Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Sistema WEB para el Área de Mantenimiento Automotriz

TESIS

Para obtener el título de:
Licenciado en Ciencias de la Computación.

Presenta:

Juan Carlos Sánchez Barradas

Asesor:
María del Rocío Boone Rojas.

Noviembre 2009

Tabla de contenido

Introducción

Se propone el desarrollo de una aplicación cliente servidor de Base de Datos (BD) Relacionales basada en el Modelo Incremental de Ingeniería de Software. Usando Mysql como administrador de la Base de Datos y PHP como lenguaje de desarrollo.

Se pretende realizar una aplicación sobre internet para automatizar la administración de los servicios del área de mantenimiento automotriz de una agencia de autos, ya que actualmente esto se realiza de forma manual.

Se pretende que el sistema sea fácil de operar y confiable para el personal que lo maneje.

El presente reporte de trabajo de tesis se encuentra estructurado en cinco capítulos.

El capítulo uno, se presenta una introducción a los conceptos fundamentales para el desarrollo de aplicaciones de Bases de Datos y Web. En el segundo capítulo encontramos la descripción de los requerimientos, además de la definición de los alcance del sistema y se plantean los diagramas de flujos de datos. En el capítulo tres se establece el diagrama E-R a usar en este sistema, también se implementa la base de datos y se explica el proceso de normalización de las mismas. En el capítulo cuatro se encuentra una explicación sobre el lenguaje de programación PHP, el manejador de base de datos MYSQL y el servidor WEB REACTOR. Por último en el capítulo cinco se discuten las pruebas e implementación del sistema.

Capítulo 1

Conceptos Fundamentales de la Ingeniería del Software y las Bases de Datos

1.1 Ingeniería del Software

La ingeniería de software es una actividad para la solución de problemas. Se usan los modelos para buscar una solución aceptable. Esta búsqueda es conducida por la experimentación. Los ingenieros de software no tienen recursos infinitos, y están restringidos por presupuesto y tiempos de entrega. Dada la falta de una teoría fundamental con frecuencia tienen que apoyarse en métodos empíricos para evaluar los beneficios de alternativas diferentes.

La ingeniería de software es una actividad para la adquisición de conocimiento. En el modelado del dominio de la aplicación y la solución, el ingeniero de software recopila datos, los organiza en información y los formaliza en conocimiento. La adquisición de conocimiento no es lineal, ya que un sólo dato puede invalidar modelos completos.

La ingeniería de software es una actividad dirigida por una fundamentación. Cuando se adquiere conocimiento y se toman decisiones acerca del sistema o sus dominios de aplicación, los ingenieros de software también necesitan captar el contexto en que se tomaron las decisiones y las razones que hay tras las mismas. La información de la fundamentación, representada como un conjunto de modelos de problemas, permite que los ingenieros de software comprendan las implicaciones descubiertas de un cambio propuesto cuando revisan una decisión. (1)

El trabajo de la ingeniería de software se puede dividir en tres fases genéricas:

1.- La fase de definición que se centra en el **¿qué?**
En esta parte de definición se debe identificar ¿Qué información será procesada?, ¿Qué función y rendimiento se desea?, ¿Qué comportamiento del sistema?, ¿Qué interfaces van a ser establecidas y qué criterios de validación se necesitan para definir un sistema correcto?

En esta fase se deben identificar claramente los requisitos claves del sistema y del software. También se llevan a cabo tres tareas principales: **ingeniería de sistemas, planificación del proyecto de software y análisis de los requisitos**.

2.- La fase de desarrollo que se centra en el **¿cómo?**
En esta fase de desarrollo se definen como han de diseñarse las estructuras de datos, cómo han de implementarse la función dentro de una arquitectura de software, como han de implementarse los detalles procedimentales, como se implementara la interface y sobre todo como ha de traducirse el diseño en un lenguaje de programación y como han de realizarse las pruebas. Las tareas específicas en esta fase de desarrollo son: **diseño del software, generación de código y prueba del software**.

3.- La fase de mantenimiento se centra en el "**cambio"**.
Esta fase está directamente relacionada a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software y a cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente. En esta fase de mantenimiento se encuentran cuatro tipos de cambios: c**orrección, adaptación, mejora y prevención**. (2)

1.1.1 Modelos del ciclo de vida

El proceso de desarrollo de un producto software puede adoptar diferentes modelos, a continuación se presenta una descripción de los más significativos de acuerdo a Pressman [3] y Boone [2]

1.1.2 Modelo Lineal Secuencial:

Llamado algunas veces "Ciclo de vida básico ", fig. 1.1 este modelo secuencial sugiere un enfoque sistemático y secuencial para el desarrollo del software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento.

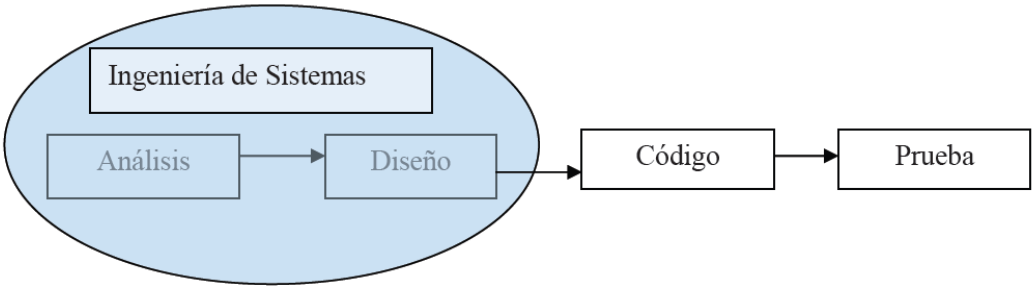

Fig. 1.1 Ciclo de vida básico.

Ingeniería de sistemas: el software siempre permanecerá a un sistema más grande (a una empresa) y el trabajo consiste en establecer todos los requisitos de todos los elementos del sistema ya que el software interconectará con otros elementos como lo son el hardware, personas, bases de datos y procesos. La ingeniería de sistemas abarca los requisitos que se recogen en el nivel de empresa estratégico y en el nivel del área de negocio.

Análisis: En esta etapa se analizan los requisitos del software, cuál será el dominio de información del software, así como la función requerida, comportamiento, rendimiento e interconexiones.

Diseño: En esta etapa del diseño del software se centra la estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental (algoritmos).

Generación de Código: Esta fase depende completamente de la fase de Diseño ya que si el diseño fue llevado a cabo de una forma detallada, la generación del código se realiza mecánicamente

Pruebas: Esta etapa consiste en probar el código generado y se centra en los procesos lógicos internos del software, asegurándose que las sentencias se han comprobado y básicamente asegurarse de que las entradas definidas producen los resultados reales de acuerdo con los resultados requeridos.

Mantenimiento: El software indudablemente sufrirá cambios después de ser entregado al cliente como mencionamos anteriormente se darán básicamente por los siguientes factores: Corrección, Adaptación, Mejora y Prevención.

Fracasos: Este Método como otros, se dice que llegan a fallar porque no son respetadas sus etapas y por lo tanto falla el proyecto, de acuerdo a Pressman (3). Dada la naturaleza secuencial de este modelo los cambios a medio proyecto en alguna de las fases regularmente causan demasiada confusión y atrasos.

Otros de los factores de fracaso de proyectos es que los clientes generalmente no exponen explícitamente todos los requisitos. Y por supuesto la secuencia lineal del modelo no deja ver errores hasta el final cuando el cliente revisa el programa (4).

1.1.3 Modelo evolutivo o de prototipos.

Él modelo evolutivo se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado figuras 1.2. más que tener actividades separadas de especificación, desarrollo y validación, estas se llevan a cabo concurrentemente y tienen retroalimentación rápida a lo largo del proceso.

Existen dos tipos de desarrollo evolutivo:

1.-Desarrollo exploratorio.
El objetivo del proceso es trabajar con el cliente para explorar sus requerimientos y entregar un sistema final. El desarrollo empieza con las partes del sistema que se comprende mejor. El sistema evoluciona agregando nuevos atributos acordes con las propuestas del cliente.

2.-Prototipos desechables.
El objetivo del proceso de desarrollo evolutivo es comprender los requerimientos del cliente y entonces desarrollar una definición mejorada de los requerimientos para el sistema. El prototipo se centra en experimentar con aquellas partes de los requerimientos del cliente que no se comprenden del todo.

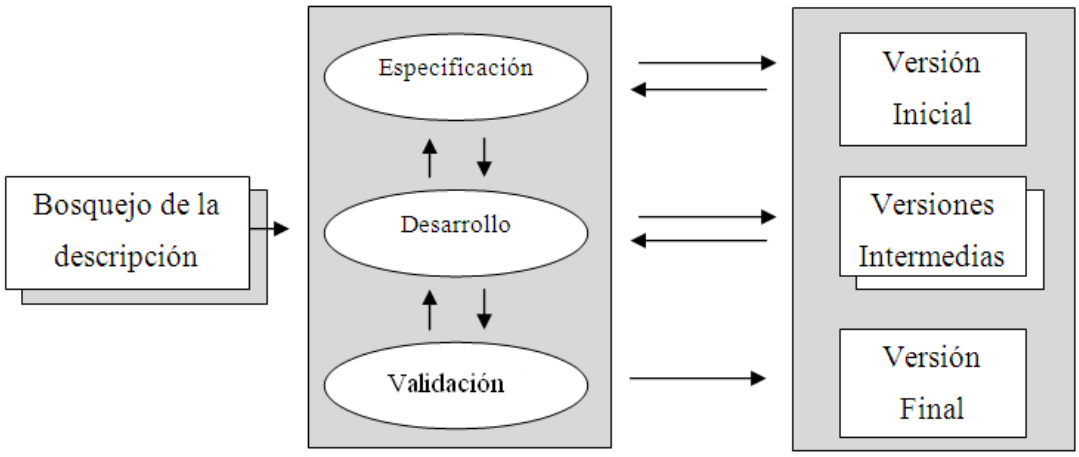

Fig. 1.2 Desarrollo evolutivo.

A menudo la producción de sistemas, un enfoque evolutivo para el desarrollo de software es más efectivo que el enfoque de cascada ya que cumple con las necesidades inmediatas de los clientes. La ventaja de un proceso de software que se basa en un enfoque evolutivo es que la especificación se puede desarrollar de forma creciente. Tan pronto como los usuarios desarrollen un mejor entendimiento de su problema, éste se reflejará en el sistema de software. Sin embargo, desde una perspectiva de ingeniería y administración, hay tres problemas:

1.- El proceso no es visible.
Los administradores tienen que hacer entregas regulares para medir el progreso. Si los sistemas se desarrollan rápidamente, es muy costoso producir documentos que reflejen cada versión del sistema.

2.- A menudo los sistemas tienen una estructura deficiente.
Los cambios continuos tienden a corromper las estructuras del software. Incorporar cambios en él se convierte en una tarea difícil y costosa.

3.- Se requieren herramientas y técnicas especiales.
Éstas permiten un desarrollo rápido pero son incompatibles con otras herramientas o técnicas y relativamente pocas personas tienen las habilidades necesarias para utilizar (5).

1.1.4 Modelo incremental

Los modelos evolutivos son iterativos. Se caracterizan por la forma en que permiten desarrollar versiones cada vez más completas del software.

El modelo incremental combina elementos del modelo lineal secuencial (aplicados repetidamente) con la filosofía interactiva de construcción de prototipos. El modelo incremental entrega el software en partes pequeñas, pero utilizables, llamadas “incrementos”.

La diferencia entre el modelo de prototipos y el modelo del proceso incremental se da en que el modelo incremental se centra en la entrega de un producto operacional con cada incremento, los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

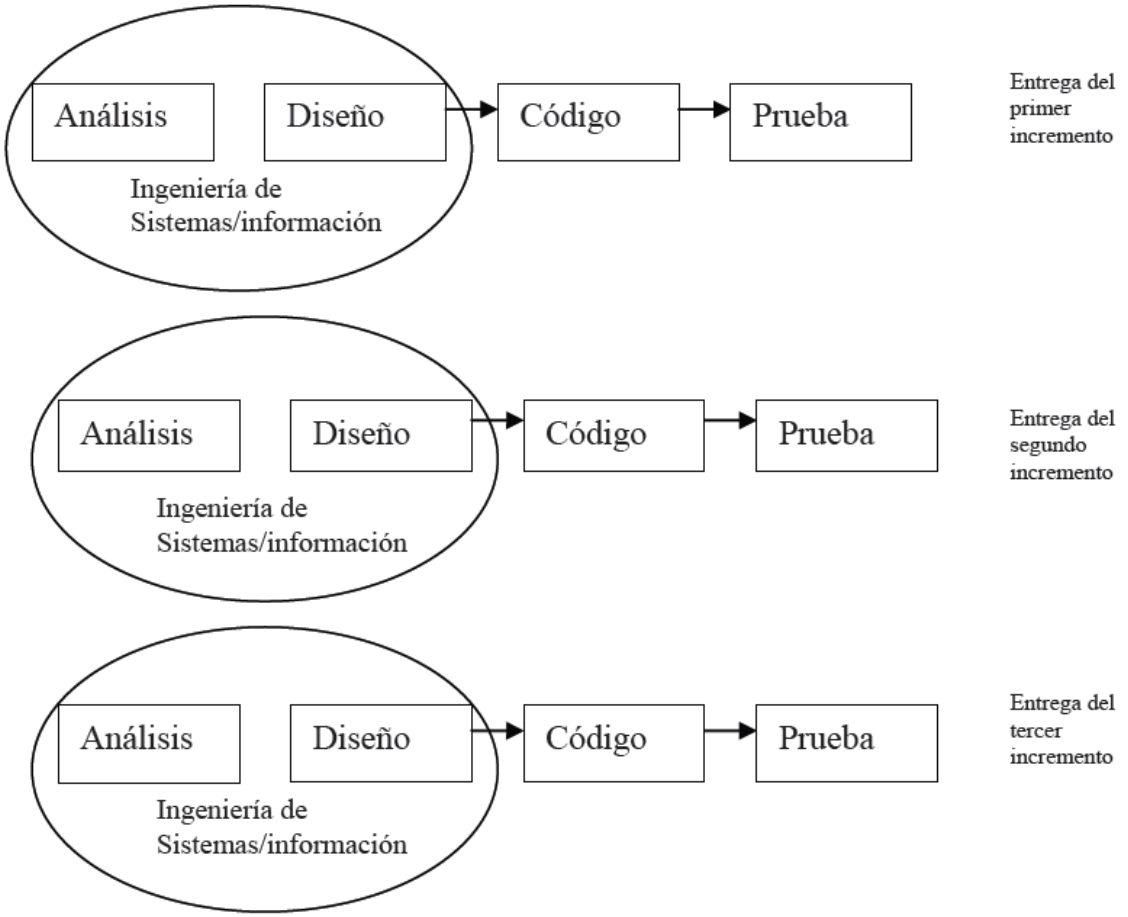

Fig. 1.3 Desarrollo incremental.

El número de incrementos que pueda llegar a tener un sistema dependerá de las fechas de entrega del proyecto, del tamaño del proyecto y de la cantidad de módulos en la que pueda ser dividido el sistema.

1.1.5 Modelo de Espiral:

El Modelo en Espiral es un modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial.

Las prioridades de este método iniciando de dentro hacia fuera son:

1. Proyecto de desarrollo de conceptos.
2. Proyecto de desarrollo de nuevos productos.
3. Proyecto de mejora de productos.
4. Proyecto de mantenimiento de productos.

En el modelo espiral, el software se desarrolla en una serie de versiones incrementales.

Durante las primeras versiones la versión puede ser un prototipo o bien un modelo en papel y durante las últimas iteraciones las versiones suelen ser cada vez más completas, hasta tener el sistema diseñado.

Este modelo se divide en un número de actividades que suelen ser 6 regiones de tareas:

Comunicación con el cliente: Las tareas requeridas para establecer comunicación entre el desarrollador y el cliente.
Planificación: las tareas requeridas para definir recursos, el tiempo y otra información relacionadas con el proyecto.

Análisis de riesgos: Las tareas requeridas para evaluar riesgos técnicos y de gestión
Ingeniería: las tareas requeridas para construir una o más representaciones de la aplicación.

Construcción y acción: Las tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario (por ejemplo: documentación y práctica).
Evaluación del cliente: las tareas requeridas para obtener la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería e implementada durante la etapa de instalación.

Cuando inicia este proceso evolutivo, el equipo de ingeniería del software gira alrededor de la espiral en la dirección de las agujas del reloj, comenzando por el centro. El primer circuito de la espiral puede producir el desarrollo de una especificación de productos; los pasos siguientes en la espiral se podrían utilizar para desarrollar un prototipo y progresivamente versiones más sofisticadas del software. Cada paso por la región de planificación produce ajustes en el plano del proyecto. El coste y la planificación se ajustan con la realimentación ante la evaluación del cliente.

Al igual que los demás métodos si no es detectado y gestionado algún riesgo importante a tiempo se van a tener serios problemas, este modelo aún no ha sido probado tanto como los modelos lineales y de prototipos así que aún no se puede asegurar con toda afectividad su eficacia, de acuerdo a Pressman (3).

1.1.6 Diseño de Sistemas

De acuerdo a Pressman (3). El diseño es una representación significativa de ingeniería de algo que se va a construir. En el contexto de la ingeniería del software el diseño se centra en cuatro áreas importantes: datos, arquitectura, interfaces y componentes.

Las metodologías de diseño del software carecen de la profundidad, flexibilidad y naturaleza cuantitativa que se asocian normalmente a las disciplinas de diseño de ingeniería más clásicas.

El diseño del software se encuentra en el núcleo técnico de la ingeniería del software y se aplica independientemente del método del diseño de software que se utilice. Una vez que se analizan y especifican los requisitos del software, el diseño del software es la primera de las tres actividades técnicas – diseño generación de código y pruebas – que se requieren para construir y verificar el software.

Los requisitos del software, manifestados por los modelos de datos funcionales y de comportamiento, alimentan la tarea del diseño. El diseño da por resultado el diseño de datos, un Diseño Arquitectónico, un diseño de interfaz y un diseño de componentes.

Diseño de datos: Este diseño surge directamente de la información que se crea durante el análisis en las estructuras de datos que se necesitarán para implementar el software. Los objetos de datos y las relaciones definidas en el diagrama entidad-relación y el contenido de datos detallado que se presenta en el diccionario de datos proporcionan la base para el diseño de datos. El diseño de datos aparece con más detalle a medida que se van diseñando cada uno de los componentes del software.

Diseño arquitectónico: Define la relación entre los elementos estructurales principales del software, los patrones de diseño que se pueden utilizar para lograr los requisitos que se han definido para el sistema y las restricciones que afectan a la manera en que se pueden aplicar los patrones de diseño arquitectónicos.

La representación del diseño arquitectónico - el marco de trabajo de un sistema basado en computadora – puede derivarse de la especificación del sistema, del modelo de análisis y de la interacción del subsistema definido dentro del modelo de análisis.

Diseño de Interfaz: Describe la manera de comunicarse el software dentro de sí mismo, con los sistemas que ínteroperan dentro de él y con las personas que lo utilizan. Una interfaz implica un flujo de información (por ejemplo: datos y/o control) y un tipo especifico de comportamiento.

El Diseño a nivel de componentes transforma los elementos estructurales de la arquitectura del software en una descripción procedimental de los componentes del software. La información que se obtiene de la Especificación de Procesos, de la Especificación de control y de la descripción de objetos de datos sirven como base para el diseño de los componentes (4).

1.2 Bases de Datos

1.2.1 Sistema Gestor de Base de Datos SGBD

Los sistemas de gestión de base de datos (SGBD); (en inglés: Database management System, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de datos.

Existen distintos objetivos que deben cumplir los SGBD:

Abstracción de la información. Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

Independencia. La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

Redundancia mínima. Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

Consistencia. En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

Seguridad. La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentra segura frente a usuarios malintencionados, que intenten leer información privilegiada; frente a ataques que deseen manipular o destruir la información; o simplemente ante las torpezas de algún usuario autorizado pero despistado. Normalmente, los SGBD disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

Integridad. Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados, o cualquier otra circunstancia capaz de corromper la información almacenada.

Respaldo y recuperación. Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

Control de la concurrencia. En la mayoría de entornos (excepto quizás el doméstico), lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información, bien para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. Así pues, un SGBD debe controlar este acceso concurrente a la información, que podría derivar en inconsistencias.

Tiempo de respuesta. Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en darnos la información solicitada y en almacenar los cambios realizados.

Ventajas:

1. Facilidad de manejo de grandes volúmenes de información.
2. Gran velocidad de ejecución de las consultas.
3. Independencia del tratamiento de información.
4. Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.
5. No hay duplicidad de información, comprobación de información en el momento de introducir la misma.
6. Integridad referencial el terminar los registros.

Inconvenientes:

1. El costo de actualización del hardware y software son muy elevados.
2. El Costo (salario o remuneración) del administrador de la base de datos es grande.
3. El mal diseño de esta puede originar problemas a futuro.
4. Un mal adiestramiento a los usuarios puede originar problemas a futuro.
5. Si no se encuentra un manual del sistema no se podrán hacer relaciones con facilidad.
6. Generan campos vacíos en exceso.
7. El mal diseño de seguridad genera problemas en la base de datos (6).

1.2.2 Modelos de Bases de Datos

Una de las características fundamentales de los sistemas de bases de datos es que proporcionan cierto nivel de abstracción de datos, al ocultar las características sobre el almacenamiento físico que la mayoría de usuarios no necesita conocer. Los modelos de datos son el instrumento principal para ofrecer dicha abstracción. Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos: los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos. Los modelos de datos contienen también un conjunto de operaciones básicas para la realización de consultas (lecturas) y actualizaciones de datos. Además, los modelos de datos más modernos incluyen conceptos para especificar comportamiento, permitiendo especificar un conjunto de operaciones definidas por el usuario.

Los modelos de datos se pueden clasificar dependiendo de los tipos de conceptos que ofrecen para describir la estructura de la base de datos. Los modelos de datos de alto nivel, o modelos conceptuales, disponen de conceptos muy cercanos al modo en que la mayoría de los usuarios percibe los datos, mientras que los modelos de datos de bajo nivel, o modelos físicos, proporcionan conceptos que describen los detalles de cómo se almacenan los datos en la computadora. Los conceptos de los modelos físicos están dirigidos al personal informático, no a los usuarios finales. Entre estos dos extremos se encuentran los modelos lógicos, cuyos conceptos pueden ser entendidos por los usuarios finales, aunque no están demasiado alejados de la forma en que los datos se organizan físicamente. Los modelos lógicos ocultan algunos detalles de cómo se almacenan los datos, pero pueden implementarse de manera directa en una computadora.

Los modelos conceptuales utilizan conceptos como entidades, atributos y relaciones. Una entidad representa un objeto o concepto del mundo real como, por ejemplo, un empleado de la empresa inmobiliaria o una oficina. Un atributo representa alguna propiedad de interés de una entidad como, por ejemplo, el nombre o el salario del empleado. Una relación describe una interacción entre dos o más entidades, por ejemplo, la relación de trabajo entre un empleado y su oficina.
Cada SGBD soporta un modelo lógico, siendo los más comunes el relacional, el de red y el jerárquico. Estos modelos representan los datos valiéndose de estructuras de registros, por lo que también se denominan modelos orientados a registros. Hay una nueva familia de modelos lógicos, son los modelos orientados a objetos, que están más próximos a los modelos conceptuales.

Los modelos físicos describen cómo se almacenan los datos en la computadora: el formato de los registros, la estructura de los ficheros (desordenados, ordenados, etc.) y los métodos de acceso utilizados (índices, etc.).

A la descripción de una base de datos mediante un modelo de datos se le denomina esquema de la base de datos. Este esquema se especifica durante el diseño, y no es de esperar que se modifique a menudo. Sin embargo, los datos que se almacenan en la base de datos pueden cambiar con mucha frecuencia: se insertan datos, se actualizan, etc. Los datos que la base de datos contiene en un determinado momento se denominan estado de la base de datos u ocurrencia de la base de datos.

La distinción entre el esquema y el estado de la base de datos es muy importante. Cuando definimos una nueva base de datos, sólo especificamos su esquema al SGBD. En ese momento, el estado de la base de datos es el "estado vacío", sin datos. Cuando se cargan datos por primera vez, la base datos pasa al "estado inicial". De ahí en adelante, siempre que se realice una operación de actualización de la base de datos, se tendrá un nuevo estado. El SGBD se encarga, en parte, de garantizar que todos los estados de la base de datos sean estados válidos que satisfagan la estructura y las restricciones especificadas en el esquema. Por lo tanto, es muy importante que el esquema que se especifique al SGBD sea correcto y se debe tener muchísimo cuidado al diseñarlo. El SGBD almacena el esquema en su catálogo o diccionario de datos, de modo que se pueda consultar siempre que sea necesario (7).

1.2.3 Modelo E/R Entidad Relación

De acuerdo a Silberschatz (8). El modelo de datos Entidad- Relación está basado en una percepción del mundo real consistente en objetos básicos llamados ENTIDADES y de RELACIONES entre objetos. Esto permite la especificación de un esquema de la empresa que representa la estructura lógica completa de una base de datos. El modelo de E/R es útil para hacer corresponder los significados e interacciones de las empresas del mundo real con un esquema conceptual.

Existen tres nociones básicas que emplea el modelo de datos E/R:

1. Conjunto de Entidades
2. Conjunto de relaciones
3. Atributos.

Conjunto de Entidades: Una entidad es una “cosa” u “objeto” en el mundo real que es distinguible de los demás objetos. Por ejemplo cada persona en un desarrollo es una entidad. Una entidad tiene un conjunto de propiedades, y los valores para algún conjunto de propiedades pueden identificar una entidad de forma unívoca. Por ejemplo en una empresa el número de identificación único, identifica a una persona de manera unívocamente.

Una entidad puede ser concreta, como persona o un libro, o puede ser abstracta, como un préstamo, unas vacaciones o un concepto.

Conjunto de Entidades es un conjunto de entidades del mismo tipo que comparten las mismas propiedades, o atributos. El conjunto de todas las personas que son clientes en un banco dado, por ejemplo, se pueden definir como el conjunto de entidades cliente.

Análogamente, el conjunto de todos los préstamos podría representar el conjunto de todos los préstamos concedidos.

Atributos: Una entidad se representa mediante un conjunto de atributos. Los atributos describen propiedades que posee cada miembro de un conjunto de entidades. La designación de un atributo para un conjunto de entidades expresa que la base de datos almacena información similar concerniente a cada entidad del conjunto de entidades; sin embargo, cada entidad puede tener su propio valor para cada atributo.

Posibles atributos del conjunto de entidades para el cliente son el Id-cliente, nombre-cliente, calle-cliente y ciudad-cliente y demás datos como código postal, provincia etc. Los posibles atributos para la entidad Préstamo son número-préstamo e importe.

Cada entidad tiene un valor para cada uno de sus atributos. Por ejemplo una entidad cliente en concreto puede tener el valor 54235 para el Id-cliente, el valor de Torres para nombre-cliente, y Maya para calle-cliente.

Para cada atributo hay un conjunto de valores permitidos, llamados el dominio, o el conjunto de valores, de ese atributo. El dominio del atributo nombre-cliente podría ser el conjunto de todas las cadenas de texto de una cierta longitud.

Así se puede ver que existe una integración del esquema con el desarrollo real de la empresa que se está modelando. Los valores de los atributos que describen una entidad constituirán una porción significante de los datos almacenados en la base de datos.
Un atributo en el modelo E/R se puede caracterizar por lo siguientes tipos de atributos:

Atributos simples y compuestos. Los atributos simples son los descritos anteriormente es decir no están divididos en subpartes. Los atributos compuestos son aquellos que se pueden dividir en subpartes por ejemplo Nombre-cliente podría estar estructurado en Nombre, primer-apellido y segundo-apellido.

Atributos monovalorados y multivalorados: Los atributos monovalorados tienen un solo valor para una entidad concreta, por ejemplo el atributo número-préstamo para una entidad préstamo especifico, referencia a un único número de préstamo.

Los atributos multivalorados son aquellos que tienen un conjunto de valores para una entidad específica, por ejemplo para un conjunto de entidades empleado con el atributo número-teléfono. Los empleados pueden tener desde cero o más números de teléfono.

Atributos Derivados. El valor para este tipo de atributos se puede derivar de los valores de otros atributos o entidades relacionadas. Por ejemplo sea la entidad "cliente" que tiene un atributo "préstamos" que representa cuántos préstamos tiene un cliente en el banco. Ese atributo se puede derivar contando el número de entidades préstamo asociadas con ese cliente. El valor de un atributo derivado no se almacena, sino que se calcula cuando sea necesario.

Atributo Nulo: Un atributo toma un valor nulo cuando una entidad no tiene un valor para atributo. El valor nulo también puede indicar "no aplicable" es decir que el valor no existe para la entidad. Por ejemplo una persona puede no tener segundo nombre de pila.

Conjunto de Relaciones:
Una "relación" es una asociación entre diferentes entidades. Por ejemplo, se puede definir una relación que asocie al cliente López con el préstamo P-15. Esta relación especifica que López es un cliente con el préstamo número P-15.

La asociación entre conjuntos de entidades se conoce como participación; es decir conjunto de entidades participan en un conjunto de relaciones.

La correspondencia de cardinalidades apropiada para un conjunto de relaciones particular depende obviamente de la situación del mundo real que el conjunto de relaciones modela.

La estructura lógica general de una base de datos se puede expresar gráficamente mediante un diagrama E/R, que consta de los siguientes componentes:

1. **Rectángulos**: que representan conjuntos de entidades
2. **Elipses**: que representan atributos.
3. **Rombos**: que representan relaciones entre conjuntos de entidades.
4. **Líneas**: que unen los atributos con los conjuntos de entidades o los conjuntos de entidades con las relaciones.

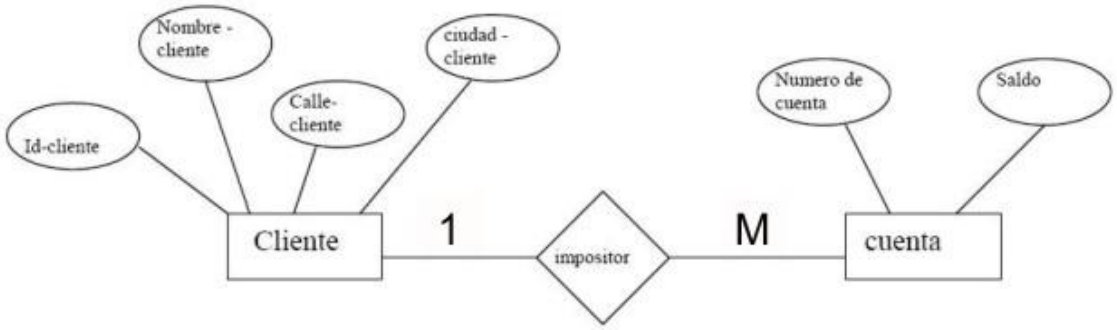

Fig. 1.4 Ejemplo de diagrama Entidad / Relación.

1.2.4 Teoría de la Normalización

De acuerdo a Date (9). La teoría de la normalización tiene como fundamento el concepto de formas normales. Se dice que una relación está en una determinada forma si satisface un cierto conjunto de restricciones.

Las formas normales garantizan que muchos de los problemas de redundancia y de anomalías en una base de datos, no ocurran.

De acuerdo a Jeffrey (10) se da el nombre de anomalías a los problemas como el de la redundancia que se presenta cuando intentamos acumular demasiada información en una relación individual. Los principales tipos de anomalías que nos encontramos en la práctica son:

• Redundancia: La información se repite innecesariamente en varias tuplas.

- Anomalías de actualización: se puede cambiar la información en una tupla y dejarla inalterada en otra.
- Anomalías de eliminación: Si un conjunto de valores queda vacío, podemos perder información adicional como efecto secundario.

Primera Forma Normal (1FN)
Una relación R está en primera forma normal (1FN) si y sólo si satisface la restricción de que sus dominios simples subyacentes contengan sólo valores atómicos.

Esta definición sólo declara que cualquier relación normalizada está en 1FN, lo cual desde luego es correcto. Una relación que sólo está en primera forma normal (o sea, una relación en 1FN que no está también en 2FN y por lo tanto tampoco en 3FN) tiene una estructura indeseable por varias razones.

Segunda Forma Normal (2FN)
Una relación está en segunda forma normal (2FN) si y sólo si está en 1FN y todos los atributos no clave dependen por completo de la clave primaria.

Tercera Forma Normal (3FN)
Una relación está en tercera forma normal (3FN) si y sólo si está en 2FN y todos los atributos no clave dependen de manera no transitiva de la clave primaria.

Una dependencia transitoria es aquella en la cual sus atributos no –llave son dependientes de otros atributos no-llave.

1.2.5 Modelo Relacional

El **modelo relacional** para la gestión de una base de datos es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos (11).

Su idea fundamental es el uso de «relaciones». Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados «tuplas». Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar, esto es, pensando en cada **relación** como si fuese una **tabla** que está compuesta por *registros* (cada fila de la tabla sería un registro), que representarían las *tuplas*, y *campos* (las columnas de una tabla).

En este modelo todos los datos son almacenados en relaciones, y como cada relación es un conjunto de datos, el orden en el que estos se almacenen no tiene mayor relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar por un usuario no experto. La información puede ser recuperada o almacenada por medio de «consultas» que ofrecen una amplia flexibilidad y poder para administrar la información.

Este modelo considera la base de datos como una colección de relaciones. De manera simple, una relación representa una tabla que no es más que un conjunto de filas, cada fila es un conjunto de campos y cada campo representa un valor que interpretado describe el mundo real. Cada fila también se puede denominar tupla o registro y a cada columna también se le puede llamar campo o atributo.

Para manipular la información utilizamos un lenguaje relacional, actualmente se cuenta con dos lenguajes formales el Álgebra relacional y el Cálculo relacional. El Álgebra relacional permite describir la forma de realizar una consulta, en cambio, el Cálculo relacional sólo indica lo que se desea devolver.

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

Un esquema, es la definición de una estructura (generalmente relaciones o tablas de una base de datos), es decir, determina la identidad de la relación y qué tipo de información podrá ser almacenada dentro de ella; en otras palabras, el esquema son los **metadatos** de la relación. Todo esquema constará de:

1. Nombre de la relación (su identificador).
2. Nombre de los atributos (o campos) de la relación y sus dominios, el dominio de un atributo o campo define los valores permitidos para el mismo, es equivalente al tipo de dato por ejemplo *character*, *integer*, *date*, *string*, etc.

Una instancia de manera formal es la aplicación de un esquema a un conjunto finito de datos. En palabras no tan técnicas, se puede definir como el contenido de una tabla en un momento dado, pero también es valido referirnos a una instancia cuando trabajamos o mostramos únicamente un subconjunto de la información contenida en una relación o tabla, como por ejemplo:

1. Ciertos caracteres y números (una sola columna de una sola fila).
2. Algunas o todas las filas con todas o algunas columnas
3. Cada fila es una tupla. El número de filas es llamado *cardinalidad*.
4. El número de columnas es llamado *aridad o grado* (11).

1.3 Aplicaciones de bases de datos y WEB

1.3.1 El Modelo Cliente – Servidor

Esta arquitectura consiste básicamente en que un programa “el cliente” que 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 (13).

1.3.2 Base de Datos y Web

En la actualidad, muchas instituciones se han dado cuenta de la importancia que el Web tiene en el desarrollo de sus potencialidades, ya que con ello pueden lograr una mejor comunicación con personas o instituciones situadas en cualquier lugar del mundo.

Gracias a la conexión con la red mundial Internet, poco a poco, cada individuo o institución va teniendo acceso a mayor cantidad de información de las diversas ramas de la ciencia con distintos formatos de almacenamiento.

La mayor parte de información es presentada de forma estática a través de documentos HTML, lo cual limita el acceso a los distintos tipos de almacenamiento en que ésta pueda encontrarse.

Pero, en la actualidad surge la posibilidad de utilizar aplicaciones que permitan acceder a información de forma dinámica, tal como a bases de datos, con contenidos y formatos muy diversos.

Una de las ventajas de utilizar el Web para este fin, es que no hay restricciones en el sistema operativo que se debe usar, permitiendo la conexión entre si, de las páginas Web desplegadas en un browser del Web que funciona en una plataforma, con servidores de bases de datos alojados en otra plataforma. Además, no hay necesidad de cambiar el formato o estructura de la información dentro de la bases de datos (14).

Capítulo 2

Análisis del Sistema y Especificación

2.1 Planteamiento del Problema

Se pretende realizar una aplicación sobre internet para automatizar la administración de los servicios del área de mantenimiento automotriz de una agencia de autos ya que actualmente esto se realiza de forma manual.

El sistema deberá permitir la concertación de citas y de información relacionada con el estado del servicio para un auto, además debe apoyar los servicios relacionados con el área de mantenimiento automotriz de la agencia, tales como, la asignación de zonas de trabajo y personal, también deberá de mantener la información relacionada con cada servicio, tal como, diagnóstico, presupuesto, tiempo estimado de entrega, pólizas de seguro y además llevará un historial de servicios al cliente.

El sistema contará con los siguientes módulos: servicios y precios, información del personal, información de clientes, zonas de trabajo, información del auto.

2.2 Especificación de Requerimientos

El sistema deberá contar con los siguientes requerimientos:

Concertación de citas
En esta parte el sistema deberá ofrecer al usuario la posibilidad de crear citas y actualizar o eliminar alguna previamente capturada.

Información relacionada con el estado del vehículo
El sistema tendrá la información centralizada, esto es, que en cada etapa de mantenimiento del auto, se va capturando el avance que se tenga sobre los servicios que fueron solicitados por el cliente.

Servicios relacionados con el área de mantenimiento
En esta parte del sistema, se debe tener la opción de dar de alta, actualizar o eliminar servicios con sus respectivos precios para que posteriormente sean ofertados al cliente.

Dichos servicios deberán contar con un tiempo estimado de realización.

Asignación de zonas de trabajo
En la agencia de autos existen zonas de trabajo en donde se realizan los servicios que fueron solicitados por el cliente, en el sistema se debe considerar la opción para poder dar de alta, actualizar o eliminar zonas de trabajo, para que puedan ser contempladas en el seguimiento que se tendrá sobre el estatus de los servicios que se van a realizar a un auto.

Asignación de personal
El sistema contempla tres tipos de usuarios administrador, empleado y cliente.

El sistema tomará datos del empleado o empleados que estarán a cargo del mantenimiento al auto, para que ésta información pueda ser agregada a la orden de servicio.

Diagnóstico
El diagnóstico es un servicio que nos permitirá generar un presupuesto, el sistema deberá poder dar de alta, actualizar o eliminar diagnósticos, para que posteriormente puedan ser utilizados en la orden de servicio.

Presupuesto
El presupuesto será generado con base en los servicios que el cliente requiera para su auto, o dependiendo de los servicios que el diagnóstico nos arroje.

También el sistema deberá tener la opción de generar, actualizar ó eliminar presupuestos.

Tiempo estimado de entrega
De acuerdo a los servicios que el auto requiera, el sistema dará un estimado del tiempo requerido para los servicios autorizados por el cliente.

Historial de servicios al auto
El sistema guardará un historial de los servicios que se hayan realizado, esto para llevar un control sobre el manteamiento para un auto y a su vez tener referencia ante posibles fallas futuras.

Información del personal
El sistema debe tener la opción de dar de alta, actualizar ó eliminar datos de los empleados que laboren en el departamento de mantenimiento, también poder ingresar los servicios que podría cubrir cada uno de los empleados.

Información del auto
El sistema guardará los datos del auto, tales como número de matrícula, color, marca, tipo, modelo, tipo de combustible, transmisión, tipo de dirección y datos del propietario.

Orden de trabajo (ingreso)
La orden de trabajo contendrá el concentrado de todo lo que conformará el mantenimiento del auto, desde los datos del auto, personal asignado, servicios solicitados por el cliente, área de trabajo asignada y los datos del propietario.

Esta orden se estará actualizando constantemente por el personal que está realizando el mantenimiento, para que el propietario pueda tener el estatus correcto de su auto en el momento que él lo desee.

2.3 Alcances

Se propone el desarrollo de una aplicación cliente servidor de Base de Datos (BD) Relacionales basada en el Modelo Incremental de Ingeniería de Software. Usando Mysql como administrador de la Base de Datos y PHP como lenguaje de desarrollo.

Se pretende realizar una aplicación sobre internet para automatizar la administración de los servicios del área de mantenimiento automotriz de una agencia de auto ya que actualmente esto se realiza de forma manual.

El sistema deberá permitir la concertación de citas y de información relacionada con el estado del servicio para un auto, además debe apoyar los servicios relacionados con el área de servicios de mantenimiento automotriz de la Agencia, tales como, la asignación de zonas de trabajo y personal, también deberá mantener la información relacionada con cada servicio, tal como, diagnóstico, presupuesto, tiempo estimado de entrega, pólizas de seguro y además llevará un historial de servicios al cliente.

El sistema contará con los siguientes módulos: servicios y precios, información del personal, información de clientes, zonas de trabajo e información del auto.

Se busca que el sistema sea fácil de operar y confiable para el personal que lo maneje.

2.4 Descripción de la Información

Caso de uso: Sistema en General	
Actor: Usuario	
Curso Normal	Alternativo
1.- El usuario accede al sistema 2.- El sistema muestra la pantalla de opciones	

Caso de uso: Acceso al sistema	
Actor: Usuario	
Curso Normal	Alternativo
1.- Se valida el acceso al sistema 2.- Se muestran las 4 diferentes opciones que tiene el sistema • Dar de alta • Consultar • Dar de baja • Actualizar 3.- Se muestran en la pantalla del usuario	

Caso de uso: Validar Usuario	
Actor: Usuario	
Curso Normal	Alternativo
1.- Se dan los datos de login y password	

2.- Se verifica que los datos existan en la base de datos 3.- Se accede al sistema 4.- Se muestra en la pantalla del usuario	Si no se encuentran los datos en la base de datos, se muestra un mensaje de error

Caso de uso: Dar de alta	
Actor: Usuario	
Curso Normal	Alternativo
1.- Se valida el acceso al sistema 2.- Se ingresan datos personales del empleado 3.- Se ingresan datos de habilidades del empleado 4.- Se muestra en la pantalla del usuario	Si hubo algún problemas se tendrán que volver a dar los datos Si hubo algún problemas se tendrán que volver a dar los datos

Caso de uso: Ingresar datos	
Actor: Usuario	
Curso Normal	Alternativo
1.- Se valida el acceso al sistema 2.- Se leen los datos a ingresar 3.- Se validan los datos a ingresar 4.- Se ingresan a la base de datos 5.- Se avisa que los datos fueron insertados correctamente 6.- Se muestra en la pantalla del Usuario	Si hubo algún problema se tendrán que volver a escribir los datos Se avisa que los datos no fueron ingresados, y se vuelven a ingresar

Caso de uso: Consultar Datos	
Actor: Usuario	
Curso Normal	Alternativo
1.- Se valida el acceso al sistema 2.- Se solicitan los datos a consultar 3.- Se consultan los datos solicitados en la Base de Datos 4.- Se ingresan a la base de datos 5.- Se muestra en la pantalla del Usuario	Se avisa que los datos no fueron consultados

Caso de uso: Eliminar Datos	
Actor: Usuario	
Curso Normal	Alternativo
1.- Se valida el acceso al sistema 2.- Se muestran los datos 3.- Se seleccionan datos a consultar 4.- Se eliminan los datos de la base de datos 5.- Se avisa que los datos fueron eliminados correctamente 5.- Se muestra en la pantalla del Usuario	Se avisa que los datos no fueron eliminados

Caso de uso: Actualizar Datos	
Actor: Usuario	
Curso Normal	Alternativo
1.- Se valida el acceso al sistema 2.- Se muestran los datos 3.- Se seleccionan datos para actualizar 4.- Se actualizan los datos de la base de datos 5.- Se avisa que los datos fueron actualizados 5.- Se muestra en la pantalla del Usuario	Se avisa que los datos no fueron actualizados

2.5 Casos de Uso

Los casos de uso son una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico (15).

2.6 Diagramas de flujo de datos

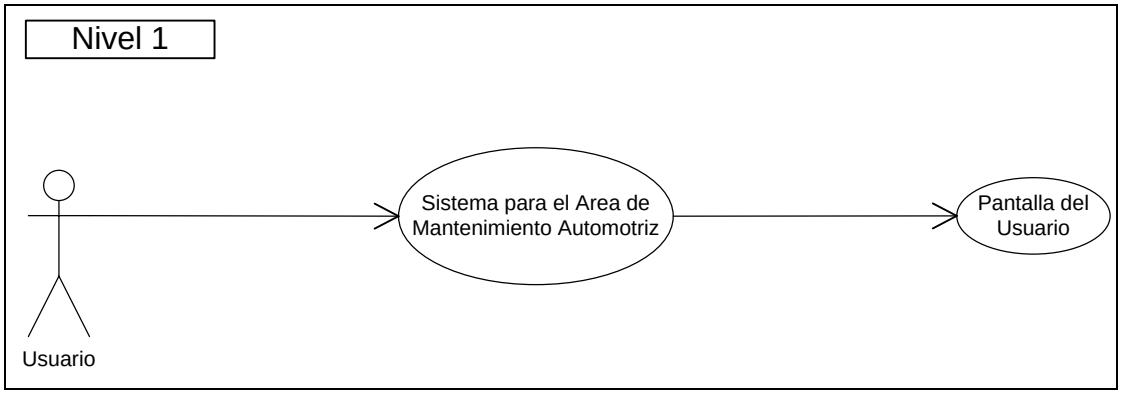

FIGURA 2.1 Sistema en General

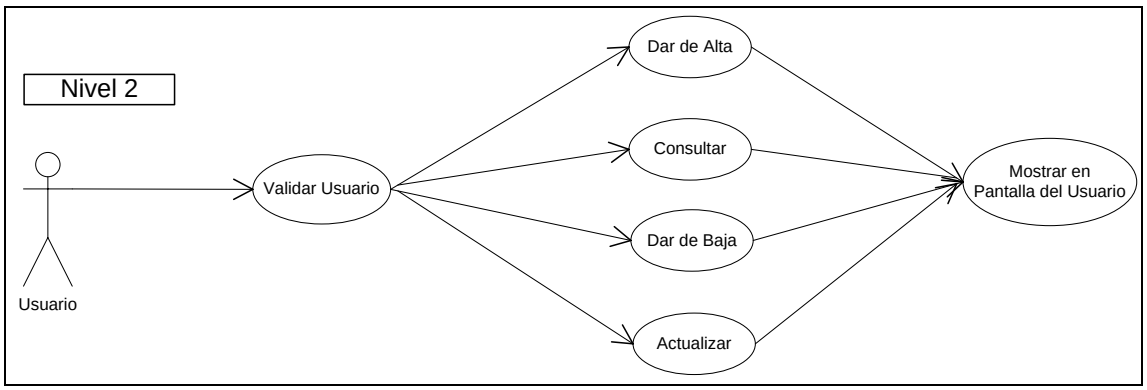

FIGURA 2.2 Acceso al sistema

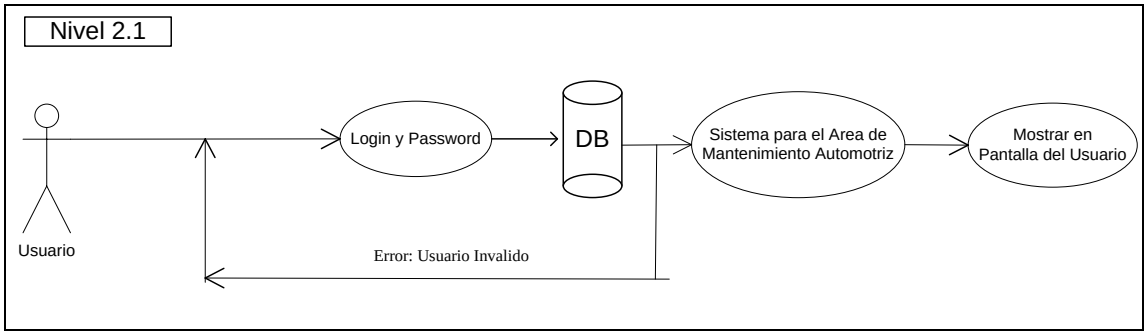

FIGURA 2.3 Validar Usuario

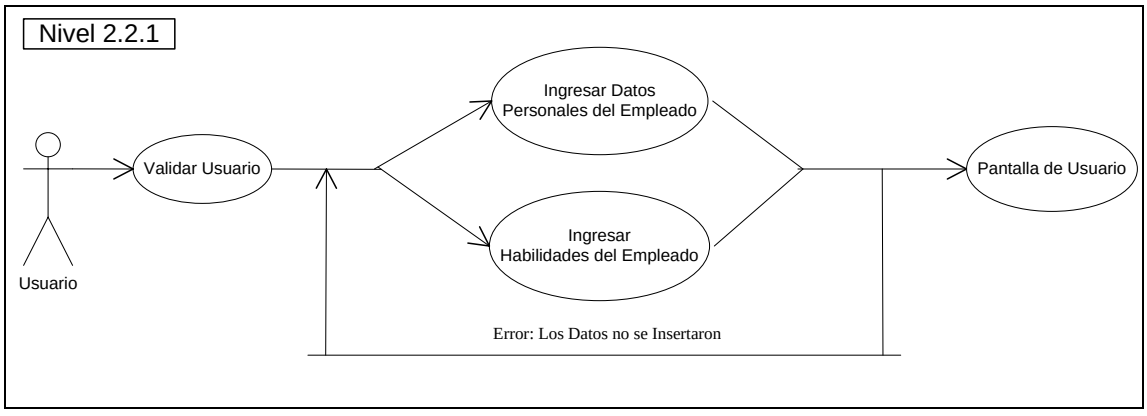

FIGURA 2.4 Dar de alta

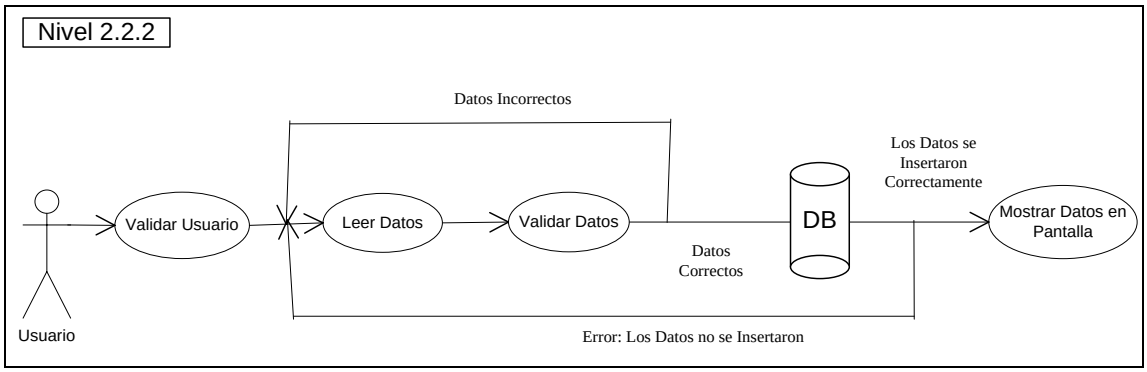

FIGURA 2.5 Ingresar datos

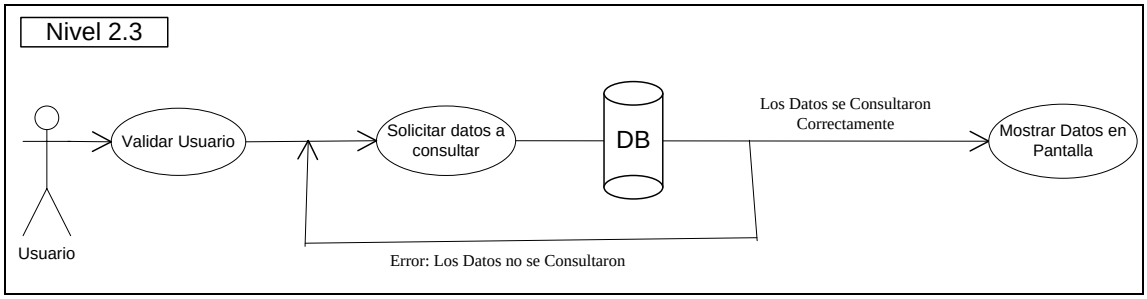

FIGURA 2.6 Consultar Datos

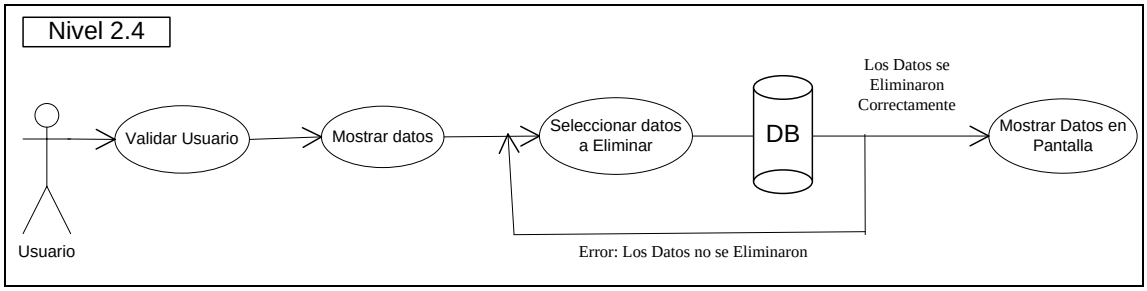

FIGURA 2.7 Eliminar Datos

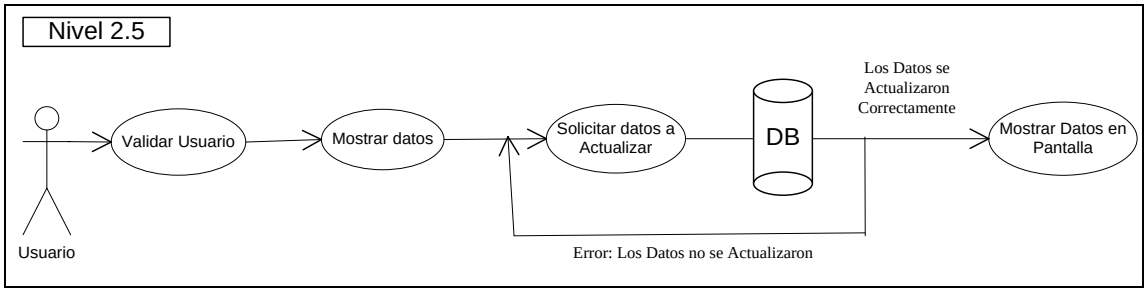

FIGURA 2.8 Actualizar Datos

Capítulo 3

Diseño de la Base de Datos

3.1 Modelo Entidad Relación

El modelo de datos más extendido es el denominado entidad-relación. En el modelo E/R se parte de una situación real a partir de la cual se definen entidades y relaciones entre dichas entidades.

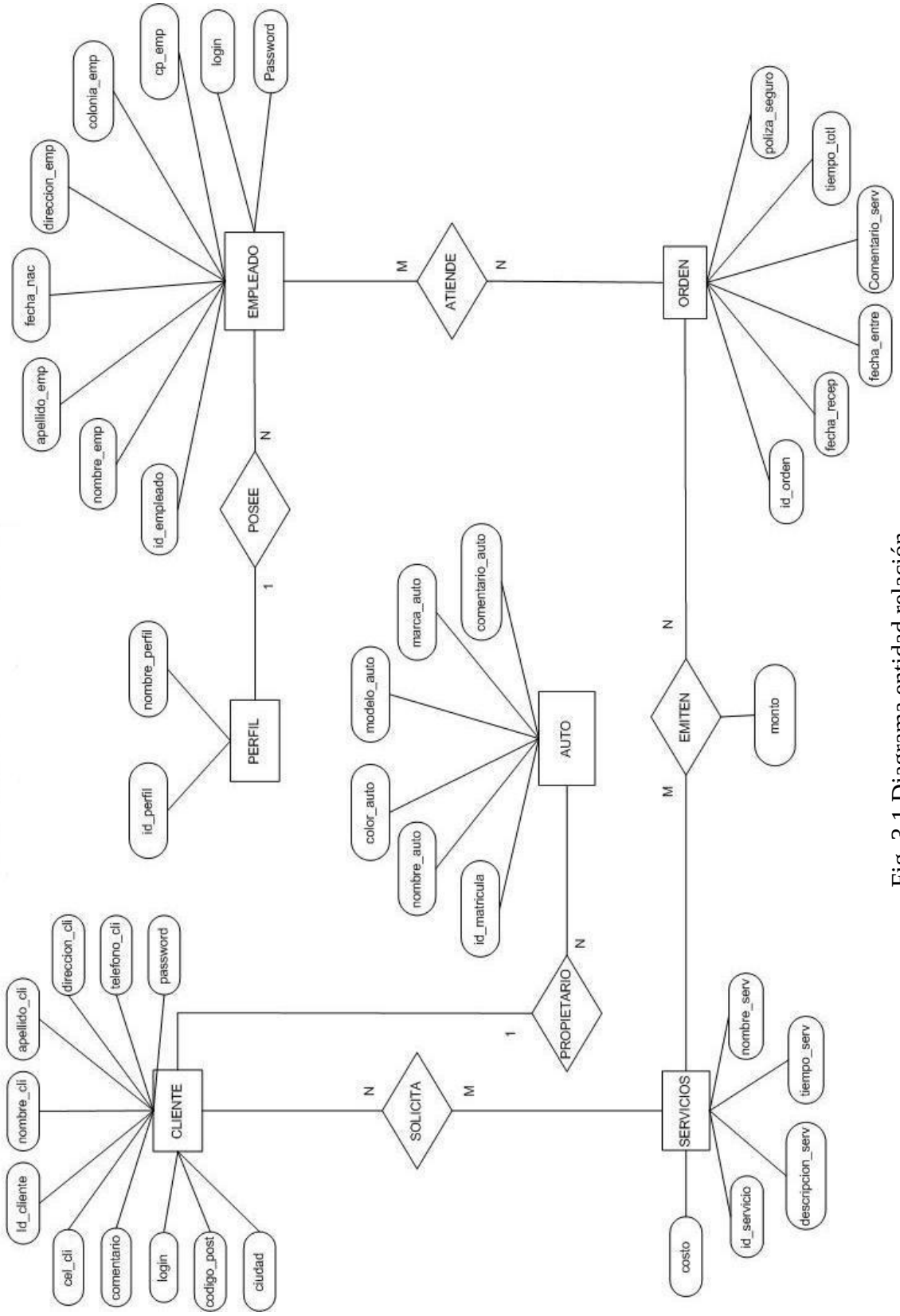

Fig. 3.1 Diagrama entidad relación.

3.2 Diccionario de Datos

Una vez que obtenemos el diagrama Entidad-Relación e implementamos la base de datos, podremos crear el diccionario de datos que contendrá.

Atiende

Campo	Tipo	Nulo	Univalorado	Comentarios
id_empleado	Entero	No	Si	Identificador del empleado
id_orden	Entero	No	Si	Identificador de la orden

Auto

Campo	Tipo	Nulo	Univalorado	Comentarios
id_matricula	Caracter	No	Si	Identificador del auto
nombre_auto	Alfanumérico	Sí	Si	Nombre del modelo del coche
color_auto	Alfanumérico	Sí	Si	Color del auto
modelo_auto	Alfanumérico	Sí	Si	Año del auto
marca_auto	Alfanumérico	Sí	Si	Marca del auto
comentario_auto	Alfanumérico	Sí	Si	Comentario sobre el auto
id_cliente	Entero	Sí	Si	Identificador del cliente

Cliente

Campo	Tipo	Nulo	Univalorado	Comentaros
id_cliente	Entero	No	Si	Identificador del cliente
nombre_cli	Alfanumérico	Sí	Si	Nombre del cliente
apellido_cli	Alfanumérico	Sí	Si	Apellido del cliente
direccion_cli	Alfanumérico	Sí	Si	Dirección del cliente
telefono_cli	Alfanumérico	Sí	Si	Teléfono del cliente
cel_cli	Alfanumérico	Sí	Si	Celular del cliente
Comentario	Alfanumérico	Sí	Si	Comentario sobre el cliente
codigo_pos	Alfanumérico	Sí	Si	Código postal del cliente
Ciudad	Alfanumérico	Sí	Si	Ciudad donde vive el cliente
Colonia	Alfanumérico	Sí	Si	Colonia donde vive el cliente

Emiten

Campo	Tipo	Nulo	Univalorado	Comentarios
id_servicio	Entero	No	Si	Identificador del servicio
id_orden	Entero	No	Si	Identificador de la orden

Empleado

Campo	Tipo	Nulo	Univalorado	Comentarios
id_empleado	Entero	No	Si	Identificador del empleado
nombre_emp	Alfanumérico	Sí	Si	Nombre del empleado
apellido_emp	Alfanumérico	Sí	Si	Apellido del empleado

fecha_nac	Datetime	Sí	Si	Fecha de nacimiento del empleado
direccion_emp	Alfanumérico	Sí	Si	Dirección del empleado
colonia_emp	Alfanumérico	Sí	Si	Colonia donde vive el empleado
cp_emp	Alfanumérico	Sí	Si	Código postal del empleado
id_perfil	Entero	Sí	Si	Identificador del perfil
Login	Alfanumérico	Sí	Si	Usuario del sistema
Password	Alfanumérico	Sí	Si	Contraseña del usuario

Orden

Campo	Tipo	Nulo	Univalorado	Comentarios
id_orden	Entero	No	Si	Identificador de la orden
fecha_recep	Datetime	Sí	Si	Fecha de recepción de la orden
fecha_entre	Datetime	Sí	Si	Fecha de entrega de la orden
comentario_serv	Alfanumérico	Sí	Si	Comentario sobre el servicio
tiempo_tot	Entero	Sí	Si	Tiempo aproximado de la orden
poliza_seg	Alfanumérico	Sí	Si	Numero de póliza de seguro

Perfil

Campo	Tipo	Nulo	Univalorado	Comentarios
id_perfil	Entero	No	Si	Identificador del perfil
nombre_per	Alfanumérico	Sí	Si	Nombre del perfil

Servicio

Campo	Tipo	Nulo	Univalorado	Comentarios
id_servicio	Entero	No	Si	Identificador del servicio
nombre_serv	Alfanumérico	Sí	Si	Nombre del servicio
descripcion_serv	Alfanumérico	Sí	Si	Descripción del servicio
tiempo_serv	Alfanumérico	Sí	Si	Tiempo aproximado del servicio
Costo	Float	Sí	Si	Costo de cada servicio

Solicita

Campo	Tipo	Nulo	Univalorado	Comentarios
id_cliente	Entero	No	Si	Identificador del cliente
id_servicio	Entero	No	Si	Identificador del servicio

3.3 Tablas Relacionales

CLIENTE (id_cliente, nombre_cli, apellido_cli, dirección_cli, teléfono_cli, cel_cli, comentario, login, password, código_post, ciudad)

SERVICIO (id_servicio, nombre_serv, descripcion_serv, tiempo_serv, costo)

SOLICITA (id_cliente, id_servicio)

AUTO (id_matricula, nombre_auto, color_auto, modelo_auto, marca_auto, comentario_auto, id_cliente)

ORDEN (id_orden, fecha_recep, fecha_entre, comentario_serv, tiempo_tot, poliza_seguro)

EMITEN (id_servicio, id_orden, monto)

EMPLEADO (id_empleado, nombre_emp, apellido_emp, fecha_nac, dirección_emp, colonia_emp, cp_emp, login, password, id_perfil)

ATIENDE (id_empleado, id_orden)

PERFIL (id_perfil, nombre_perfil)

3.4 Diagrama de Base de datos

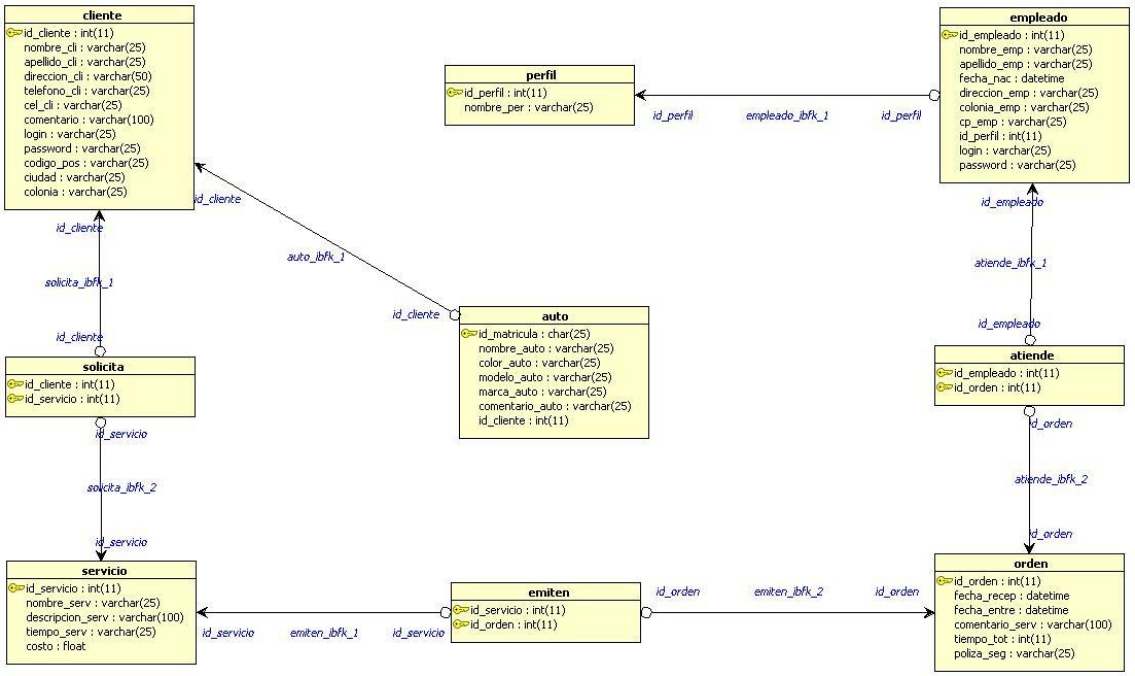

Fig. 3.2 Diagrama de base de datos.

3.5 Normalización de la base de datos

En esta sección justificaremos la normalización de las tablas de la base de datos:

Tabla Cliente. Esta tabla está en primera forma normal, puesto que no se repite algún atributo dentro de ella, como la forma normal lo dice, cada elemento es atómico. Por otra parte, tampoco tiene múltiples valores cada uno de los atributos. Con esta justificación se da por hecho que está en primera forma normal.

Analizando la tabla también está en segunda forma normal ya que además de estar en primera forma normal, todos sus atributos dependen de la llave primaria, es decir no hay algún otro atributo que dependa de alguna otra llave, pues no tiene alguna otra llave foránea.

También la tabla no tiene dependencias transitivas, por esa razón, se deduce que también está en tercera forma normal, y por consiguiente está tabla está normalizada.

Tabla Solicita. Como sólo son dos llaves foráneas, no llevan más atributos que se puedan repetir. Tampoco dependen de alguna otra llave, y mucho menos hay dependencias transitivas, por esas razones esta en primera, segunda y tercera forma normal.

Tabla Servicio. De igual manera esta tabla cumple con la primera forma normal pues ningún campo se repite, cada dato es atómico, como lo dice la regla.

También está en segunda forma normal, puesto que sus atributos dependen de la llave principal.

Como todos los atributos dependen de la llave primaria, no puede haber ninguna dependencia transitiva, por esta razón también está en tercera forma normal y de esta manera la tabla queda normalizada.

Tabla Emiten. Como sólo son dos llaves foráneas, no llevan más atributos que se puedan repetir. Tampoco dependen de alguna otra llave, y mucho menos hay dependencias transitivas, por esas razones está en primera, segunda y tercera forma normal.

Tabla Orden. En esta tabla también como sólo son dos llaves foráneas, no llevan más atributos que se puedan repetir. Tampoco dependen de alguna otra llave, y mucho menos hay dependencias transitivas, por esas razones esta en primera, segunda y tercera forma normal.

Tabla Atiende. La siguiente tabla es muy parecida a las anteriores sólo son dos llaves foráneas, no llevan más atributos que se puedan repetir. Tampoco dependen de alguna otra llave, y mucho menos hay dependencias transitivas, por esas razones está en primera, segunda y tercera forma normal.

Tabla Empleado. Esta tabla está en primera forma normal, porque no se repite algún atributo dentro de ella.

También está en segunda forma normal ya que además de estar en primera forma normal, todos sus atributos dependen de la llave primaria, es decir no hay algún otro atributo que dependa de alguna otra llave.

La tabla no tiene dependencias transitivas, por esa razón, se deduce que está en tercera forma normal, y por consiguiente esta tabla está normalizada.

Tabla Perfil. Esta tabla no tiene múltiples valores en los atributos ni datos repetidos asi que está en primera forma normal, también está en segunda forma normal porque sus atributos dependen de la llave primaria, y no tiene dependencias transitivas, así que está tabla también está normalizada.

Tabla Auto. Por último la tabla auto no tiene atributos repetidos, cuenta con una llave primaria y además no tiene dependencias transitivas, entonces por estas razones se dice que la tabla está normalizada.

Capítulo 4

Implantación del Sistema

4.1 MySQL

MySQL será el manejador de base de datos que utilizaremos para la implementación de nuestras tablas ya que es sencillo de usar e increíblemente rápido.

También es uno de los motores de base de datos más usados en internet, la principal razón de esto es que es gratuito para aplicaciones no comerciales.
Las características principales de MySQL son:

- **Es un gestor de base de datos.** Una base de datos es un conjunto de datos y un gestor de base de datos es una aplicación capaz de manejar este conjunto de datos de manera eficiente y cómoda.
- **Es una base de datos relacional.** Una base de datos relacional es un conjunto de datos que están almacenados en tablas entre las cuales se establecen unas relaciones para manejar los datos de una forma eficiente y segura. Para gestionar una base de datos relacional se usa el lenguaje estándar de programación SQL.
- **Es Open Source.** El código fuente de MySQL se puede descargar y está accesible a cualquiera, por otra parte, usa la licencia GPL para aplicaciones comerciales.
- **Es una base de datos muy rápida.** Segura y fácil de usar. Gracias a la colaboración de muchos usuarios, la base de datos se ha ido mejorando optimizándose en velocidad. Por eso es una base de datos más usada en internet.
- **Existe una gran cantidad de software que la utiliza.**

4.2 PHP

PHP será el lenguaje de programación que utilizaremos para el desarrollo del sistema, el cual nos permite controlar de forma segura y confiable el acceso a la base de datos.

Es el heredero de un producto anterior, llamado PHP/FI. PHP/FI fue creado por Rasmus Lerdorf en 1995, inicialmente como un simple conjunto de scripts de PERL para controlar los accesos de su trabajo online. Llamo a ese conjunto de scripts "Personal Home Page Tools". Según se requiera más funcionalidad, Rasmus eligió liberar el código fuente de PHP/FI para que cualquiera pudiese utilizarlo, así como arreglar errores y mejorar el código.

PHP permite embeber pequeños fragmentos de código dentro de la página HTML y realizar determinadas acciones de una forma fácil y eficaz sin tener que generar programas íntegramente en un lenguaje distinto al HTML. Por otra parte, y es aquí donde reside su mayor interés con respecto a los lenguajes pensados para los CGI, PHP ofrece un sinfín de funciones para la explotación de base de datos de una manera llana, sin complicaciones.

Como funcionalidades primordiales en PHP se consideran:

- 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 base de datos: el lenguaje PHP ofrece interfaces para el acceso a la mayoría de 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, envió de lotes de imágenes y funciones de graficado.

Usos de PHP
Los principales usos del PHP son los siguientes:

- Programación de páginas Web dinámicas, habitualmente en combinación con el motor de base de datos de MySQL, aunque cuenta con el soporte nativo para otros motores, incluyendo el estándar ODBC, lo que amplía en gran medida sus posibilidades de conexión.
- Programación en consola, al estilo de Perl o Shell scripting.
- Creación de aplicaciones gráficas independientes del navegador, por medio de la combinación de PHP y GTK (GIMP Tool Kit), lo que permite desarrollar aplicaciones de escritorio en los sistemas operativos en los que está soportado.

Ventajas de PHP
- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destacada su conectividad con MySQL
- Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Permite crear los fomularios para la Web.
- Biblioteca nativa de funciones sumamente amplia e incluida
- No requiere definición de tipos de variables ni manejo detallado de bajo nivel (12).

4.3 REACTOR

Reactor es un paquete completo de software que sirve para instalar y configurar un servidor Web Apache, contiene el lenguaje de programación PHP y el servidor de base de datos MySQL.

Estas son las tres herramientas principales que necesitamos para la implementación de nuestro sistema.

Capítulo 5

Pruebas del sistema

5.1 Página de Validación

Esta es la página en la cual el usuario dependiendo del perfil que tenga, deberá ingresar sus datos de usuario y contraseña para poder ingresar al sistema, si los datos que ingrese son incorrectos su acceso no será permitido.

Los perfiles que se manejan son los siguientes:

- Administrador
- Empleado
- Cliente

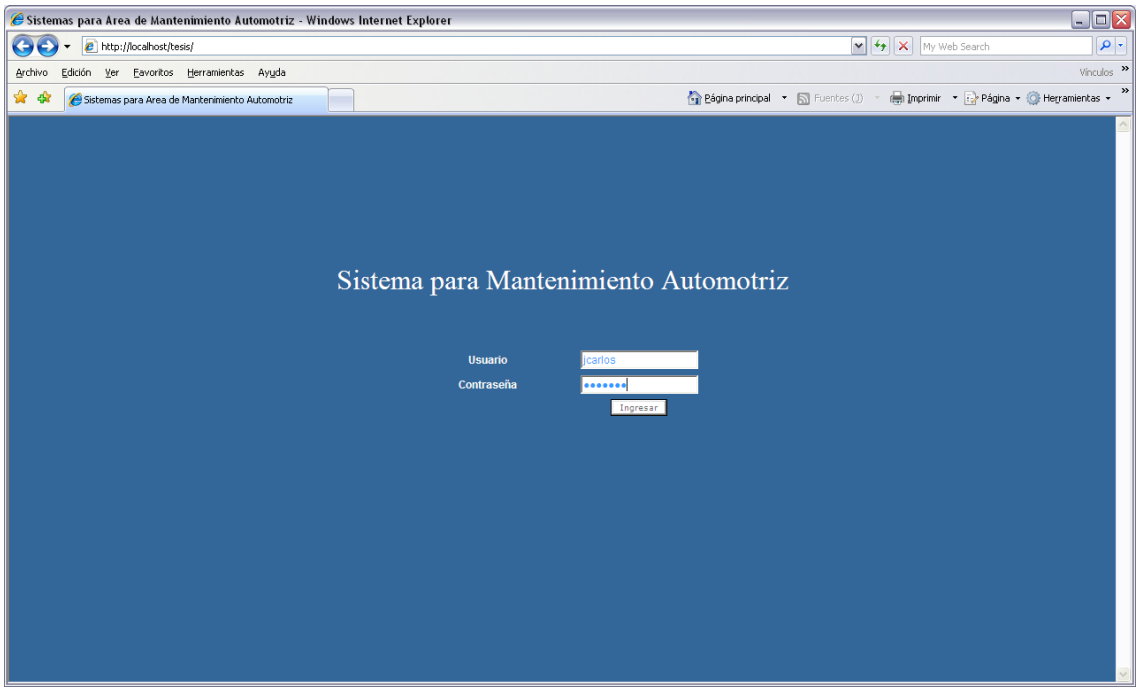

Figura 5.1 Ventana de Administrador.

5.2 Perfil de administrador

El administrador es aquella persona que tiene los permisos necesarios para poder dar de alta nuevos empleados en la base de datos, puede también dar de alta nuevos servicios, clientes y de igual forma que cualquier empleado puede generar órdenes de trabajo.

Figura 5.2 Ventana de opciones del Administrador.

En la figura 5.3 el administrador debe ingresar los datos de su personal, tales como el nombre del empleado, apellido, fecha de nacimiento, dirección, colonia, código postal, usuario y contraseña para su futuro acceso al sistema.

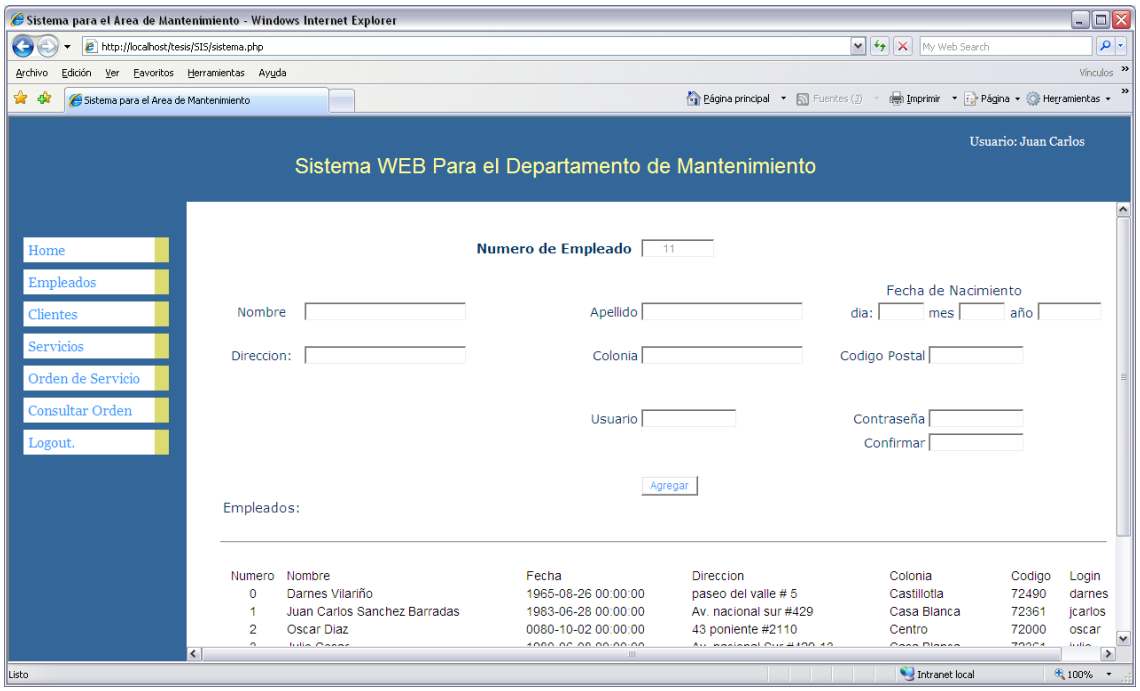

Figura 5.3 Ventana de creación de empleados.

En la figura 5.4 podemos dar de alta los datos de los clientes, esto es para evitar tomar los datos de un mismo cliente varias veces, los datos que guardamos es el nombre del cliente, apellido, fecha de nacimiento, dirección, Colonia, Ciudad, Teléfono, Celular, Código postal, Usuario, Contraseña y alguna observación.

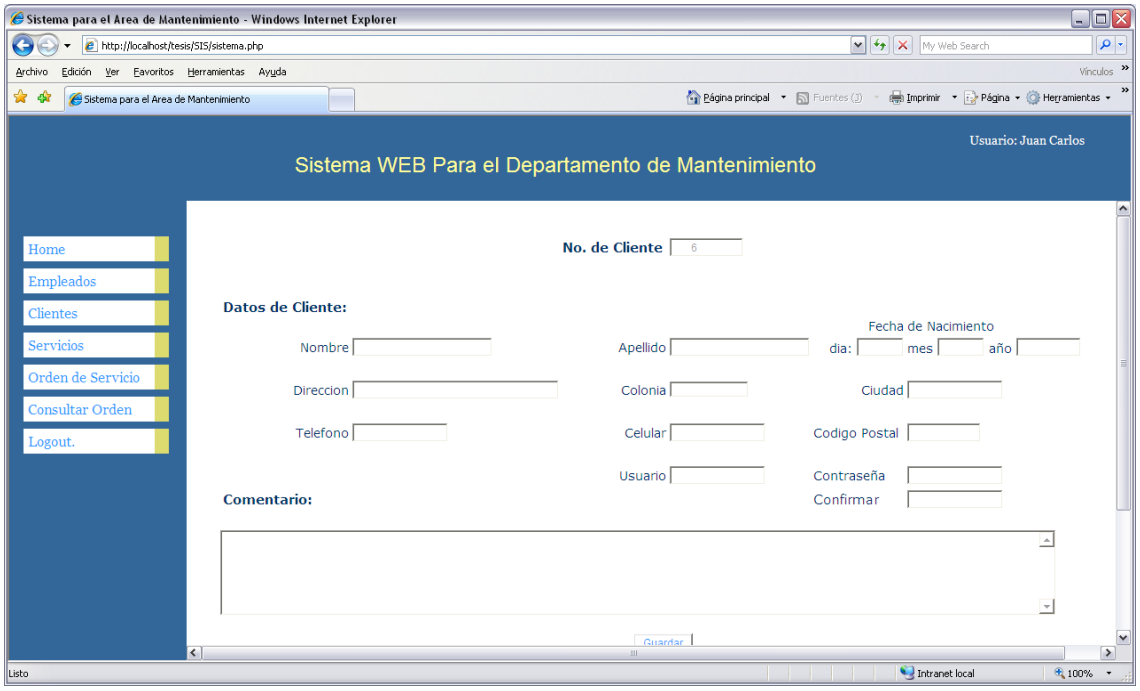

Figura 5.4 Ventana de creación de clientes.

En la figura 5.5 sólo el administrador puede ingresar datos de un nuevo servicio, los datos que debe ingresar son nombre del servicio, el tiempo aproximado en el que se realiza ese trabajo, el costo que tendrá para los clientes y se agrega una descripción en la cual se explica en lo que consiste el servicio.

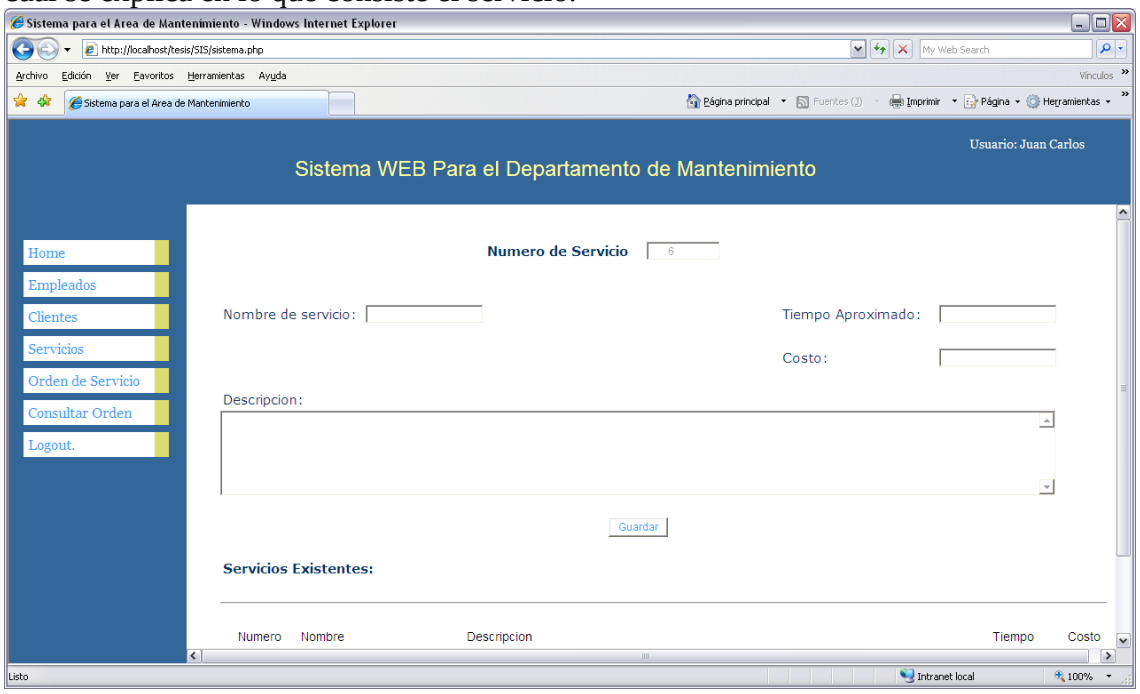

Figura 5.5 Ventana de creación de servicios.

Esta es la parte más importante del sistema, en esta página es donde se deben ingresar los datos de una nueva orden de trabajo, tales como el nombre del cliente que previamente debemos haber ingresado para después sólo elegirlo de una lista, posteriormente tomamos los datos del automóvil matricula del auto, marca, modelo, color, nombre y por si fuera necesario también un comentario.

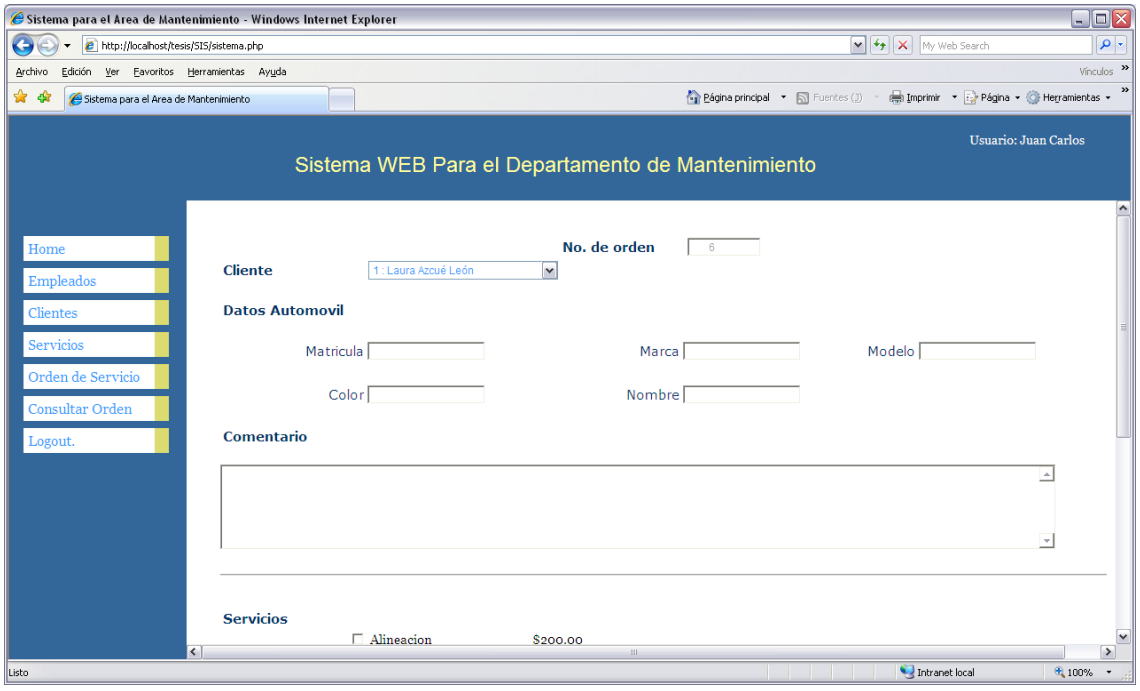

Figura 5.6 Ventana de creación de órdenes.

Continuando con la generación de una orden, aquí elegimos los servicios que requiera el cliente para su auto, los servicios que aquí se muestran son los que guardamos en la sección de servicios.

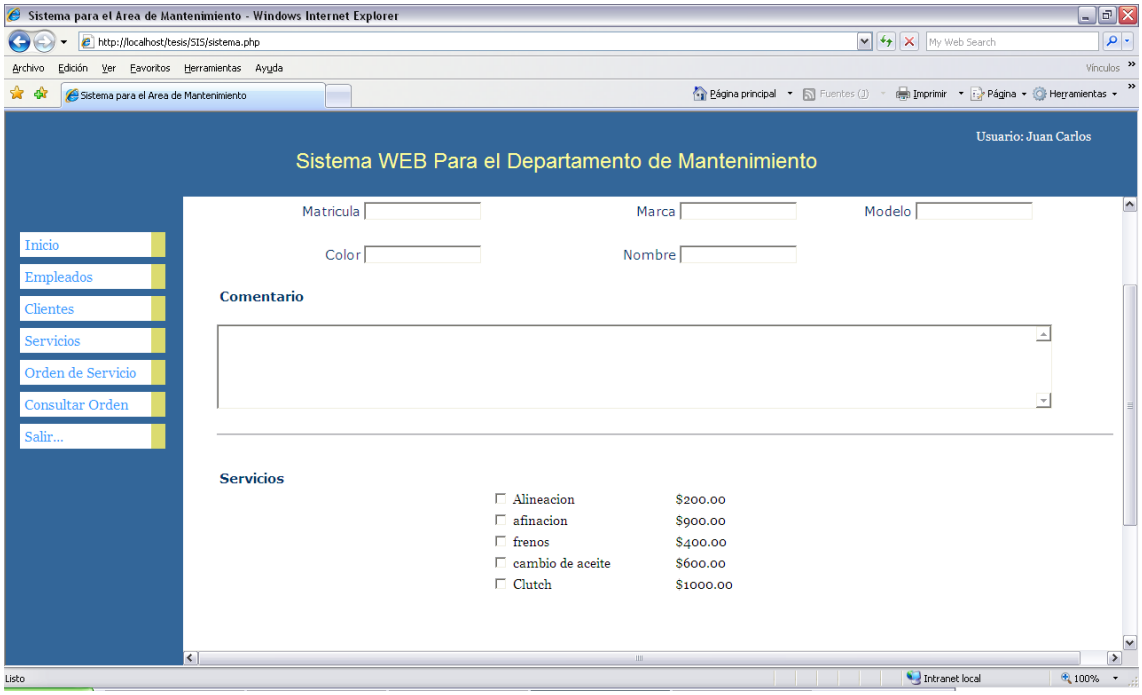

Figura 5.7 Ventana de creación de órdenes.

Esta es la página en donde si fuera necesario hacer la consulta de una orden previamente generada, sólo debemos ingresar el número de la orden para que sea desplegada.

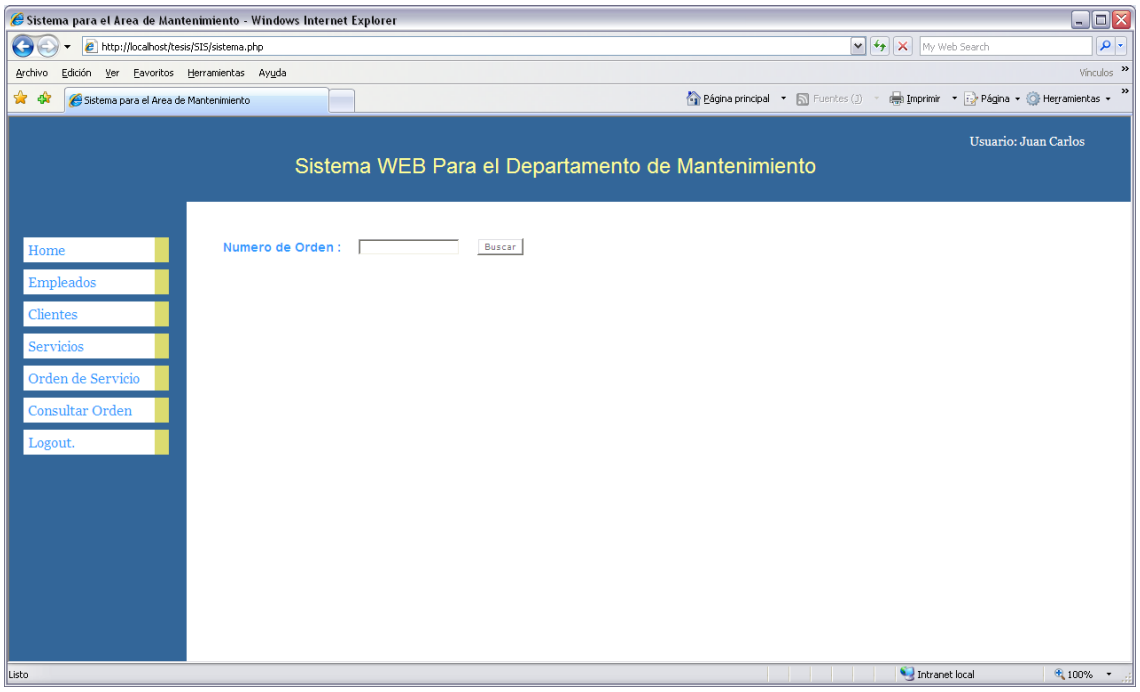

Figura 5.8 Ventana de consulta de órdenes.

5.3 Perfil de Empleado

En esta página el empleado debe ingresar el usuario y contraseña que le fue asignado a la hora de la captura de sus datos personales.

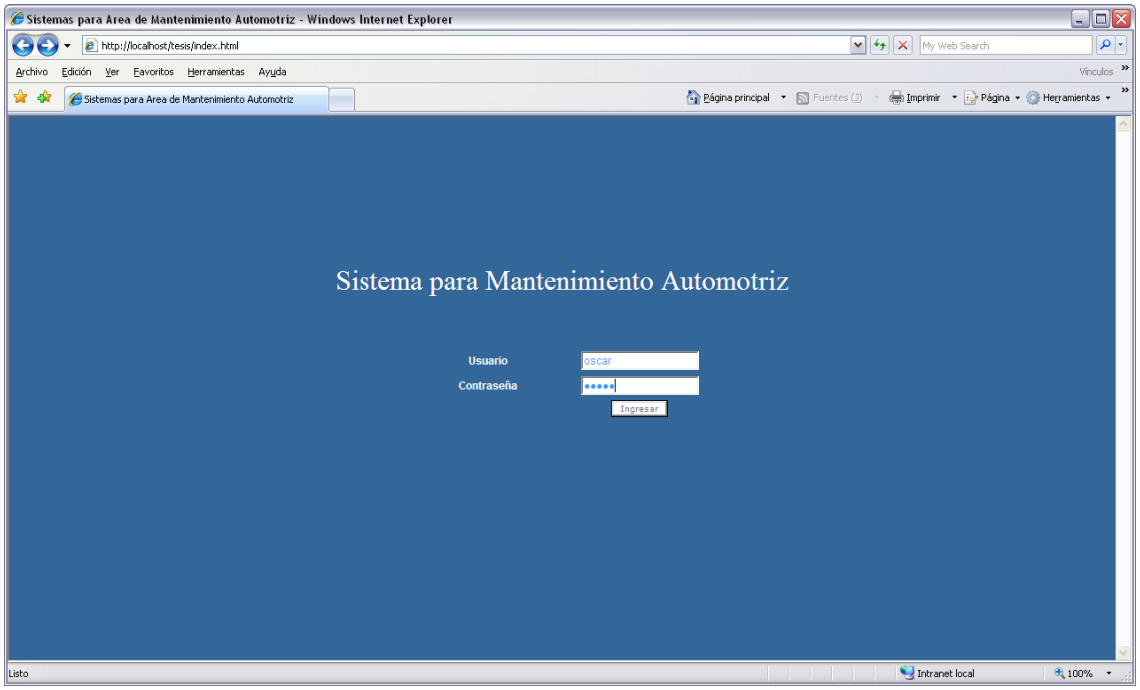

Figura 5.9 Ventana de empleado.

En la figura 5.10 podemos notar la diferencia que existe entre las opciones que se le muestran al administrador y al empleado, en este caso sólo se da opción a agregar clientes, generar y consultar órdenes de servicio.

Figura 5.10 Ventana de opciones de empleado.

5.4 Perfil de cliente

En la figura 5.11 el cliente ingresará los datos de usuario y contraseña que le fueron asignados al momento de la captura de sus datos personales.

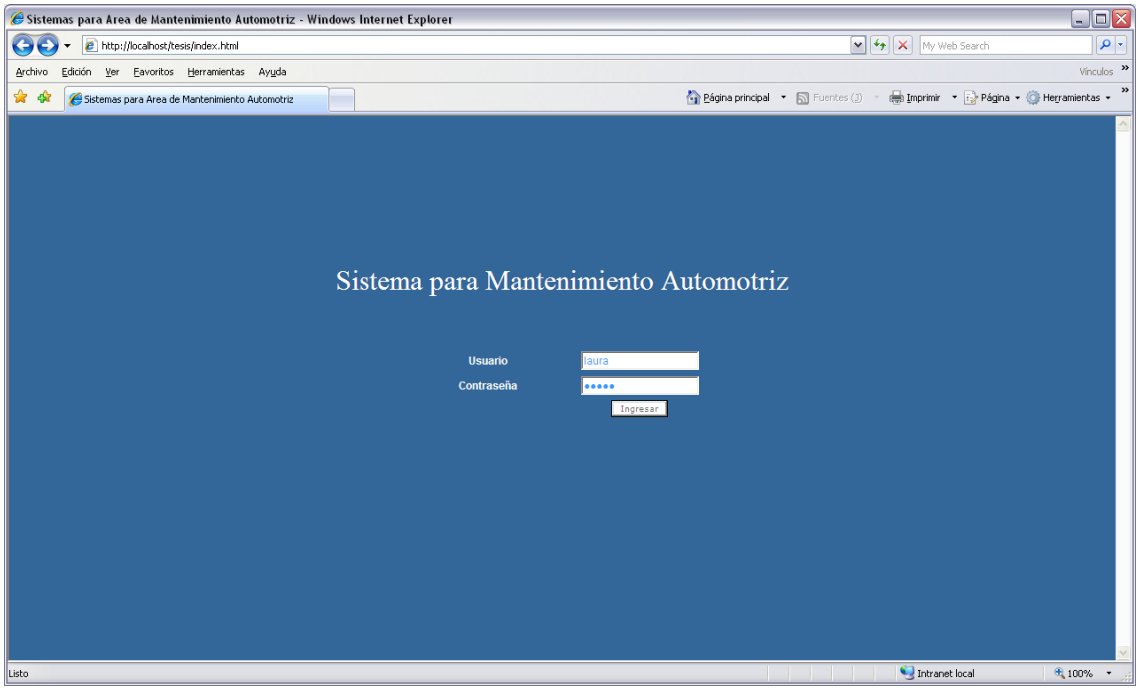

Figura 5.11 Ventana de clientes

Al cliente sólo se le da la opción de poder consultar el número de orden que le fue generado al momento de dejar su auto.

Figura 5.12 Ventana de opciones de cliente

El cliente desde la comodidad de su casa podrá saber el estado real de su auto, y sabrá el avance de los diferentes servicios que fueron contratados.

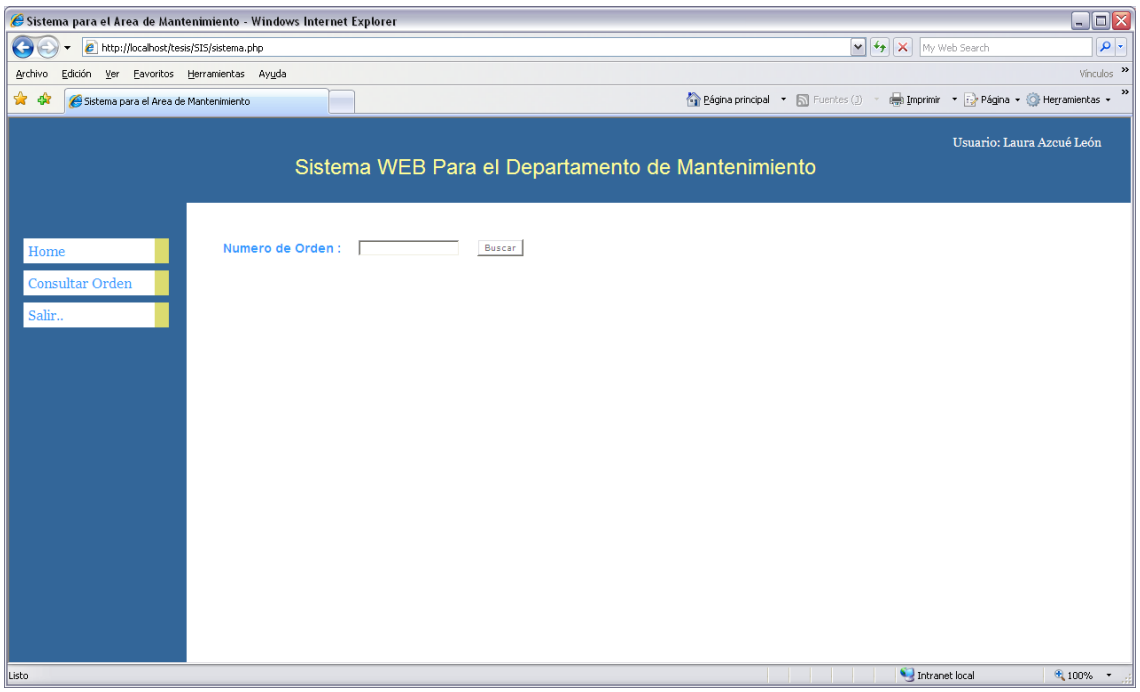

Figura 5.13 Ventana de consulta de orden de cliente.

Conclusiones

Al realizar este proyecto, he aprendido las diferentes fases por las que se tiene que pasar para poder realizar un proyecto de software.

El sistema que se desarrolló en esta tesis, fue un sistema para el departamento de mantenimiento automotriz de una agencia de autos, que podrá ayudar a hacer más eficiente el trabajo, además de ofrecer un mejor servicio al cliente, dándole la opción que desde su propia casa pueda consultar el estado de su coche, evitando el tener que estar dando vueltas o preguntando por teléfono.

Este sistema fue desarrollado usando software libre, MYSQL como manejador de la base de datos y PHP para el desarrollo de la aplicación.

Como experiencia personal, puedo decir que he obtenido habilidad para futuros sistemas, además de que ya tengo claros los pasos que hay que seguir para hacer todo el desarrollo de un sistema de forma correcta.

Una de las perspectivas para este sistemas seria agregarle un módulo que se encargue de administrar las pólizas de seguro, además de un modulo que maneje el inventario de herramientas y refacciones.

Bibliografía

(1) Shari Lawrence Pfleeger
Ingeniería de software orientado a objetos
Prentiece Hall, Buenos aires, Primera edición, 2002.

(2) Ma. Del Roció Boone Rojas
Notas "Ingeniería de software"
BUAP, Puebla México; 2008

(3) Roger S. Pressman
Ingeniería de Software. Un enfoque práctico
Mc. Grall Hill, México. Quinta Edición, c2002.

(4) Lorena Flores Huerta
Tesis "Base de Datos para el Control de Inventarios y Punto de Venta de Farmacias Universitarias, Alexander Fleming". Asesor: Ma. Del Rocío Boone Rojas
Fac. Cs. de la Computación, Puebla México

(5) Ian Sommerville
Ingeniería de Software
Pearson Educación, México, Sexta edición, 2002.

(6) http://es.wikipedia.org/wiki/DBMS

(7) Yolanda Moyao Martínez
Antología "Introducción a los Sistemas de Bases de Datos"
BUAP, Puebla México, 2008.

(8)Abraham Silberschatz, Henry F. Korth, S. Sudarshan
Fundamentos de Bases de Datos
Mc. Graw Hill. Cuarta edición, 2006.

(9) C.J. Date.
Introducción a los Sistemas Administradores de Bases de Datos.
Pearson Education Quinta Edición, 2001.

(10) Jeffrey Ullman, Jennifer Widom
Introducción a los sistemas de Bases de Datos
Mc. Graw Hill, 1999.

(11) http://es.wikipedia.org/wiki/Modelo_relacional

(12) Jever Dolores Astorga
Tesis "Sistema WEB: herramientas de software libre para los cursos de la FCC"
BUAP, Puebla México 2007, Asesor: Ma. Josefina Somodevilla García

(13) http://es.wikipedia.org/wiki/Cliente-servidor
(14) http://www.uca.edu.sv/investigacion/bdweb/intbdweb.html
(15) http://es.wikipedia.org/wiki/Caso_de_uso