia de la entidad genérica que no corresponde con ninguna ocurrencia de ninguna sub-entidad. Una jerarquía es exclusiva si cada ocurrencia de la entidad genérica corresponde, como mucho, con una ocurrencia de una sola de las sub-entidades.

Es superpuesta si existe alguna ocurrencia de la entidad genérica que corresponde a ocurrencias de dos o más sub-entidades diferentes.

2.6. Modelado de Bases de Datos con UML

Los objetos brindan confiabilidad, flexibilidad y eficiencia a los sistemas de software, haciéndoles frente a los diseñadores y a los arquitectos de hoy en día con muchas opciones. Desde el punto de vista de la tecnología, la opción esta generalmente entre orientación a objetos pura, híbrido relacional-objeto, relacional puro, y soluciones basadas en los formatos de archivos abiertos o propietarios. Oracle, IBM, Microsoft, POET ofrecen soluciones similares pero a menudo incompatibles [1].

Para cada dominio veremos solamente las características principales que afectan nuestra tarea. Entonces miraremos las técnicas y las ediciones implicadas en el mapeo del modelo de clases al modelo de base de datos, incluyendo persistencia del objeto, comportamiento del objeto, relaciones entre los objetos e identidad de los objetos. Concluiremos con una revisión del perfil de los datos de UML (según lo propuesto por Rational Software).Una cierta familiarización con diseño orientado a objetos, UML y modelado de base de datos relacional es asumida.

El modelo de clase en UML es el artefacto principal producido para representar la estructura lógica de un sistema de software. Captura los requisitos de los datos y el comportamiento de objetos dentro del dominio de modelo.

2.6.1. Modelo Relacional

El **modelo relacional** para la gestión de una base de datos es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos.

Su idea fundamental es el uso de relaciones. Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados tuplas.

El modelo propuesto por Codd presentaba los siguientes objetivos:

Independencia Física. El modo en que se almacenan los datos no debe influir en su manipulación lógica y por lo tanto, los usuarios que acceden a los datos no han de modificar sus programas por cambios en el almacenamiento físico.

Independencia Lógica. Anadir, eliminar o modificar cualquier elemento de la base de datos no debe repercutir en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos.

Flexibilidad. Poder ofrecer a cada usuario los datos de la forma más adecuada a su aplicación.

Uniformidad. Las estructuras lógicas de los datos presentan un aspecto uniforme (tablas), lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.

Sencillez. Las características anteriores, así como unos lenguajes de usuario sencillos, producen el resultado de que el modelo de datos relacional es fácil de comprender y de utilizar.

El modelo relacional está basado en la relación como elemento básico y puede ser representado como una tabla.

2.6.2. Concepto de Valor Nulo en el Modelo Relacional.

Si bien los valores nulos no son un concepto exclusivo del modelo relacional, ha sido en el contexto de este modelo donde se ha abordado su estudio de manera más sistemática y donde se están realizando más investigaciones a fin de formalizar su tratamiento.

Se puede definir el valor nulo (también denominado valor ausente) como una señal utilizada para representar información desconocida, inaplicable, inexistente, no valida, no proporcionada, indefinida, etc.

Su necesidad en las bases de datos es evidente por diversas razones:

1. Crear tuplas (filas) con ciertos atributos desconocidos en ese momento, por ejemplo, el año de edición de un libro.
2. Anadir un nuevo atributo a una relación existente; atributo que, en el momento de añadirse, no tendría ningún valor para las tuplas de la relación.
3. Atributos inaplicables a ciertas tuplas, por ejemplo, la editorial para un artículo o la profesión para un menor.

2.6.3. Llaves Primarias y Foráneas

2.6.3.1. Llave Primaria

En base de datos, una llave primaria es un conjunto de uno o más atributos de una tabla, que tomados colectivamente nos permiten identificar un registro como único, es decir, en una tabla podemos saber cual es un registro en específico sólo con conocer la llave primaria.

En una arquitectura entidad-relación la llave primaria permite las relaciones de la tabla que tiene la llave primaria, con otras tablas que van a utilizar la información de esta tabla.

La llave primaria debe permitirle a un Sistema de Gestión de Base de Datos (SGBD), correctamente proyectado, generar un error si un usuario intenta incluir un nuevo registro cuya llave primaria coincida con la de otro registro ya existente en el archivo.

En consecuencia en cada archivo solo podrá existir un único registro que posea un valor determinado para su llave primaria. En otras palabras no puede existir en un archivo un registro que cuente con el mismo valor de otro registro en el campo de la llave primaria; la llave primaria no puede tener valores repetidos para distintos registros.

2.6.3.2. Llave Foránea

En el contexto de bases de datos relacionales, una clave foránea o clave ajena (o Foreign Key FK) es una limitación referencial entre dos tablas. La clave foránea identifica una columna o grupo de columnas en una tabla (tabla hija o referendo) que se refiere a una columna o grupo de columnas en otra tabla (tabla maestra o referenciada). Las columnas en la tabla referendo deben ser la clave primaria u otra clave candidata en la tabla referenciada.

Los valores en una fila de las columnas referendo deben existir solo en una fila en la tabla referenciada. Así, una fila en la tabla referendo no puede contener valores que no existen en la tabla referenciada.

De esta forma, las referencias pueden ser creadas para vincular o relacionar información. Esto es una parte esencial de la normalización de base de datos. Múltiples filas en la tabla referendo pueden hacer referencia, vincularse o relacionarse a la misma fila en la tabla referenciada. Mayormente esto se ve reflejado en una relación uno (tabla maestra o referenciada) a muchos (tabla hija o referendo).

Los valores en una fila de las columnas referendo deben existir solo en una fila en la tabla referenciada. Así, una fila en la tabla referendo no puede contener valores que no existen en la tabla referenciada. De esta forma, las referencias pueden ser creadas para vincular o relacionar información. Esto es una parte esencial de la normalización de base de datos.

Múltiples filas en la tabla referendo pueden hacer referencia, vincularse o relacionarse a la misma fila en la tabla referenciada. Mayormente esto se ve reflejado en una relación uno (tabla maestra o referenciada) a muchos (tabla hija o referendo).

La tabla referendo y la tabla referenciada pueden ser la misma, esto es, la clave foránea remite o hace referencia a la misma tabla. Una tabla puede tener múltiples claves foráneas y cada una puede tener diferentes tablas referenciadas. Cada clave foránea es forzada independientemente por el sistema de base de datos. Por tanto, las relaciones en cascada entre tablas pueden realizarse usando claves foráneas.

Configuraciones impropias de las claves foráneas o primarias o no forzar esas relaciones son frecuentemente la fuente de muchos problemas para la base de datos o para el modelamiento de los mismos.

2.7. Normalización

Normalización es un proceso que clasifica relaciones, objetos, formas de relación y demás elementos en grupos, en base a las características que cada uno posee. Si se identifican ciertas reglas, se aplica una categoría; si se definen otras reglas, se aplicará otra categoría [4].

2.7.1. Definición de Normalización

La normalización es el proceso de organizar los datos de una base de datos. Se incluye la creación de tablas y el establecimiento de relaciones entre ellas según reglas diseñadas tanto para proteger los datos como para hacer que la base de datos sea más flexible al eliminar la redundancia y las dependencias incoherentes.

Otra ventaja de la normalización de su base de datos es el consumo de espacio. Una base de datos normalizada puede ocupar menos espacio en disco que una no normalizada. Hay menos repetición de datos, lo que tiene como consecuencia un mucho menor uso de espacio en disco. Mientras que los datos redundantes desperdician el espacio de disco y crean problemas de mantenimiento. Si hay que cambiar datos que existen en más de un lugar, se deben cambiar de la misma forma exactamente en todas sus ubicaciones. Un cambio en la dirección de un cliente es mucho más fácil de implementar si los datos sólo se almacenan en la tabla Clientes y no en algún otro lugar de la base de datos.

2.7.2. Formas Normales

El principal objetivo común de las formas normales aplicadas a las bases de datos es optimizar las tablas, dividiendo los datos ingresados en estas, pero sin perder ninguno de estos, para así

mejorar la respuesta de las tablas a las consultas que el usuario pudiese generar en el uso de una base de dato “x”.

Las Formas Normales (NF) nos proporcionan los criterios para determinar el grado de vulnerabilidad de una tabla en cuanto a inconsistencias y anomalías lógicas. Mientras más alta sea la forma normal aplicada a una tabla, es menos vulnerable a inconsistencias y anomalías. Cada tabla tiene una "forma normal más alta" (Hight Normal Form): por definición, una tabla siempre satisface los requisitos de su forma normal más alta; también por definición, una tabla no puede satisfacer los requisitos de ninguna forma normal más arriba que su forma normal más alta.

Las NF son aplicables a tablas individuales; decir que una base de datos entera está en la forma normal x es decir que todas sus tablas están en la forma normal x.

2.7.2.1. Primera Forma Normal

La primera forma normal (1FN) es una forma normal usada en normalización de bases de datos. Una tabla de base de datos relacional que se adhiere a la 1FN es una que satisface cierto conjunto mínimo de criterios. Estos criterios se refieren básicamente a asegurarse que la tabla es una representación fiel de una relación y está libre de grupos repetitivos.

Una relación R se encuentra en 1FN si y solo sí por cada renglón columna contiene valor es atómicos.

Una tabla está en Primera Forma Normal si:

1. Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son indivisibles, mínimos.
2. La tabla contiene una llave primaria única.
3. La llave primaria no contiene atributos nulos.
4. No debe existir variación en el número de columnas.
5. Los Campos no llave deben identificarse por la llave (Dependencia Funcional)
6. Debe existir una independencia del orden tanto de las filas como de las columnas, es decir, si los datos cambian de orden no deben cambiar sus significados.
7. Una tabla no puede tener múltiples valores en cada columna.
8. Los datos son atómicos (a cada valor de X le pertenece un valor de Y y viceversa).

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

2.7.2.2. Segunda Forma Normal

Una relación R se encuentra en 2FN si y solo si esta en 1FN, y los atributos no primos dependen de la llave primaria.

Una relación está en 2FN si está 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.

Una relación se encuentra en segunda forma normal, cuando cumple con las reglas de la primera forma normal y todos sus atributos que no son claves (llaves) dependen por completo de

la clave. Cada tabla que tiene un atributo único como clave, está en segunda forma normal.

2.7.2.3. Tercera Forma Normal

Una relación estará en 3FN si y sólo si está en 2FN y además existen atributos no claves que dependen de otros atributos no claves de la entidad compleja. Estos atributos no claves tienen relación transitiva con la entidad principal [3].

2.8. Arquitectura- Cliente Servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios llamados servidores y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor le da respuesta [6].

Cliente: Es una aplicación informática o un computador que consume un servicio remoto en otro computador, conocido como servidor, normalmente a través de una red de telecomunicaciones.1

Servidor: Es una computadora que, formando parte de una red, provee servicios a otras computadoras denominadas clientes.

2.8.1. Características de Cliente

1. Es quien inicia solicitudes o peticiones, tienen por tanto un papel activo en la comunicación (dispositivo maestro o amo).
2. Espera y recibe las respuestas del servidor.
3. Por lo general, puede conectarse a varios servidores a la vez.
4. Normalmente interactúa directamente con los usuarios finales mediante una interfaz gráfica de usuario.

5. Al contratar un servicio de redes, se debe tener en cuenta la velocidad de conexión que le otorga al cliente y el tipo de cable que utiliza.

2.8.2. Características de Servidor

1. Al iniciarse esperan a que lleguen las solicitudes de los clientes, desempeñan entonces un papel pasivo en la comunicación (dispositivo esclavo).
2. Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente.
3. Por lo general, aceptan conexiones desde un gran número de clientes (en ciertos casos el número máximo de peticiones puede estar limitado).
4. No es frecuente que interactúen directamente con los usuarios finales.

2.8.3. Código Abierto (Open Source)

Código abierto es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones éticas y morales las cuales destacan en el llamado software libre.

Su uso nació por primera vez en 1998 de la mano de algunos usuarios de la comunidad del software libre, tratando de usarlo como reemplazo al ambiguo nombre original en inglés del software libre (free software).

El término para algunos no resultó apropiado como reemplazo para el ya tradicional free software, pues eliminaba la idea de libertad, confundida usualmente con la simple gratuidad.

El término código abierto continúa siendo ambivalente, puesto que se usa en la actualidad por parte de programadores que no ofrecen software libre pero en cambio, sí ofrecen el código fuente de los programas para su revisión o modificación previamente autorizada por parte de sus pares académicos.

Sin embargo, hay que diferenciar los programas de código abierto, que dan a los usuarios la libertad de mejorarlos, de los programas que simplemente tienen el código fuente disponible, previa restricciones sobre su uso o modificación.

En la actualidad el código abierto se utiliza para definir un movimiento nuevo de software (la Iniciativa Open Source), diferente al movimiento del software libre, incompatible con este último desde el punto de vista filosófico, y completamente equivalente desde el punto de vista práctico, de hecho, ambos movimientos trabajan juntos en el desarrollo práctico de proyectos.

2.8.4. Apache

Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa.

La historia de Apache se remonta a febrero de 1995, donde empieza el proyecto del grupo Apache, el cual está basado en el servidor Apache http de la aplicación original de NCSA.

El desarrollo de esta aplicación original se estancó por algún tiempo tras la marcha de Rob McCool por lo que varios web máster siguieron creando sus parches para sus servidores web hasta que

se contactaron vía email para seguir en conjunto el mantenimiento del servidor web, fue ahí cuando formaron el grupo Apache.

Fueron Brian Behlendorf y Cliff Skolnick quienes a través de una lista de correo coordinaron el trabajo y lograron establecer un espacio compartido de libre acceso para los desarrolladores.

Fue así como fue creciendo el grupo Apache, hasta lo que es hoy. Aquella primera versión y sus sucesivas evoluciones y mejoras alcanzaron una gran implantación como software de servidor inicialmente solo para sistemas operativos UNIX y fruto de esa evolución es la versión para Windows.

Apache es una muestra, al igual que el sistema operativo Linux (un Unix desarrollado inicialmente para PC), de que el trabajo voluntario y cooperativo dentro de Internet es capaz de producir aplicaciones de calidad profesional difíciles de igualar.

2.8.5. PHP

PHP es el acrónimo de Hipertext Preprocesor; es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación, creado originalmente en 1994 por Rasmus Lerdorf, pero como PHP está desarrollado en política de código abierto, a lo largo de su historia ha tenido muchas contribuciones de otros desarrolladores [6].

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente.

El cliente solamente recibe una página con el código HTML resultante de la ejecución de la PHP.

PHP se escribe dentro del código HTML, lo que lo hace realmente fácil de utilizar, al igual que ocurre con el popular ASP de Microsoft, pero con algunas ventajas como su gratuidad, independencia de plataforma, rapidez y seguridad.

Este lenguaje de programación está preparado para realizar muchos tipos de aplicaciones web gracias a la extensa librería de funciones con la que está dotado. La librería de funciones cubre desde cálculos matemáticos complejos hasta tratamiento de conexiones de red, por poner dos ejemplos.

Capacidades de PHP:

Compatibilidad con las bases de datos más comunes como MySQL, MySQL, Oracle y ODBC, por ejemplo. Incluye funciones para el envío de correo electrónico, upload de archivos, crear dinámicamente en el servidor imágenes en formato GIF, incluso animadas y una lista interminable de utilidades adicionales.

2.8.6. MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQL AB desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009 desarrolla MySQL como software libre en un esquema de licenciamiento dual [5].

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

SQL (lenguaje de consulta estructurado) fue comercializado por primera vez en 1981 por IBM, el cual fue presentado a ANSI y desde entonces ha sido considerado como un estándar para las bases de datos relacionales. Desde 1986, el estándar SQL ha aparecido en diferentes versiones como por ejemplo: SQL:92, SQL:99, SQL:2003. MySQL es una idea originaria de la empresa open source MySQL AB establecida inicialmente en Suecia en 1995 y cuyos fundadores son David Axmark, Allan Larsson, y Michael "Monty" Widenius.

MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación.

2.8.7. RIA

Las **Rich Internet Applications**, o **RIA** (en español "aplicaciones de Internet enriquecidas"), son aplicaciones web que tienen la mayoría de las características de las aplicaciones de escritorio tradicionales.

Las RIA surgen como una combinación de las ventajas que ofrecen las aplicaciones web y las aplicaciones tradicionales. Normalmente en las aplicaciones web hay una recarga continua de páginas cada vez que el usuario pulsa sobre un enlace. De esta forma se produce un tráfico muy alto entre el cliente y el servidor, llegando muchas veces a recargar la misma página con un cambio mínimo.

En los entornos RIA, en cambio, no se producen recargas de página, ya que desde el principio se carga toda la aplicación, y sólo se produce comunicación con el servidor cuando se necesitan datos externos como datos de una base de datos o de otros ficheros externos.

Las aplicaciones RIA son un nuevo tipo de aplicaciones con más ventajas que las tradicionales aplicaciones web. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales.

Otra de las desventajas de las tradicionales aplicaciones Web es la poca capacidad multimedia que posee. Para ver un vídeo es necesario usar un programa externo para su reproducción.

Las capacidades multimedia son totales gracias a que estos entornos tienen reproductores internos y no hace falta ningún reproductor del sistema operativo del usuario.

Hay muchas herramientas para la creación de entornos RIA. Entre estas se puede mencionar las plataformas Adobe Flash, Adobe Flex y Adobe AIR de Adobe, AJAX, OpenLaszlo, Silverlight de Microsoft, JavaFX Script de Sun Microsystems.

Arquitectura: Generalmente se tiene una aplicación cliente "stateful" y una capa de servicios separada. Las RIA se apoya más sobre un desarrollo "cliente-servidor" en vez de un desarrollo web tradicional, en donde el estado se mantiene en el servidor en sesiones.

Cliente: Se maneja la interacción entre el usuario y la interfaz de usuario, el usuario invoca comandos, actualiza vistas y carga datos.

Servidor: Aquí se manejan y se procesan todas las peticiones de la aplicación cliente y delega las acciones en el servidor, estas pueden ser, guardar datos en la base de datos, actualizar los archivos del sistema, retornar datos al servidor, o algún tipo de proceso analítico. Determina y le da formato a los datos que son retornados al cliente.

Aplicaciones: Consisten en el aprovechamiento de la experiencia del usuario en herramientas y funciones de escritorio tan naturales como copiar, cortar y pegar, redimensionar columnas y ordenar etc., con el alcance y la flexibilidad de presentación y despliegue que ofrecen las aplicaciones o páginas Web junto con lo mejor de la multimedia (voz, vídeo, etc.).

Se puede decir que las RIA son la nueva generación de las aplicaciones y es una tendencia ya impuesta por empresas como Macromedia, Magic Software, Sun o Microsoft que se encuentran

desarrollando recursos para hacer de este tipo de aplicaciones una realidad.

Entre los beneficios principales de aplicaciones RIA tenemos una mejora importante en la experiencia visual, que hacen del uso de la aplicación algo muy sencillo, ofrece mejoras en la conectividad y despliegue instantáneo de la aplicación, agilizando su acceso, garantizan la desvinculación de la capa de presentación es decir acceso a la aplicación desde cualquier computador en cualquier lugar del mundo.

2.8.7.1. Características Clave

Accesibilidad: AJAX en nativo en los navegadores web y es el único RIA framework que puede ser encontrado por los diferentes motores de búsqueda. Aunque Adobe Flash ha dado grandes pasos en esta dirección.

Comunicaciones Avanzadas: con servidores que soporten nuevas tecnologías se puede mejorar la experiencia del usuario al utilizar protocolos de red optimizados y entradas y salidas asíncronas. Se requiere de una conexión de banda ancha confiable.

Complejidad: soluciones avanzadas puede ser más difíciles de diseñar, desarrollar, implementar y depurar que las aplicaciones web tradicionales.

Consistencia: la interfaz de usuario y las experiencias pueden ser controlada por el sistema operativo, el monitoreo del rendimiento y diagnostico de errores puede ser difícil.

Instalación y Mantenimiento: se requiere de la instalación de un plugin o una máquina virtual o sandbox, que generalmente es más rápida que la instalación de una aplicación tradicional y esta no se puede automatizar. Las actualizaciones son automáticas.

Seguridad: se mejora la seguridad por medio de actualizaciones automáticas y sandbox.

3. CAPITULO: Análisis y Diseño

3.1. Introducción

La Planta Física de la BUAP es un sistema sencillo capaz de almacenar gran infinidad de información de cada uno de los inmuebles de la BUAP.

Este sistema está diseñado para facilitar al usuario la ubicación y reconocimiento de ciertos edificios e incluso salones y áreas deportivas.

3.2. Planteamiento del Problema

Desarrollar un sistema administrativo que muestre cada una de las instalaciones de la BUAP donde se pueda mostrar una imagen de dicho inmueble para así tener una mejor ubicación de cada edificio o salón de la universidad y así poder ofrecer al público en general una forma más sencilla de ubicarse en las instalaciones de la BUAP.

3.3. Objetivo General

Desarrollar un Sistema Administrativo que muestre cada una de las instalaciones de la BUAP considerando el número de edificio y la facultad considerando la búsqueda de salones y edificios.

3.4. Objetivo Específico

Ofrecer al público en general la ubicación exacta de algún edificio en particular.

3.5. Metodología

En el desarrollo de este sistema se usará la metodología cascada modificado.

El sistema no es muy complejo entonces esta metodología es adecuada para el seguimiento de cada fase de desarrollo.

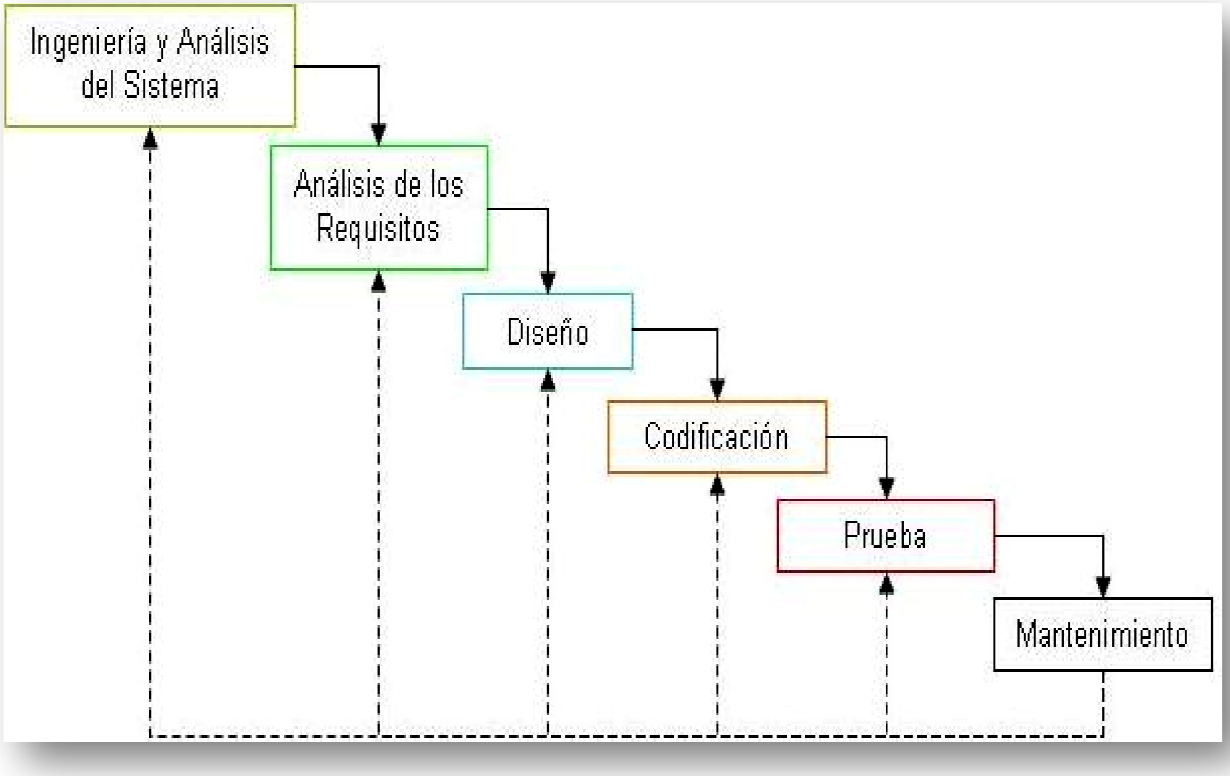

Figura 3.5 Modelo de Cascada Modificado.

3.5.1. Actores

Administrador: Es aquel usuario registrado encargado de consultar, modificar, eliminar y de crear las categorías (permisos).

Secretario Administrativo: Es aquel usuario registrado encargado de consultar, modificar, eliminar y agregar imágenes de cada instalación ingresada así como información adicional (dirección, número de edificio, tel., etc.).

Público: Es aquel usuario que podrá consultar el inmueble de acuerdo a sus necesidades y así tener una ubicación exacta para lo que le convenga.

3.5.2. Diagrama de Casos de Uso

El modelo de casos de uso describe la funcionalidad propuesta del nuevo sistema.

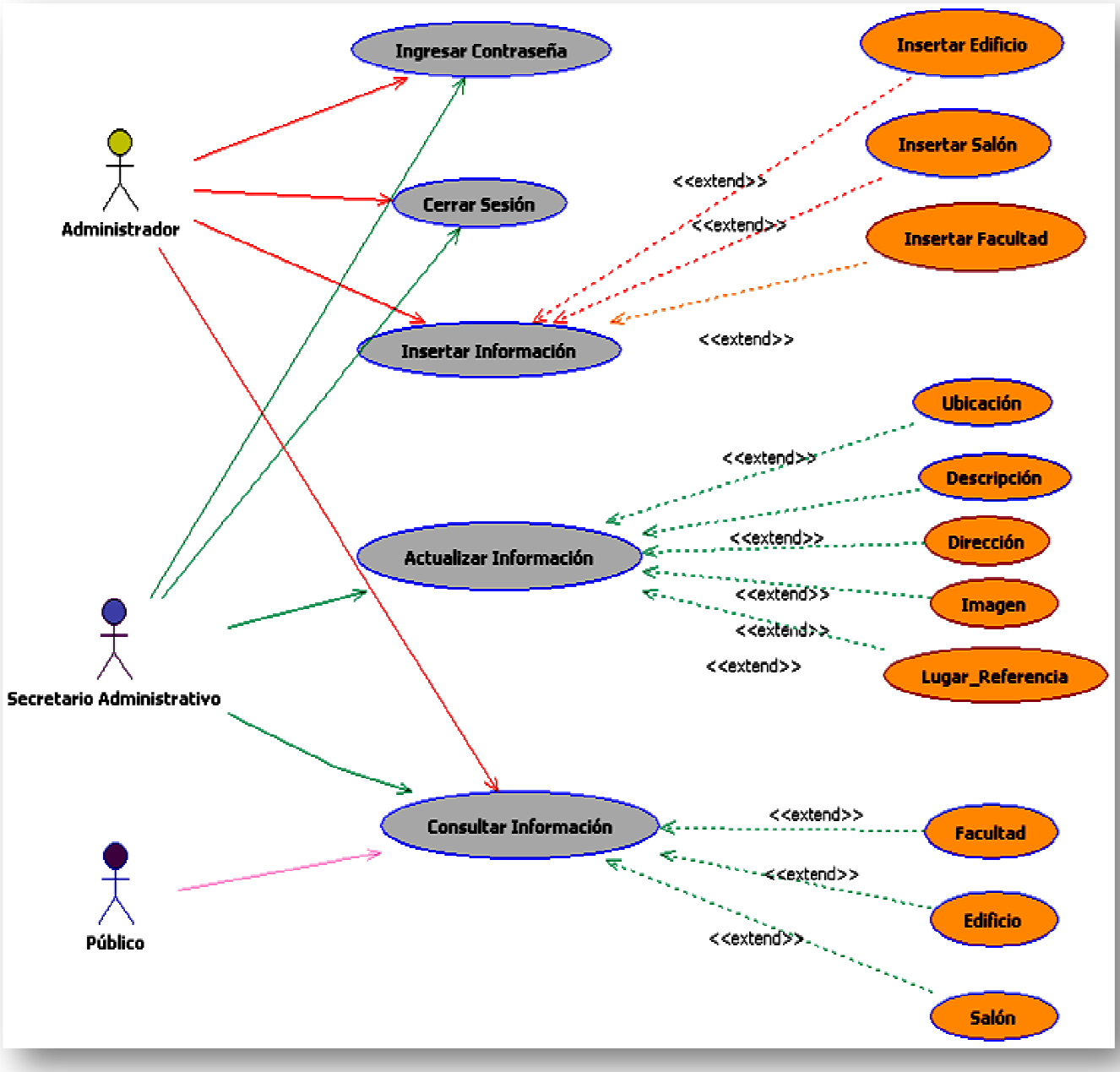

Figura 3.5.2 Diagrama General de Casos de Uso.

3.5.3. Diagramas de Clase

Un diagrama de Clases representa las clases que serán utilizadas dentro del sistema.

Figura 3.5.3 Diagrama de Clases.

3.5.4. Diagramas de Secuencia

Un diagrama de secuencia muestra que objetos se comunican con otros objetos y que mensajes disparan esas comunicaciones.

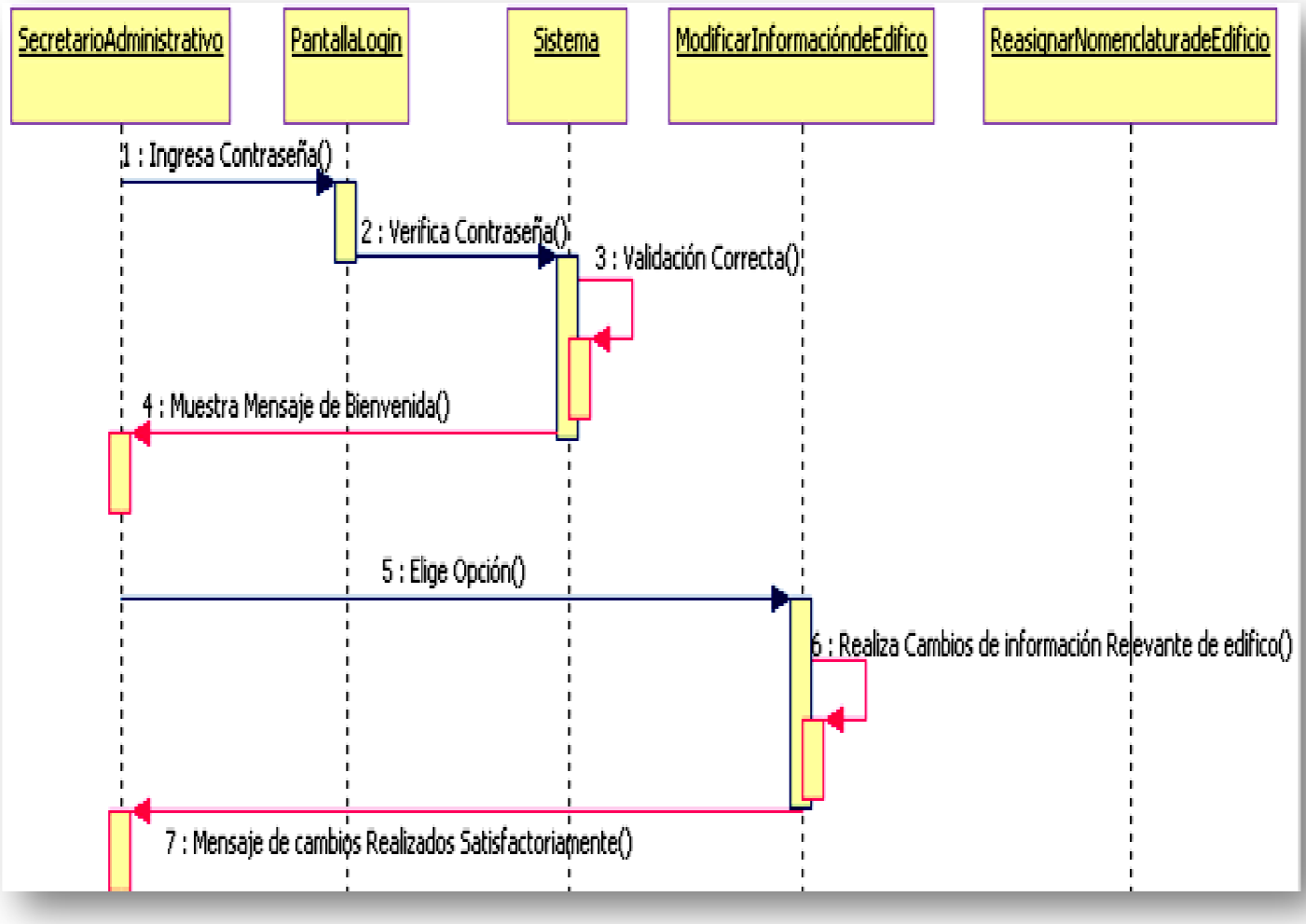

Figura 3.5.4 Diagrama de Secuencia para Secretario Administrativo.

Diagramas de Secuencia

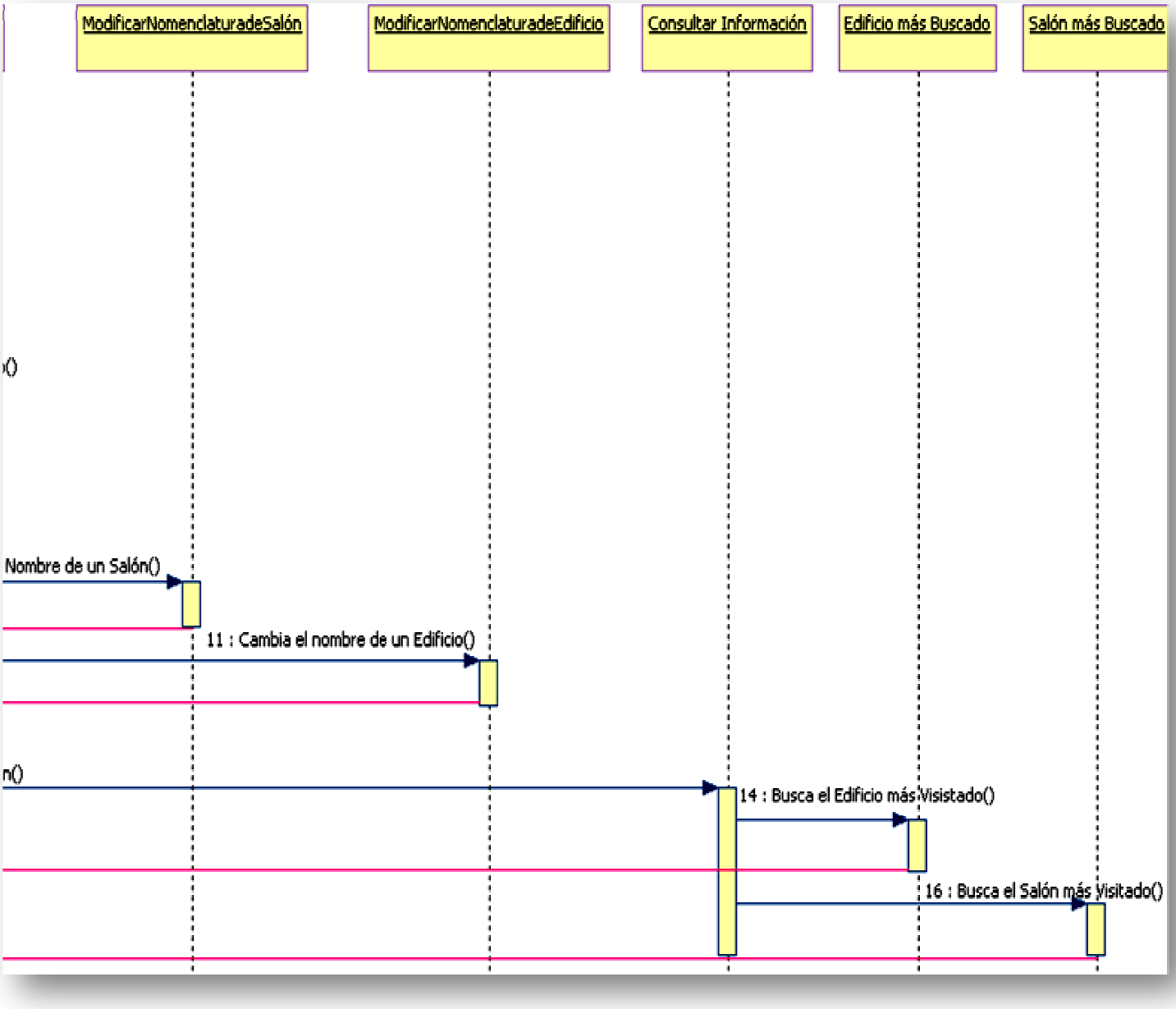

Figura 3.5.4 a) Continuación de Diagrama de Secuencia para Secretario Administrativo.

Diagramas de Secuencia

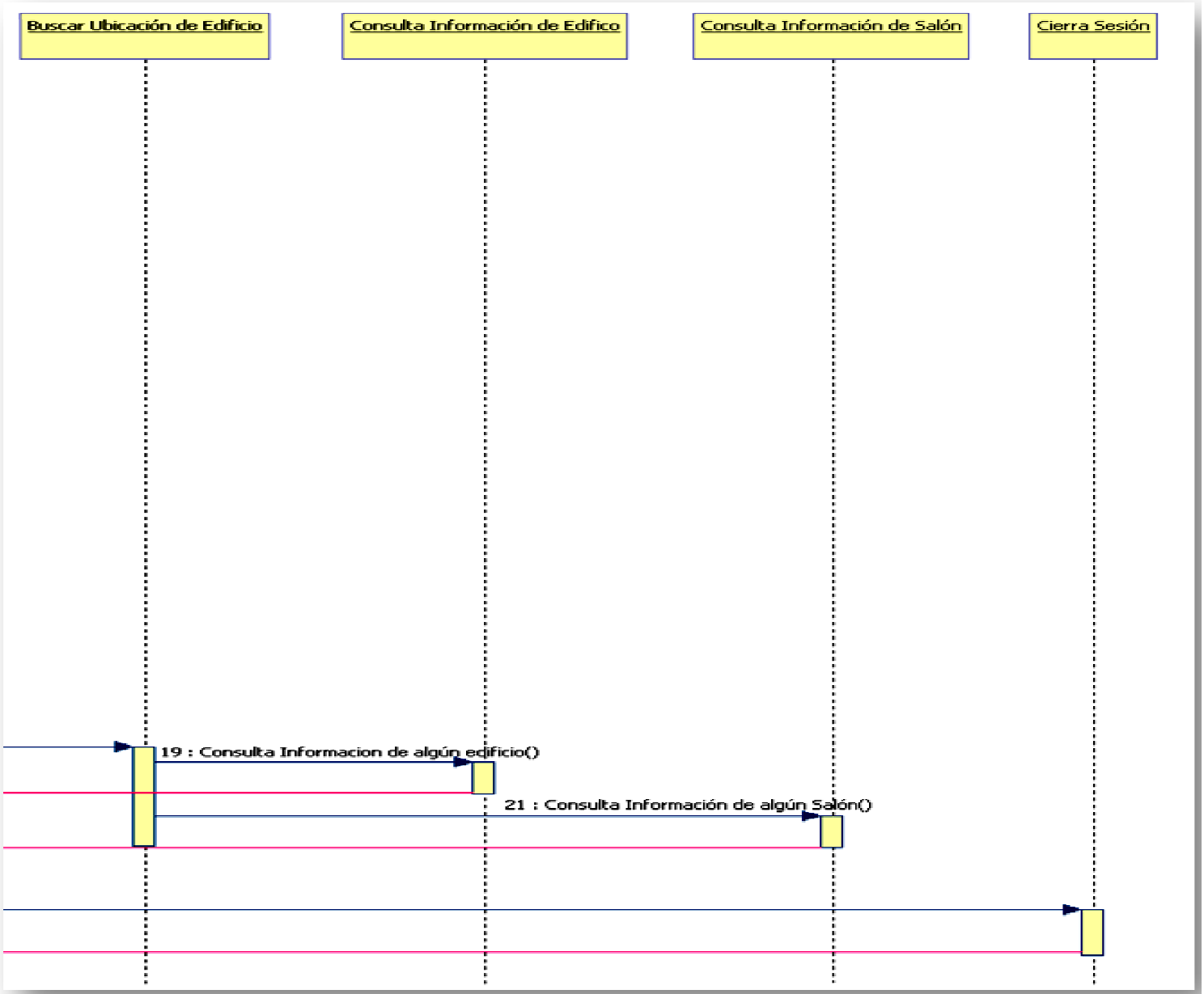

Figura 3.5.4 b) Continuación de Diagrama de Secuencia para Secretario Administrativo.

Diagramas de Secuencia

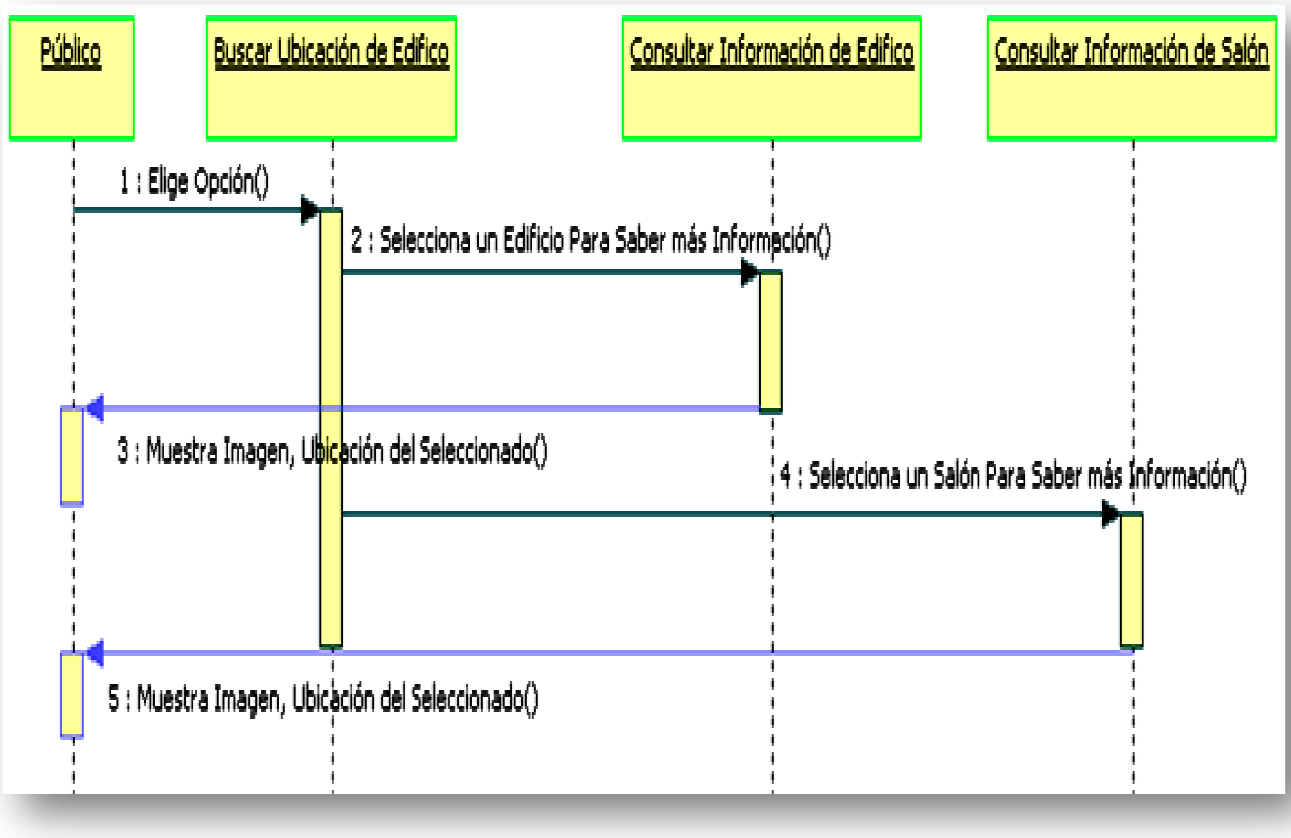

Figura 3.5.4 c) Continuación de Diagrama de Secuencia para Público en General.

Diagramas de Secuencia

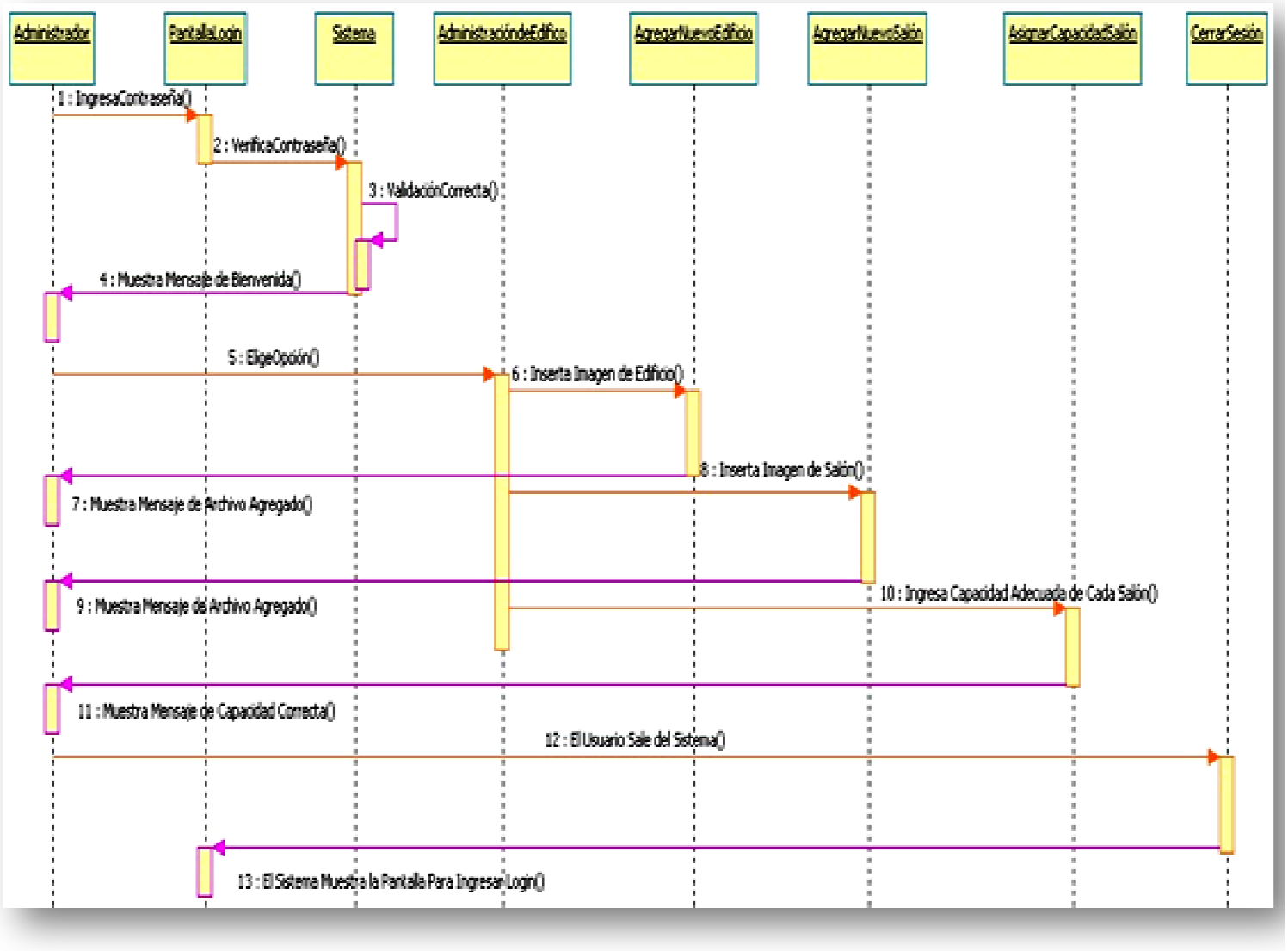

Figura 3.5.4 d) Continuación de Diagrama de Secuencia para Administrador.

3.5.5. Modelo de Análisis

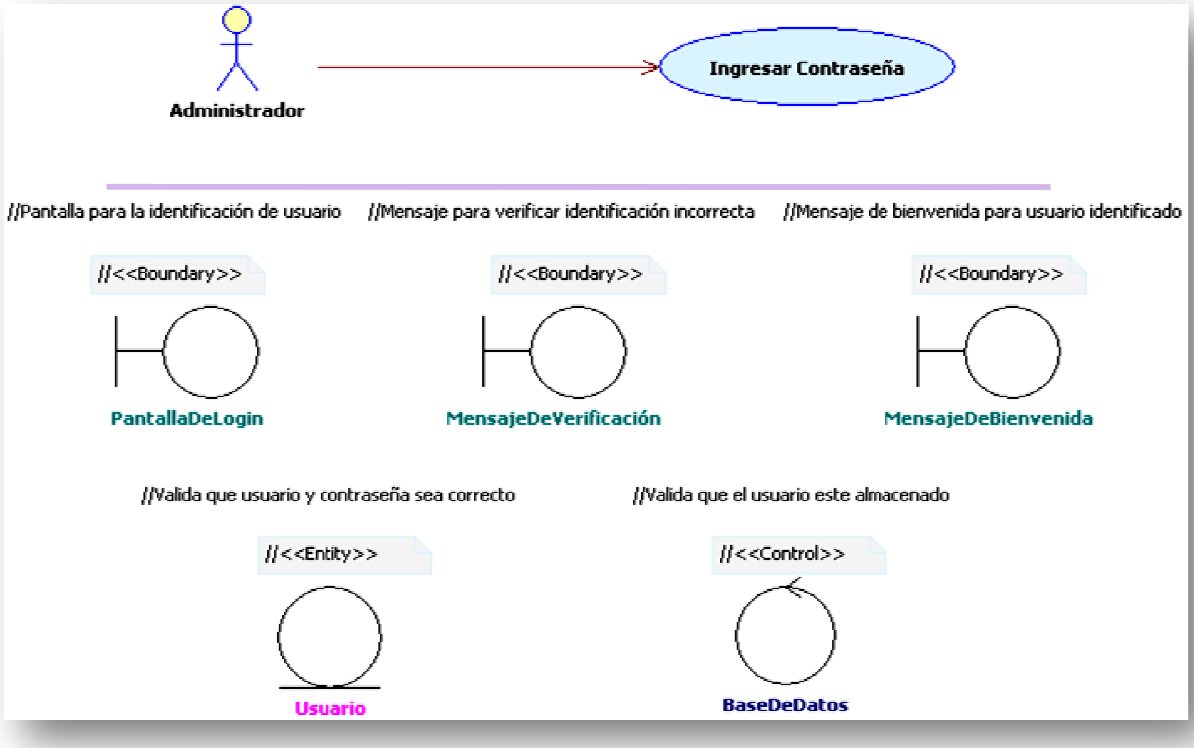

Figura 3.5.5 Diagrama de Clases de Análisis para Caso de Uso "Ingresar Contraseña".

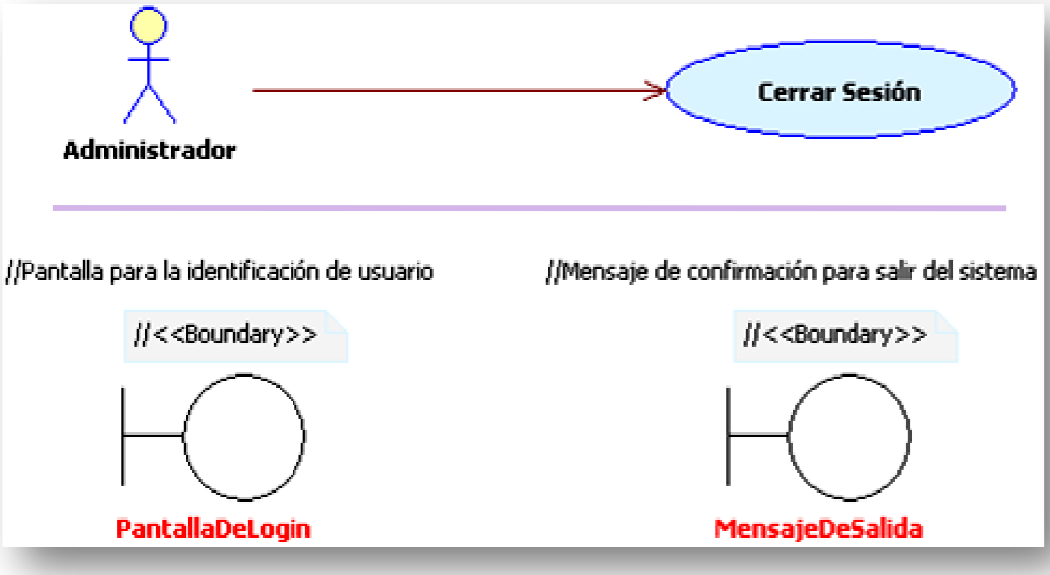

Figura 3.5.5 a) Diagrama de Clases de Análisis para Caso de Uso "Cerrar Sesión".

3.5.6. Modelo Conceptual

1. Diccionario del Modelo

1.1. Nombre: Usuario

1.1.1. **Definición de Trabajo.-** Persona que puede tener una contraseña y usuario para poder ingresar al sistema y así agregar información como: lugar de referencia, información relevante del edifico o salón y también agregar nuevos edificios y salones de la infraestructura de la BUAP.

1.2. Nombre: Facultad

1.2.1. **Definición de Trabajo.-** Se agregará información como: nombre, imagen, lugar ubicación y referencia.

1.3. Nombre: Área

1.3.1. **Definición de Trabajo.-** Almacenará información de áreas donde se ubican instalaciones de la BUAP.

1.4. Nombre: Edificio

1.4.1. **Definición de Trabajo.-** Se agregará información de ubicación, lugar de referencia y como llegar.

1.5. Nombre: Espacio

1.5.1. **Definición de Trabajo.-** Se compone de áreas verdes (laguna, parque botánico), complejo deportivo (canchas, albercas, estadio, polideportivo) que existen en la BUAP.

1.6. Nombre: Consulta

1.6.1. **Definición de Trabajo.-** Se compone de laboratorios, cubículos, baños, dirección, auditorio, biblioteca, y cafetería todos estos ubicados en un edificio de la BUAP.

3.5.7. Especificación de Casos de Uso

INGRESAR CONTRASEÑA (CU01).

1. Iniciar Sesión

1.1. Descripción Breve

1.1.1. Este caso de uso el usuario ingresa su contraseña para tener acceso al sistema y realizar actividades de acuerdo a su necesidad del usuario.

2. Flujo de Eventos

2.1. Pre-Condiciones

2.2. Flujo Básico

2.2.1. El sistema muestra las opciones de "Inicio", "C.U", "Centro", "Regionales", "Salud", "Foráneas", "Consultas" y "Salir".

2.2.2. El sistema muestra en pantalla un apartado para la identificación de usuario **(A-1)**.

2.2.3. El usuario captura los datos de: usuario y contraseña **(E-1)**.

2.2.4. El sistema verifica que el usuario y contraseña estén escritos correctamente.

2.2.5. El usuario presiona el botón "Aceptar" y entra al sistema con sus correspondientes opciones.

2.2.6. Termina el caso de uso.

2.3. Flujos Alternos

2.3.1. **A-1 El usuario selecciona salir.**

2.3.1.1. El sistema regresa a la página anterior.

2.4. Flujos de Excepción

2.4.1. **E-1 Error en la captura de datos.**

2.4.1.1. El sistema despliega el mensaje "información incorrecta: Verifique sus datos" y al paso 2.2.3 del flujo básico.

CERRAR SESIÓN (CU02).

1. Cerrar Sesión.

1.1. Descripción Breve

1.1.1. Este caso de uso le permite al usuario salir del sistema en el momento deseado.

2. Flujo de Eventos

2.1. Pre-Condiciones

2.1.1. El usuario debió haber completado el caso de uso iniciar sesión.

2.2. Flujo Básico

2.2.1. El sistema presenta en pantalla una opción de "Salir" para poder salir del sistema.

2.2.2. El usuario elige la opción "cerrar sesión".

2.2.3. El sistema muestra una pantalla donde le pregunta al usuario si "desea salir del sistema" **(A-1)**.

2.2.4. El usuario presiona el botón "aceptar".

2.2.5. Termina el caso de uso.

2.3. Flujos Alternos

2.3.1. **A-1 El usuario selecciona no salir del sistema.**

2.3.1.1. El sistema regresa a la página anterior.

INSERTAR INFORMACIÓN (CU03).

1. Ingresar Información.

1.1. Descripción Breve

1.1.1. Este caso de uso le permite al usuario realizar diferentes actividades como: "Insertar Edificio", "Insertar Salón" e "Insertar Facultad".

2. Flujo de Eventos

2.1. Pre-Condiciones

2.1.1. El usuario debe haber completado el caso de uso "Ingresar Contraseña" como Administrador.

2.2. Flujo Básico

2.2.1. El sistema muestra en pantalla las opciones de "Inicio", "C.U", "Centro", "Regionales", "Salud", "Foráneas", "Consultas" y "Salir".

2.2.2. El usuario selecciona una opción donde desee ingresar información y despliega en pantalla un menú en el cual le permite al usuario "Insertar", "Consultar" y "Actualizar".

2.2.3. El usuario elige la opción "Insertar"

2.2.4. El sistema muestra en pantalla 3 botones para elegir alguna opción de acuerdo a la necesidad del usuario: "Facultad", "Edificio", "Salón".

2.2.5. El usuario elige la opción de "Facultad".

2.2.6. El sistema muestra en pantalla una serie de campos donde el usuario ingresa información de la Facultad a insertar. **(A-1)**.

2.2.7. El usuario presiona el botón "Agregar".

2.2.8. El sistema muestra en pantalla un mensaje de: "La Facultad fue agregada satisfactoriamente".

2.2.9. El sistema guarda los cambios, regresa a Flujo básico 2.2.4.

2.2.10. Termina el caso de uso.

2.2.11. El usuario elige la opción de "Edificio".

2.2.12. El sistema muestra en pantalla una serie de campos donde el usuario ingresa datos para el nuevo Edificio a insertar. **(A-1)**.

2.2.13. El usuario presiona el botón "Agregar".

2.2.14. El sistema muestra en pantalla un mensaje de: "El Edificio fue agregado satisfactoriamente".

2.2.15. El sistema guarda los cambios, regresa a Flujo básico 2.2.4.
2.2.16. Termina el caso de uso.
2.2.17. El usuario elige la opción de "Salón".
2.2.18. El sistema muestra en pantalla una serie de campos donde el usuario ingresa información del Salón a insertar. **(A-1)**.
2.2.19. El usuario presiona el botón "Agregar".
2.2.20. El sistema muestra en pantalla un mensaje de: "El Salón se agrego satisfactoriamente".
2.2.21. El sistema guarda los cambios, regresa a Flujo básico 2.2.4.
2.2.22. Termina el caso de uso.

ACTUALIZAR INFORMACIÓN (CU04).

1. Actualizar Información.

1.1. Descripción Breve

1.1.1. Este caso de uso le permite al usuario modificar información detallada de un edificio o salón, esta información puede ser: "Dirección, Nombre, Lugar de Referencia, Capacidad, Nomenclatura e Imagen".

2. Flujo de Eventos

2.1. Pre-Condiciones

2.1.1. El usuario debió haber completado el caso de uso iniciar sesión.

2.2. Flujo Básico

2.2.1. El usuario selecciona la opción de "Actualizar".
2.2.2. El sistema muestra en pantalla 3 botones para elegir alguna opción de acuerdo a la necesidad del usuario: "Facultad", "Edificio", "Salón".
2.2.3. El usuario selecciona la opción "Facultad".
2.2.4. El sistema muestra en pantalla una lista desplegable de los edificios que tiene la facultad seleccionada.
2.2.5. El usuario elije el nombre de una facultad para su modificación.
2.2.6. El sistema muestra en pantalla datos de la facultad seleccionada como: nomenclatura, ubicación, dirección, lugar de referencia e imagen para que el usuario modifique información.
2.2.7. El usuario selecciona el botón "Examinar" para modificar la imagen.
2.2.8. El sistema muestra en pantalla un recuadro para seleccionar la imagen deseada.
2.2.9. El usuario elije la imagen.
2.2.10. El usuario da click en el botón "Actualizar" (A-1).
2.2.11. El sistema muestra en pantalla un mensaje de: "La actualización se ha realizado satisfactoriamente".
2.2.12. El sistema guarda los cambios, regresa a Flujo básico 2.2.4.
2.2.13. Termina el caso de uso.
2.2.14. El usuario selecciona la opción "Edificio".
2.2.15. El sistema muestra en pantalla una lista desplegable
2.2.16. para que el usuario elija una facultad para poder modificar un edificio.
2.2.17. El sistema muestra en pantalla una lista desplegable de los edificios que tiene la facultad seleccionada.
2.2.18. El usuario elije el nombre de una facultad para su modificación.

2.2.19. El sistema muestra en pantalla datos de la facultad seleccionada como: nomenclatura, ubicación, dirección, lugar de referencia e imagen para que el usuario modifique información.
2.2.20. El usuario selecciona el botón "Examinar" para modificar la imagen.
2.2.21. El sistema muestra en pantalla un recuadro para seleccionar la imagen deseada.
2.2.22. El usuario elije la imagen.
2.2.23. El usuario da click en el botón "Actualizar" **(A-1)**.
2.2.24. El sistema muestra en pantalla un mensaje de: "La actualización se ha realizado satisfactoriamente".
2.2.25. El sistema guarda los cambios, regresa a Flujo básico 2.2.15.
2.2.26. Termina el caso de uso.
2.2.27. El usuario selecciona la opción "Salón".
2.2.28. El sistema muestra en pantalla una lista desplegable llamado Facultad.
2.2.29. El usuario da click en la lista la Facultad a la cual pertenece el salón a actualizar.
2.2.30. El sistema activa otra lista desplegable de los edificios que tiene la facultad seleccionada.
2.2.31. El usuario elije y da click sobre el edificio al cual pertenece el salón que desea actualizar.
2.2.32. El sistema activa la última lista desplegable donde muestra los salones que están relacionados con el edificio seleccionado.
2.2.33. El usuario da click sobre el salón que desea actualizar.
2.2.34. El sistema muestra en pantalla datos del salón seleccionado como: nombre, capacidad e imagen para que el usuario modifique información.
2.2.35. Si el usuario modifica la imagen del salón, da click en el botón "Seleccionar".
2.2.36. El sistema muestra en pantalla un recuadro para seleccionar la imagen deseada.
2.2.37. El usuario elije la imagen **E-1**.
2.2.38. El usuario da click en el botón "Actualizar" **(A-1)**.
2.2.39. El sistema muestra en pantalla un mensaje de: "La actualización se ha realizado satisfactoriamente".
2.2.40. El sistema guarda los cambios, regresa a Flujo básico 2.2.27.
2.2.41. El usuario elige la opción de "cerrar sesión".
2.2.42. El sistema regresa a la página principal.
2.2.43. Termina el caso de uso.

2.3. Flujos Alternos

2.3.1. **A-1. El usuario selecciona salir.**

2.3.1.1. El sistema regresa a la página anterior sin efectuar algún cambio.

2.4. Flujos de Excepción

2.4.1. **E-1. Error**

CONSULTAR INFORMACIÓN (CU05).

1. Consultar Información.

1.1. Descripción Breve

1.1.1. Este caso de uso le permite al usuario consultar información como: ubicación, lugar de referencia, imagen, dirección nomenclatura de algún salón, edificio, o Facultad de todos los inmuebles de la BUAP.

2. Flujo de Eventos

2.1. Pre-Condiciones

2.1.1. El usuario debe entrar al sistema.

2.2. Flujo Básico

2.2.1. El sistema muestra las opciones de "Inicio", "C.U", "Centro", "Regionales", "Salud", "Foráneas", "Consultas" y "Salir".

2.2.2. El usuario elige la opción de "consultas" **(A-1)**.

2.2.3. El sistema despliega un sub-menú para "consultar edificio más buscado" y "consultar salón más buscado".

2.2.4. El usuario elige la opción de "consultar edificio más buscado" **(E-1)**.

2.2.5. El sistema muestra en pantalla una lista con los nombres de los edificios más visitados por los usuarios.

2.2.6. El usuario elige un nombre de la lista.

2.2.7. El sistema muestra en pantalla una imagen del edificio y muestra un poco de información acerca de este mismo.

2.2.8. Termina el caso de uso.

2.2.9. El usuario elige la opción de "consultar salón más buscado".

2.2.10. El sistema muestra en pantalla una lista con los nombres de los salones más visitados por los usuarios.

2.2.11. El usuario elige un nombre de la lista.

2.2.12. El sistema muestra en pantalla una imagen del salón y muestra un poco de información acerca de este mismo.

2.2.13. El sistema muestra en pantalla datos de la facultad o salón elegido y muestra información referente a lo seleccionado.

2.2.14. El usuario selecciona "Consultas" (regresa al paso 2.2.2).

2.2.15. Termina el caso de uso.

2.3. Flujos Alternos

2.3.1. **A-1 El usuario no selecciona nada.**

2.3.1.1. El sistema no despliega información en pantalla.

2.4. Flujos de Excepción

2.4.1. **E-1 Error en la selección de edificio o salón.**

2.4.1.1. El usuario cierra la ventana y vuelve a elegir un nuevo edificio o salón.

3.5.8. Escenarios

Un escenario es una secuencia específica de acciones que ilustran comportamiento, un escenario es una instancia de un caso de uso.

"INSERTAR INFORMACIÓN"

La Administradora del sistema María Pérez Gómez ya completo el caso de uso "Ingresar Contraseña" y elije la opción "C.U", se despliega varias opciones como "Actualizar, Insertar y Consultar".

María elije la opción "Insertar" del Sub-menú de C.U, la cual le muestra en pantalla 3 botones: "FACULTAD", "EDIFICIO" y "SALÓN.

María elige la opción de "FACULTAD", esto le permite ingresar al sistema datos generales de una nueva Facultad que no existía.

María ingresa datos en los campos que se le solicitan.

El sistema verifica que el formato de la imagen sea el correcto y se almacena en la base de datos.

El sistema le muestra una pantalla un mensaje donde se le informa que los datos están correctos.

María termina el caso de uso.

Figura 3.5.8 Ejemplo de Escenario para caso de uso Insertar Información.

Escenarios

"INSERTAR INFORMACIÓN"

María elige la opción de "EDIFICIO", esto le permite ingresar al sistema datos generales de un nuevo Edifico perteneciente a una Facultad.

María ingresa datos en los campos que se le solicitan.

El sistema verifica que el formato de la imagen sea el correcto y se almacena en la base de datos.

El sistema le muestra una pantalla un mensaje donde se le informa que los datos están correctos.

María termina el caso de uso.

María elige la opción de "SALÓN" ", esto le permite ingresar al sistema datos generales de un nuevo Salón perteneciente a un Edificio o Facultad.

María ingresa datos en los campos que se le solicitan.

El sistema verifica que el formato de la imagen sea el correcto y se almacena en la base de datos.

El sistema le muestra una pantalla un mensaje donde se le informa que los datos están correctos.

María termina el caso de uso.

Figura 3.5.8 a) Continuación de Ejemplo Escenario para caso de uso Insertar Información.

Escenarios

"ACTUALIZAR INFORMACIÓN"

El Secretario Administrativo Pedro López García ya completo el caso de uso "Ingresar Contraseña" y escoge la opción "C.U., se despliega varias opciones como "Actualizar, Insertar y Consultar".

Pedro elige la opción "Actualizar" la cual muestra en pantalla 3 botones: "FACULTAD", "EDIFICIO" y "SALÓN.

Pedro elige la opción de "FACULTAD", el sistema muestra una lista desplegable de todas las Facultades existentes.

Pedro selecciona una Facultad, enseguida el sistema le muestra en pantalla datos de la Facultad seleccionada.

Pedro modifica datos en los campos que necesite y guarda los cambios.

El sistema le muestra una pantalla un mensaje donde se le informa que los datos están correctos.

Pedro termina el caso de uso.

Pedro elige la opción de "EDIFICIO", el sistema muestra una lista desplegable de todas las Facultades existentes.

Pedro selecciona una Facultad, enseguida el sistema muestra una lista desplegable donde le muestra todos los edificios pertenecientes a la facultad seleccionada.

Figura 3.5.8 b) Ejemplo de Escenario para caso de uso Actualizar Información.

Escenarios

"ACTUALIZAR INFORMACIÓN"

Pedro selecciona el edifico a modificar y el sistema muestra en pantalla datos del edificio seleccionado.

Pedro modifica datos en los campos que necesite y guarda los cambios.

El sistema le muestra una pantalla un mensaje donde se le informa que los datos están correctos.

Pedro termina el caso de uso.

Pedro elige la opción de "SALON", el sistema muestra una lista desplegable de todas las Facultades existentes.

Pedro selecciona una Facultad, enseguida el sistema muestra una lista desplegable donde le muestra todos los edificios pertenecientes a la facultad seleccionada.

Pedro selecciona el edifico, enseguida el sistema muestra en pantalla todos los salones pertenecientes a dicho edificio.

Pedro elije el salón y modifica datos en los campos que necesite y guarda los cambios.

El sistema le muestra una pantalla un mensaje donde se le informa que los datos están correctos.

Pedro termina el caso de uso.

Figura 3.5.8 b) Continuación de Ejemplo de Escenario para caso de uso Actualizar Información.

Escenarios

"BUSCAR INFORMACIÓN"

Sol Torres Mendieta ingresa en su navegador la dirección de la página del Sistema Multimedia de la Planta Física de la BUAP.

El sistema le muestra en pantalla varias opciones con los nombres de las áreas de los campus de la universidad como son: "C.U.". "SALUD", "CENTRO", "PREPARATORIAS", "REGIONALES" y "CONSULTAS".

Sol elige la opción de "SALUD" porque desea saber la ubicación y conocer la fachada de la facultad de estomatología.

El sistema le muestra un Sub-menú y Sol da click en la opción "Consultar".

El sistema le muestra en pantalla una serie de listas desplegables, Sol da click en la facultad de estomatología, el sistema muestra en pantalla una imagen de la facultad e incluso le muestra la ubicación, el cómo llegar y algún lugar de referencia.

Sol escribe esta información en un papel.

Figura 3.5.8 c) Ejemplo de Escenario para caso de uso Buscar Información.

4. CAPITULO: Diseño de la Base de Datos

4.1. Introducción

El diagrama Entidad-Relación facilita el diseño de la Base de Datos de un sistema desde el más simple hasta el más complejo.

Por ello la importancia de hacer un diagrama Entidad-Relación.

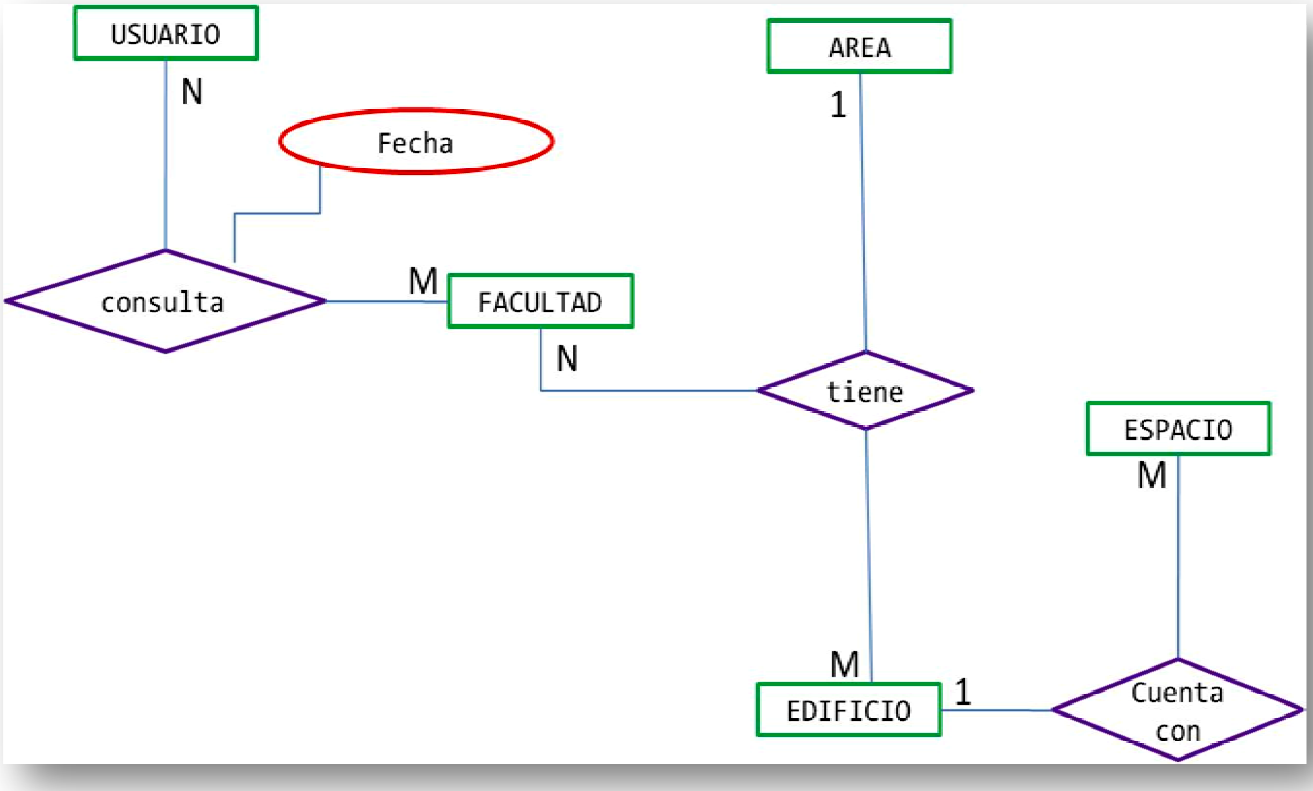

Figura 4.1 Diagrama Entidad-Relación

4.2. Atributos

Una vez identificadas las entidades se obtienen los atributos para con ello obtener las llaves primarias, foráneas para con ello poder armar la base de datos.

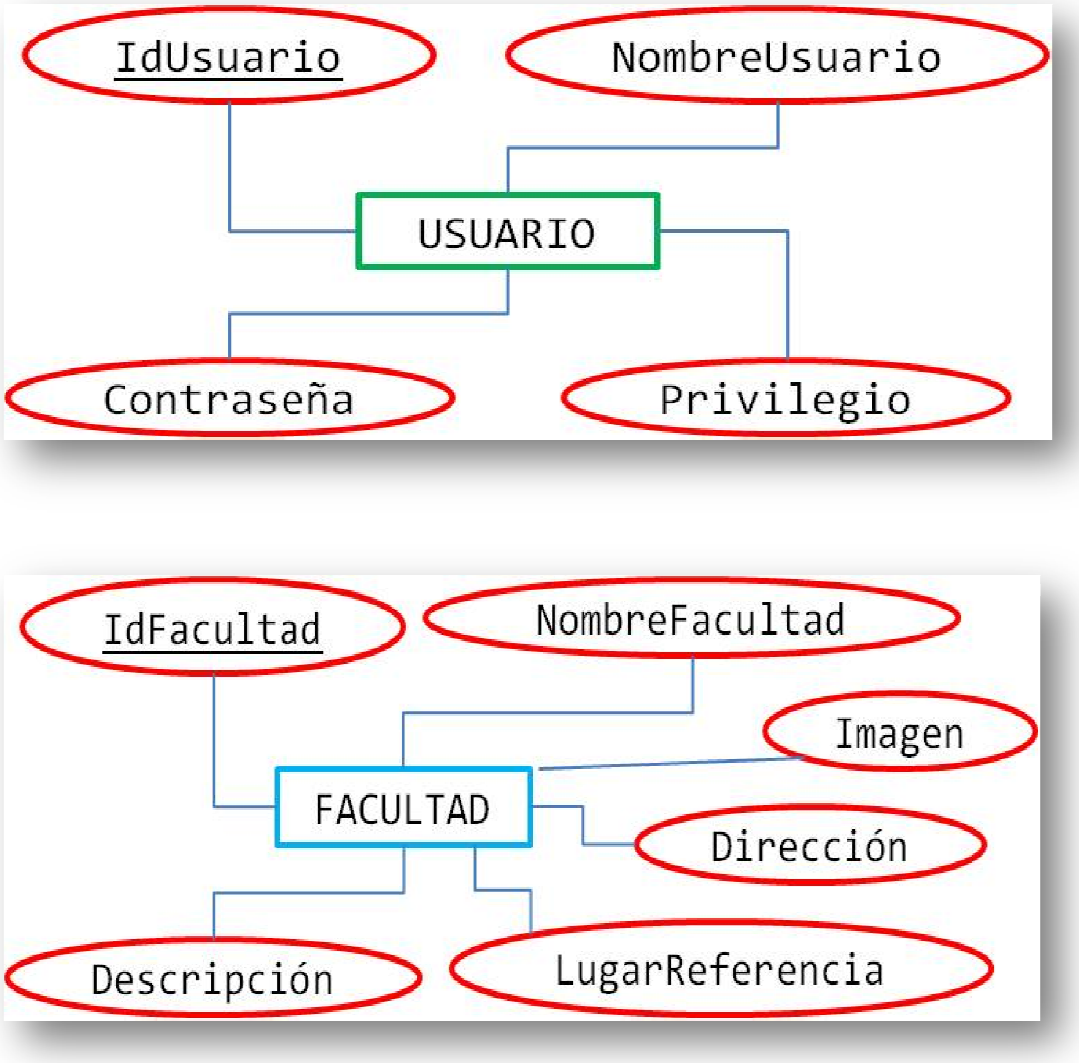

Figura 4.2 Atributos Identificados

4.3. Llaves Primarias

Una vez identificadas las entidades se obtienen los atributos para con ello obtener las llaves primarias, foráneas para con ello poder armar la base de datos.

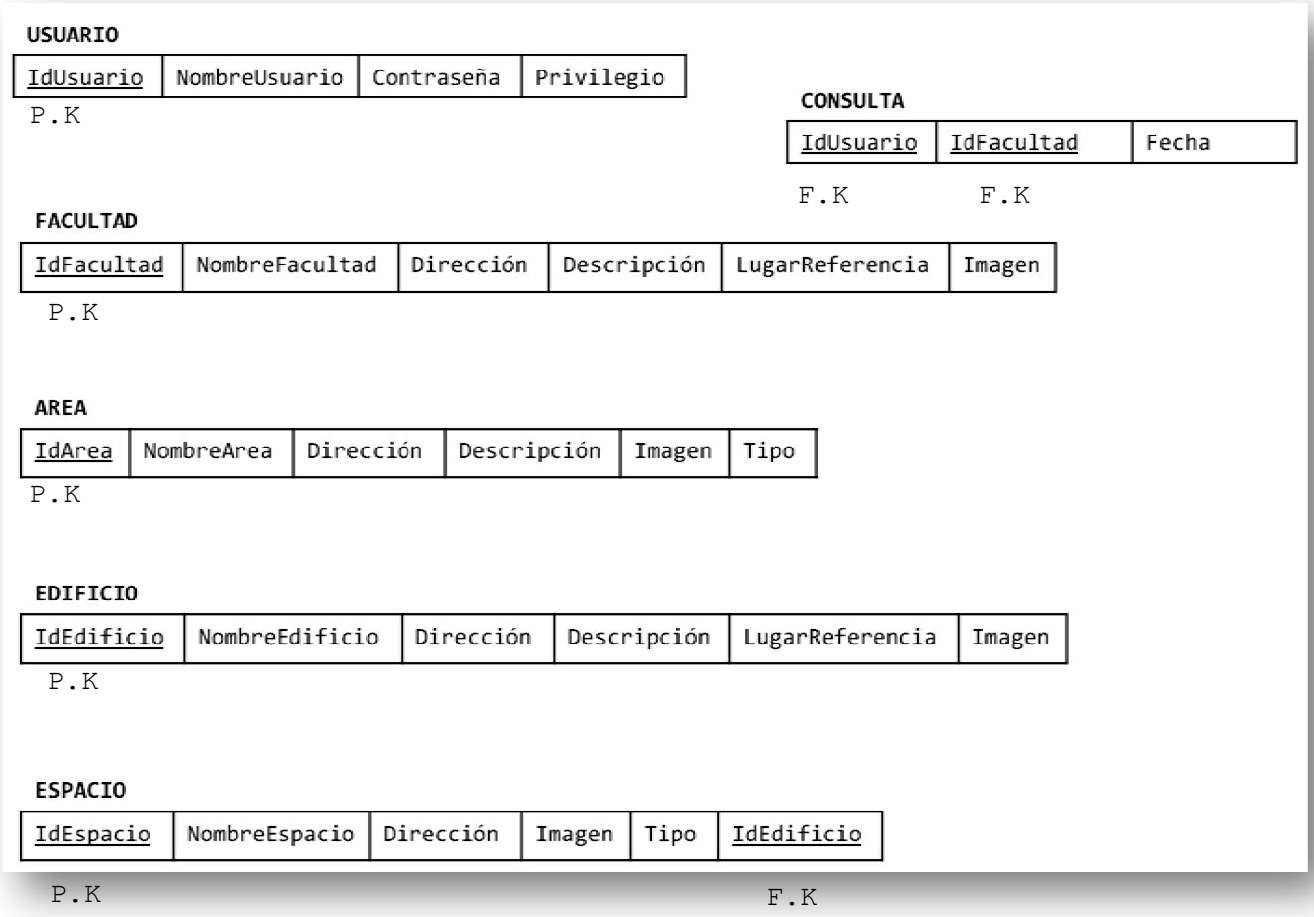

Figura 4.3 Identificando Llaves Primarias y Foráneas

4.4. Tablas

Ejemplo de algunas tablas con campos que conforman la Base de Datos del Sistema

edificio

Comentarios de la tabla: InnoDB free: 3072 kB

Campo	Tipo	Nulo
id_edificio	int(11)	No
nombre_edificio	varchar(120)	No
direccion	varchar(120)	No
descripcion	varchar(120)	No
lugar_referencia	varchar(120)	No
imagen	varchar(120)	No

area

Comentarios de la tabla: InnoDB free: 3072 kB

Campo	Tipo	Nulo
id_area	int(11)	No
nombre_area	varchar(120)	No
direccion	varchar(120)	No
descripcion	varchar(120)	No
tipo	int(2)	No
imagen	varchar(120)	No

usuario

Comentarios de la tabla: InnoDB free: 3072 kB

Campo	Tipo	Nulo
id_usuario	int(11)	No
nombre_usuario	varchar(50)	No
contrasena	varchar(50)	No
privilegio	int(2)	No

Figura 4.4 Algunas tablas que conforman la base de datos con sus atributos.

5. CAPITULO: Implementación y Pruebas del Sistema

5.1. Implementación

En esta etapa se muestran los resultados obtenidos después de analizar el problema, plantear soluciones y plasmar en lenguaje PHP el diseño de la Planta Física de la BUAP.

5.1.1. Software para el Desarrollo del Sistema

1. Php y MySQL para administración de la Base de Datos.
2. Flash 8 para diseño gráfico del sistema.
3. WampServer para pruebas del sistema.
4. Dreamweaver cs4 para diseño y programación del sistema.
5. XAJAX para programación del sistema.

5.2. Implementación de la Interfaz

La interfaz de un sistema debe ser agradable para el público y basada en todos los requerimientos del sistema, así como dar al usuario confianza y facilidad de usar la interfaz.

Figura 5.2 Interfaz del Sistema.

5.3. Página de Inicio

La figura 5.3 y 5.3 a) muestra la página de inicio del sistema como producto final, en esta página el usuario (Administrador, secretario académico, público) puede realizar todas las actividades como ingresar nuevas imágenes de los edificios, actualizar la capacidad de un salón o simplemente visualizar una facultad donde se puede saber la ubicación del mismo.

Figura 5.3 Página de Inicio del Sistema.

Figura 5.3 a) Menú Desplegable del Sistema.

5.4. Pruebas del Sistema

Desde el punto de vista de la programación, nos interesa la ausencia de errores, la confiabilidad y la eficiencia, realizando pruebas de caja blanca y caja negra los cuales nos ayudaran a la detección de errores del sistema y así poder corregirlos antes de entregar el producto final.

En la figura 5.4 y 5.4 a) se muestra el funcionamiento del sistema para una consulta.

El usuario elije la Facultad que desea ubicar si lo requiere el edificio e incluso el salón, el sistema le muestra imagen del inmueble seleccionado.

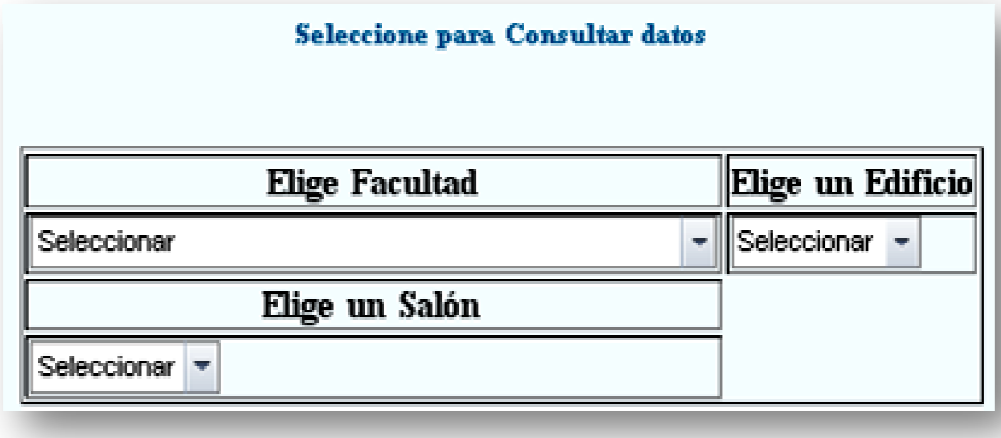

Figura 5.4 Bloque para consultas

Figura 5.4 a) Bloque para consultas

Cuando el usuario desea consultar un edificio, el sistema muestra en pantalla la imagen del mismo, así como algunas características que le servirán al usuario de guía, ejemplo figura 5.4 b).

Figura 5.4 b) Imagen con Detalles

5.4.1. Actualizar Datos

Cuando el usuario necesite realizar una actualización o modificación de la información de algún inmueble, selecciona en el menú la opción de "Actualizar" y el sistema le muestra 3 botones para que seleccione lo que desea modificar.

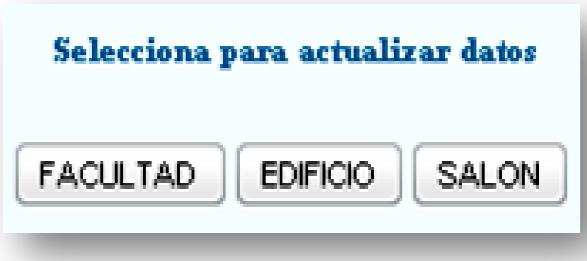

Figura 5.4.1 Bloque para actualización.

En este caso el usuario elije el botón "FACULTAD" enseguida el sistema le muestra en pantalla la siguiente imagen:

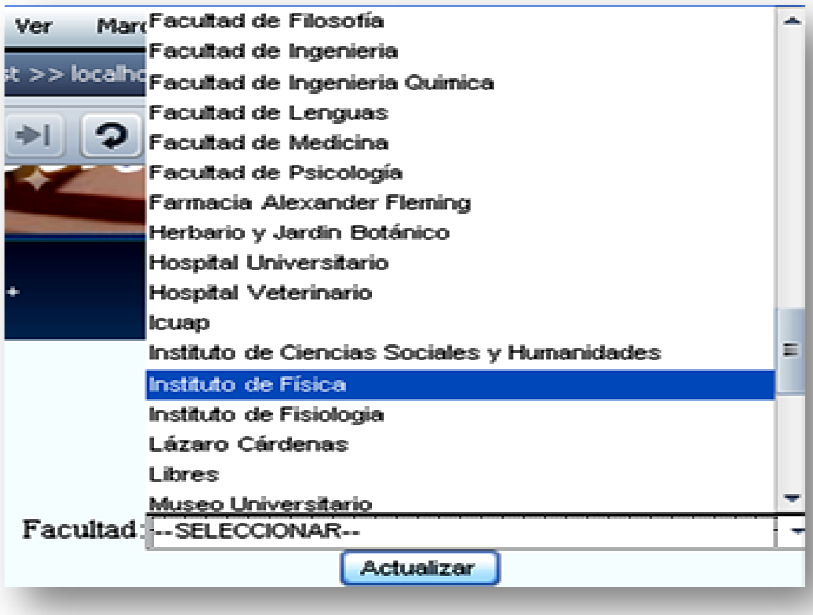

Figura 5.4.1 a) Bloque donde el usuario elije que facultad va a actualizar.

Una vez que se haya elegido la Facultad, el sistema muestra en pantalla un bloque con las características de la Facultad deseada, ahí el usuario modifica datos y guarda los cambios.

Figura 5.4.1 b) Bloque que muestra información de la Facultad elegida.

5.4.2. Insertar Datos

Cuando el usuario da click en el menú y elige un área, el sistema le muestra en pantalla un submenú.

El usuario elige la opción Insertar, el sistema le muestra en pantalla el siguiente bloque.

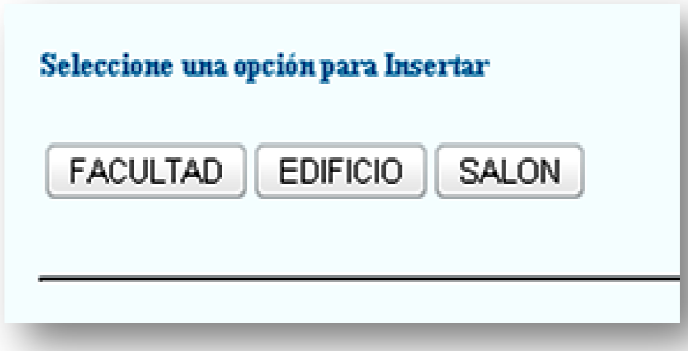

Figura 5.4.2 Bloque donde el usuario elije que facultad va a actualizar.

El usuario da click en el botón "Facultad", el sistema le muestra en pantalla varios campos donde ingresa la información para dar de alta una nueva Facultad, así mismo sucede con edificio y salón.

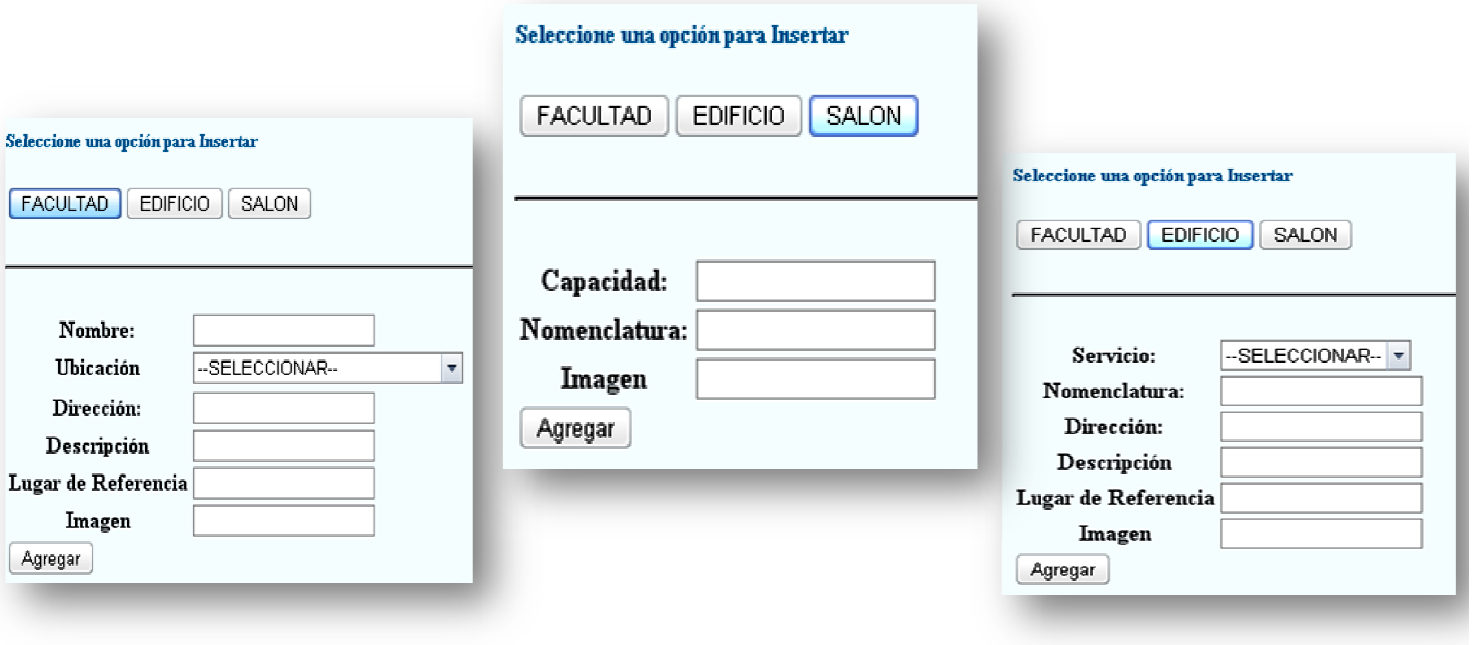

Figura 5.4.2 a) Bloques para dar de alta facultades, edifico y salones.

Cuando el usuario ingresa la información, para este ejemplo salón se ve reflejado en las opciones de Salón y automáticamente lo guarda en la base de datos y se verá reflejado en el sistema como se muestra en la figura 26.0.

IdFacultad	IdZonaF	nombreF	direccion	descripcion	lugar_referencia	imagen
1	1	Facultad de Contaduria Publica				
2	1	Instituto de Fisiologia				
3	1	Icuap				
4	1	Facultad de Cs. de la Computación				
5	1	Facultad de Cs. Quimicas				
6	1	Facultad de Ingenieria Quimica				
7	1	Facultad de Arquitectura				
8	1	Facultad de Ingenieria				
9	1	Facultad de Cs. de la Electronica	cu	esta en cu	polideportivo	
10	1	Instituto de Física	prueba	prueba inserta	prueba actualiza	sup.jpg

Figura 5.4.2 b) Base de datos donde se guarda la información ingresada por el usuario.

5.5. Seguridad del sistema

La seguridad en la Ingeniería de Software es un tema amplio.

La seguridad de software aplica los principios de la seguridad de información al desarrollo de software. Information security (La seguridad de información) se refiere a la seguridad de información comúnmente como la protección de sistemas de información contra el acceso desautorizado o la modificación de información, si está en una fase de almacenamiento, procesamiento o tránsito. También la protege contra la negación de servicios a usuarios desautorizados y la provisión de servicio a usuarios desautorizados, incluyendo las medidas necesarias para detectar, documentar, y contrarear tales amenazas [10].

La seguridad requiere más manejo y riesgo de mitigación, de la que requiere la tecnología. Como un desarrollador, uno primero debe de determinar los riesgos de una aplicación particular. Por ejemplo, el Website típico de hoy puede ser sujeto de una variedad de riesgos; la desfiguración o la negación distribuida de ataques del servicio.

Una vez que se identifiquen los riesgos, identificar medidas de seguridad apropiadas llega a ser manejable.

En particular, al definir los requisitos, es importante considerar cómo la aplicación será utilizada.

6. Conclusiones

Hoy en día la tecnología ha tenido un gran avance en cuanto a la forma de desarrollar sistemas, páginas, plataformas, etc., es por ello que los desarrolladores web se deben actualizar constantemente para la realización de los proyectos que se les presenten.

Es tanta la demanda que ahora el usuario exige sistemas más llamativos al público pero que sean seguros, amigables y fáciles de usar de ahí que se crean tantos programas que ayudan a la programación.

También es muy importante tener conocimiento en diseño de imágenes, animación 3D ya que estos complementan el sistema o páginas web para que sean atractivas.

7. Trabajo a Futuro

Se pretende contar a futuro con un Sistema Administrativo para la Planta Física de la BUAP interactivo donde el usuario pueda visitar las instalaciones de alguna facultad desde su computadora, con ello logrará reconocer inmediatamente el edificio y también podrá saber la ubicación exacta del mismo.

Con esto la BUAP brindara un mejor servicio a la comunidad estudiantil e incluso al público en general.

8. Glosario de Términos

Área.- Nombre asignado a un lugar físico donde se localizan inmuebles de la BUAP.

Instalaciones.- Sinónimo para referirse a edificio y salones de la BUAP.

Zona.- Nombre asignado para lugares recreativos o de servicio como: canchas, albercas, polideportivo, laguna, parque botánico, terminal stu (transporte universitario), estadio.

Infraestructura.- Nombre que se le asigna al conjunto en general de edificios que conforman a la BUAP.

Inmueble.- Nombre asignado para referenciarse a algún edificio o salón en específico de la BUAP.

Edificio.- Nombre asignado para referirse a un inmueble más particular.

Facultad.- Nombre asignado al conjunto de edificios que conforman el área de estudio de alguna carrera de la BUAP.

Información.- Se refiere a datos específicos de un salón o edificio como: dirección, lugar de referencia, imagen, nombre (se utiliza una nomenclatura) y la forma de accesar a dicho inmueble. Esta información le servirá de guía al usuario para ubicar lugares específicos de la infraestructura de la BUAP.

9. Bibliografía

[1] DATE. Introducción a los Sistemas de Bases de Datos. Pearson Educación.

[2] SILBERSHATE. Fundamentos de Bases de Datos. Mc Graw Hill.

[3] DATE. Sistemas de Bases de Datos. Pearson Educación.

[4] ULLMAN. Sistemas de Bases de Datos. Pearson Educación.

[5] DUBOIS PAUL. Programación en MySQL. Tercera Edición. Madrid: Anaya Multimedia, 2005.

[6] L. WELLING, L. THOMSON. Desarrollo Web con PHP y MySQL. Tercera Edición. Madrid: Anaya Multimedia. 2005.

[7] SOMMERVILLE IAN. Ingeniería de software. Sexta Edición. Madrid: Pearson Educación, 2005.

[8] S. PRESSMAN ROGER. Ingeniería de Software. Un enfoque práctico. Quinta Edición. Madrid: Mc Graw Hill, 2002.

[9] INGENIERIA DEL SOFWARE, Ian Sommerville (Prentice Hall), 7ma Edicion

[10] SECURING PHP WEB APPLICATIONS, Dec. 2008, Addison Wesley, PDF

Material de Apoyo Visitado en la Web

http://es.wikipedia.org/wiki/MySQL

http://es.wikipedia.org/wiki/Cifrado_de_Vigen%C3%A8re

http://www.desarrolloweb.com/articulos/xajax-libreria-php.html

http://www.programacion.com/articulo/tutorial_basico_de_mysql_189/11

http://es.wikipedia.org/wiki/Servidor_HTTP_Apache