Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

"Sistema de Registro de Incidencias como parte del Sistema Integral de Seguridad Universitaria".

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

PRESENTA**:**
Erick Albertho Euan Waldestrand.

ASESOR**:**
Dra. Darnes Vilariño Ayala.

Puebla, Pue. Marzo 2008

Agradecimiento

A Dios, por su compañía y permitirme cumplir una de mis grandes metas.

A mis padres, Delfina Waldestrand Cazarín y Diego Alberto Euan Estrella, por la oportunidad que me dieron de estudiar y lograr tener una formación profesional.

A mi extraordinaria hermana, por escucharme y aportar en mí maravillosos consejos.

A mis abuelos, porque con sus experiencias me orientaron hacia el mejor camino.

A la Dra. Darnes Vilariño Ayala, la Mtra. Mireya Tovar Vidal y el Dr. Luís Enrique Colmenares Guillén por su apoyo académico.

A mis familiares y amigos, por todos los momentos que compartimos juntos y su invaluable apoyo en los momentos difíciles.

Dedicatoria

Dedicado a mi familia en especial a ti Gisey, ustedes son lo mejor que tengo, juntos hemos aprendido que con dedicación y esfuerzo podemos lograr las metas que nos propongamos.

INDICE:

INTRODUCCIÓN

En todos los centros educativos de nivel superior, centros de investigación, empresas industriales, dependencias de gobierno, ocurren hechos relacionados con la seguridad, los hechos ocurridos son atendidos por gente encargada de la seguridad de cada organización, las organizaciones que tienen la tarea de atender los hechos no todas cuentan con un sistema para registrar las diferentes incidencias, analizarlas y describir su comportamiento.

En nuestra universidad existe una dependencia encargada de la seguridad, esta es la Dirección de Protección Universitaria, dicha organización es dependiente de la Secretaría Administrativa en la Benemérita Universidad Autónoma de Puebla, tiene la función de proteger y salvaguardar la integridad de la comunidad Universitaria, visitantes y el patrimonio de la Institución, implementando para ello medidas de seguridad, protección civil, atención a emergencias, siendo un gestor de la prevención de incidencias.

Esta dependencia clasifica los hechos y les llama incidencias, las incidencias son hechos ocurridos que tienen que ver con la seguridad en la universidad, por ejemplo un vehiculo abierto, una amenaza de bomba, un derrame de sustancia peligrosa por mencionar algunos, cada uno de los anteriores representara un tipo de incidencia.

La universidad tiene la necesidad de tener un sistema en donde registre los reportes de las incidencias ocurridas, debido a esto, construiremos un sistema que cumpla con sus necesidades aplicando metodologías, modelos y herramientas de desarrollo, que nos permitan analizar, diseñar e implementar un sistema que tenga las características deseadas para el facilitarles el manejo de su información.

Elaboraremos y describiremos los diagramas correspondientes a cada metodología, para la creación de la base de datos elegiremos un gestor de base de datos y para el desarrollo del sistema un lenguaje basado en implementaciones Web.

Objetivo general.

El objetivo es elaborar un sistema Web para llevar un control de todos los reportes de incidencias que puedan ocurrir en áreas que pertenecen a la universidad, con este sistema podremos automatizar este proceso ya que los integrantes del área de seguridad podrán registrar los reportes desde cualquier computadora que tenga acceso a Internet, el acceso al sistema tendrá restricciones por lo tanto cada uno deberá tener una clave de acceso para capturar los registros correspondientes.

Objetivos particulares.

A continuación describiremos los objetivos particulares de la tesis.

- Analizar la problemática del almacenamiento de la información y proponer una solución.
- Caracterizar el sistema basado en la Teoría de la Ingeniería de Software, Bases de Datos y arquitectura Cliente-Servidor para su desarrollo.
- Aplicación del Proceso Unificado de desarrollo de Software para la construcción del Sistema.
- Diseñar la base de datos siguiendo el Modelo Relacional.
- Implementación de la base de datos en MySql.
- Implementación de sistema utilizando el lenguaje de programación Php.
- Realizar las pruebas del Sistema.
- Escritura de la Tesis.

Organización de la tesis:

Capítulo 1.

En el capítulo uno presentamos el marco teórico sobre el cual se fundamentara el sistema; se mencionan conceptos fundamentales de la ingeniería de software como son; alternativas de modelos de ciclo de vida y proceso de desarrollo de software. Explicaremos también que es una base de datos, ventajas, desventajas de las bases de datos, en que consisten los Sistemas de Administración de las Bases de Datos (DBMS), características de los (DBMS), la arquitectura Cliente-Servidor, la estructura del enfoque relacional y la Jerarquía de Generalización.

Capítulo 2.

En el capítulo dos se establece el planteamiento del problema, la especificación de los requerimientos de sistema, la elaboración de casos de uso, el diagrama (Entidad-Relación), el Modelo Relacional, el Diseño Físico de la Base de Datos, la Normalización de la Base de datos y diseño de la interfaz del sistema.

Capítulo 3.

En el capítulo tres se eligen las herramientas de implementación con las que realizaremos el sistema (PHP y MySQL), las funcionalidades que proporcionan las herramientas elegidas, la creación de la base de datos con el DBMS y explicaremos las sentencias SQL ocupadas para la creación de la base de datos en MySQL.

Capítulo 4.

En el capítulo cuatro mostraremos el desarrollo de las diferentes interfaces del sistema, realizaremos pruebas con valores reales, explicaremos la manera detallada como utilizar correctamente en sistema.

CAPÍTULO 1: MARCO TEÓRICO

Introducción

La ingeniería de software, los conceptos de bases de datos y la arquitectura Cliente-Servidor son temas fundamentales para el desarrollo de esta tesis a continuación veremos lo mas importante de cada una de estas metodologías y posteriormente las aplicaremos en el sistema de base de datos que desarrollaremos.

1.1.-INGENIERÍA DE SOFTWARE.

La Ingeniería de Software es el establecimiento y uso de principios de ingeniería para obtener software que sea confiable y que funcione eficientemente en máquinas reales.

La Ingeniería de Software es relativamente nueva ya que aparece a finales de los años sesenta y principios de los setenta, comenzando con las Técnicas de Programación Estructurada, incorporándolas a las fases del ciclo vital de software. La Programación Estructurada fue seguida por otros métodos estructurados de análisis y también métodos estructurados de diseño. Además, comenzaron a usarse tecnologías orientadas a objetos. En un principio la programación era la tarea de oro de la Ingeniería de Software pero ahora la ingeniería y el diseño de requisitos son más importantes.

En la década pasada, los estándares de la Ingeniería de Software y la madurez de proceso han caracterizado la industria del software como una disciplina madura.

En un nivel más técnico, la Ingeniería de Software comienza con una serie de tareas que hacen modelos y que resultan en una especificación completa de requisitos y una representación comprensiva de diseño del software que será construido. Se han desarrollado muchos métodos para hacer modelos de sistemas de información. Sin embargo, los métodos Orientados a Objeto (OO) van a llegar a ser el estándar.

Para ciertos sistemas de información críticos, el uso de métodos formales es recomendado en el ciclo de vida del software, para producir sistemas con la integridad más alta. Los métodos formales confían en las técnicas matemáticas que expresan y modelan los requisitos de cualquier producto en el ciclo vital del software. [1]

1.1.1.- Objetivos de la Ingeniería de Software

En la construcción y desarrollo de proyectos se aplican métodos y técnicas para resolver los problemas, la informática aporta herramientas y procedimientos sobre los que se apoya la ingeniería de software.

- Mejorar la calidad de los productos de software
- Aumentar la productividad y trabajo de los ingenieros del software.
- Facilitar el control del proceso de desarrollo de software.
- Suministrar a los desarrolladores las bases para construir software de alta calidad en una forma eficiente.

- Definir una disciplina que garantice la producción y el mantenimiento de los productos de software desarrollados en el plazo fijado y dentro del costo estimado.

1.1.2.-Etapas del Proceso de Software.

La ingeniería de software requiere llevar a cabo muchas tareas, sobre todo las siguientes**:**

Análisis de requisitos

Extraer los requisitos de un producto de software es la primera etapa para crearlo. Mientras que los clientes piensan que ellos saben lo que el software tiene que hacer, se requiere de habilidad y experiencia en la ingeniería de software para reconocer requisitos incompletos, ambiguos o contradictorios. El resultado del análisis de requisitos con el cliente se plasma en el documento ERS, *Especificación de Requerimientos del Sistema*, cuya estructura puede venir definida por varios estándares, tales como CMM-I. Asimismo, se define un diagrama de Entidad/Relación, en el que se plasman las principales entidades que participarán en el desarrollo del software.

Especificación

Es la tarea de describir detalladamente el software a ser escrito, en una forma matemáticamente rigurosa. En la realidad, la mayoría de las buenas especificaciones han sido escritas para entender y afinar aplicaciones que ya estaban desarrolladas. Las especificaciones son más importantes para las interfaces externas, que deben permanecer estables.

Diseño y arquitectura

Se refiere a determinar como funcionará de forma general sin entrar en detalles. Según Yourdon consiste en incorporar consideraciones de la implementación tecnológica, como el hardware, la red, etc. Se definen los casos de uso para cubrir las funciones que realizará el sistema, y se transforman las entidades definidas en el análisis de requisitos en clases de diseño, obteniendo un modelo cercano a la programación orientada a objetos.

Programación

Reducir un diseño a código puede ser la parte más obvia del trabajo de ingeniería de software, pero no es necesariamente la porción más larga.

Prueba

Consiste en comprobar que el software realice correctamente las tareas indicadas en la especificación. Una técnica de prueba es probar por separado cada módulo del software, y luego probarlo de forma integral.

Documentación

Realización del manual de usuario, y posiblemente un manual técnico con el propósito de lograr dar mantenimiento futuro y ampliaciones al sistema.

Mantenimiento

Mantener y mejorar el software para enfrentar errores descubiertos y nuevos requisitos. Esto puede llevar más tiempo incluso que el desarrollo inicial del software. Alrededor de 2/3 de toda la ingeniería de software tiene que ver con dar

mantenimiento. Una pequeña parte de este trabajo consiste en arreglar errores, o *bugs*. La mayor parte consiste en extender el sistema para hacer nuevas cosas.

1.1.3. – Modelos de ciclo de vida.

Un modelo de ciclo de vida de software es una vista de las actividades que ocurren durante el desarrollo de software, intenta determinar el orden de las etapas involucradas y los criterios de transición asociadas entre estas etapas.

Un modelo de ciclo de vida del software:

- Describe las fases principales de desarrollo de software.
- Define las fases primarias esperadas de ser ejecutadas durante esas fases.
- Ayuda a administrar el progreso del desarrollo
- Provee un espacio de trabajo para la definición de un detallado proceso de desarrollo de software.

Todo proyecto de ingeniería tiene unos fines ligados a la obtención de un producto, proceso o servicio que es necesario generar a través de diversas actividades. Algunas de estas actividades pueden agruparse en fases porque globalmente contribuyen a obtener un producto intermedio, necesario para continuar hacia el producto final y facilitar la gestión del proyecto. Al conjunto de las fases empleadas se le denomina **"ciclo de vida"**.

Sin embargo, la forma de agrupar las actividades, los objetivos de cada fase, los tipos de productos intermedios que se generan, etc. pueden ser muy diferentes dependiendo del tipo de producto o proceso a generar y de las tecnologías empleadas.

La complejidad de las relaciones entre las distintas actividades crece exponencialmente con el tamaño, con lo que rápidamente se haría inabordable si no fuera por la vieja táctica de "divide y vencerás". De esta forma la división de los proyectos en fases sucesivas es un primer paso para la reducción de su complejidad, tratándose de escoger las partes de manera que sus relaciones entre sí sean lo más simples posibles.

La definición de un ciclo de vida facilita el **control sobre los tiempos** en que es necesario aplicar recursos de todo tipo (personal, equipos, suministros, etc.) al proyecto. Si el proyecto incluye subcontratación de partes a otras organizaciones, el **control del trabajo subcontratado** se facilita en la medida en que esas partes encajen bien en la estructura de las fases. El **control de calidad** también se ve facilitado si la separación entre fases se hace corresponder con puntos en los que ésta deba verificarse (mediante comprobaciones sobre los productos parciales obtenidos).

De la misma forma, la práctica acumulada en el diseño de modelos de ciclo de vida para situaciones muy diversas permite que nos beneficiemos de la **experiencia adquirida** utilizando el enfoque que mejor se adapte a nuestros requerimientos. [1]

1.1.3.1.- Alternativas de Modelos de ciclo de Vida.

Hay diferentes modelos de ciclos de vida que podemos utilizar para el desarrollo de nuestro sistema, en la definición del plan de proyecto el modelo de ciclo de vida que seleccionemos influye en el éxito que logre el proyecto como cualquier otra decisión de planificación que se tome.

El modelo de ciclo de vida apropiado puede orientar su proyecto y ayudarte a asegurar que cada paso se acerque más a la consecución del objetivo.

Dependiendo el modelo de ciclo de vida que seleccionemos se puede aumentar la velocidad de desarrollo, mejorar la calidad, el control y seguimiento del proyecto, minimizar gastos y riesgos o mejorar la relación con los clientes.

Si seleccionamos un modelo ineficaz, corremos el riesgo de que nuestro trabajo sea lento, repetitivo, innecesario y frustrante. Los mismos efectos pueden afectar si no elegimos un modelo se ciclo de vida.

1.1.3.2.- Modelo de Cascada.

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

Un ejemplo de una metodología de desarrollo en cascada es:

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

De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo. La palabra *cascada* sugiere, mediante la metáfora de la fuerza de la gravedad, el esfuerzo necesario para introducir un cambio en las fases más avanzadas de un proyecto, en la **Figura 1.1** vemos su esquema.

Si bien ha sido ampliamente criticado desde el ámbito académico y la industria, sigue siendo el paradigma más seguido al día de hoy.

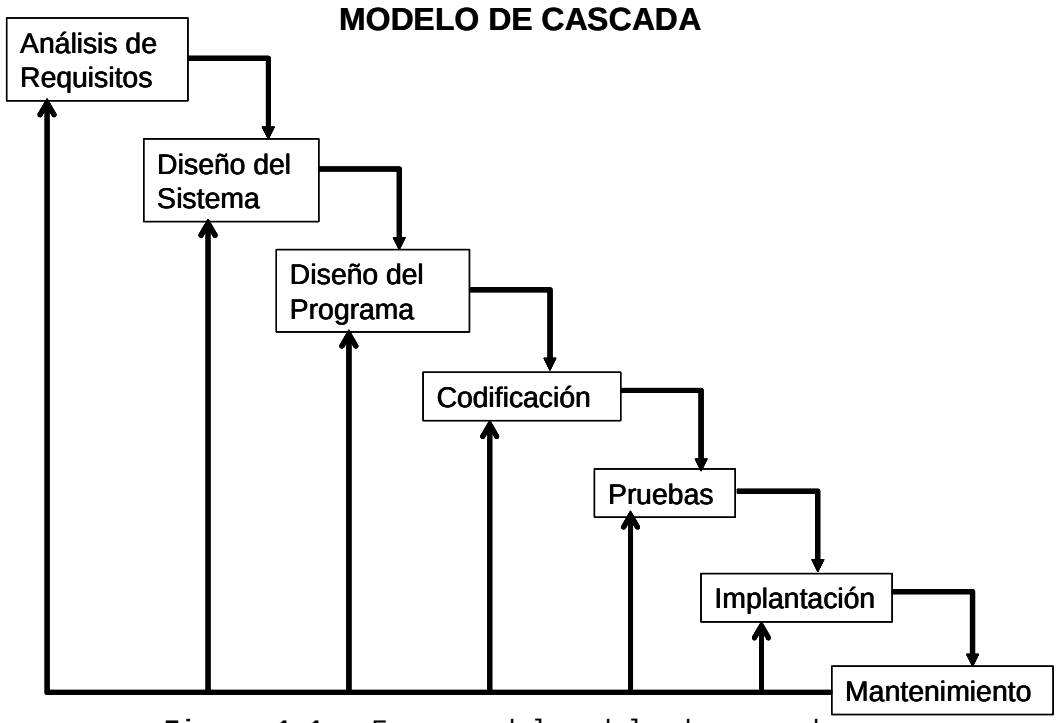


Figura 1.1.- Esquema del modelo de cascada.

Fases del Modelo de Cascada:

Análisis de Requisitos:

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

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

Diseño del Sistema:

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

Diseño del Programa:

Es la fase en donde se realizan los algoritmos necesarios para el cumplimiento de los requerimientos del usuario así como también los análisis necesarios para saber que herramientas usar en la etapa de Codificación.

Codificación:

Es la fase de programación propiamente dicha. Aquí se desarrolla el código fuente, haciendo uso de prototipos así como pruebas y ensayos para corregir errores. Dependiendo del lenguaje de programación y su versión se crean las librerías y componentes reutilizables dentro del mismo proyecto para hacer que la programación sea un proceso mucho más rápido.

Pruebas:

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

Implantación:

El software obtenido se pone en producción. Se implementan los niveles software y hardware que componen el proyecto. La implantación es la fase con más duración y con más cambios en el ciclo de elaboración de un proyecto. Es una de las fases finales del proyecto. Durante la explotación del sistema software pueden surgir cambios, bien para corregir errores o bien para introducir mejoras. Todo ello se recoge en los Documentos de Cambios.

Variantes:

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

Dificultad del Modelo:

La dificultad de este modelo reside en hacer cambios entre etapas.

1.1.3.3.- Modelo de Espiral.

En este modelo el esfuerzo de desarrollo es iterativo. Tan pronto como uno completa un esfuerzo de desarrollo otro comienza. Tiene una ventaja este modelo en el análisis de riesgos, ya que tiene la opción de seguir o no, según las necesidades del cliente.

El modelo en espiral que mostramos en la **`Figura 1.2`** contiene algunos principios básicos que mencionamos a continuación.

- Planteamiento de objetivos.
- Se identifican los objetivos específicos para cada fase del proyecto.
- Identificación y reducción de riesgos.
- Los riesgos claves se identifican, analizan y la información sirve para minimizar los riesgos.
- Desarrollo y Validación.
- Se elige un modelo apropiado para la siguiente fase de desarrollo
- Planeación.
- Se revisa el proyecto y se trazan planes para la siguiente ronda del espiral.

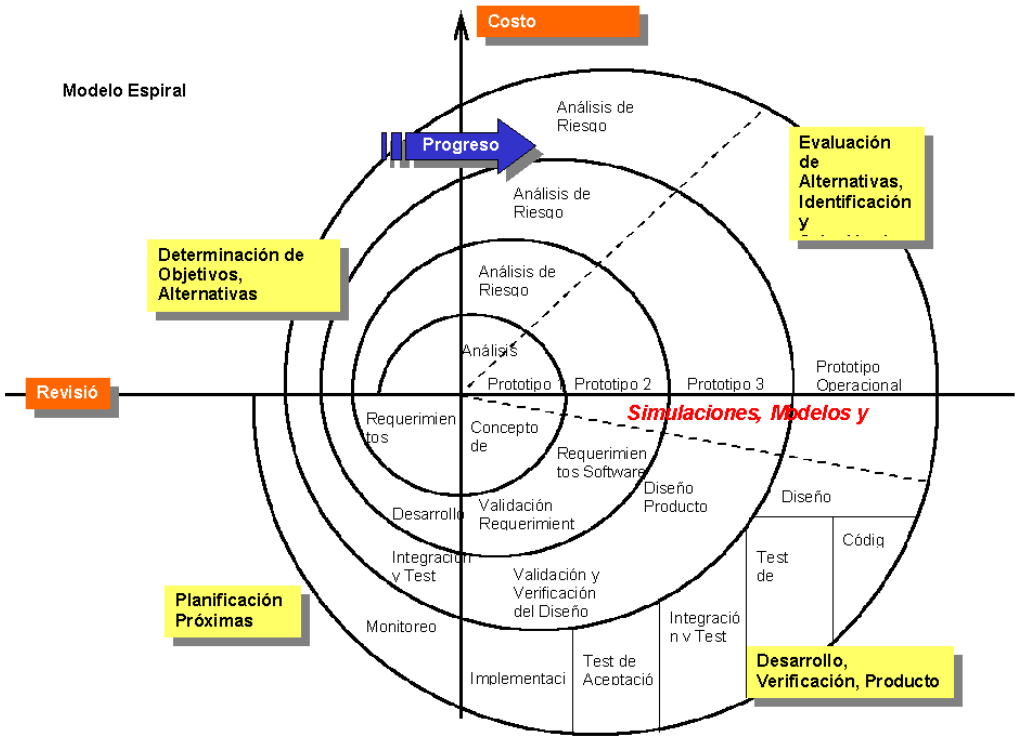


Figura 1.2.- Esquema del modelo en Espiral.

1.1.3.4.- Modelo de Prototipo.

El modelo de prototipo que presentamos en la **Figura 1.3.-**, se utiliza cuando los requerimientos del sistema no son muy claros ó no se identifican en forma detallada los requerimientos de entrada, salida y funciones. Puede tomar alguna de las siguientes formas.

- Un escenario (simulación del uso del sistema).
- Una demostración (porciones de código que realizan algunas funciones).
- Una Versión 0 (aplicada liberada que puede usarse bajo condiciones preliminares añadiendo, cambiando o quitando funciones existentes y creándole su documentación).

MODELO DE PROTOTIPO

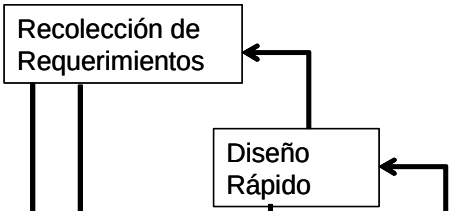


Figura 1.3.- Esquema del modelo de Prototipo.

1.1.4.- Herramientas Case.

CASE es una sigla, que corresponde a las iniciales de: **C**omputer **A**ided **S**oftware **E**ngineering; y en su traducción al Español significa Ingeniería de Software Asistida por Computación.

Podemos definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software (Investigación Preliminar, Análisis, Diseño, Implementación e Instalación.).

CASE es también definido como el Conjunto de métodos, utilidades y técnicas que facilitan el mejoramiento del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.

Se puede ver al CASE como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales.

Existe también el CASE integrado que fue comenzando a tener un impacto muy Significativo en los negocios y sistemas de información de las organizaciones, además con este CASE integrado las compañías pueden desarrollar rápidamente sistemas de mejor calidad para soportar procesos críticos del negocio y asistir en el desarrollo y promoción intensiva de la información de productos y servicios.

Concentrando nuestra atención en el uso de estas herramientas, para el desarrollo de proyectos informáticos que tengan como objetivo la automatización de procedimientos administrativos; podemos decir que:

Las herramientas CASE representan una forma que permite Modelar los Procesos de Negocios de las empresas y desarrollar los Sistemas de Información Gerenciales.

1.1.4.1.- Metodología CASE.

El método CASE encierra un conjunto integrado de técnicas, estas incluyen entrevistas con los usuarios finales, secciones de retroalimentación, análisis de la organización del sistema además de técnicas para modelar y representar las funciones de la organización, los diagramas Entidad-Relación y el flujo de datos a través del sistema.

El método CASE divide el ciclo de vida de los sistemas dentro de seis fases diferentes y les da un peso de acuerdo a la cantidad promedio de tiempo gastado en cada uno, tal y como se lista a continuación:

1.- Estrategias
2.- Análisis
3.- Diseño
4.- Construcción
5.- Transición
6.-Producción.

A través de cada fase, el método CASE controla los movimientos y acciones d los diseñadotes de la aplicación para refinar sus diseños y para hacer uso de su tiempo mas productivo. CASE fuerza el uso de modelos para expresar las funciones de la organización y los requerimientos del sistema, además usa técnicas que facilitan la comunicación del diseñador por un lado y los usuarios finales por otro.

Una documentación establece metas que conducen al éxito del sistema. Un análisis, para llegar al éxito, requiere de lo siguiente:

- Selección del blanco óptimo
- Generación de una documentación detallada de tal manera que subsecuentes modificaciones puedan ser evaluadas con base a la afectación al blanco principal.
- Predecir parámetros importantes asociados, factibilidad del sistema, costos, beneficios, programación o calendarización de proyectos y funcionamientos.
- Coordinación entre los puntos anteriores.

1.1.5.- Proceso unificado de desarrollo de 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, a continuación mencionamos las etapas del proceso.

1. **Proceso de Desarrollo de Software:** Proceso de negocio, o caso de uso de negocio, de un negocio de desarrollo de software. Conjunto total de actividades necesarias para transformar los requisitos de un cliente en un conjunto consistente de artefactos que representan un producto de software y en punto posterior en el tiempo para transformar cambios en dichos requisitos en nuevas versiones del producto.
2. **Lenguaje Unificado de Modelado (UML):** Lenguaje estándar para el modelado de software lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema con gran cantidad de software. Lenguaje usado por el Proceso Unificado. Lenguaje que permite a los desarrolladores visualizar el producto de su trabajo (Artefactos) en esquemas o diagramas estandarizados.
3. **Proceso Unificado:** Proceso de desarrollo de software basado en el Lenguaje Unificado de Modelado y que es iterativo, centrado en la arquitectura y dirigido por los casos de uso y los riesgos. Proceso que se organiza en cuatro fases: inicio, elaboración, construcción y transición, y que se estructura en torno a cinco flujos de trabajo fundamentales: recopilación de requisitos, análisis, diseño, implementación y pruebas. Proceso que se describe en términos de un modelo de negocio, el cual esta a su vez estructurado en función de tres bloques de construcción primordiales trabajadores, actividades y artefactos.
4. **Requisitos:** Flujo de trabajo fundamental cuyo propósito esencial es orientado al desarrollado hacia el sistema correcto. Esto se lleva a cabo mediante la descripción de los requisitos del sistema de forma tal que se pueda llegar a un acuerdo entre el cliente (incluyendo los usuarios) y los desarrolladores del sistema, acerca de lo que el sistema debe hacer y lo que no.
5. **Análisis:** Flujos de trabajo fundamental cuyo propósito principal es analizar los requisitos descritos en la captura de requisitos, mediante su refinamiento y estructuración. El objetivo de esto es (1) lograr una comprensión mas precisa de los requisitos, y (2) obtener una descripción de los requisitos que sea fácil de mantener y que nos ayude a dar estructura al sistema en su conjunto incluyendo su arquitectura.
6. **Diseño:** Flujo de trabajo fundamental cuyo propósito principal es la de formular modelos que se centran en los requisitos no funcionales y el dominio de la solución y que prepara para la implementación y pruebas del sistema.
7. **Implementación:** Flujo de trabajo fundamental cuyo propósito esencial es implementar el sistema en términos de componentes, es decir código fuente guiones, ficheros binarios, ejecutables, et.
8. **Prueba:** Flujo de trabajo fundamental cuyo propósito esencial es comprobar el resultado de la implementación mediante las pruebas de cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema que van a ser entregadas a terceras personas.
9. **Fase de Inicio:** Primera fase del ciclo de vida del software, en la que la idea inicial para el desarrollo es refinada hasta el punto de quedar lo suficientemente bien establecida como para garantizar la entrada en la base de elaboración.
10. **Fase de Elaboración:** Segunda fase del ciclo de vida, en la que se define la arquitectura.
11. **Fase de Construcción:** Tercera fase del ciclo de vida del software, en la que el software es desarrollado a partir de una línea base de la arquitectura ejecutable, hasta el punto en el que se esta listo para ser transmitido a las comunidades de usuarios.

12. **Fase de Transición:** Cuarta fase del ciclo de vida del software es puesto en manos de la comunidad de usuarios.
13. **Arquitectura:** Conjunto de decisiones significativas, acerca de la organización de un sistema software, la selección de los elementos estructurales a partir de los cuales se compone el sistema y las interfaces entre ellos, junto con su comportamiento, tal y como se especifica en las colaboraciones entre estos elementos, la composición de estos elementos estructurales y de comportamiento de subsistemas progresivamente mayores, y el estilo arquitectónico que guía esta organización, estos elementos y sus interfaces, sus colaboraciones y su composición. La arquitectura de software se interesa no solo por la estructura y el comportamiento, sino también por las restricciones y compromisos de uso, funcionamiento, flexibilidad al cambio, reutilización, comprensión, economía y tecnología, así como aspectos estéticos.
14. **Vista Arquitectónica del Modelo de Casos de Uso:** Vista de la arquitectura de un sistema abarcando los casos de uso significativos desde un punto de vista arquitectónico.
15. **Vista Arquitectónica del Modelo de Análisis:** Vista arquitectónica de un sistema, abarcando las clases, paquetes y realizaciones de casos de uso del análisis, vista que fundamentalmente aborda el refinamiento y estructuración de los requisitos del sistema. La estructura de esta vista se preserva en la medida de lo posible cuando se diseña e implementa la arquitectura del sistema.
16. **Vista Arquitectónica del Modelo de Diseño:** Vista de la arquitectura de un sistema, abarcando las clases, subsistemas, interfaces y realizaciones de casos de uso del diseño que forman el vocabulario del dominio de la solución del sistema, vista que abarca también los hilos y procesos que establecen la concurrencia y mecanismos de sincronización del sistema, vista que aborda los requisitos no funcionales, incluyendo los requisitos de rendimiento y capacidad de crecimiento de un sistema.
17. **Vista Arquitectónica del Modelo de Despliegue:** Vista de la arquitectura de un sistema abarcando los nodos que forman la topología hardware sobre la que se ejecuta el sistema, vista que aborda la distribución, entrega e instalación de las partes que constituyen el sistema físico.
18. **Vista Arquitectónica del Modelo de Implementación:** Vista arquitectónica de un sistema, abarcando los componentes usados para el ensamblado y lanzamiento del sistema físico, vista que aborda la gestión de la configuración de las versiones del sistema, constituida por componentes independientes que pueden ser ensambladas de varias formas para producir un sistema ejecutable.[2]

1.2.- BASES DE DATOS

El término de bases de datos fue escuchado por primera vez en 1963, en un simposio celebrado en California –USA.

Una base de datos se puede definir como un conjunto de información relacionada que se encuentra agrupada ó estructurada.

Desde el punto de vista de la informático, una base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Una base de datos tiene mucha importancia en el ritmo de vida que llevamos en la actualidad momentos, ya que, está acelera los procesos en donde se maneja una gran cantidad de información.

A continuación escribimos algunos conceptos de las Bases de Datos**:**

Base de Datos: es la colección de datos aparentes usados por el sistema de aplicaciones de una determinada empresa.

Base de Datos: es un conjunto de información relacionada que se encuentra agrupada o estructurada. Un archivo por sí mismo no constituye una base de datos, sino más bien la forma en que está organizada la información es la que da origen a la base de datos.

Base de Datos: colección de datos organizada para dar servicio a muchas aplicaciones al mismo tiempo al combinar los datos, de manera que aparezcan estar en una sola ubicación.

1.2.1.- Conceptos fundamentales de las bases de datos.

Al entender la arquitectura interna de una base de datos es una ayuda muy importante de cara al diseño y la construcción de Bases de Datos relacionales. Conocer la teoría nos permite realizar consultas más eficientes y detectar los problemas rápidamente, así como elaborar una solución para los mismos por eso se mencionan a continuación varios conceptos fundamentales de las bases de datos y definiciones:

Datos persistentes: Son aquellos datos que mantienen a la base de datos existiendo 2 tipos denominados *Datos de Entrada y Datos de Salida***.**

Datos de Entrada: Se refiere específicamente a la información que entra al sistema por primera vez, generalmente a través del teclado, esta información va alterar los datos persistentes y por consecuencia esta información no forma parte de la base de datos (Son parámetros para extraer información).

Datos de Salida: Es la información que se obtiene a través de la manipulación, puede ser conocida a través de una consulta del usuario ó por parte del programa ó del sistema.

Entidad: Se define por entidad a cualquier objeto distinguible que pueda representarse en una base de datos.

Relación: Es la correspondencia entre entidades en base a un atributo, generalmente estas son bi-direccionales y se clasifican de la siguiente manera:

1 – 1 Relación Uno a Uno.
1 – N Relación Uno a Muchos.
N - 1 Relación Muchos a Uno.
N – M Relación Muchos a Muchos.

Atributo: De cada entidad se almacenan una serie de datos que se denominan atributos de la entidad. Pueden ser Atributos de una entidad cualquier característica ó propiedad de esta. Por ejemplo, son atributos de la Entidad Usuario: Nombre, contraseña, correo.

Dominio: (Atributo, valor del dato) Un par para cada atributo del conjunto de entidades. Por ejemplo. Entidad Usuarios, Dominio (Nombre, A…..Z).

1.2.2.- Ventajas de utilizar bases de datos.

- Se evita la inconsistencia de la información
- Se evita la redundancia de la información, manteniendo integrada la base de datos y bien definidas las tablas.
- Evita la duplicidad de datos.
- Los datos pueden ser compartidos por varias aplicaciones ó varios usuarios.
- Se mantienen Niveles de Seguridad en los datos.

1.2.3.- Desventajas de utilizar bases de datos.

- Al perder o dañarse la base de datos de hacen irrecuperables las tablas, informes y consultas, que en ella existen.
- Al tener una base de datos con demasiada información su acceso es lento debido a que varios usuarios pueden estar haciendo uso de ella.

1.2.4.- Independencia de los datos

Se habla de datos independientes cuando en una base de datos se pueden alterar la estructura de almacenamiento o la forma de acceder a los datos sin afectar gravemente la aplicación, se dice que la independencia de los datos es parte de los objetivos de las bases de datos y de pude definir como la inmunidad de las aplicaciones ante los cambios de estructura de datos (almacenamiento) y a la técnica de acceso, es importante mencionar que en un sistema de base de datos no es recomendable tener aplicaciones dependientes de los datos.

1.2.5.- Arquitectura de un sistema de base de datos.

La arquitectura de un sistema de base de datos se divide en 3 niveles comunes: Nivel Interno, Conceptual y Externo.

- **Nivel Interno**: Es el mas cercano al almacenamiento físico, es decir, es el que se ocupa de la forma como se almacenan físicamente los datos.
- **Nivel Externo (Aplicaciones)**: Es el mas cercano a los usuarios, es decir, es el que se ocupe de la forma como los usuarios reciben los datos.
- **Nivel Conceptual (Modelo Entidad-Relación)**: Es el nivel de mediación entre los dos anteriores.

Nivel Externo: Es el nivel del usuario individual, es decir, los usuarios pueden ser programadores en algunos casos los usuarios finales, cada usuario dispone de un lenguaje y de un programador dependiendo del caso.

Dispone de un lenguaje convencional. En el caso de un usuario final, será un lenguaje de consulta o un lenguaje orientado hacia los usuarios. El punto importante de todos estos lenguajes es que debe incluir un sublenguaje de datos del cual estará inmenso o dentro de un lenguaje anfitrión, un lenguaje dado cualquiera, va ha permitir el empleo de varios lenguajes anfitriones y varios sublenguajes para todos.

Nivel Conceptual: La vista conceptual es una presentación de toda la información contenida en la base de datos. Además puede ser muy diferente en la forma en que percibe los datos cualquier usuario final, es decir, debe ser un panorama de los datos. Tal como son y no como los percibe los usuarios. Debido a las limitaciones del lenguaje o bien al equipo que se esta utilizando.

El nivel conceptual se define mediante un esquema conceptual el cual incluye la definición de cada uno de los tipos de registro (entidades), además, el esquema conceptual no debe asociarse a representaciones de campos almacenados tales como punteros, índices, etc.; si el esquema conceptual se desarrolla en forma independiente de los datos entonces el esquema externo definido en base al esquema conceptual será también independiente de los datos.

Nivel Interno: Representación de bajo nivel de toda la base de datos, se compone de varias ocurrencias de varios tipos de registros, el nivel interno todavía está a un paso del nivel físico ya que no se manejan los registros fijos. La vista interna se define a través de un esquema interno el cual no solo define los diversos tipos de registros almacenados, si no, también especifica los índices asociados, representación de los campos almacenados, secuencia física de los registros, etc.

1.2.6.- Sistemas de Administración de Bases de Datos (DBMS).

Un sistema de administración de bases de datos DBMS (Database Management System, por sus siglas en Inglés) 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, por sus siglas en ingles).

1.2.6.1.- Usos y Funciones de un DBMS.

Los sistemas de administración de bases de datos son usados principalmente 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.
- Proveer a los administradores las herramientas que les permitan ejecutar tareas de mantenimiento y administración de los datos.

Algunas de las funciones de un DBMS son:

- Definición de la base de datos - como la información va a ser almacenada y organizada.
- Creación de la base de datos - almacenamiento de datos en una base de datos definida.
- Recuperación de los datos - consultas y reportes.
- Actualización de los datos - cambiar los contenidos de la base de datos.
- Programación de aplicaciones para el desarrollo de software.
- Control de la integridad de la base de datos.
- Monitoreo del comportamiento de la base de datos.

1.2.6.2.- Características de un DBMS

Control de la redundancia de datos

Este consiste en lograr una mínima cantidad de espacio de almacenamiento para almacenar los datos evitando la duplicación de la información. De esta manera se logran ahorros en el tiempo de procesamiento de la información, se tendrán menos inconsistencias, menores costos operativos y hará el mantenimiento más fácil.

Compartimiento de datos

Una de las principales características de las bases de datos, es que los datos pueden ser compartidos entre muchos usuarios simultáneamente, proveyendo, de esta manera, máxima eficiencia.

Mantenimiento de la integridad

La integridad de los datos es la que garantiza la precisión o exactitud de la información contenida en una base de datos. Los datos interrelacionados deben siempre representar información correcta a los usuarios.

Soporte para control de transacciones y recuperación de fallas.

Se conoce como transacción toda operación que se haga sobre la base de datos. Las transacciones deben por lo tanto ser controladas de manera que no alteren la integridad de la base de datos. La recuperación de fallas tiene que ver con la capacidad de un sistema DBMS de recuperar la información que se haya perdido durante una falla en el software o en el hardware.

Independencia de los datos.

En las aplicaciones basadas en archivos, el programa de aplicación debe conocer tanto la organización de los datos como las técnicas que le permiten acceder a los datos. En los sistemas DBMS los programas de aplicación no necesitan conocer la organización de los datos en el disco duro. Este es totalmente independiente de ello.

Seguridad

La disponibilidad de los datos puede ser restringida a ciertos usuarios. Según los privilegios que posea cada usuario de la base de datos, podrá acceder a mayor información que otros.

Velocidad

Los sistemas DBMS modernos poseen altas velocidades de respuesta y proceso.

Independencia del hardware

La mayoría de los sistemas DBMS están disponibles para ser instalados en múltiples plataformas de hardware.

Los sistemas de bases de datos relacionales RDBMS (Relational Database Management System, por sus siglas en Inglés) tales como Oracle, MySQL, SQL Server, PostgreSQL, Informix, entre otros, le permiten ejecutar las tareas que se mencionan a continuación, de una forma entendible y razonablemente sencilla:

- Le permiten ingresar datos al sistema.
- Le permiten almacenar los datos.
- Le permiten recuperar los datos y trabajar con ellos.

- Le proveen herramientas para capturar, editar y manipular datos.
- Le permiten aplicar seguridad.
- Le permiten crear reportes e informes con los datos.

1.2.7.- Bases de datos relacionales.

Es un concepto lógico basado en una colección de objetos relacionados. En las bases de datos relacionales la información se organiza en tablas. Las tablas se organizan agrupando datos relacionados con el mismo tema y contienen columnas (atributos) y filas (tuplas) de información. Luego cuando las bases de datos son solicitadas, el motor de bases de datos las relaciona entre si. Las tablas son lo que se conoce como entidades en los libros de teoría de bases de datos.

1.2.7.1.- Enfoque relacional.

El modelo relacional es una forma de ver los datos, es decir, una receta para representar los datos mediante tablas y para manipular su representación. El modelo relacional como todo modelo de datos, tiene que ver con tres aspectos de los datos.

1.- Su Estructura.
2.- Su Integridad.
3.- Su Manipulación.

1.2.7.2.- Estructura del enfoque relacional.

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

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

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

A continuación mostramos la **Tabla 1.10** con la terminología de la estructura de datos relacional.

Término Relacional Formal	**Equivalente Informal**
Relación	Tabla
Tupla	Fila o registro
Cardinalidad	Número de filas o registros
Atributo	Columna o campo
Grado	Número de columnas o campos
Clave primaria	Identificador único
Dominio	Fondos de valores legales

Tabla 1.1.- Terminología de la estructura de datos relacional.

Las relaciones están compuestas por dos partes una cabecera y un cuerpo, se define la relación sobre un conjunto de dominios. La cabecera esta formada por un conjunto fijo de atributos tal que cada atributo A(j) corresponde a cada uno de los dominios de J, con J variando hasta N. El cuerpo esta formado por un conjunto de tuplas el cual varía con el tiempo, cada tupla esta formada por un conjunto de pares (atributo, valor) DONDE (i) varia de 1 hasta N y N es el numero de tuplas del conjunto(A1,V1), (A2,V2), (A3,V3)…… (An,Vn) N va ser el grado de la relación.

Por mencionar algunos de los diferentes tipos de relaciones que pueden existir en un sistema relacional tenemos:

Relaciones Base: Son relaciones reales que tienen nombre y forman parte directa de la base de datos almacenada (son autónomas).

Vistas: También denominadas relaciones virtuales, son relaciones con nombre y derivadas: se representan mediante su definición en términos de otras relaciones con nombre, no poseen datos almacenados propios.

Instantáneas: Son relaciones con nombre y derivadas. Pero a diferencia de las vistas, son reales, no son virtuales: están representadas no solo por su definición en términos de otras relaciones con nombre, sino también por sus propios datos almacenados. Son relaciones de sólo lectura y se refrescan periódicamente.

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

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

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

La **nulidad** de una columna se refiere en sí a que se requiere ó no de una entrada para esa columna. Si desea permitir que la columna indique que el valor es desconocido, se especifica NULL. Si se desea insistir que cada fila tenga una entrada en esa columna, se especifica NOT NULL.

1.2.7.3.- Jerarquía de generalización.

Una entidad E es una generalización de un grupo de entidades $E_1, E_2, \ldots, E_n$, si cada objeto de estas es también un objeto de la entidad E. Ejemplo: el tipo de entidad VEHÍCULO es una generalización del tipo de entidad BICICLETA, ya que todas las bicicletas son vehículos. El tipo de entidad PERSONA es una generalización de las entidades HOMBRE y MUJER. ***Se puede decir que estos son subconjuntos de la generalización (Es_un o Es_parte_de)***.[3]

1.2.8.- Reglas generales de integridad.

Reglas de integridad

Una vez definida la estructura de datos del modelo relacional, pasamos a estudiar las reglas de integridad que los datos almacenados en dicha estructura deben cumplir para garantizar que son correctos.

Al definir cada atributo sobre un dominio se impone una restricción sobre el conjunto de valores permitidos para cada atributo. A este tipo de restricciones se les denomina *restricciones de dominios*. Hay además dos reglas de integridad muy importantes que son restricciones que se deben cumplir en todas las bases de datos relacionales y en todos sus estados o instancias (las reglas se deben cumplir todo el tiempo). Estas reglas son la *regla de integridad de entidades* y la *regla de integridad referencial*. Antes de definirlas, es preciso conocer el concepto de *nulo*.

Nulos

Cuando en una tupla un atributo es desconocido, se dice que es *nulo*. Un nulo no representa el valor cero ni la cadena vacía, éstos son valores que tienen significado. El nulo

implica ausencia de información, bien porque al insertar la tupla se desconocía el valor del atributo, o bien porque para dicha tupla el atributo no tiene sentido.

Ya que los nulos no son valores, deben tratarse de modo diferente, lo que causa problemas de implementación. De hecho, no todos los SGBD relacionales soportan los nulos.

Regla de integridad de entidades

La primera regla de integridad se aplica a las claves primarias de las relaciones base: *ninguno de los atributos que componen la clave primaria puede ser nulo.*

Por definición, una clave primaria es un identificador irreducible que se utiliza para identificar de modo único las tuplas. Que es irreducible significa que ningún subconjunto de la clave primaria sirve para identificar las tuplas de modo único. Si se permite que parte de la clave primaria sea nula, se está diciendo que no todos sus atributos son necesarios para distinguir las tuplas, con lo que se contradice la irreducibilidad.

Nótese que esta regla sólo se aplica a las relaciones base y a las claves primarias, no a las claves alternativas.

Regla de integridad referencial

La segunda regla de integridad se aplica a las claves ajenas: *si en una relación hay alguna clave ajena, sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o bien, deben ser completamente nulos*.

La regla de integridad referencial se enmarca en términos de estados de la base de datos: indica lo que es un estado ilegal, pero no dice cómo puede evitarse. La cuestión es ¿qué hacer si estando en un estado legal, llega una petición para realizar una operación que conduce a un estado ilegal? Existen dos opciones: *rechazar* la operación, o bien *aceptar* la operación y realizar operaciones adicionales compensatorias que conduzcan a un estado legal.

Por lo tanto, para cada clave ajena de la base de datos habrá que contestar a tres preguntas:

- ***Regla de los nulos**:* ¿Tiene sentido que la clave ajena acepte nulos?
- ***Regla de borrado**:* ¿Qué ocurre si se intenta borrar la tupla referenciada por la clave ajena?
    - ***Restringir**:* no se permite borrar la tupla referenciada.
    - ***Propagar**:* se borra la tupla referenciada y se propaga el borrado a las tuplas que la referencian mediante la clave ajena.
    - ***Anular**:* se borra la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos).
- ***Regla de modificación**:* ¿Qué ocurre si se intenta modificar el valor de la clave primaria de la tupla referenciada por la clave ajena?

- ***Restringir:*** no se permite modificar el valor de la clave primaria de la tupla referenciada.
- ***Propagar:*** se modifica el valor de la clave primaria de la tupla referenciada y se propaga la modificación a las tuplas que la referencian mediante la clave ajena.
- ***Anular:*** se modifica la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos). [4]

1.2.9.- Algebra relacional.

Consiste en un conjunto de operadores de alto nivel que operan sobre relaciones, cada uno de estos operadores toma una o dos relaciones como entrada y produce una nueva relación como salida. Los operadores se dividen en dos grupos:

1.- El primero involucra las operaciones tradicionales de conjuntos tales como unión, intersección, producto cartesiano y diferencia.

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

Unión: Constituye una relación formada por todas las tuplas que aparecen en cualquiera de las dos relaciones involucradas.

Intersección: Constituye una relación formada por aquellas tuplas que aparezcan en las dos relaciones involucradas.

Diferencia: Constituye una relación formada por todas las tuplas de la primera relación que no aparezcan en la segunda relación.

Producto: A partir de 2 relaciones especificadas constituye una relación que contiene todas las tuplas combinadas (una de cada una de las relaciones).

Restricción: Extrae las tuplas especificadas en una relación dada, es decir restringe la relación a tuplas que satisfagan una condición.

Proyección: Extrae los atributos especificados de una relación dada, (salida ó lista).

Reunión: A partir de 2 relaciones especificadas constituye una relación que contiene todas las posibles combinaciones de tuplas, una de cada una de las relaciones tales que las dos tuplas participantes en una combinación dada satisfagan alguna condición especifica. Esta es muy parecida a la restricción pero depende de los elementos de intervienen.

1.2.10.- Normalización de una base de datos.

El proceso de **normalización de una base de datos** consiste en aplicar una serie de reglas a las relaciones obtenidas tras el paso del modelo E-R (entidad-relación) al modelo relacional.

Las bases de datos relacionales se normalizan para:

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

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

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

Las ventajas de la normalización son las siguientes:

- Evita anomalías en inserciones, modificaciones y borrados.
- Mejora la independencia de datos.
- No establece restricciones artificiales en la estructura de los datos.[5]

1.2.11.- Arquitectura Cliente-Servidor.

Esta arquitectura consiste básicamente en que un programa -el cliente- realiza peticiones a otro programa -el servidor- que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores Web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes **computadoras** aumentando así el grado de distribución del sistema. [11]

Para la realización de sistema utilizaremos la arquitectura Cliente-Servidor debido a las siguientes razones: Por un lado están los clientes y administradores que visualizan el sistema a través de un navegador Internet, con total independencia del sistema operativo que posean y por el otro lado, se localiza el *servidor Web*, en el cual está almacenado el sistema de registro de reportes y la base de datos que contiene la información de los reportes.

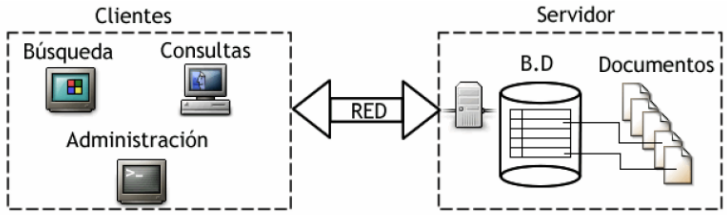


Figura 1.4.- Arquitectura Cliente-Servidor.

1.2.11.1.-Características de un cliente.

En la arquitectura C/S el **remitente de una solicitud** es conocido como cliente. Sus características son:

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

1.2.11.2.-Características de un servidor

En los sistemas C/S el **receptor de la solicitud** enviada por cliente se conoce como servidor. Sus características son:

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

CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA

Introducción

A continuación estableceremos el planteamiento del problema, analizaremos la especificación de los requerimientos proporcionados, elaboraremos los casos de uso detectados, identificaremos las entidades con sus atributos correspondientes, para hacer el diagrama (Entidad-Relación), Modelo Relacional y Diseño Físico de la Base de Datos, aplicaremos las etapas de Normalización de la Base de datos y diseñaremos la interfaz del sistema.

2.1. PLANTEAMIENTO DEL PROBLEMA.

La Dirección de Protección Universitaria almacena todos sus reportes de las incidencias que ocurren cotidianamente en hojas de papel, estas son llenadas por el personal encargado de vigilar todas las zonas de ciudad universitaria, al finalizar el día los reportes son entregados a la Jefatura de Seguridad, el personal de la dependencia lee los reportes, los captura en un archivo de Excel y posteriormente las hojas son guardadas en archiveros. Esto ocasiona que se tenga una información desordenada, que sea difícil extraer datos necesarios concisos de algún reporte determinado además de que tendrán montones de papel conforme transcurra el tiempo.

Las Personas encargadas de llevar el control hacen la captura de la información en un libro de excel asignándole a cada hoja un tipo de incidencia, esto tiene limitantes, cuando desean realizar tareas como generar informes, consultar reportes, hacer búsquedas, extraer datos concretos, elaboración de gráficos que reflejen en comportamiento de la información etc.

La dependencia sólo lleva el control de los reportes de incidencias por número en una hoja de excel, es decir solamente tienen el numero de reportes de incidencias ocurridas cada mes, pero no pueden consultar su descripción, si quisieran consultar su descripción tendrían que buscar en los montones de papel acumulados durante el mes ó cuatrimestre según la fecha del reporte.

Ponemos la tarea de elaborar un informe mensual que son entregados a la secretaria administrativa el cual consiste en desglosar la descripción de la incidencia y la cantidad de veces que ocurrió en un determinado periodo de tiempo.

La solución al problema es crear un sistema que automatice el proceso de almacenamiento de los reportes de incidencias en la dirección de protección universitaria, para ello registraremos todas los reportes de incidencias ocurridas en una base de datos para después generar los informes y hacer las consultas necesarias.

Para resolver este problema nos apoyaremos en conceptos y estrategias que actualmente son aplicados en Bases de Datos, la base de datos a diseñar debe almacenar información clasificada acerca de los diferentes reportes de incidencias ocurridas.

2.1.1.- Especificación de requerimientos.

De manera específica se necesita un sistema que sea capaz de automatizar el proceso de almacenamiento de los reportes para agilizar los informes emitidos por la dependencia así como su fácil consulta. Debemos tomar en cuenta que el conteo de los reportes se hacia de forma manual y no se tenia un informe actualizado con el numero de incidencias ocurridas cotidianamente.

Los reportes tienen datos generales y datos descriptivos de la incidencia.

Los datos similares que contendrán todos los reportes se denominan datos generales y son los siguientes:

Datos Generales: (clave, fecha del reporte, fecha del sistema, hora del reporte, turno, sector, clave de la incidencia, dependencia, edificio, estacionamiento, vialidad), los datos generales serán llenados obligatoriamente en todos los reportes.

Los datos diferentes que tendrán los reportes de incidencias se denominan **Datos Descriptivos**, cuando se registra un reporte en la base de datos llenaremos los datos generales y los datos descriptivos.

Dependiendo el número de incidencia seleccionado el sistema mostrara el formulario correspondiente para registrar el reporte.

Los reportes de incidencia que podremos registrar son los siguientes: Amenaza de bomba, agresiones, emergencias médicas, bicicletas inseguras, conatos de incendio, derrames de sustancias peligrosas, fugas de gas, ingresos no autorizados, intoxicados, juegos prohibidos, objetos perdidos, patrimonio afectado, percances vehiculares, sismos y vehículos expuestos, a estas incidencias se sumaran las que posteriormente se deseen clasificar.

Para manipular la información de la base de datos se requiere de operaciones básicas como: altas de reportes, borrar reportes, consultar reportes, imprimir reportes. En las consultas se pueden extraer todos los registros de los reportes, además el sistema debe tener un espacio donde el administrador del sistema pueda realizar tareas como agregar usuarios, editar usuarios, agregar dependencias, editar dependencias entre otras.

De los usuarios debemos saber su nombre, su correo, de que dependencia es, además de su nombre de acceso (nick) y password.

De las dependencias tendremos el nombre, dirección, teléfono y director.

En la tabla incidencias contendremos el nombre de la incidencia, su clave y una descripción de la incidencia.

El sistema generara los siguientes reportes y consultas:

- Una hoja de informe que contendrá una lista de incidencias indicando cuantas ocurrieron cada mes y generando un total durante el año.

- Un informe descriptivo, en este debemos indicar al sistema el número de la incidencia de la cual esperamos el informe descriptivo.

- El usuario podrá consultar los reportes registrados:
  - Por Fecha
  - Por Periodo
  - Por Incidencia
  - Por Dependencia

2.1.2.- Definición conceptual del sistema.

El sistema mostrara una pantalla de autentificación de usuarios, una ves autentificados los datos de los usuarios podrán capturar reportes de las diferentes incidencias ocurridas.

El sistema mostrara un menú de opciones el cual contiene lo siguiente:(registrar reportes, consultar reportes, informe general, informe descriptivo y una sección en donde el administrador solo tendrá acceso).

El sistema llevara un registro de todos los reportes de las incidencias ocurridas.

Veremos a continuación las funciones más importantes del sistema**:**

- Agregar Usuarios
- Editar Usuarios
- Eliminar Usuarios
- Agregar Dependencias
- Editar Dependencias
- Eliminar Dependencias
- Registrar Reportes.
- Consultar Reportes.
- Eliminar Reportes.
- Imprimir Reportes.
- Obtener Hoja de Informe General e Informe Descriptivo.
- Obtener Datos de una Incidencia en específico.

La recopilación de la información para el desarrollo de sistema se hizo a través de entrevista con personal de la Dirección de Protección Universitaria, ya que ellos solicitaban un sistema que cubriera sus necesidades, les explique que sí era posible realizar la automatización de su proceso de registro de reportes y obtendrían algunos beneficios como son el orden de la información, información actualizada disponible para cualquier consulta además de ahorro tiempo.

2.1.2.1.- Casos de Uso. (Descripción de procesos)

Los casos de uso están diseñados para cumplir los deseos del usuario cuando utiliza el sistema.

De manera precisa, un caso de uso es una descripción de un conjunto de secuencias de acciones, incluyendo variaciones, que un sistema lleva acabo y que conduce a un resultado observable de interés para un actor determinado. [6]

A continuación se describen los actores que intervienen en el sistema y los casos de uso correspondientes.

Identificación de Actores:

En la **Figura 2.2** mostramos la primera aproximación donde identificamos a los actores que interactúan con el sistema:

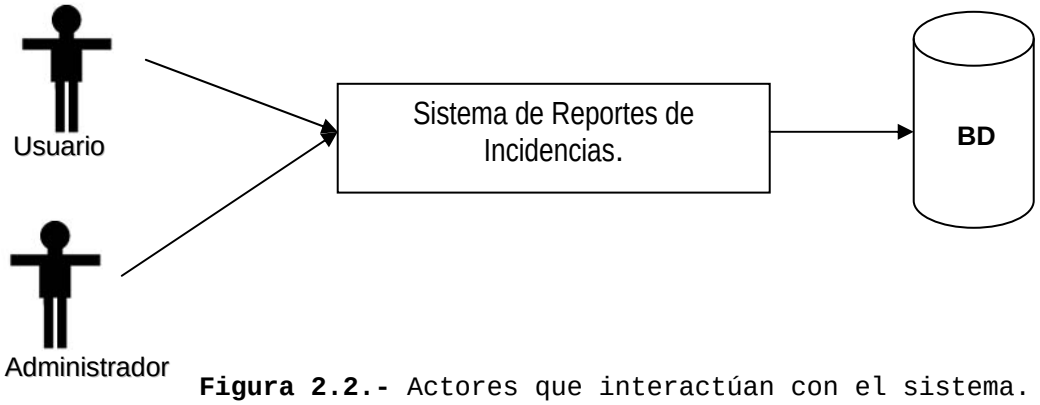


Figura 2.2.- Actores que interactúan con el sistema.

Los actores involucrados en el sistema en el contexto de la función y el comportamiento son los siguientes: usuario, administrador y el Sistema de Base de Datos. Cuyas actividades de cada uno de los actores dentro del sistema se describen a continuación.

Usuarios:
Registrar Reportes.
Consultar Reportes.
Imprimir Reportes.
Imprimir Informes

Administrador:
Gestión de Usuarios.
Gestión de Dependencias
Gestión de Hojas de Informes.
Registrar Reportes.

Consultar Reportes.
Imprimir Reportes.
Imprimir Hojas de Informe.

Sistema de Base de Datos.

Genera Informe General de todos los reportes de incidencias ocurridas.
Generar Informe Descriptivo de la incidencia seleccionada por el usuario.

Identificación de casos de uso del sistema.

Basándose en la secuencia de tareas de los escenarios de uso se crearon los casos de uso de manera que se pueda tener una idea clara de que es lo que se quiere funcionalmente del sistema y de la forma en la que se realizan los procesos. A continuación de presentan los casos de uso del sistema:

Caso de Uso:	**Registrar Reporte.**
Actores:	Usuario ó Administrador.
Descripción:	El Usuario ó Administrador puede llenar el formulario de un reporte en el sistema para posteriormente guardarlo en la base de datos.

Caso de Uso:	**Consultar Reporte.**
Actores:	Usuario ó Administrador.
Descripción:	El Usuario ó Administrador realiza una consulta de los reportes de incidencias que desea, eligiendo una de las opciones de búsqueda.

Caso de Uso:	**Eliminar Reporte.**
Actores:	Usuario ó Administrador.
Descripción:	El Usuario ó Administrador puede borrar un reporte seleccionado almacenado en el sistema.

Caso de Uso:	**Imprimir Reporte.**
Actores:	Usuario ó Administrador.
Descripción:	El Usuario ó Administrador puede imprimir un reporte seleccionado.

Caso de Uso:	**Imprimir Informe.**
Actores:	Usuario ó Administrador.
Descripción:	El Usuario ó Administrador pueden imprimir los informes que genera el sistema.

Caso de Uso:	**Agregar Usuarios.**
Actores:	Administrador.
Descripción:	El administrador deberá proporcionar su clave, para poder agregar un nuevo usuario y este interactué con el sistema.

Caso de Uso:	**Autentificar Usuario.**
Actores:	Sistema de Base de datos.
Descripción:	El Usuario o administrador proporcionan al sistema datos de acceso para ser validados, si son correctos los datos proporcionados, el sistema les permitirá el acceso.

Caso de Uso:	**Generar Informe General.**
Actores:	Sistema de Base de Datos.
Descripción:	El Sistema de Base de Datos Genera una tabla con el numero reportes de incidencias ocurridas mensualmente hasta el momento.

Caso de Uso:	**Generar Informe Descriptivo.**
Actores:	Sistema de Base de Datos.
Descripción:	El Sistema de Base de Datos Genera una tabla mostrando las dependencias y el numero de ocurrencias según la incidencia seleccionada por el usuario.

2.1.2.2.-Diagrama de caso de uso del sistema.

En el diagrama se muestra como una segunda aproximación el sistema de forma general, en este diagrama **Figura 2.3,** se presentan los actores que van a interactuar directamente con el sistema y sus casos de usos sobre los cuales interactúa:

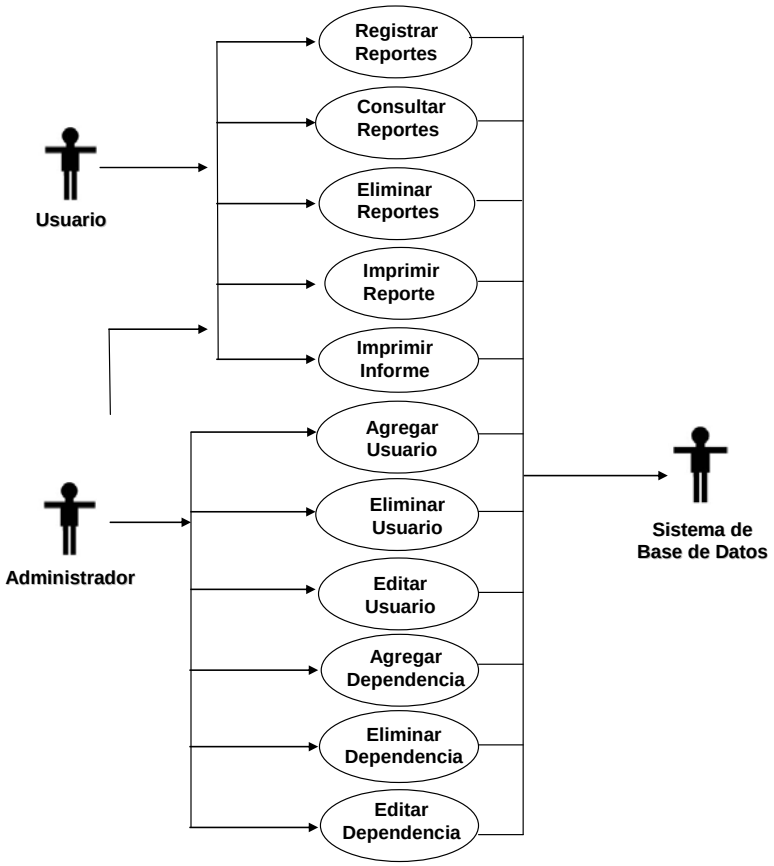


Figura 2.3.- Diagrama general de casos de uso del sistema.

2.1.2.3.- Flujo de eventos.

El propósito del flujo de eventos es documentar el flujo lógico que seguirán los casos de uso. En el flujo de eventos se describe a detalle lo que realizan los usuarios del sistema y como responde éste a las solicitudes que le son requeridas.
El flujo de eventos incluye:

Descripción. Se refiere a un resumen concreto de lo que realiza el caso de uso.

Actores. Son aquellos que interactúan con el sistema.

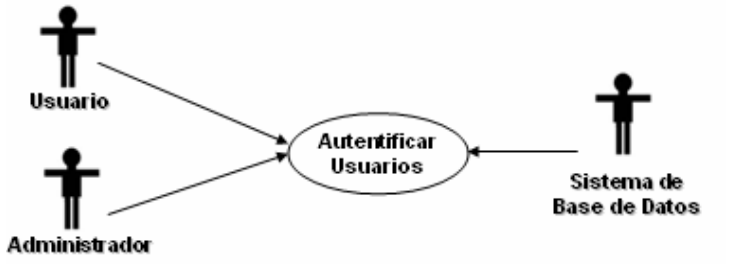
Precondiciones. Son los hechos que se han de cumplir para que el flujo de evento se pueda llevar a cabo.

Flujo normal. Describe paso a paso la funcionalidad que ejecuta cada caso de uso, situándose en las condiciones del escenario ideal.

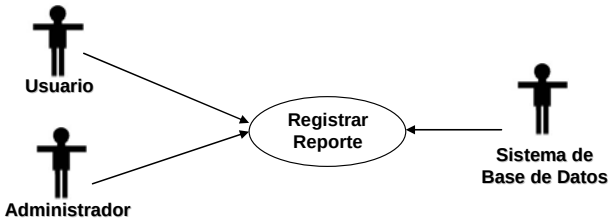
Flujo alternativo. Son los que nos permiten indicar qué es lo que hace el sistema en los casos menos frecuentes e inesperados.

Poscondiciones. Son los hechos que se ha de cumplir si el flujo de eventos normal se ha ejecutado correctamente.

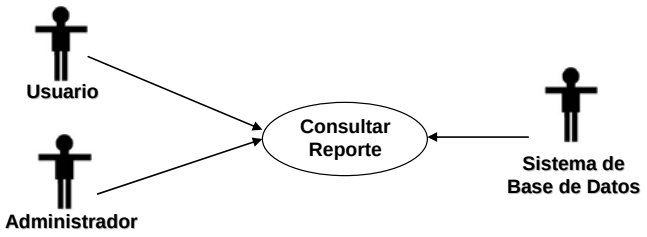
A continuación se presenta el flujo de eventos para cada caso de uso del sistema, representados en la fig. `1(c)` (Diagrama de casos de uso). Se representan primeramente de manera gráfica los actores y casos de uso que intervienen y a continuación el flujo correspondiente.

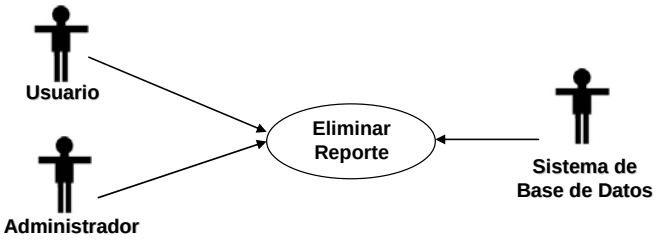
CASO DE USO: Autentificar Usuarios	
	
Descripción:	Permite validar el acceso a los usuarios y al administrador a través del sistema de Bases de Datos.
Actores:	Usuarios Registrados en el Sistema y administrador del Sistema.

Precondiciones:	El usuario deberá estar registrado en el sistema, en el caso del administrador deberá contar con claves de acceso.
Flujo Normal:	1.- El sistema solicita nombre de usuario y contraseña del actor. (Usuario ó administrador). 2.-El actor proporciona su nombre de usuario y clave. 3.-El sistema valida con el sistema de base de datos ó con el propio sistema los datos del usuario ó administrador según sea el caso. 4.- El sistema despliega una página de bienvenida al actor.
Flujo Alternativo:	3.- El sistema comprueba la validez de los datos, si los datos no son correctos, envía un mensaje al actor permitiéndole proporcionarlos nuevamente.
Poscondiciones:	El actor ha ingresado correctamente al Sistema.

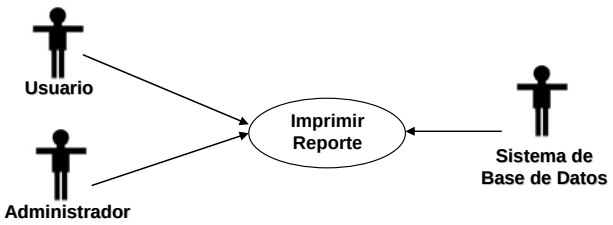
CASO DE USO: Registrar Reportes	
	
Descripción:	Permite a los actores registrar reportes de Incidencias en el sistema.
Actores:	Usuarios Registrados en el Sistema y administrador del Sistema.
Precondiciones:	Los actores deberán ser autentificados entes de poder registrar incidencias.
Flujo Normal:	1.- El actor pulsa el vinculo Registrar Reportes en el Menú que esta en la página de bienvenida. 2.-El Sistema despliega un menú de todas las incidencias que podemos registrar. 3.-El actor debe elegir que incidencia quiere registrar y pulsar el vínculo. 4.- El Sistema mostrara el formulario que tendrá que llenar el actor para guardar la incidencia. 5.- El actor debe llenar los campos requeridos para guardar la incidencia después pulsar el botón guardar. 6.- El sistema informara al actor que los datos han sido guardados correctamente. 7.- El registro ha sido guardado correctamente.
Flujo Alternativo:	6.- El sistema verificara que los datos obligatorios estén llenos, de no ser así el sistema te indicara cuales datos tienes que llenar para

	poder guardar el registro. 7.- El registro ha sido guardado correctamente.
Poscondiciones:	El registro ha sido almacenado en el Sistema.

CASO DE USO: Consultar Reporte.	
	
Descripción:	Permite a los actores consultar las incidencias registradas en el sistema.
Actores:	Usuarios registrados en el sistema y administrador.
Precondiciones:	Los actores deberán ser autentificados para poder consultar incidencias. El sistema debe tener almacenadas incidencias ocurridas.
Flujo Normal:	1.- El actor pulsa el vinculo consultar incidencias en el menú que esta en la página de bienvenida. 2.-El Sistema despliega las diferentes maneras de cómo puede consultar el actor. 3.-El actor debe elegir de que manera quiere hacer la consulta de la incidencia y pulsar el vinculo buscar. 4.- El Sistema mostrara la lista de las incidencias que cumplan con la condición de búsqueda del actor. 5.- El actor podrá pulsar sobre la incidencia deseada para ver toda su descripción.
Flujo Alternativo:	4.- El sistema desplegara vació en caso de no tener incidencias que cumplan con la condición de búsqueda.
Poscondiciones:	El actor podrá eliminar ó imprimir las incidencias consultadas.

Caso de uso: Eliminar Reporte.	
	
Descripción:	Permite a los a los actores Eliminar registros del Sistema.
Actores:	Usuarios Registrados en el Sistema y administrador del Sistema.
Precondiciones:	Los actores deberán ser autentificados para poder Eliminar registros incidencias. El sistema debe tener almacenadas registros de incidencias ocurridas.
Flujo Normal:	1.- El actor pulsa el vinculo Consultar Incidencias en el Menú que esta en la página de bienvenida. 2.-El Sistema despliega las diferentes maneras de cómo puede consultar el actor. 3.-El actor debe elegir de que manera quiere hacer la consulta de la incidencia y pulsar el vinculo buscar. 4.- El Sistema mostrara la lista de las incidencias que cumplan con la condición de búsqueda del actor. 5.- El actor podrá pulsar sobre la incidencia deseada para ver toda su descripción. 6.- El Sistema mostrara toda la información que contenga el registro seleccionado. 7.- El Sistema tendrá mostrara las opciones de Eliminar e Imprimir. 8.- El actor pulsara las opción Eliminar. 9.- El sistema preguntara al actor: (¿Estas seguro de Eliminar el Registro?). 10.- El actor tomara la decisión de eliminar ó no el registro.

Flujo Alternativo:	4.- El sistema desplegara vació en caso de no tener incidencias que cumplan con la condición de búsqueda.
Poscondiciones:	El registro ya no estará almacenado en el sistema.

Caso de uso: Imprimir Reporte.	
	
Descripción:	Permite a los a los actores Imprimir reportes almacenados en la base de datos.
Actores:	Usuarios Registrados en el Sistema y administrador del Sistema.
Precondiciones:	Los actores deberán ser autentificados para poder Imprimir reportes de incidencias. El sistema debe tener almacenadas registros de los reportes de las incidencias ocurridas.
Flujo Normal:	1.- El actor pulsa el vinculo Consultas en el Menú que esta en la página de bienvenida. 2.-El Sistema despliega las diferentes maneras de cómo puede consultar el actor. 3.-El actor debe elegir de que manera quiere hacer la consulta de la incidencia y pulsar el vinculo buscar. 4.- El Sistema mostrara la lista de las incidencias que cumplan con la condición de búsqueda del actor. 5.- El actor podrá pulsar sobre la incidencia deseada para ver toda su descripción. 6.- El Sistema mostrara toda la información que contenga el registro seleccionado. 7.- El Sistema tendrá mostrara las opciones de Eliminar e Imprimir. 8.- El actor pulsara las opción Imprimir.

	9.- El sistema mostrara una ventana para elegir la impresora y poder imprimir. 10.- El actor configurara las opciones de impresión y dará clic en imprimir.
Flujo Alternativo:	4.- El sistema desplegara vació en caso de no tener incidencias que cumplan con la condición de búsqueda.
Poscondiciones:	El actor contara con la información impresa del reporte seleccionado.

2.2. -DISEÑO DEL SISTEMA.

Esta etapa consta de tres fases: diseño conceptual, diseño lógico y diseño físico de la base de datos.

Los objetivos del diseño de la base de datos son:

- Representar los datos que requieren las principales áreas de aplicación y los grupos de usuarios, y representar las relaciones entre dichos datos.
- Proporcionar un modelo de datos que soporte las transacciones que se vayan a realizar sobre los datos.
- Especificar un esquema que alcance las prestaciones requeridas para el sistema.

2.2.1.- Diseño conceptual.

En esta etapa se debe construir un esquema de la información que se usará, 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: encuentran entidades, atributos y relaciones. [7] El objetivo es comprender:

Las tareas a realizar en el diseño conceptual son las siguientes:

- Identificar las entidades.
- Identificar las relaciones.
- Identificar los atributos y asociarlos a entidades y relaciones.
- Determinar los dominios de los atributos.
- Determinar los identificadores.
- Determinar las jerarquías de generalización (si las hay).
- Dibujar el diagrama entidad-relación.
- Revisar el esquema conceptual local con el usuario.

Identificación de Entidades

Entidades:

- Universidad
- Dependencia
- Incidencias
- Usuarios
- Reportes
- Involucrados

Una vez analizado en problema identificaremos las entidades y sus atributos, se procedemos a crear el diagrama entidad-relación correspondiente.

2.2.1.1.- Diagrama entidad relación.

El **modelado entidad-relación** **Figura 2.4** es una *técnica* para el modelado de datos utilizando diagramas entidad relación. No es la única técnica pero sí la más utilizada. Observemos en la parte inferior se tiene una jerarquía de generalización, la entidad **REPORTE** es la entidad padre contendrá los datos generales de los reportes y las subentidades hijos son: Incidencia01, Incidencia02, incidencia 03, ya que contendrán las descripciones de las incidencias ocurridas.

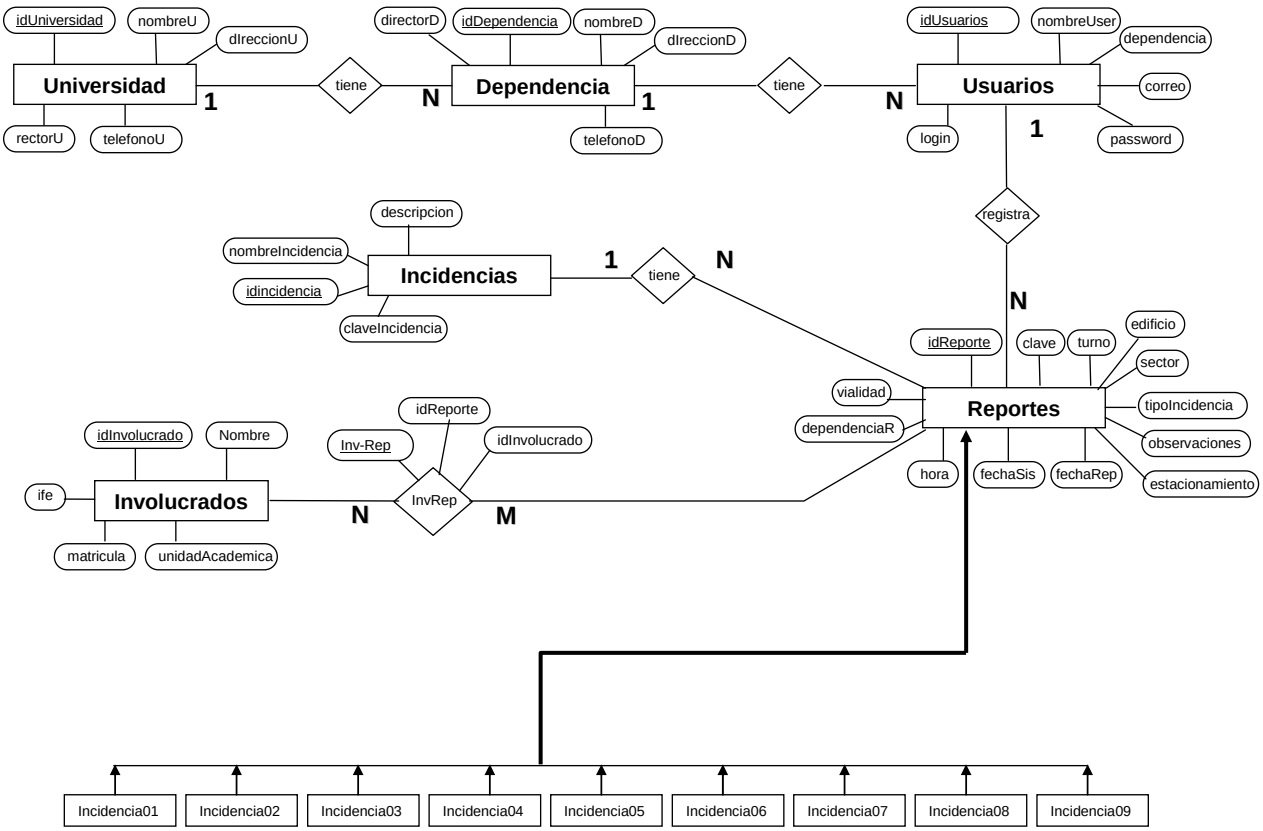


Figura 2.4.- Diagrama **Entidad - Relación.**

2.2.2.- Diseño lógico.

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

En esta etapa, se transforma el esquema conceptual en 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.

Tanto el diseño conceptual, como el diseño lógico, son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. Ambos se deben ver como un proceso de aprendizaje en el que el diseñador va comprendiendo el funcionamiento de la empresa y el significado de los datos que maneja. 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, sino imposible, definir todas las vistas de usuario (esquemas externos), o mantener la integridad de la base de datos. También puede ser difícil definir la implementación física o el mantener unas prestaciones aceptables del sistema. Además, hay que tener en cuenta que la capacidad de ajustarse a futuros cambios es un sello que identifica a los buenos diseños de bases de datos. Por todo esto, es fundamental dedicar el tiempo y las energías necesarias para producir el mejor esquema que sea posible.

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

1. Toda entidad se convierte en una tabla.

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

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

2.2.2.1.- Modelo relacional.

La estructura fundamental del modelo relacional es precisamente esa, "relación", es decir una tabla bidimensional constituida por líneas (tuplas) y columnas (atributos). Las relaciones representan las entidades que se consideran interesantes en la base de datos. Cada instancia de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representarán las propiedades de la entidad. Por ejemplo, si en la base de datos se tienen que representar usuarios, se podrá definir una relación llamada "Usuarios", cuyos atributos describen las características de los Usuarios (tabla siguiente). Cada tupla de la relación "Usuarios" representará un usuario en concreto, en la **Figura 2.5** mostramos el modelo relacional de nuestro sistema.

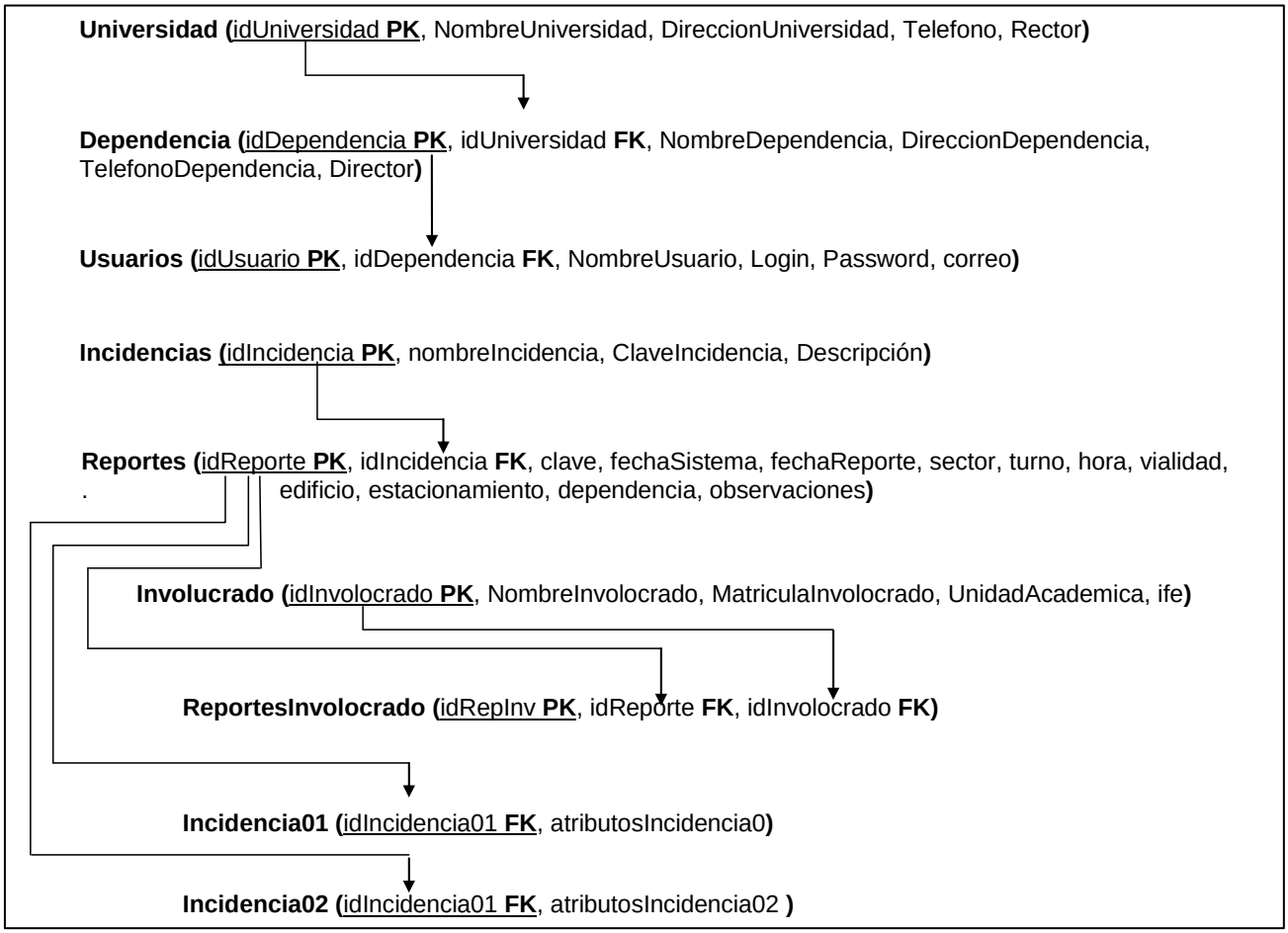


Figura 2.5.- Modelo Relacional.

2.2.2.2.-Normalización de la BD.

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

Primera forma normal. (1FN):

Una relación está en primera forma normal si, y sólo si, todo los dominios de la misma contienen valores atómicos, es decir, no hay grupos repetidos.

Como cada una de las relaciones maneja valores atómicos, ya se encuentra en primera forma normal, lo cual se ejemplifica con tabla Dependencia.

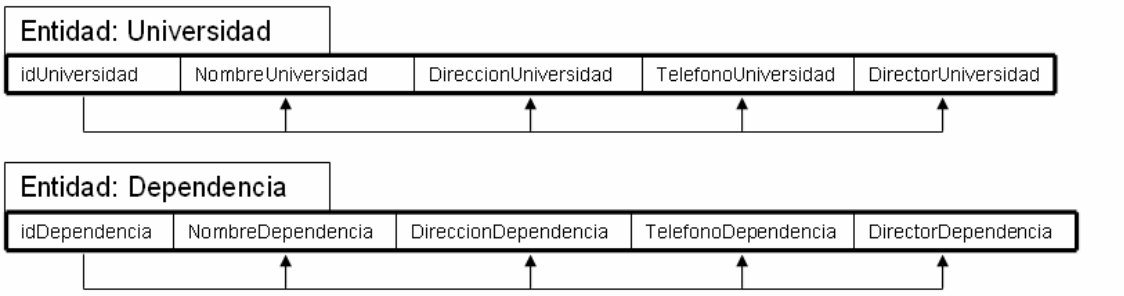
Entidad: Dependencia

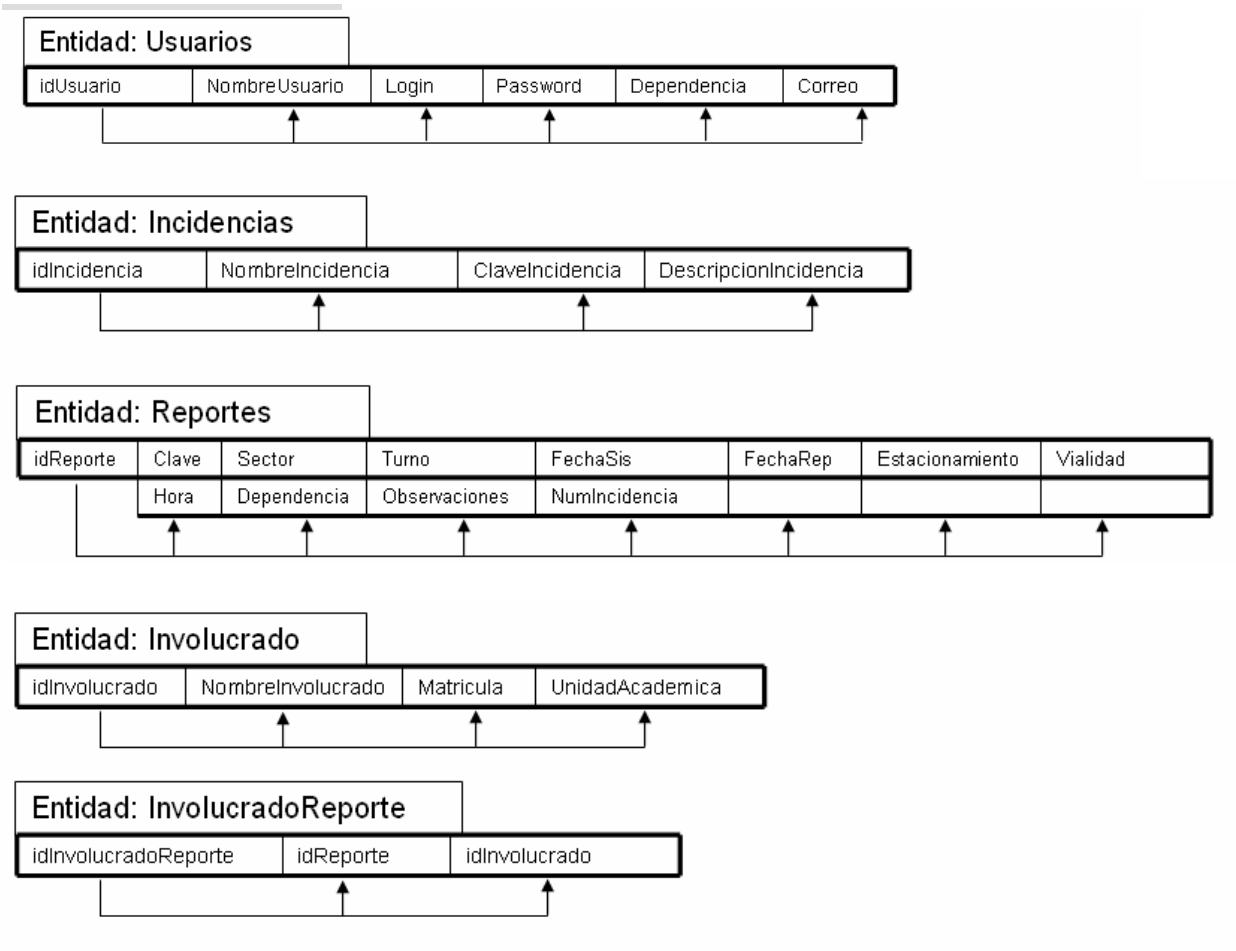
idDependencia	NombreDependencia	DireccionDependencia	TelefonoDependencia	DirectorDependencia
01	Dirección de Protección Universitaria	Boulevard Valsequillo, Ciudad Universitaria	2295500 Ext. 7997	Ing. Octavio Efrén Vázquez López.
02	Vicerrectoría de Docencia	Av.Juan de Palafox y Mendoza No. 212 Altos	2295500 Ext. 5568	Mtro. Jose Jaime Vázquez López.

Segunda forma normal. (2FN):

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.

En todas y cada una de las tablas se cumple esta regla, ya que todos los atributos son totalmente dependientes funcionalmente de la llave primaria como se muestra a continuación:





Tercera forma normal. (3FN):

Una relación esta en tercera forma normal si, y sólo si, está en segunda forma normal y, además no existen dependencias transitivas entre los atributos, se refiere a dependencias transitivas cuando existe más de una forma de llegar a referenciar a un atributo de una relación.

Como se pudo apreciar anteriormente todos y cada uno de los atributos son dependientes solo de la llave primaria, por tanto ya se encuentra en tercera forma normal.

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 obtenido en la fase anterior. Concretamente, en el modelo relacional, esto consiste en:

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

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

En la **Figura 2.6** se muestra la construcción física de la base de datos, observemos las diferentes entidades con sus atributos y claves primarias.

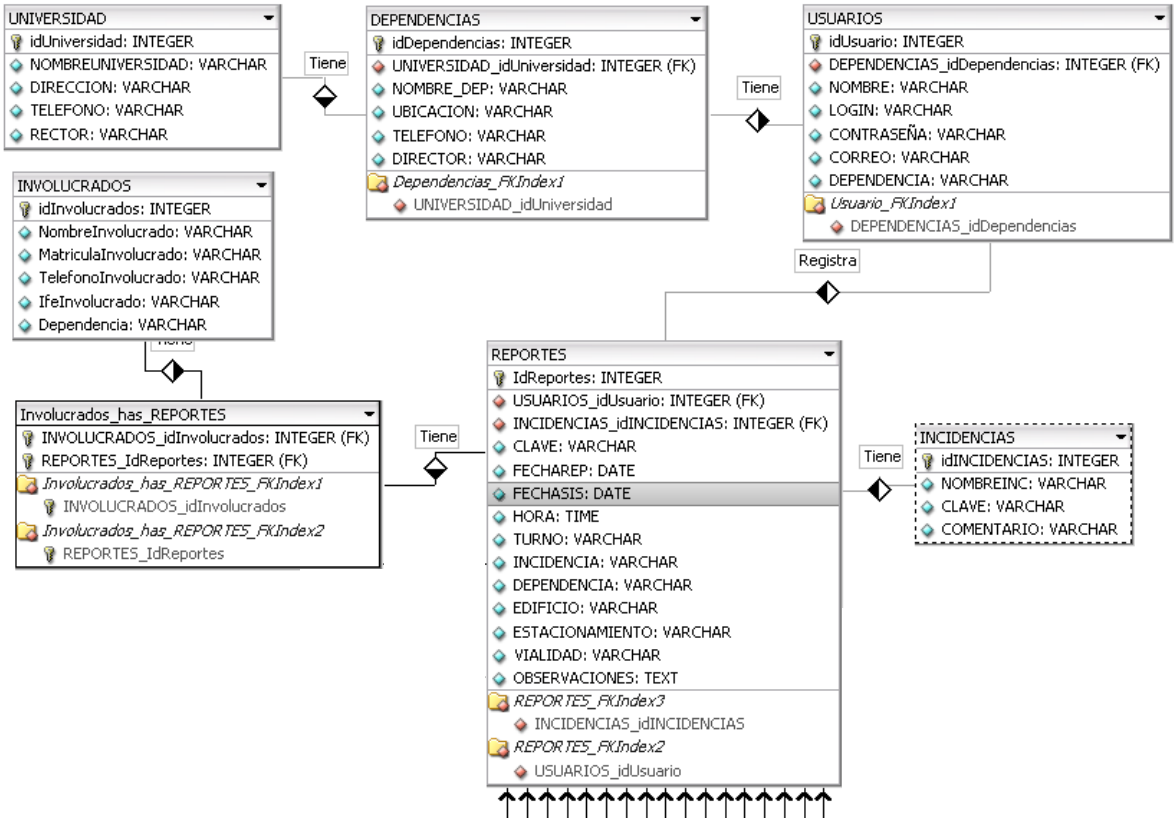


Figura 2.6.- Diseño físico de la Base de Datos del Sistema.

El diseño de la base de datos contiene una jerarquía de generalización, la tabla padre es **REPORTES** y las tablas hijos son las que contienen los datos descriptivos de los reportes de incidencias, las cuales presentamos a continuación en la **Figura 2.7**.

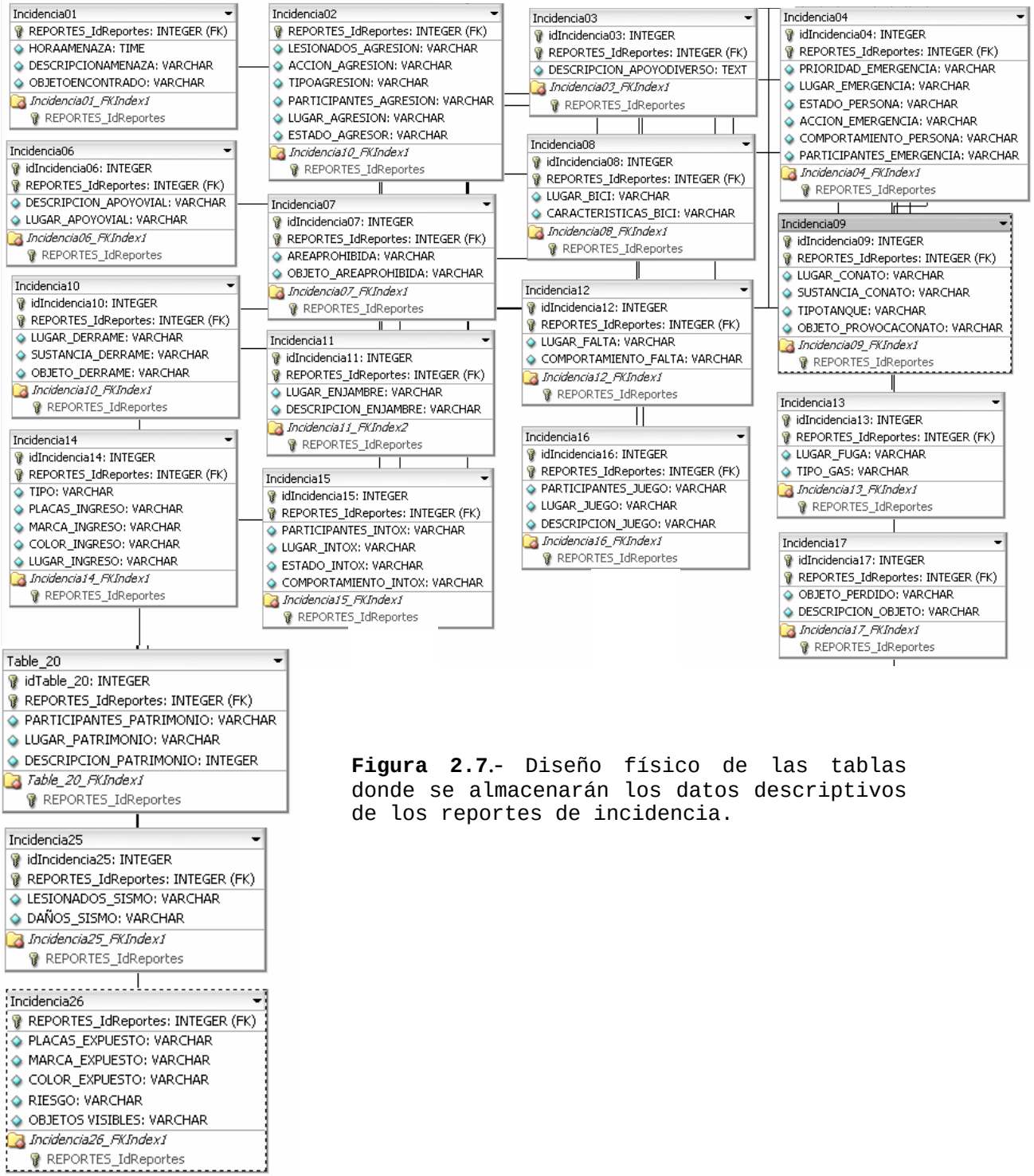


Figura 2.7.- Diseño físico de las tablas donde se almacenarán los datos descriptivos de los reportes de incidencia.

2.2.4.- Diccionario de datos.

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

Estos diccionarios se desarrollan durante el análisis de flujo de datos y ayuda a los analistas que participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño del proyecto.

2.2.4.1. Identificación de entidades.

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

Entidad	Descripción
Universidad	Contiene la información de las universidades.
Dependencia	Contiene información de las dependencias que hay en las universidades.
Reportes	Contiene los datos generales de los reportes de las incidencias.
Usuarios	Contiene información de los usuarios registrados en el sistema.
Incidencias	Contiene la información acerca de las incidencias que existen en el sistema de base de datos.
Involucrados	Contiene información de los involucrados en el reporte.

2.2.4.2. Identificación de relaciones.

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

Relaciones	Relación	Cardinalidad
Universidad-Dependencia	Tiene	Uno a Muchos
Dependencia - Usuarios	Tiene	Uno a Muchos
Incidencia-Reportes	Tiene	Uno a Muchos
Dependencia –Usuarios	Tiene	Uno a Muchos
Reporte-Involucrado	Tiene	Muchos a Muchos

2.2.4.3. Identificación de atributos y dominios.

De igual forma que con las entidades, se buscan nombres en el análisis de requerimientos. Son atributos los nombres que identifican propiedades, cualidades, identificadores o características de entidades o relaciones. El dominio de un atributo es el conjunto de valores que puede tomar el atributo. A continuación se enlistan los atributos y dominios para cada una de las entidades:

Entidad: UNIVERSIDAD

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
Id_universidad	Identificador único de Universidad	Int(5)	0…..9	No
Nombre_Uni	Especificamos el nombre la Universidad	Varchar(40)	A……Z	No
Direccion_Uni	Especificamos Dirección de la Universidad	Varchar(60)	A……Z	No
Telefono_Uni	Especificamos Teléfono de la Universidad	Varchar(25)	A……Z	No
Director_Uni	Especificamos el nombre del director de la Universidad	Varchar(30)	A……Z	No

Entidad: DEPENDENCIA

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
Id_dependencia	Identificador único de la Dependencia	Int(5)	0…..9	No
Nombre_dep	Especificamos el nombre la Dependencia	Varchar(40)	A……Z	No
Direccion_dep	Especificamos Dirección de la Dependencia	Varchar(60)	A……Z	No
Telefono_dep	Especificamos Teléfono de la Dependencia	Varchar(25)	A……Z	No
Director_dep	Especificamos el nombre del director de la Dependencia	Varchar(30)	A……Z	No

Entidad: USUARIOS

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
Id_usuario	Identificador único del usuario	Int(5)	0…..9	No
Nombre_Usuario	Especificamos el nombre del usuario.	Varchar(40)	A……Z	No
Dependencia_Usuario	Especificamos la dependencia a la que pertenece el usuario.	Varchar(60)	A……Z	No
Login_Usuario	Especificamos el Login del usuario.	Varchar(30)	A……Z	No
Password Usuario	Especificamos el password del usuario.	Varchar(30)	A…Z y/o 0…9	No
Correo_usuario	Especificamos el correo del usuario	Varchar(30)	A…Z y/o 0…9	Si

Entidad: INCIDENCIAS

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
Id_incidencia	Identificador único de la incidencia.	Int(5)	0…..9	No
Nombre Incidencia	Especificamos el nombre de la incidencia.	Varchar(40)	A……Z	No
Clave_incidencia	Especificamos la clave de la incidencia.	Varchar(10)	A…Z y/o 0…9	No
Descripción_incidencia	Especificamos una descripción de la incidencia.	Text()	A…Z y/o 0…9	No

Entidad: INVOLUCRADO

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
Id_Involucrado	Identificador único del usuario	Int(5)	0…9	No
Nombre_Involucrado	Escribimos el nombre del involucrado en la incidencia.	Varchar(30)	A…Z	Si
Matricula_Involucrado	Escribimos la matricula del involucrado en la incidencia.	Varchar(12)	A…Z y/o 0…9	Si
Ife_Involucrado	Escribimos el número de IFE del involucrado en la incidencia.	Varchar(20)	A…Z y/o 0…9	Si
UnidadAcademica_Involucrado	Escribimos la unidad académica a la cual pertenece el involucrado en la incidencia.	Varchar(60)	A…Z	Si

Entidad: REPORTES

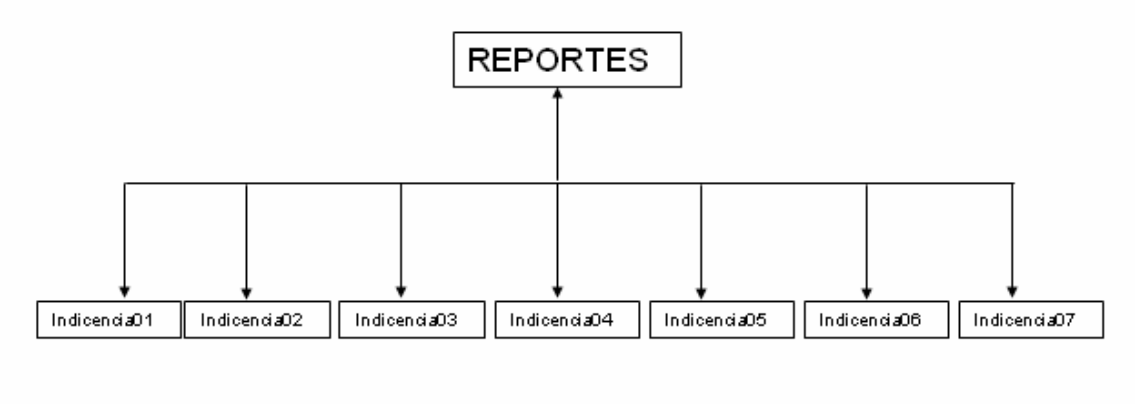
ATRIBUTO	DESCRIPCION	TIPO/LONGITUD	DOMINIO	NULO
Id_reporte	Identificador único del usuario	Int(5)	0…9	No
Clave_reporte	Especificamos la clave del reporte.	Varchar(40)	A…Z y/o 0…9	No
Fecha_reportes	Especificamos la fecha en que ocurrió el reporte.	Varchar(60)	DATETIME	No
Fehc_sistema	Especificamos la fecha de la captura del reporte.	Varchar(25)	DATETIME	No
Sector_reporte	Especificamos el sector del reporte.	Varchar(30)	A…Z y/o 0…9	No
Hora_reporte	Especificamos la hora del reporte.	Varchar(30)	TIME	No
Turno_reporte	Especificamos el turno en el cual ocurrió la incidencia.	Varchar(30)	A…Z y/o 0…9	No
Dependencia_reporte	Escribimos la dependencia en donde ocurrió la incidencia.	Varchar(60)	A…Z	No
Clave_incidencia	Escribimos la clave de la Incidencia.	Varchar(10)	A…Z y/o 0…9	No
Estacionamiento_reporte	Escribimos en que estacionamiento ocurrió la incidencia.	Varchar(10)	A…Z y/o 0…9	Si
Edificio_reporte	Escribimos en que edificio ocurrió la incidencia.	Varchar(10)	A…Z y/o 0…9	Si
Vialidad_reporte	Escribimos en que vialidad ocurrió la incidencia.	Varchar(20)	A…Z y/o 0…9	Si
Observaciones_reporte	Escribimos las observaciones de la incidencia.	Text	A…Z y/o 0…9	Si

Entidad: INVOLUCRADO_REPORTE

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
Id_InvolucradroReporte	Identificador único del usuario	Int(10)	0…9	No
IdInvolucrado	Obtenemos el id del reporte.	Int(5)	0….9	No
IdReporte	Obtenemos en id del involucrado.	int(5)	0…9	No

Sub Entidades.

En el sistema contiene una **Jerarquía de Generalización**, la Entidad padre es "REPORTES" y las sub entidades son las descripciones de las incidencias que registraremos en el sistema.



Sub Entidad: Incidencia01

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Hora_Amenaza	Especificamos la hora en la cual ocurrió la amenaza de bomba.	TIME	00:00:00	No
Descripción_Amenaza	Escribimos un relato de la amenaza.	Varchar(100)	A…Z y/o 0…9	No
Objeto_Encontrado	Especificamos si hay algún objeto encontrado en donde indicaron en la amenaza.	Varchar(100)	A…Z y/o 0…9	SI

Sub Entidad: Incidencia02

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Lesionados_Agresión	Contendrá el numero de lesionados que resultaron de la agresión.	Varchar(20)	A…Z y/o 0…9	No
Acción_Agresión	Contendrá la acción que se tomo con el agredido.	Varchar(50)	A…Z y/o 0…9	No
Participantes_Agresion	Contendrá el número de participantes en la agresión.	Varchar(20)	A…Z y/o 0…9	No
Tipo_Agresión	Contendrá el tipo de agresión fue.	Varchar(30)	A…..Z	No
Lugar_Agresión	Contendrá el lugar donde ocurre la agresión.	Varchar(50)	A…Z y/o 0…9	No
Estado_Agresor	Contendrá el estado el que se encuentra el agresor.	Varchar(50)	A…Z y/o 0…9	No

Sub Entidad: Incidencia03

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Descipción_Apoyo	Contendrá la descripción del apoyo.	Varchar(50)	A…Z y/o 0…9	No

Sub Entidad: Incidencia04

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Prioridad_Emergencia	Contendrá de qué prioridad fue la emergencia.	Varchar(20)	A…Z y/o 0…9	No
Lugar_Emergencia	Contendrá el lugar donde ocurrió la emergencia.	Varchar(50)	A…Z y/o 0…9	No
Estado_Persona	Contendrá el estado de la persona atendida.	Varchar(20)	A…Z	No
Acción_Emergencia	Contendrá la acción tomada.	Varchar(50)	A…..Z	No
Comportamiento_Persona	Contendrá el comportamiento de la persona atendida.	Varchar(30)	A….Z	No
Participantes_Emergencia	Contendrá el número de participantes de la Emergencia.	Varchar(20)	A…Z y/o 0…9	No

Sub Entidad: Incidencia06

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Descripción_ApoyoVial	Contendrá la descripción del apoyo vial.	Text	A…Z y/o 0…9	No
Lugar_ApoyoVial	Contendrá el Lugar ó descripción de lugar donde ocurre el apoyo vial.	Varchar(50)	A…Z y/o 0…9	No

Sub Entidad: Incidencia07

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Área_Prohibida	Contendrá el tipo ó el nombre del área prohibida.	Varchar(50)	A…Z y/o 0…9	No
Objeto_AreaProhibida	Contendrá el objeto que se introdujo al área prohibida.	Varchar(50)	A…Z y/o 0…9	No

Sub Entidad: Incidencia08

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Lugar_Bici	Contendrá el lugar donde se encontró la Bicicleta.	Varchar(50)	A…Z y/o 0…9	No
Caracteristicas_Bici	Contendrá las características de la bicicleta.	Varchar(255)	A…Z y/o 0…9	No

Sub Entidad: Incidencia09

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Lugar_Conato	Contendrá descripción del lugar donde ocurre el conato.	Varchar(20)	A…Z y/o 0…9	No
Sustancia_Conato	Contendrá la sustancia detectada en el conato de incendio.	Varchar(50)	A…Z y/o 0…9	No
Tipo_Tanque	Contendrá el tipo de tanque que provoca el conato de incendio.	Varchar(20)	A…Z y/o 0…9	Si

Objeto_ProvocaConato	Contendrá la descripción del objeto que provoco el conato de incendio.	Varchar(50)	A…Z y/o 0…9	Si

Sub Entidad: Incidencia10

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Lugar_Derrame	Contendrá descripción del lugar donde ocurre el derrame.	Varchar(20)	A…Z y/o 0…9	No
Sustancia_Derrame	Contendrá que sustancia se esta derramando.	Varchar(50)	A…Z y/o 0…9	No
Objeto_Derrame	Contendrá la descripción del objeto la cual esta provocando el derrame de sustancias peligrosas.	Varchar(20)	A…Z	Si

Sub Entidad: Incidencia11

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Lugar_Enjambre	Contendrá descripción del lugar donde ocurre el enjambre.	Varchar(20)	A…Z y/o 0…9	No
Descripción_Enjambre	Contendrá la descripción del enjambre.	Varchar(50)	A…Z	No

Sub Entidad: Incidencia12

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Lugar_Falta	Contendrá descripción del lugar donde ocurrió la falta.	Varchar(30	A…Z y/o 0…9	No
Comportamiento_Falta	Contendrá el comportamiento de la persona que cometió la falta.	Varchar(50)	A…Z y/o 0…9	No

Sub Entidad: Incidencia13

Atributo	Descripción	Tipo/longitud	Dominio	Nulo

FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Lugar_Fuga	Contendrá descripción del lugar de la fuga.	Varchar(30	A…Z y/o 0…9	No
Tipo_Gas	Contendrá descripción del tipo de gas de la fuga.	Varchar(50)	A…Z y/o 0…9	No

Sub Entidad: Incidencia14

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Tipo	Contendrá descripción de quien ingreso sin autorización.(Persona ó Vehiculo).	Varchar(20)	A…Z y/o 0…9	No
Placas_Ingreso	Contendrá el numero de placas del vehiculo que ingreso.	Varchar(20)	A…Z y/o 0…9	Si
Marca_Ingreso	Contendrá la marca del vehiculo que ingreso.	Varchar(20)	A…Z y/o 0…9	Si
Color_Ingreso	Contendrá el color del vehiculo que ingreso.	Varchar(20)	A…..Z	Si
Lugar_Ingreso	Contendrá descripción del lugar donde ingreso.	Varchar(30)	A…Z y/o 0…9	Si

Sub Entidad: Incidencia15

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Participantes_Intox	Contendrá el número de participantes durante la intoxicación.	Varchar(20)	A…Z y/o 0…9	No
Lugar_Intox	Contendrá descripción del lugar donde ocurre la intoxicación.	Varchar(30)	A…Z y/o 0…9	No
Estado_Intox	Contendrá el estado en el que se encuentra el intoxicado.	Varchar(30	A…Z y/o 0…9	No
Comportamiento_Intox	Contendrá descripción del comportamiento del intoxicado.	Varchar(30)	A…..Z	No

Sub Entidad: Incidencia16

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Participantes_Juego	Contendrá el número de participantes en el juego prohibido.	Varchar(20)	A…Z y/o 0…9	No

Lugar_Juego	Contendrá descripción del lugar donde ocurre el juego prohibido.	Varchar(30)	A…Z y/o 0…9	No
Descripción_Juego	Contendrá descripción del juego prohibido.	Varchar(50)	A…Z y/o 0…9	No

Sub Entidad: Incidencia17

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Objeto_Perdido	Contendrá el nombre del objeto perdido.	Varchar(20)	A…Z y/o 0…9	No
Descripción del Objeto	Contendrá descripción del objeto perdido.	Varchar(255)	A…Z y/o 0…9	No

Sub Entidad: Incidencia20

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Participantes_patrimonio	Contendrá el número de participantes que afectaron el patrimonio.	Varchar(20)	A…Z y/o 0…9	No
Lugar_patrimonio	Contendrá descripción del lugar donde ocurre la incidencia patrimonio afectado.	Varchar(30)	A…Z y/o 0…9	No
Descripción_Patrimonio	Contendrá descripción del patrimonio afectado.	Varchar(255)	A…Z y/o 0…9	No

Sub Entidad: Incidencia25

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Lesinados_Sismo	Contendrá el número de lesionados que dejo el sismo.	Varchar(20)	A…Z y/o 0…9	No
Daños_Sismo	Contendrá descripción de los daños del sismo	Varchar(255)	A…Z y/o 0…9	No

Sub Entidad: Incidencia26

Atributo	Descripción	Tipo/longitud	Dominio	Nulo
FK_ Id_reporte	Identificador único de la incidencia.	Int(5)	0…..9	No
Placas_Expuesto	Contendrá el número placas del vehiculo expuesto.	Varchar(20)	A…Z y/o 0…9	No
Marca_Expuesto	Contendrá la marca del vehiculo expuesto.	Varchar(30)	A…Z y/o 0…9	No
Color_Expuesto	Contendrá el color del vehiculo expuesto.	Varchar(255)	A…Z	No

Riesgo	Contendrá el tipo de riesgo que tiene el vehiculo expuesto.	Varchar(10)	A…Z y/o 0…9	No
Objetos_Visibles	Contendrá descripción de los objetos visibles del vehiculo expuesto.	Varchar(255)	A…Z y/o 0…9	Si

2.4.4.- Identificación de claves primarias.

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

Entidad: Universidad

Atributo	**Identificador**
Id_Universidad	PK

Entidad: Dependencia

Atributo	**Identificador**
Id_dependencia	PK

Entidad: Usuarios

Atributo	**Identificador**
Id_Usuario	PK

Entidad: Incidencia

Atributo	**Identificador**
Id_Incidencia	PK

Entidad: Reportes

Atributo	**Identificador**
Id_reporte	PK

Entidad: Involocrado

Atributo	**Identificador**
Id_Involocrado	PK

La siguiente entidad tiene clave primaria (PK) compuesta por los identificadores de la Entidad Involucrado y Reportes.

Entidad: InvolocradoReporte

Atributo	**Identificador**
IdInvolocradoReporte	PK

2.2.5.- Diseño de la Interfaz del Sistema.

El sistema será una aplicación Web la cual será visualizada a través de un navegador de Internet por lo tanto, requerimos diseñar una interfaz que sea sencilla, eficiente, que no distraiga al usuario con elementos innecesarios, que sea minimalista y que cumpla con los estándares de diseño de Web XHTML, CSS, Ajax y JavaScript.

A continuación en la **Figura 2.7** mostramos un Bosquejo General de la Interfaz Web del Sistema.

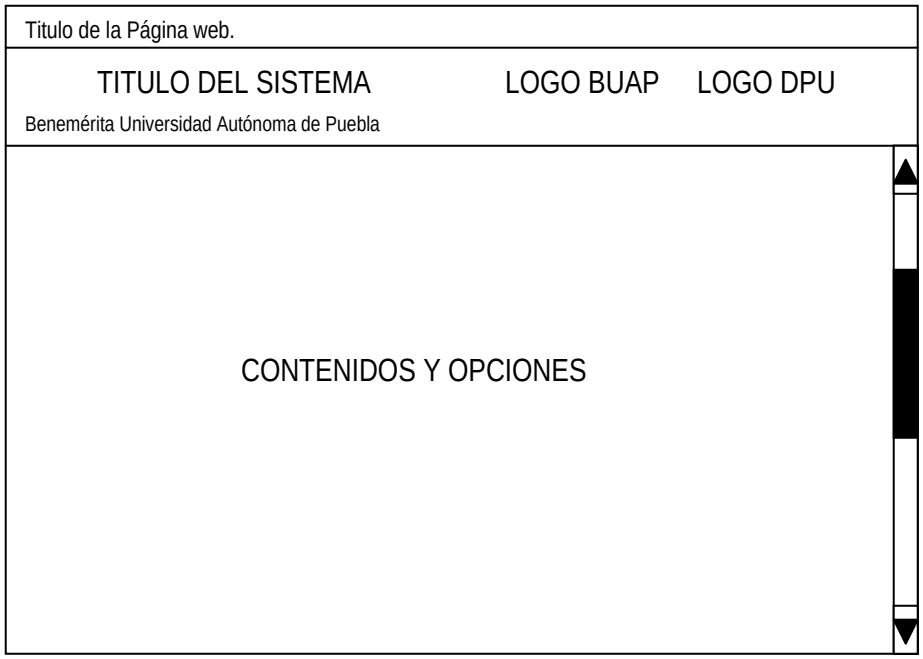


Figura 2.7.- Bosquejo General del diseño de la página web del sistema de base de datos.

Como podemos observar el diseño de la página Web del sistema contendrá dos marcos (*frames*), el marco superior contendrá el nombre del sistema, nombre de la universidad, los logotipos de la universidad y de la Dirección de Protección Universitaria. El marco inferior ó marco de contenidos contendrá la interfaz para poder acceder al sistema, las diferentes opciones que se podrán hacer, los formularios que llenaran los usuarios, consultas esta opciones irán apareciendo conforme a las acciones que vayan ocurriendo en el sistema.

De esta manera los usuarios interactuaran con el sistema en el marco inferior ó marco de contenidos.

CAPÍTULO 3.- IMPLEMENTACIÓN DEL SISTEMA

Introducción

En todos los desarrollos de sistemas de bases de datos se hace un análisis para decidir que herramientas son las apropiadas, estas deben cubrir las necesidades del sistema y llevar a cabo una correcta implementación.

3.1.- HERRAMIENTAS DE IMPLEMENTACIÓN

Las herramientas que utilizaremos para la implementación del sistema son Php y MySQL, ya que cubren la necesidades del sistema, haremos una descripción breve de ellas a continuación.

3.1.1.- Php.

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de programación usado principalmente como herramienta de desarrollo de aplicaciones Web [10]. PHP nos permite diseñar páginas dinámicamente de servidor, es decir, generar páginas bajo petición capaces de responder de manera inteligente a las demandas del cliente y que nos permitan la automatización de gran cantidad de tareas.

PHP está muy relacionado con el lenguaje de hipertexto HTML; tanto es así que el código PHP aparece normalmente insertado dentro de un documento HTML. El documento PHP, una vez interpretado correctamente en el servidor, genera una página HTML que será enviada al cliente.

En PHP se combinan muchas características que contribuyen notablemente su utilización masiva; entre otras, está el hecho de ser un software de libre distribución y multiplataforma que sigue la filosofía de Open Source. Una de las características que más ha influido en su popularización es la sencillez de uso que presenta a los programadores principiantes.

PHP proporciona las siguientes funcionalidades:

- **Funciones de correo electrónico:** envío de correos individual o por grupos, parametrizando toda una serie de aspectos tales como el e-mail de procedencia, asunto, destinatario.

- **Gestión de bases de datos:** El lenguaje PHP ofrece interfaces para el acceso a la mayoría de las bases de datos comerciales y por ODBC a todas las bases de datos posibles en sistemas Microsoft.

- **Gestión de archivos:** mediante operaciones de creación, borrado, modificación, además de ofrecer transferencia de archivos.

- **Tratamiento de imágenes:** mediante funciones de automatización de formato, envío de lotes de imágenes y funciones de graficado.

PHP además proporciona las siguientes posibilidades:

- Soporte para múltiples sistemas operativos: Unix, Microsoft Windows, Mac OS X, RISC OS.

- Soporte para múltiples servidores Web: Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro Server, Caudium, Xitami, OmniHTTPd y muchos otros.

- Soporte para múltiples base de datos: Adabas D, Ingres, Oracle, dBase, InterBase, Ovrimos, Empress, FrontBase, PostgresSQL, mSQL, Solid, Hyperwave, Direct MS-SQL, Sybase, IBM DB2, Informix, Unix dbm y MySQL, entre otras.
- Soporte para ODBC y extensiones DBX.

- Soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM y muchos más.

- Generación de resultados en multiples formatos como XHTML, XML, ficheros de imágenes, ficheros PDF Y películas Flash.

- Funciones de comercio electrónico, como Cybercash, CyberMUT, VeriSign Playflow Pro y CCVS para las pasarelas de pago.

Y un sin fin de posibilidades que van en aumento cada día.

3.1.2.- MySQL.

MySQL es un sistema gestor de base de datos relacionales, que además ofrece compatibilidad con PHP, Perl, C y HTML, y funciones avanzadas de administración y optimización de base de datos para facilitar las tareas habituales.

Implementa funcionalidades Web, permitiendo un acceso seguro y sencillo a los datos a través de Internet. Este sistema gestor de base de datos incluye capacidades de análisis integradas, servicios de transformación y duplicación de datos y funciones de programación mejoradas.

Se puede decir que MySQL es un sistema cliente servidor de administración de base de datos relacionales diseñado para el trabajo tanto en los sistemas operativos Windows como en

los sistemas Unix/Linux. Además, determinadas sentencias de MySQL pueden ser embebidas en código PHP y HTML para diseñar aplicaciones Web dinámicas que incorporan la información de las tablas de MySQL a páginas Web. Asimismo, MySQL es compatible con el software más potente de diseño Web, como Dreamweaver MX.

MySQL proporciona las siguientes características y ventajas:

- Disponibilidad en gran cantidad de plataformas y sistemas.
- Diferentes opciones de almacenamiento según si se desea velocidad en las operaciones o el mayor número de operaciones disponibles.
- Transacciones y claves foráneas.
- Conectividad segura.
- Replicación.
- Búsqueda e indexación de campos de texto.
- Acceso a las bases de datos de forma simultánea por varios usuarios y/o aplicaciones.
- Seguridad, en forma de permisos y privilegios.
- Portabilidad: SQL es también un lenguaje estandarizado, de modo que las consultas hechas usando SQL son fácilmente portables a otros sistemas y plataformas.
- Escalabilidad: es posible manipular bases de datos enormes, del orden de seis mil tablas y alrededor de cincuenta millones de registros, y hasta 32 índices por tabla.

3.2. Implementación de la base de datos en MySQL.

Como ya hemos elegido hacer la implementación en MyQSL mostraremos como crear la base de datos que utilizara el sistema, ejecutaremos las siguientes sentencias en lenguaje en SQL.

La siguiente sentencia nos permite ingresar como administrador de MySQL.

```
Mysql –u root –p
```

Después de ejecutar la sentencia mysql pedirá la contraseña del usuario que esta deseando entrar, en este caso el usuario es el administrador (root), procedemos a escribir la contraseña correcta, se debe tener en cuenta que MySQL distingue mayúsculas y minúsculas.

```
Enter password: ***********
```

Por cada carater que escribamos de la contraseña aparecerá un asterisco para enmascarar la llave y sea mas seguro.

Después de haber escrito correctamente la contraseña, aparecerá el *prompt* ó indicador de MySQL, lo cual indica que esta listo para recibir y ejecutar sentencias en el lenguaje SQL.

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.21

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

Mysql>_
```

A continuación escribiremos la sentencia SQL para crear la base de datos del Sistema.

```
Mysql> CREATE DATABASE SISU;
```

Ya que hemos creado la base de datos para el sistema, debemos indicar a MySQL que base de datos utilizaremos para continuar con la creación de las tablas.

```
Mysql> USE SISU;
```

Si hemos elegido correctamente la base de datos aparecerá un mensaje como el siguiente:

```
Database changed
```

Ya que hemos indicado a MySQL en que base de datos trabajaremos, procedemos a la creación de las tablas del sistema.

Creamos la Tabla **UNIVERSIDAD**

```
mysql> CREATE TABLE UNIVERSIDAD (
      ID_UNI INT AUTO_INCREMENT NOT NULL,
      NOMBRE VARCHAR(100) NOT NULL,
      DIRECCION VARCHAR(100) NULL,
      TELEFONO VARCHAR(30) NULL,
      RECTOR VARCHAR(50) NULL,
      KEY ID (ID));
```

Si la sentencia SQL se ejecuto la sentencia correctamente, aparecerá un mensaje como este:

```
Query OK, 0 rows affected (0.30 sec)
```

Creamos la tabla **USUARIOS**

```
mysql> CREATE TABLE USUARIOS (
      idUsuario int(7) NOT NULL AUTO_INCREMENT,
      LOGIN Varchar(100) NOT NULL,
      PASSWORD Varchar(100) NOT NULL,
      NOMBRE Varchar(255) DEFAULT NULL,
      DEPENDENCIA Varchar(50) NOT NULL,
      CORREO Varchar(100) DEFAULT NULL,
      KEY id (id));
```

Creamos la Tabla **DEPENDENCIAS**

```
mysql> CREATE TABLE DEPENDENCIAS(
      ID_DEP INT AUTO_INCREMENT NOT NULL,
      NOMBRE VARCHAR(50) NOT NULL,
      DIRECCION VARCHAR(50) NULL,
      TELEFONO VARCHAR(30) NULL,
      DIRECTOR VARCHAR(50),
      KEY ID (ID));
```

Creamos la tabla **INCIDENCIAS**

```
mysql> CREATE TABLE INCIDENCIAS (
      ID INT AUTO_INCREMENT NOT NULL,
      NOMBRE_INC VARCHAR(100) NOT NULL,
      CLAVE_INC VARCHAR(100) NULL,
      DESCRIPCION_INC VARCHAR(30) NULL,
      KEY ID (ID));
```

Creamos la tabla **REPORTES**

```
mysql>CREATE TABLE REPORTES
(
      ID INT AUTO_INCREMENT NOT NULL,
      CLAVE VARCHAR(10) NOT NULL,
      FECHAREP DATE NOT NULL,
      FECHASIS DATE NOT NULL,
      HORA TIME NOT NULL,
      TURNO VARCHAR(15) NOT NULL,
      SECTOR VARCHAR(15) NOT NULL,
      INCIDENCIA VARCHAR(15) NOT NULL,
      DEPENDENCIA VARCHAR(60) NOT NULL,
      EDIFICIO VARCHAR(20) NOT NULL,
      ESTACIONAMIENTO VARCHAR(20),
      VIALIDAD VARCHAR(20),
      OBSERVACIONES TEXT NULL,
      PRIMARY KEY(ID)
);
```

Creamos la Tabla **INVOLUCRADO**

```
mysql>CREATE TABLE INVOLOCRADO
(
      IDINVOLOCRADO INT AUTO_INCREMENT NOT NULL,
      CLAVE VARCHAR(10) NOT NULL,
      NOMBREINVOLUCRADO VARCHAR(100) NULL,
      MATRICULA VARCHAR(12) NULL,
      IFE VARCHAR(20) NULL,
      TELEFONOINVOLICRADO VARCHAR(20) NULL,
      UNIDADACADEMICA VARCHAR(60) NULL,
      PRIMARY KEY(IDINVOLOCRADO)
);
```

A continuación crearemos las tablas de las sub entidades de la tabla REPORTES.

Creamos la tabla **INCIDENCIA01** (AMENAZA DE BOMBA)

```
mysql> CREATE TABLE INCIDENCIA01
(
      IDINCIDENCIA01 INT NOT NULL,
      HORABOMBA TIME NOT NULL,
      DESCRIPCIONOBJETO VARCHAR(255) NULL,
      PRIMARY KEY(ID INCIDENCIA01),
      FOREIGN KEY(ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA02** (AGRESIONES Ó RIÑAS)

```
mysql >CREATE TABLE INCIDENCIA02
(
      IDINCIDENCIA02 INT NOT NULL,
      LESIONADOS VARCHAR(20) NULL,
      ACCION VARCHAR(30) NULL,
      PARTICIPANTES VARCHAR(20) NULL,
      TIPOAGRESION VARCHAR(30) NULL,
      AGRESOR TEXT NULL,
      LUGAR VARCHAR(50) NULL,
      ESTADO VARCHAR(50),
      PRIMARY KEY (IDINCIDENCIA02) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA03** (APOYO DIVERSO)

```
mysql>CREATE TABLE INCIDENCIA03
(
      IDINCIDENCIA03 INT NOT NULL,
      DESCRIPCION TEXT NULL,
      PRIMARY KEY (ID INCIDENCIA03) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA04** (EMERGENCIA MEDICA)

```
mysql>CREATE TABLE INCIDENCIA04
(
      IDINCIDENCIA04 INT NOT NULL,
      ACCION VARCHAR(50) NULL,
      PRIORIDAD VARCHAR(10) NULL,
      PARTICIPANTES VARCHAR(50) NULL,
      LUGAR VARCHAR(100) NULL,
      ESTADO VARCHAR(10) NULL,
      COMPORTAMIENTO VARCHAR(50) NULL,
      PRIMARY KEY (IDINCIDENCIA04) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA06** (APOYO VIAL)

```
mysql>CREATE TABLE INCIDENCIA06
(
      IDINCIDENCIA06 INT NOT NULL,
      DESCRIPCION TEXT NULL,
      LUGAR VARCHAR(50),
      PRIMARY KEY (IDINCIDENCIA06) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA07** (AREAS PROHIBIDAS)

```
mysql>CREATE TABLE INCIDENCIA07
(
      IDINCIDENCIA07 INT NOT NULL,
      AREAPROHIBIDA VARCHAR(100) NULL,
      OBJETO VARCHAR(100) NULL,
      PRIMARY KEY (IDINCIDENCIA07) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA08** (BICICLETA INSEGURA)

```
mysql>CREATE TABLE INCIDENCIA08
(
      IDINCIDENCIA08 INT NOT NULL,
      LUGAR VARCHAR(100),
      CARACTERISTICAS TEXT NULL,
      PRIMARY KEY (IDINCIDENCIA08) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA09** (CONATO DE INCENDIO)

```
mysql>CREATE TABLE INCIDENCIA09
(
      IDINCIDENCIA09 INT NOT NULL,
      LUGAR VARCHAR(100) NULL,
      SUSTANCIA VARCHAR(100) NULL,
      TIPOTANQUE VARCHAR(100) NULL,
      OBJETO      VARCHAR(100) NULL,
      PRIMARY KEY (IDINCIDENCIA09) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA10** (DERRAME DE SUSTANCIA PELIGROSA)

```
mysql>CREATE TABLE INCIDENCIA10
(
      IDINCIDENCIA10 INT NOT NULL,
      LUGAR VARCHAR (100) NULL,
      SUSTANCIA VARCHAR(100) NULL,
      PRIMARY KEY (IDINCIDENCIA10) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA11** (ENJAMBRE)

```
mysql>CREATE TABLE INCIDENCIA11
(
      IDINCIDENCIA11 INT NOT NULL,
      LUGAR VARCHAR (100) NULL,
      DESCRIPCION VARCHAR(100) NULL,
      PRIMARY KEY (IDINCIDENCIA11) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA12** (FALTAS A LA MORAL)

```
mysql>CREATE TABLE INCIDENCIA12
(
      IDINCIDENCIA12 INT NOT NULL,
      LUGAR VARCHAR(100) NULL,
      COMPORTAMIENTO VARCHAR(100) NULL,
      PRIMARY KEY (IDINCIDENCIA12) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA13** (FUGA DE GAS)

```
mysql>CREATE TABLE INCIDENCIA13
(
      IDINCIDENCIA13 INT NOT NULL,
      LUGAR VARCHAR(100) NULL,
      TIPOGAS VARCHAR(100) NULL,
      PRIMARY KEY (IDINCIDENCIA13) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
```

```
);
```

Creamos la tabla **INCIDENCIA14** (INGRESO NO AUTORIZADO)

```
mysql>CREATE TABLE INCIDENCIA14
(
      IDINCIDENCIA14 INT NOT NULL,
      TIPO VARCHAR(100) NULL,
      PLACAS VARCHAR(50) NULL,
      MARCA VARCHAR(50) NULL,
      COLOR VARCHAR(50) NULL,
      LUGAR VARCHAR(100) NULL,
      PRIMARY KEY (IDINCIDENCIA14) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
      );
```

Creamos la tabla **INCIDENCIA15** (INTOXICACION POR ESTUPEFACIENTES)

```
mysql>CREATE TABLE INCIDENCIA15
(
      IDINCIDENCIA15 INT NOT NULL,
      PARTICIPANTES VARCHAR(100) NULL,
      LUGAR VARCHAR(100) NULL,
      ESTADO VARCHAR(100) NULL,
      COMPORTAMIENTO VARCHAR(100) NULL,
      PRIMARY KEY (IDINCIDENCIA15) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA16** (JUEGOS PROHIBIDOS)

```
mysql>CREATE TABLE INCIDENCIA16
(
      IDINCIDENCIA16 INT NOT NULL,
      PARTICIPANTES VARCHAR(30) NULL,
      LUGAR VARCHAR(100) NULL,
      DESCRIPCION VARCHAR(255) NULL,
      PRIMARY KEY (IDINCIDENCIA16) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA17** (OBJETOS PERDIDOS)

```
mysql>CREATE TABLE INCIDENCIA17
(
      IDINCIDENCIA17 INT NOT NULL,
      OBJETO VARCHAR(100) NULL,
      LUGAR VARCHAR(100) NULL,
      PRIMARY KEY (IDINCIDENCIA17) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA20** (PATRIMONIO AFECTADO)

```
mysql>CREATE TABLE INCIDENCIA20
(
      IDINCIDENCIA20 INT NOT NULL,
      PARTICIPANTES VARCHAR(30) NULL,
      LUGAR VARCHAR(100) NULL,
      DESCRIPCION VARCHAR(255) NULL,
      PRIMARY KEY (IDINCIDENCIA20) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA25** (SISMO)

```
mysql>CREATE TABLE INCIDENCIA25
(
      IDINCIDENCIA25 INT NOT NULL,
      LESIONADOS VARCHAR(20) NULL,
      DANOS TEXT NULL,
      PRIMARY KEY (IDINCIDENCIA25) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Creamos la tabla **INCIDENCIA26** (VEHICULO EXPUESTO)

```
mysql>CREATE TABLE INCIDENCIA26(
      IDINCIDENCIA26 INT NOT NULL,
      PLACAS VARCHAR(12) NULL,
      MARCA VARCHAR(20) NULL,
      COLOR VARCHAR(20) NULL,
      RIESGO VARCHAR(2) NULL,
      VISIBLES VARCHAR(100) NULL,
      PRIMARY KEY (IDINCIDENCIA26) ,
      FOREIGN KEY (ID) REFERENCES REPORTES(ID)
);
```

Hemos escrito el script SQL para cada tabla de nuestra base de datos, lo que vemos en la **Figura 3.1** es una tabla ya creada en MySQL.

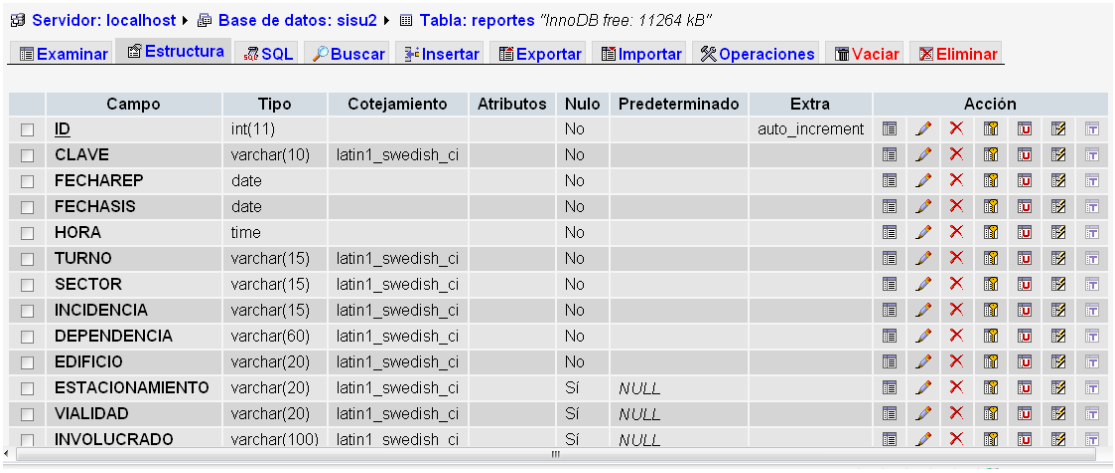


Figura 3.1.- Tabla reportes implementada en MySQL.

3.3. Implementación del Sistema.

La implementación del sistema se refiere a la programación de la versión final del sistema, en esta etapa, se muestran las herramientas para la captura y validación de la información, los métodos de seguridad implementados y la interacción con la base de datos.

Como ya hemos mencionado el sistema de registro de incidencias será accedido a través de Internet por lo cual necesitamos programar una interfaz que sea visible desde la Web para que los usuarios interactúen con el sistema de base de datos por medio de un navegador. En la **Figura 3.2 y Figura 3.3** vemos la implementación del sistema logrando la interacción con a base de datos**.**

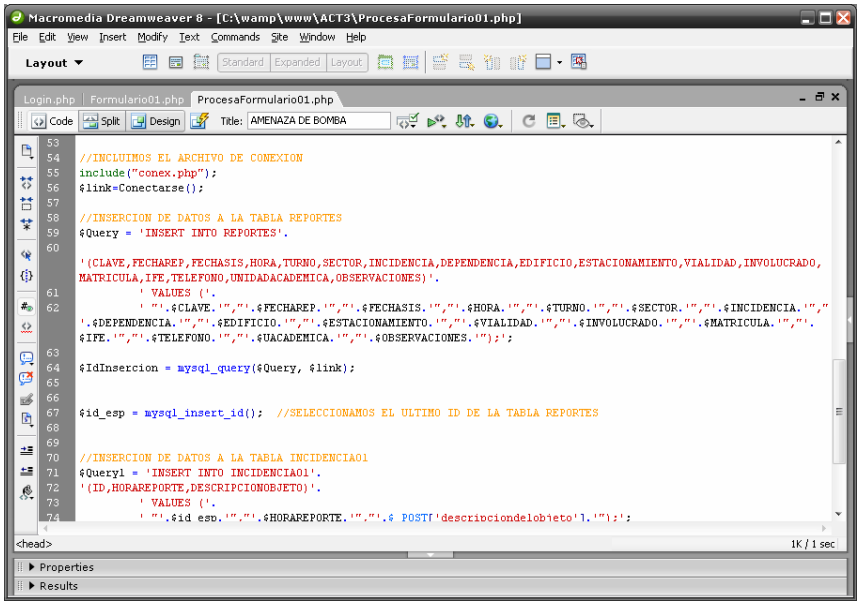


Figura 3.2.- Implementación de la interacción con la base de datos.

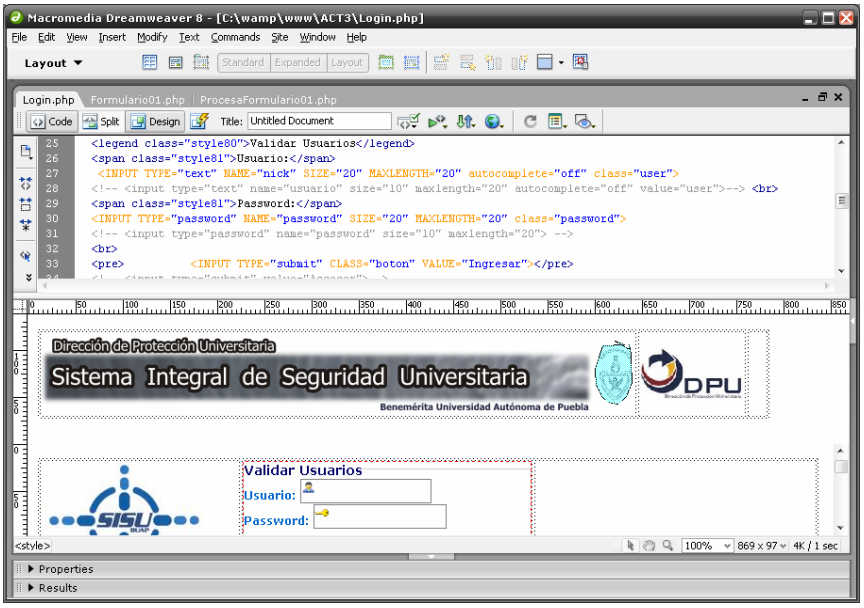


Figura 3.3.- Implementación de la interfaz del sistema.

CAPÍTULO 4: INTERFAZ Y PRUEBAS DEL SISTEMA

4.1.-INTERFAZ DEL SISTEMA.

El desarrollo de la interfaz del sistema está basado en los estándares Web 2.0 y es por medio de ella que los usuarios pueden comunicarse e interactuar. Las principales funciones de la interfaz Web son las siguientes:

- Control de acceso.
- Envió de Información.
- Búsqueda de Información.
- Actualización de la información.
- Eliminar Información.

4.2.1.- Interfaz de Inicial.

En la Pantalla Inicial **Figura 4.1** esta pantalla permitirá a los usuarios autentificarse para poder acceder al sistema, el usuario deberá poner su nombre de usuario y password en los campos establecidos y depuse dar clic en el botón ingresar.

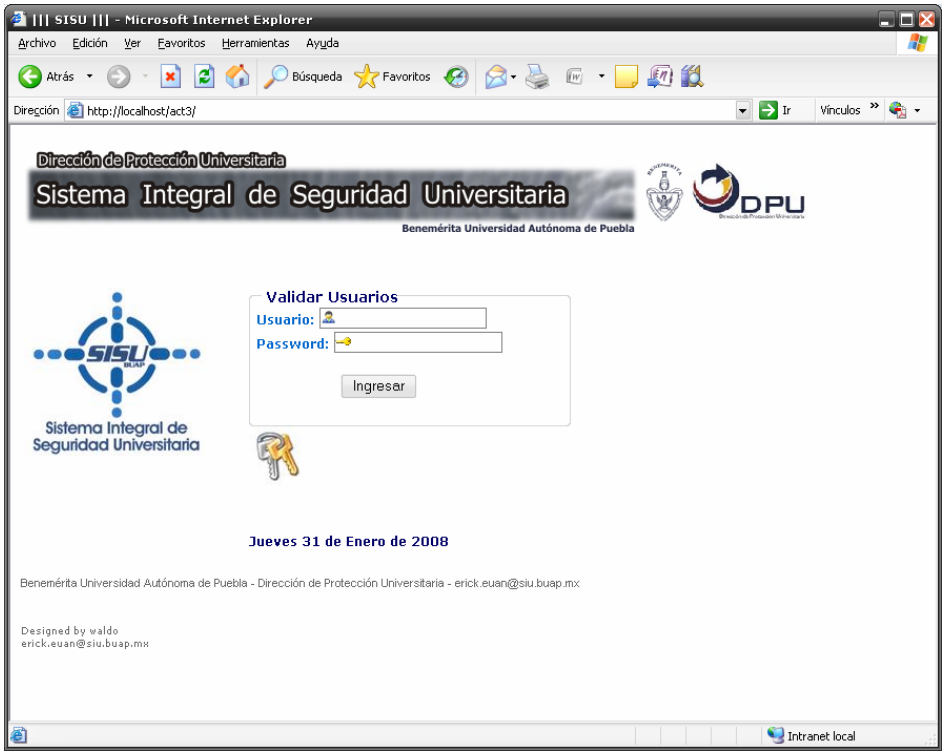


Figura 4.1.- Pantalla de Inicial del Sistema.

El sistema revisara la información proporcionada por el usuario en la base de datos. Si el usuario da clic en el botón ingresar sin escribir los datos para identificarse, el sistema mostrara un mensaje informando que tiene que escribir un nombre de usuario y contraseña para poder acceder al sistema como se muestra en la **Figura 4.2**.

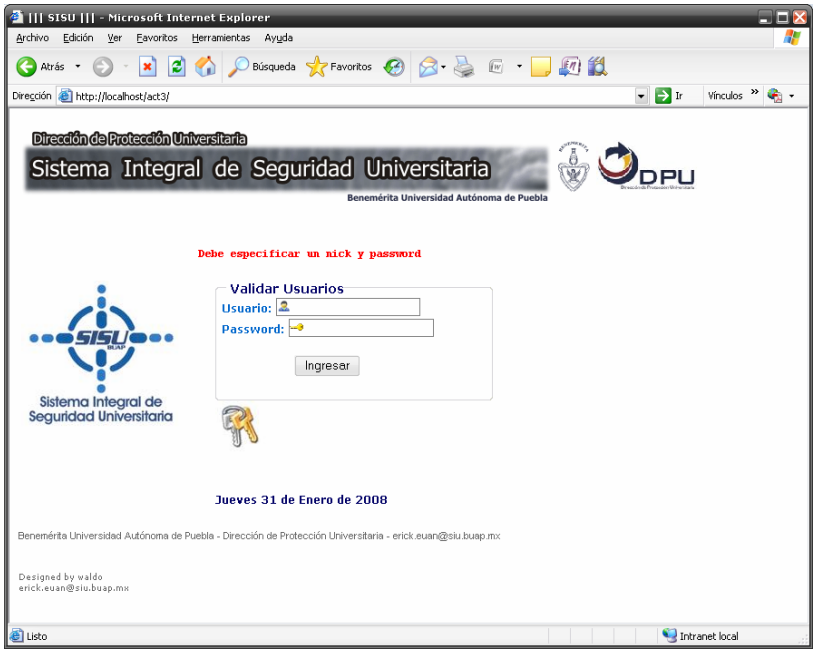


Figura 4.2.- Pantalla de validación de usuarios.

En caso de no poner un nombre de usuario valido para acceder, el sistema mostrara un mensaje informando que el usuario no existe en la base de datos como se muestra en la **Figura 4.3**.

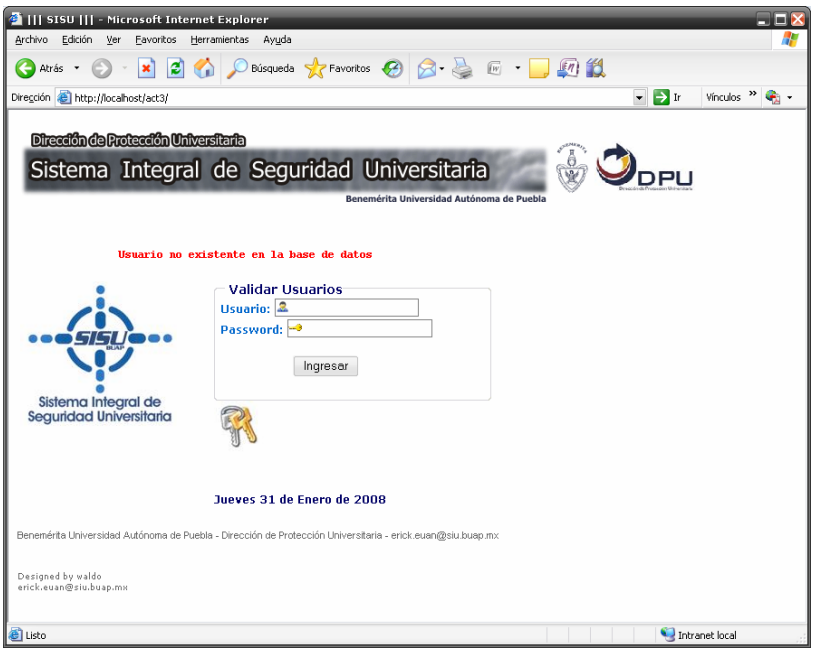


Figura 4.3.- Pantalla de Usuario no valido.

Si el usuario existe en la base de datos pero no haya escrito correctamente su contraseña el sistema de avisara que se ha escrito una incorrecta **Figura 4.4**.

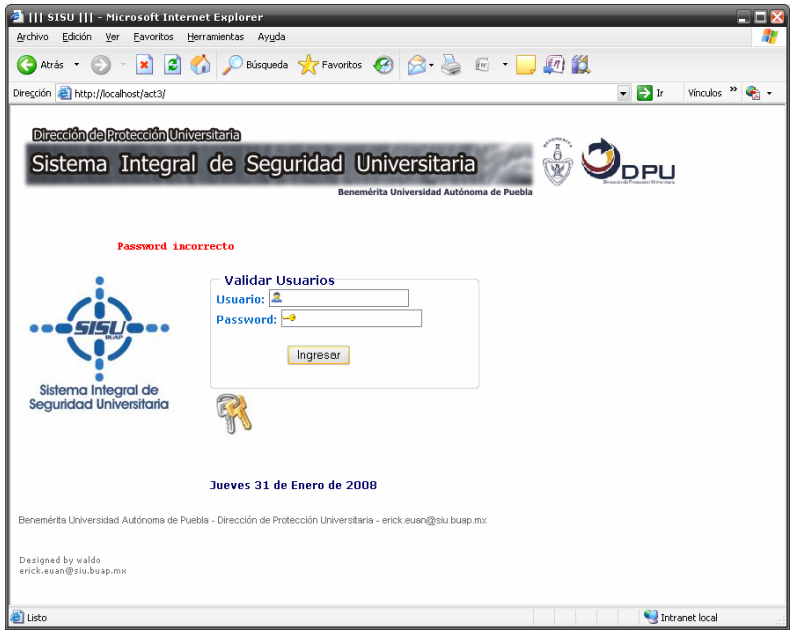


Figura 4.4.-El sistema especificará al usuario que su contraseña no es la correcta.

4.2.2.- Interfaz del menú.

En la interfaz donde mostramos las opciones principales ó menú se podrá acceder una vez autentificado el usuario de manera correcta, en la **Figura 4.5** mostramos la pantalla con las diferentes opciones.

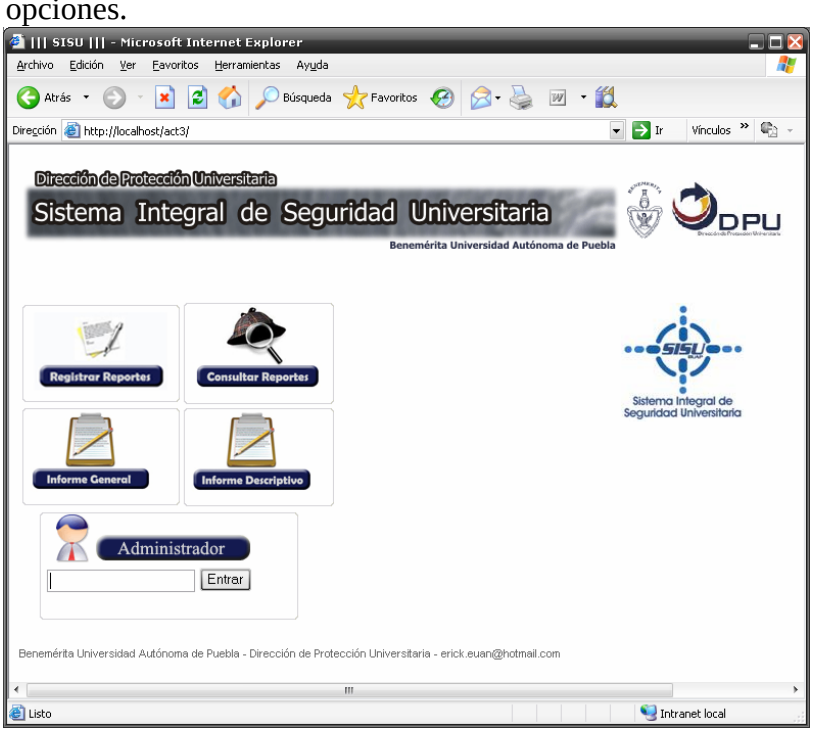


Figura 4.5.- Pantalla del Menú Principal.

A continuación las descripciones de las opciones principales ó menú del sistema.

Registrar Reportes: Esta opción permitirá mostrara a los usuarios una pantalla que contendrá las incidencias que puede registrar en el sistema.

Consultas: Esta opción permitirá a los usuarios consultar las incidencias ya registradas en el sistema.

Informe General: El sistema mostrara una tabla donde veremos el total de las incidencias ocurridas en cada mes según el año que seleccionemos, además mostrara los totales por incidencias y por mes, al final veremos el total anual.

***Informe Descriptivo*:** En sistema mostrara una tabla donde veremos cual es el comportamiento de las incidencias con respecto a la Dependencias, para generar esta tabla elegiremos la incidencia y el año, para poder hacer la consulta.

***Administrador*:** En esta opción el administrador podrá ingresar su clave, para efectos de mantenimiento por ejemplo, Agregar Usuarios, Eliminar Usuarios, Editar Información de los Usuarios, Agregar Dependencias y Editar Información de las Dependencias.

4.2.3.- Interfaz del administrador.

El Administrar el sistema implicar muchas tareas como planear, instalar y mantener actualizados los sistemas. En la siguiente **Figura 4.6** mostramos el espacio donde el administrador debe poner su contraseña para poder acceder al menú y poder dar mantenimiento.

Figura 4.6.- Contraseña del Administrador

La siguiente pantalla **Figura 4.7** muestra las opciones que puede realizar el administrador.

Usuarios:
- Agregar
- Editar
- Eliminar

Dependencias:
- Agregar
- Editar
- Eliminar

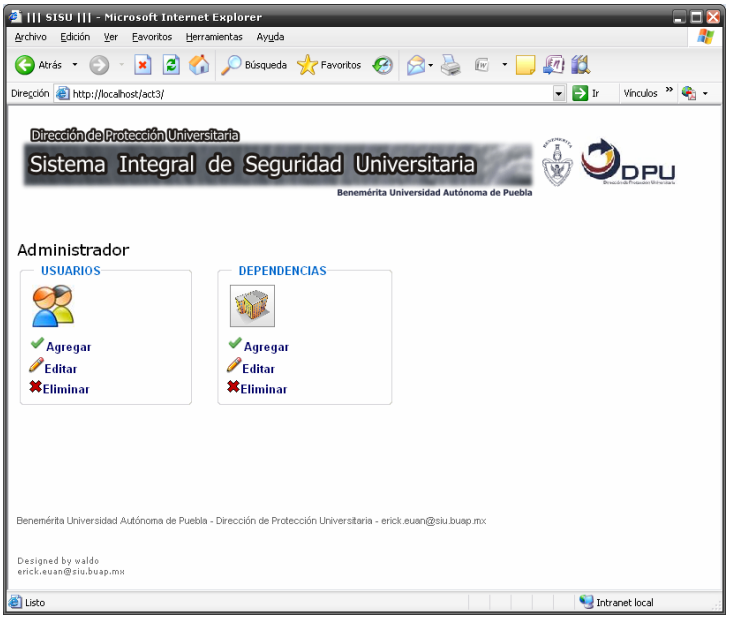


Figura 4.7.-Pantalla Opciones del Administrador.

4.2.3.- Interfaz de registro de incidencias.

El sistema contiene una pantalla donde se muestran las claves de las incidencias **Figura 4.8,** debemos tener presente que el grupo de usuarios que utilizara el sistema conocen a la perfección cada clave de las incidencias que registran.

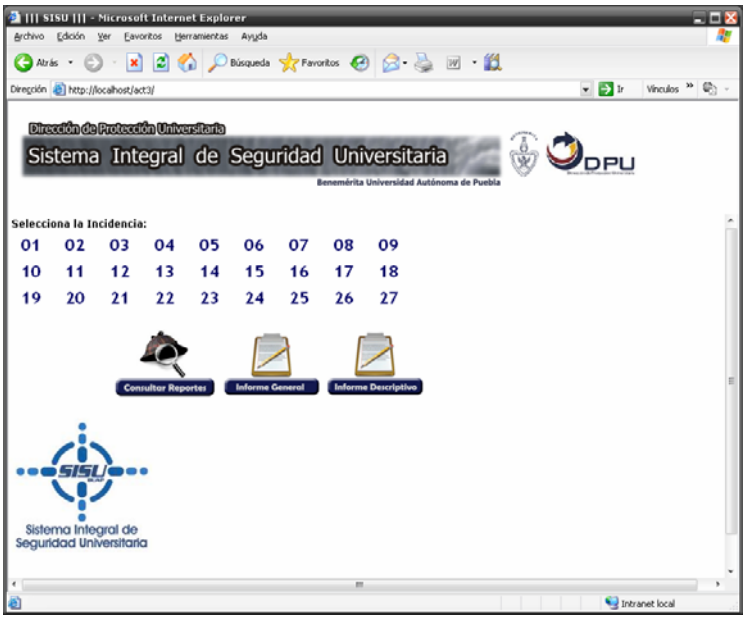


Figura 4.8.-Pantalla de selección de incidencias.

El usuario deberá dar clic en el número de incidencia para poder registrar un reporte de la incidencia seleccionada, posteriormente aparezca el formulario, en la pantalla siguiente **Figura 4.9** vemos como el número de incidencia 01 aparece en amarillo indicando que estamos a punto de seleccionarla.

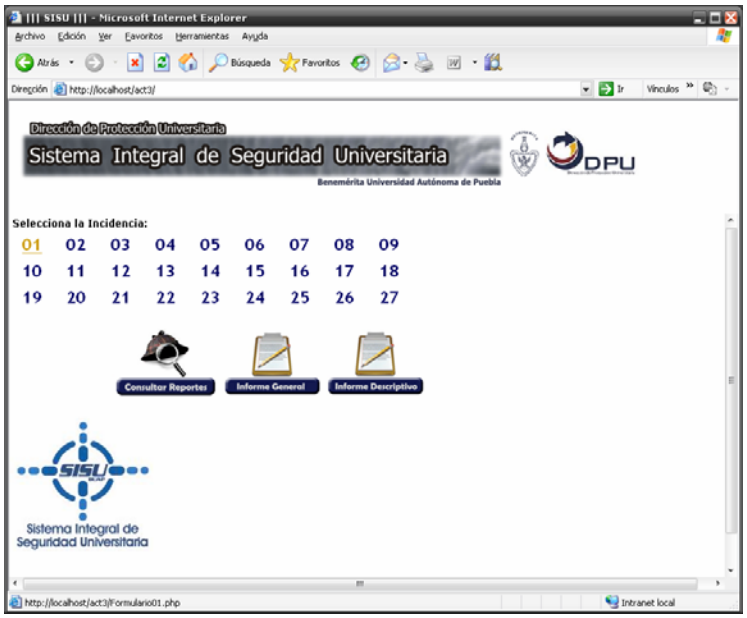


Figura 4.9.-Incidencia Seleccionada.

4.2.3.- Interfaz de los formularios.

Una ves que el usuario dio clic en el número de la incidencia que desea registrar aparecerá el formulario correspondiente como mostramos en la **Figura 4.10**.

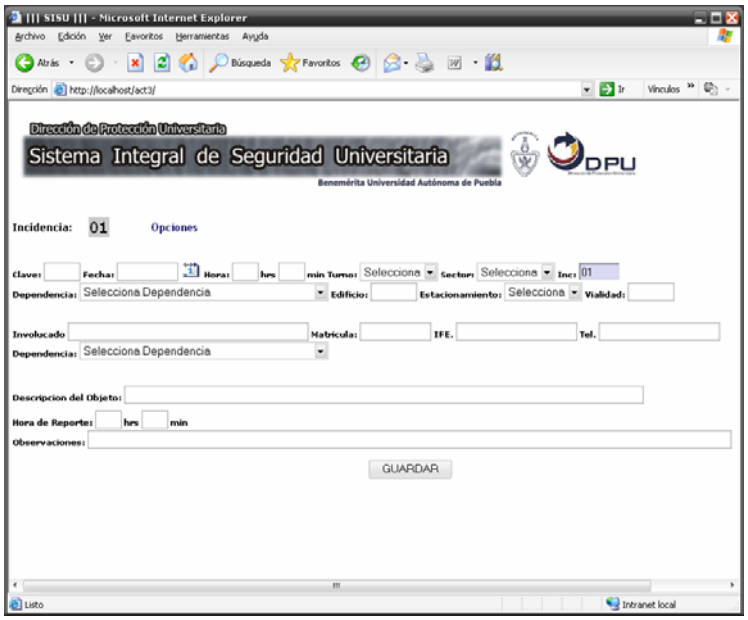


Figura 4.10.- Formulario de la incidencia seleccionad.

El usuario deberá llenar el formulario con los datos de la incidencia, cabe mencionar que algunos campos son obligatorios en caso de no llenarlos el sistema informara que faltan datos obligatorios que debe llenar en la **Figura 4.11**, vemos como aparece el mensaje de advertencia.

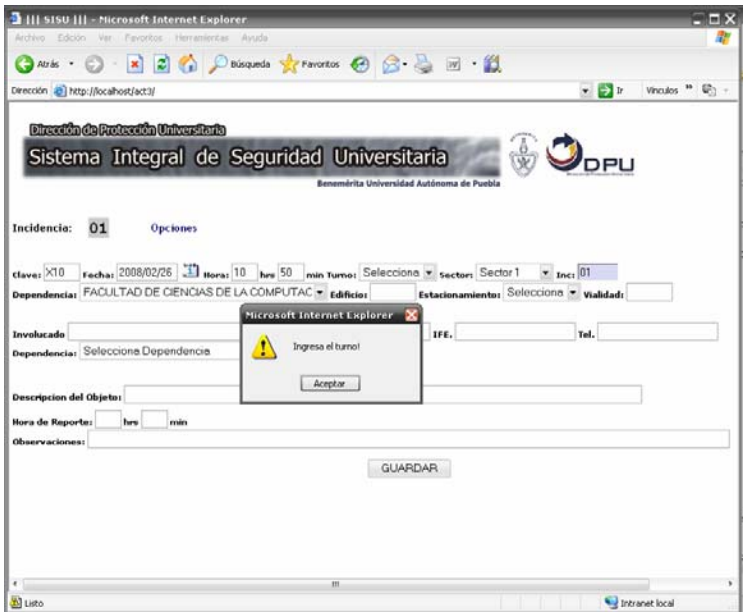


Figura 4.11.-El sistema avisa que faltan campos por llenar.

Una vez ingresado los datos de la incidencia y haber llenado los campos obligatorios el usuario debe dar clic en el botón guardar como se muestra en **Figura 4.12.**

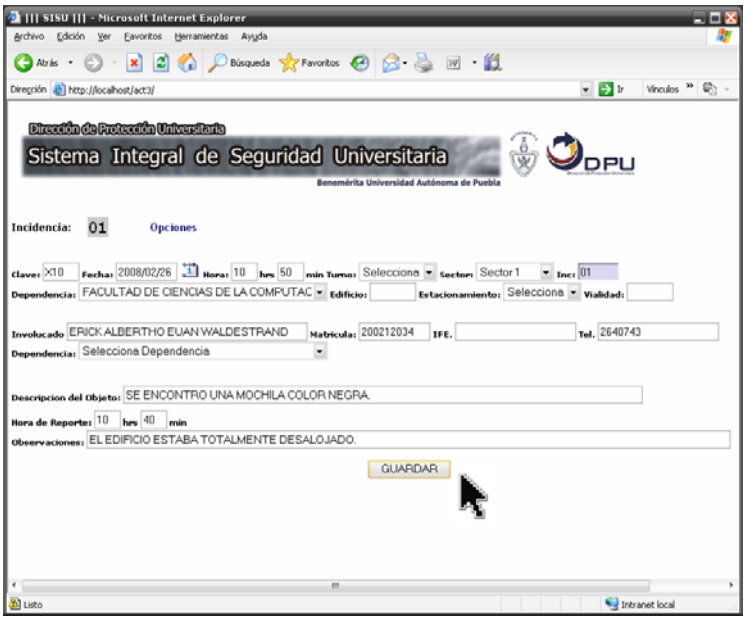


Figura 4.12.-Clic en el botón guardar para registrar el reporte.

Después de dar clic en guardar el sistema informara que el reporte ha sido registrado correctamente **Figura 4.13.**

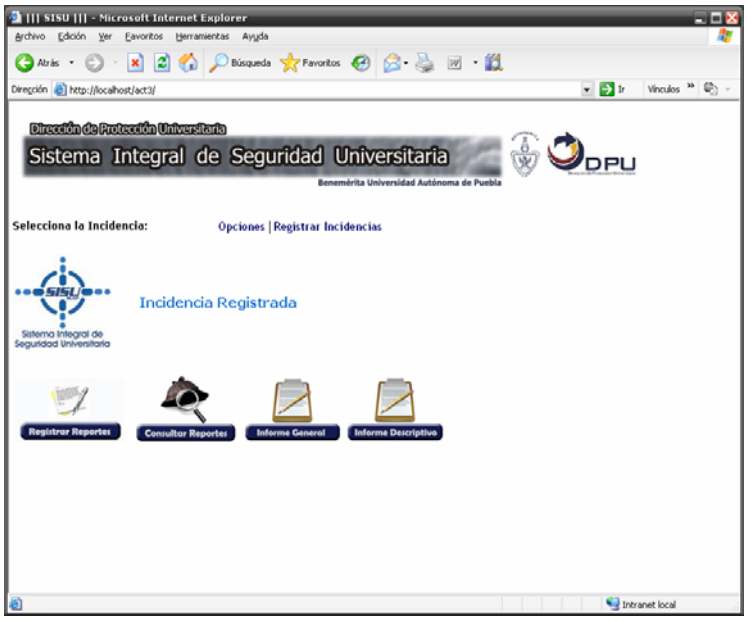


Figura 4.13.-Pantalla de confirmación de reporte registrado.

Una vez registrada la incidencia el usuario podrá dar clic en el vínculo opciones ó registrar incidencias, según desee.

4.2.4.- Interfaz de consulta de reportes de incidencia.

En la opción consultas en el menú del sistema el usuario podrá buscar reportes registrados en el sistema.

En la **Figura 4.14** el sistema mostrara diferentes opciones para poder realizar las consultas.

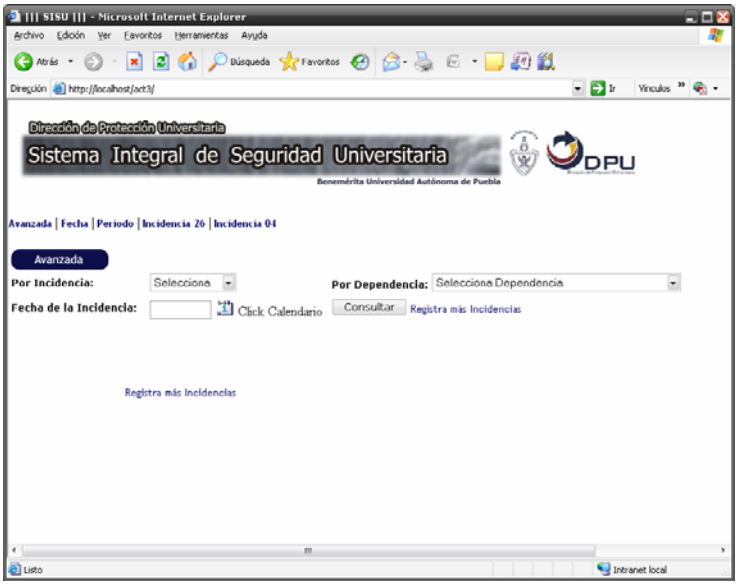


Figura 4.14.-Pantalla de consulta de Incidencias.

El usuario podrá elegir una de las opciones y dar clic en el botón “consultar” para realizar la búsqueda.

En la **Figura 4.15** vemos el ejemplo de elección de consulta por **Fecha**.

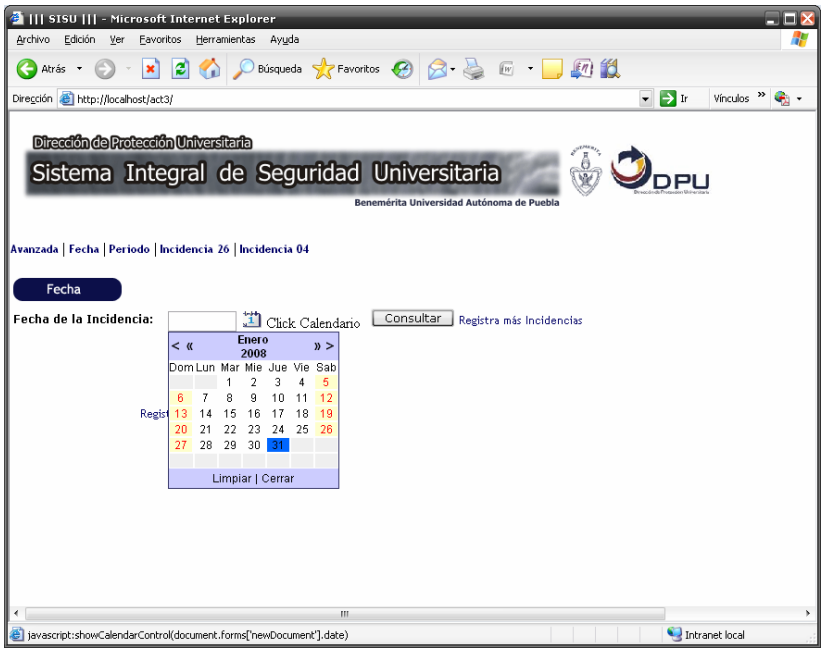


Figura 4.15.-Consulta por Fecha.

Ejemplo de elección de consulta por **Periodo** **Figura 4.16**, aquí el usuario ingresara las fechas para que el sistema busque los reportes que ocurrieron en ese periodo.

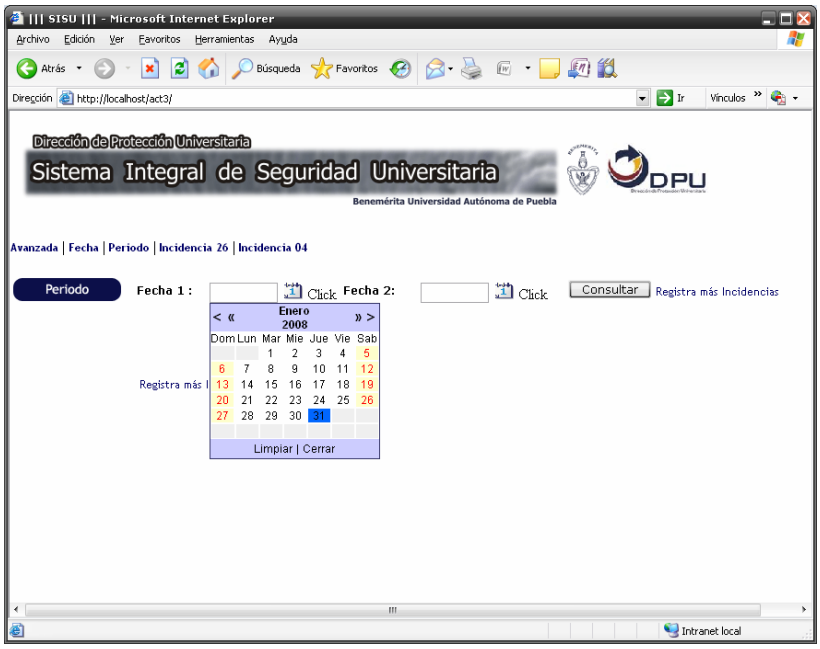


Figura 4.16.-Consulta por Periodo.

En la consulta **Avanzada** el sistema buscara aquellos reportes que cumplan con los datos de búsqueda que ha seleccionado el usuario observemos la **Figura 4.17** el ejemplo.

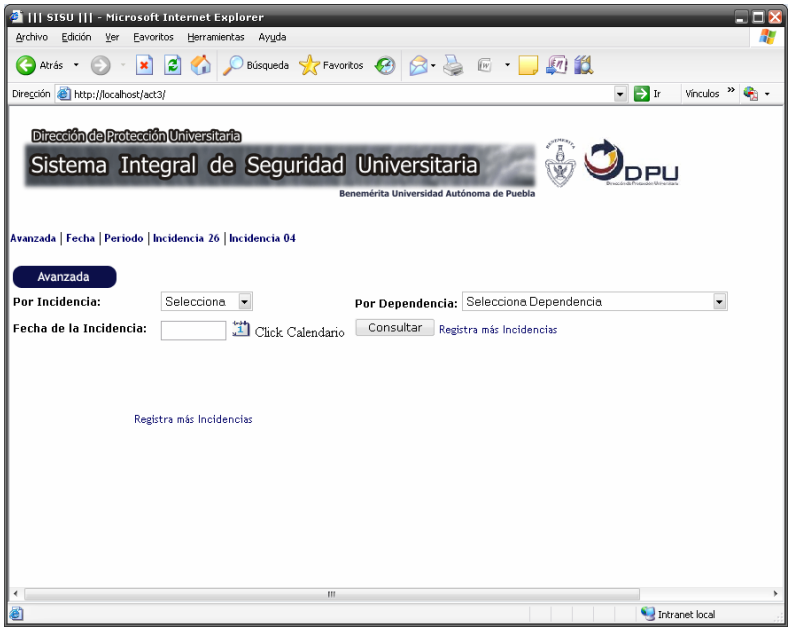
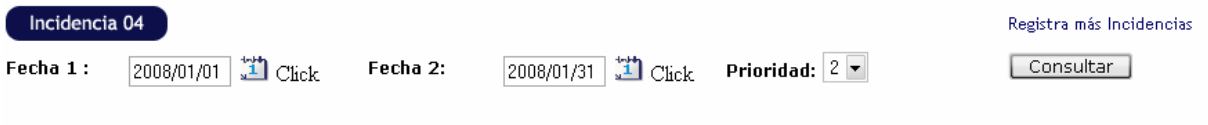


Figura 4.17.-Búsqueda Avanzada.

Si seleccionamos consultar "**Incidencia 04**" podremos elegir en que rangos de fechas queremos buscar los reportes además de indicar la prioridad que queremos buscar.



Lo mismo haremos con la incidencia 26 seleccionaremos el rango de fechas para generar la búsqueda y la prioridad.

Después de haber seleccionado la opción de búsqueda de reportes de incidencia el sistema hará un despliegue de los reportes encontrados según la condición de búsqueda que hay pedido el usuario como se muestra en la **Figura 4.18**.

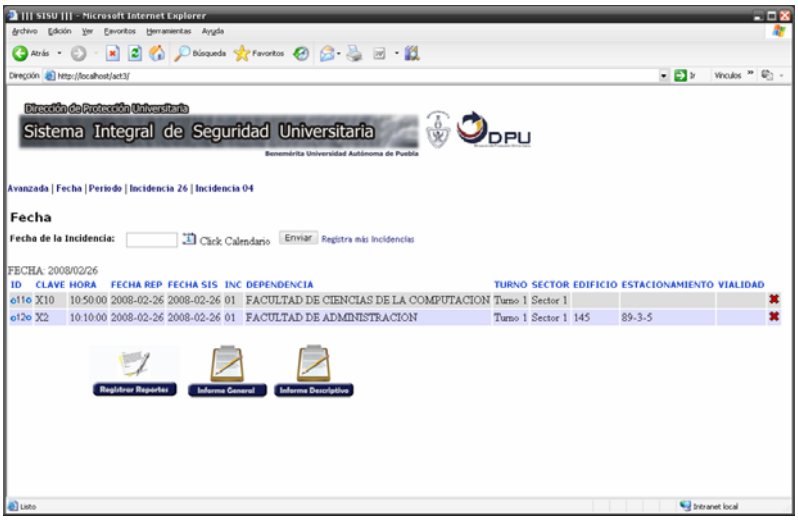


Figura 4.18.- Resultado de la Búsqueda.

El usuario podrá ver toda la información de la incidencia deseada dando clic en el número de reporte de la incidencia ó eliminar la incidencia dando clic en el vínculo (borrar) que tiene cada registro.

4.2.4.- Interfaz de descripción del reporte.

Si el usuario da clic en el número de reporte el sistema desplegara toda la información del reporte seleccionado, esto permitirá ver la información general y descriptiva del reporte **Figura 4.20**.

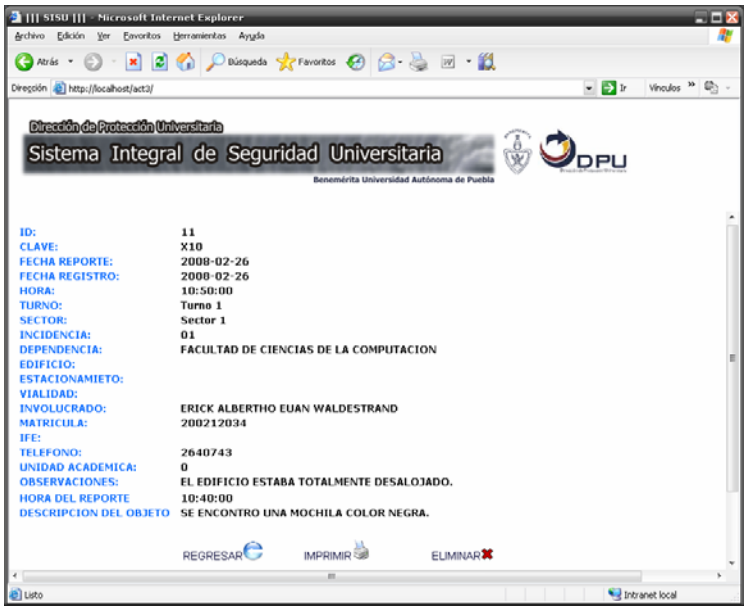


Figura 4.20.-Descripción del Reporte.

Una ves que el sistema haya desplegado la información del reporte el usuario tendrá la opción de imprimir el reporte ó eliminar el reporte.

Si el usuario elige IMPRIMIR será necesario tener una impresora conectada localmente ó una compartida en red para imprimir el reporte.

Si el usuario elige ELIMINAR , el sistema mostrara una pantalla **Figura 4.21** para confirmar el deseo de eliminar el reporte.

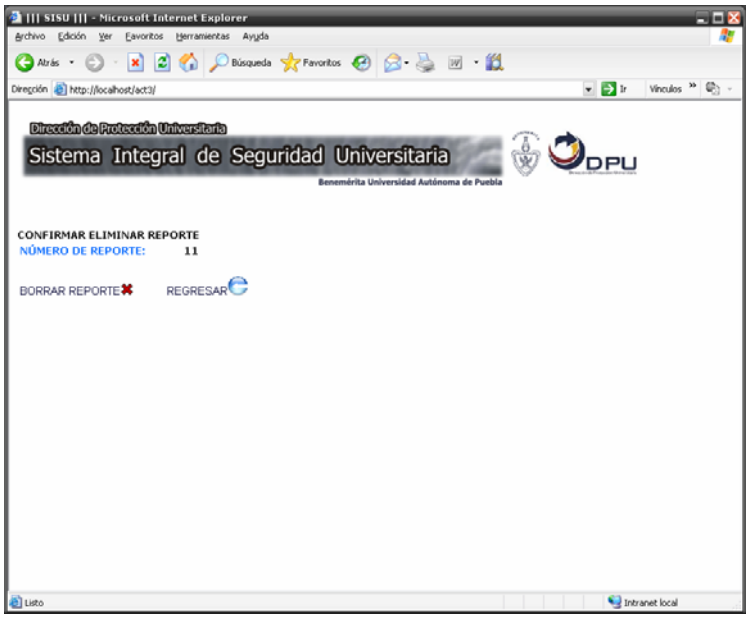


Figura 4.21.- Pantalla de confirmación para eliminar un reporte.

Si eliges la opción ELIMINAR el reporte será eliminado completamente de la base de datos.

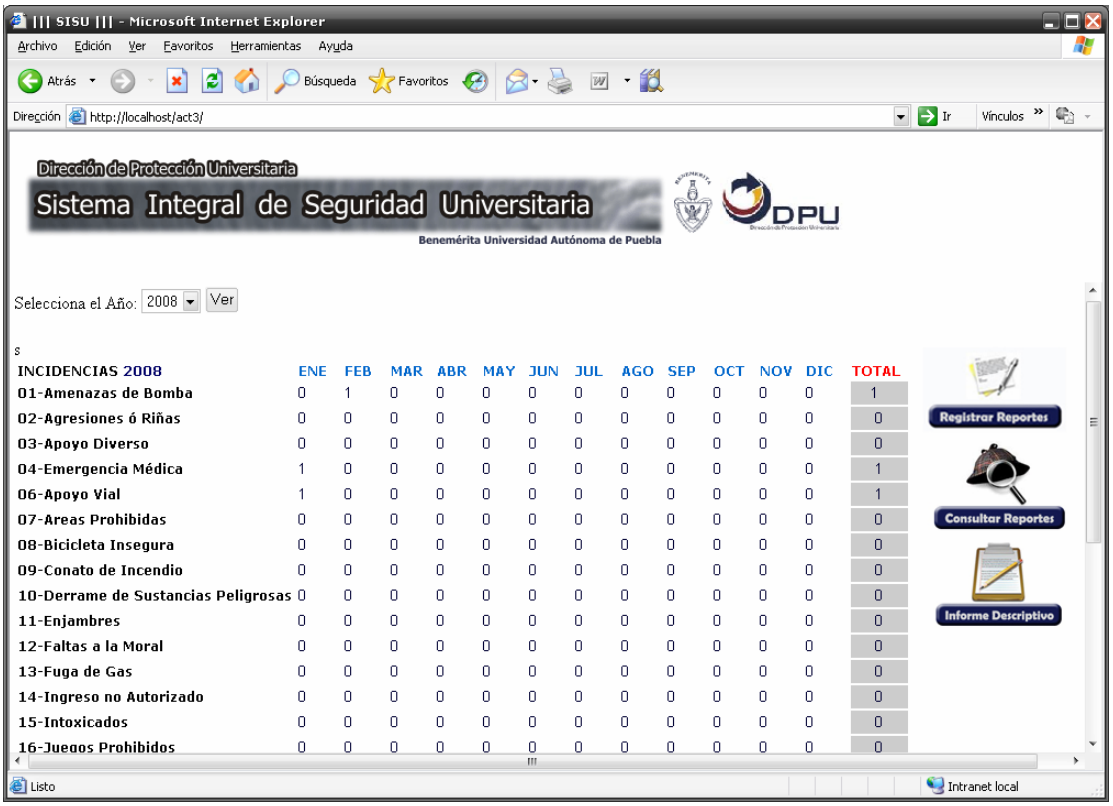
Si das clic en la opción REGRESAR , regresaras a la descripción del reporte seleccionado.

4.2.5.- Interfaz de hoja de informe general.

En el menú del sistema encontramos la opción **Hoja de Informe General** si accedemos a ella el sistema mostrara la siguiente pantalla **Figura 4.22,** en la Hoja de Informe General se podrá elegir el año que se desea verificar.

El la pantalla se muestra el número de incidencia y las veces que ocurrieron por mes, con esto logramos ver el comportamiento de las incidencias y al final de la tabla mostrando el total de las ocurrencias.

En esta hoja de Informe podremos seleccionar los números generados para ver la descripción de los reportes.



INCIDENCIAS 2008	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SEP	OCT	NOV	DIC	TOTAL
01-Amenazas de Bomba	0	1	0	0	0	0	0	0	0	0	0	0	1
02-Agresiones ó Riñas	0	0	0	0	0	0	0	0	0	0	0	0	0
03-Apoyo Diverso	0	0	0	0	0	0	0	0	0	0	0	0	0
04-Emergencia Médica	1	0	0	0	0	0	0	0	0	0	0	0	1
06-Apoyo Vial	1	0	0	0	0	0	0	0	0	0	0	0	1
07-Areas Prohibidas	0	0	0	0	0	0	0	0	0	0	0	0	0
08-Bicicleta Insegura	0	0	0	0	0	0	0	0	0	0	0	0	0
09-Conato de Incendio	0	0	0	0	0	0	0	0	0	0	0	0	0
10-Derrame de Sustancias Peligrosas	0	0	0	0	0	0	0	0	0	0	0	0	0
11-Enjambres	0	0	0	0	0	0	0	0	0	0	0	0	0
12-Faltas a la Moral	0	0	0	0	0	0	0	0	0	0	0	0	0
13-Fuga de Gas	0	0	0	0	0	0	0	0	0	0	0	0	0
14-Ingreso no Autorizado	0	0	0	0	0	0	0	0	0	0	0	0	0
15-Intoxicados	0	0	0	0	0	0	0	0	0	0	0	0	0
16-Juegos Prohibidos	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 4.22.-Pantalla de Informe General

4.2.6.- Interfaz del informe descriptivo.

En esta pantalla **Figura 4.23.-** el sistema mostrara una descripción por dependencias de la incidencia y año seleccionados por el usuario.

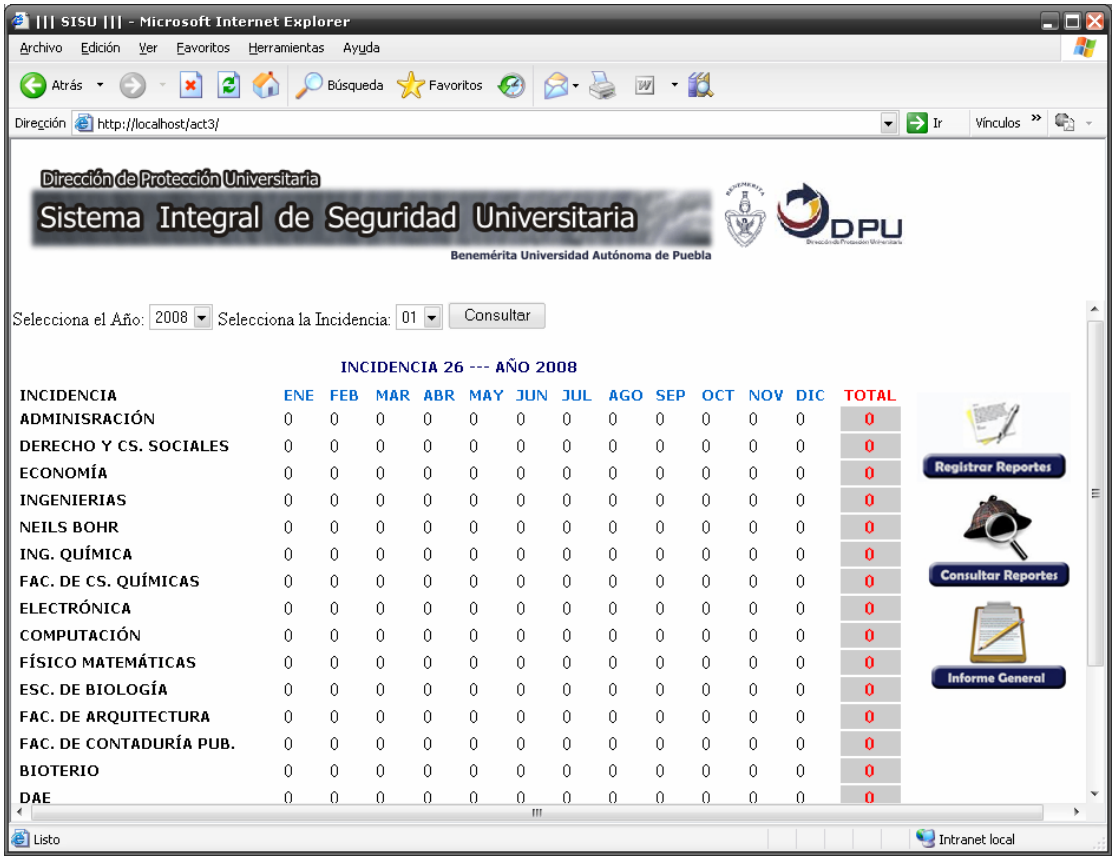


Figura 4.23.-Pantalla de informe descriptivo.

De esta manera hemos explicado las pantallas con las que deberán interactuar los usuarios.

CONCLUSIONES

El análisis, diseño e implementación del sistema cumplió con el objetivo general ya que ahora tenemos un sistema capaz de almacenar reportes de incidencias, el cual facilita el manejo de la información de los hechos ocurridos en la universidad.

La implementación Cliente /Servidor, se uso principalmente porque teníamos la necesidad de tener almacenados todos los reportes en una base de datos y que los usuarios desde sus computadoras con acceso a Internet pudieran registrar reportes logrando así que el proceso de registro de los reportes fuese mas rápido y eficiente.

Se decidió utilizar MySQL, PHP y APACHE, debido a que se puede instalar el sistema en distintas plataformas, es decir podremos instalarlo en distintos Sistemas Operativos, además de que forman parte del software libre la cual es una de las principales ventajas y proporcionan confiabilidad a las aplicaciones desarrolladas en un ambiente Web.

La metodología del Proceso Unificado de Desarrollo de Software, la elegí por basarse en casos de uso, además de ser iterativo e incremental y de fácil adaptación al proyecto realizado.

Para modelar el problema y construir correctamente la base de datos utilice el modelo de bases de datos relacional, ya que permite modelar problemas reales y administrar datos dinámicamente debido a su idea fundamental el uso de las relaciones.

LIMITACIONES

Una limitación es que debido a que se utilizara un servidor para almacenar el sistema en ocasiones puede haber indisponibilidad.

En la última fase de desarrollo el sistema fue sometido a pruebas de escritorio con la finalidad de evaluar su funcionalidad y usabilidad, el sistema no esta exento de errores para esto llevaremos un seguimiento del comportamiento del sistema con el fin de corregir los que se presenten.

PERSPECTIVAS

A continuación mencionaremos algunas de las perspectivas del sistema de bases de datos que pudiéramos implementar:

- Que el sistema sea capaz de generar reportes gráficos para mostrar historiales de las incidencias ocurridas.

- Poder integrar reportes del personal de Servicios Generales de la Universidad, ya que ellos llenan sus propios reportes y están involucrados con el Sistema Integral de Seguridad Universitaria pero no forman parte de la Dirección de Protección Universitaria y ellos son los que tienen el total acceso al Sistema de Registro de Incidencias.

- Debido a su crecimiento poder hacer una migración de Gestor de Base de datos como Oracle.

- Poder descentralizar la información para no tener indisponibilidades del servidor.

- Considerando la red de radio, red telefónica e Internet podemos fusionar arquitecturas con protocolos y algoritmos que nos faciliten una comunicación inmediata y rápida.

BIBLIOGRAFÍA

[1] Dirección de Sistemas, Universidad Nacional Autónoma de México, Ingeniería de Software http://www.sistemas.unam.mx/software.html

[2] Universidad Politécnica de Madrid, Gestión de Proyectos
http://www.getec.etsit.upm.es/docencia/gproyectos/planificacion/cvida.htm

[3] Grady Booch, Ivar Jacobson, James Rumbaugh. El Proceso Unificado de Desarrollo de Software. Addison Wesley, 2000.

[4] Jerarquía de Generalización
http://www.unalmed.edu.co/~mstabare/diseno_conceptual.htm

[5] Reglas de Integridad Referencial
http://www3.uji.es/~mmarques/f47/apun/node52.html

[6] A Relational Model of Data for Large Shared Data Banks Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377-387.
http://es.wikipedia.org/wiki/Normalizaci%C3%B3n_de_una_base_de_datos

[7] Vilariño ayala., Darnes. "Bases de Datos Cliente_Servidor". Diplomado en Tecnologías de la Información, Facultad de Ciencias de la Computación, BUAP 2007.

[8] G.W. Hansen, J.V. Hansen *Diseño y Administración de Bases de Datos*
Segunda Edición (1997) Prentice Hall
http://www3.uji.es/~mmarques/f47/apun/node69.html

[9] Silberschatz Korth Sudarshan, Fundamentos de Bases de Datos, McGraw Hill, 4ta. Edición, 2002, 787 pp.

[10] Tovar Vidal Mireya. "Desarrollo de Aplicaciones web.". Diplomado en Tecnologías de la Información, Facultad de Ciencias de la Computación, BUAP 2007.

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

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

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

[14] Richard York, CSS Practico, Anaya Multimedia, 2da Edición 2007, 400pp.