B *ENEMERITA* ***U*** *NIVERSIDAD* ***A*** *UTONOMA DE* ***P*** *UEBLA*

F *ACULTAD DE* ***C*** *IENCIAS DE LA* ***C*** *OMPUTACION*

"Sistema Gestor de Solicitudes"

TESIS

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

C. KARLA MONTIEL MORALES

ASESOR

DRA. MARÍA JOSEFA SOMODEVILLA GARCÍA

PUEBLA PUE. MARZO DE 2009

Este peldaño se lo debo

a grandes personas

que me apoyaron

a lo largo y al

final de mi carrera,

pero en esencial

a mis padres

y hermanitos

por su incondicionalidad

y nunca dudar

de este logro.

PROLOGO

Este documento tiene como propósito mostrar análisis, diseño e implementación de la parte aplicativa de la Ingeniería de Software de un Sistema de Gestor de Solicitudes, para el Sistema Operador de los Servicios de Agua Potable y Alcantarillado del Municipio de Puebla (SOAPAP) el cual utiliza una base datos relacional con tareas asignadas a cada empleado de un departamento para su consulta precisa por computadora vía Web.

INDICE

INTRODUCCIÓN

En la actualidad existe una cierta unanimidad, que el atributo que contribuye fundamentalmente a determinar la posición de la empresa en el largo plazo, es la opinión de los clientes sobre el producto o servicio que reciben. Para que para los clientes se formen una opinión positiva, la empresa debe satisfacer sobradamente todas sus necesidades y expectativas. Lo anterior es lo que se ha dado en llamar calidad del servicio.[1]

Por tanto, si queremos satisfacer las expectativas del cliente es necesario disponer de información adecuada que contenga aspectos relacionados con sus necesidades, con los atributos en los que se fijan para determinar el nivel de calidad conseguido.

La calidad, y más concretamente la calidad del servicio, se está convirtiendo en nuestros días en un requisito imprescindible para competir en las organizaciones industriales y comerciales de todo el mundo, ya que las implicaciones que tiene en la cuenta de resultados, tanto en el corto como en el largo plazo, son muy positivas. De esta forma, la calidad del servicio se convierte en un elemento estratégico que confiere una ventaja haciendo la diferencia y convirtiéndola perdurable en el tiempo contra aquellas que traten de alcanzarla.

Tanto la investigación académica, como la práctica empresarial vienen sugiriendo, desde hace ya algún tiempo, que un elevado nivel de calidad de servicio proporciona a las empresas considerables beneficios en cuanto a cuota de mercado, productividad, costos, motivación del personal, diferenciación respecto a la competencia, lealtad y capacitación de nuevos clientes[2], por citar algunos de los más importantes. Como resultado de esta evidencia, la gestión de la calidad de servicio se ha convertido en una estrategia prioritaria y cada vez son más los que tratan de definirla, medirla y, finalmente, mejorarla.

Desafortunadamente, la definición y medida de la calidad han resultado ser particularmente complejas en el ámbito de los servicios. La calidad significa satisfacción de las necesidades y expectativas del cliente, teniendo en cuenta la concepción de clientes internos y clientes externos, una dirección de calidad con responsabilidad que no solo busque el producto sino que sea aplicable en todas las funciones de la organización, siempre tomando en cuenta participación del personal para la aplicación de principios y herramientas que reflejen el mejoramiento continuo de los productos y servicios.

La calidad es una estrategia que busca garantizar, a largo plazo, la supervivencia, el crecimiento y la rentabilidad de una organización, optimizando su competitividad, mediante la satisfacción del cliente y la eliminación de todo tipo de desperdicios. Esto se logra con la participación activa de todo el personal, bajo nuevos estilos de liderazgo.[3]

La calidad la determina el consumidor, él es quien califica la calidad del producto o servicio; de allí que la calidad es un valor relativo, en función al consumidor. Es necesario identificar con precisión las variantes en las necesidades y expectativas de los consumidores y su grado de satisfacción con relación a los productos y servicios. Las expectativas de los consumidores están dadas en términos de calidad en sus diferentes aspectos (calidad del producto o servicio en sí, calidad de la atención, costos razonables, etc). Una organización mejora hacia su calidad cuando los clientes externos e internos sienten que se están cumpliendo con sus requerimientos. Esta se inicia en la demanda y culmina con la satisfacción de los consumidores.[3]

El proceso de producción esta en toda la organización, no sólo es toda la línea de producción, sino toda la empresa. Los que hacen bien su trabajo lubrican el proceso, los que lo hacen mal crean cuellos de botella en el proceso. La calidad de los productos y servicios es resultado de la calidad de los procesos, esto promueve la eliminación de todo tipo de mal inversión en los inventarios, personal, exceso de informes y reuniones, es decir, controles internos innecesarios.[3]

Establecer la mentalidad de la eliminación total de los defectos, tiene como propósito erradicar el desperdicio, eliminando las actividades que no agregan valor. “Cero defectos” consiste en tener una actitud sistemática hacia el no-error. Busca despertar la conciencia de no equivocarse. La ventaja competitiva esta en la reducción de errores y en el mejoramiento continuo: allí radica la reducción de costos. Con el resultado de “Menores Costos” se puede: bajar precios, mejorar utilidades, mejorar el producto, etc.[3]

La importancia de implementar un sistema de gestión de la calidad, radica en el hecho de servir como plataforma para desarrollar una serie de actividades, procesos y procedimientos, encaminados a lograr que las características del producto o del servicio cumplan con los requisitos del cliente, es decir, que sean de calidad, lo cual nos da mayores posibilidades de que sean adquiridos, logrando así el porcentaje de ventas planificado por la organización.

La implantación de este sistema de gestión de calidad es de acuerdo al tamaño o complejidad de una entidad, tomando como estrategia la división por departamentos. Esto es imprescindible, ya que ello facilita la conducción de las operaciones y premie establecer responsabilidades que deben ser acordes al grado de delegación de autoridad.

La responsabilidad y el grado de delegación de autoridad deben quedar claramente definidos hasta donde sea posible, ya sea en un organigrama, en una nota escrita o en un procedimiento al que tengan alcance los interesados involucrados en la tarea.[4]

Dentro de un departamento determinado, la responsabilidad debe ir hacia abajo, pero siempre dentro de una línea continua, la base para la separación de funciones descansa en la premisa de que ningún departamento deberá controlar los registros contables relativos a sus propias operaciones.[4]

Para una entidad no es conveniente que sea una sola persona quien realice todas las tareas de una transacción, ya que los registros contables pueden ser manipulados en tal forma que la localización de errores y fraudes sea difícil, y muchas veces imposible. Para ello es necesario un sistema de automatizado y procedimientos de registro. El funcionamiento adecuado de un sistema de control interno no responde solamente de la organización efectiva y de la definición de procedimientos y prácticas correctas sino de la existencia de personal responsable y con convicción acerca de su importancia para el desarrollo de la empresa.[4]

Hasta la década de los sesenta, las empresas no sabían ni conocían nada acerca de los sistemas y la automatización, todas sus operaciones las llevaban manuales y consumían un sin número de horas hombre para llevarlas a cabo, su eficiencia era muy pobre y su confiabilidad muy poca. Con la introducción de la computadora, las empresas empiezan a reforzar sus procesos y a producir con menos costos y menos esfuerzos, aunque la automatización seguía siendo un tabú, pero poco a poco las organizaciones empezaron hacer cada vez más uso de estas bondades, forzando a la tecnología a concebirse de una manera más práctica y tangible.

El trabajo, dará a conocer a cada uno de los departamentos desde su estructura organizacional y mostrará los diferentes servicios que ofrece, haciendo hincapié en lo importante que estos son y la aplicación práctica que tienen.[4]

Para lograr nuestro principal objetivo, es necesario tomar en cuenta que los sistemas de información quedan delineados de acuerdo con la estructura organizacional, la cultura de los procesos políticos y la administración, ya que la tecnología de la información puede influir también a las instituciones. Un sistema de información puede definirse técnicamente como un conjunto de componentes interrelacionados que permiten capturar, procesar, almacenar y distribuir la información para apoyar la toma de decisiones y el control en una institución. Además, de la toma de las decisiones, la coordinación y el control, los sistemas de información pueden también ayudar a los administradores y al personal a analizar problemas, visualizar cuestiones complejas, crear nuevos productos y automatizar.[4]

Bajo este contexto y teniendo en cuenta la importancia de la calidad en los servicios públicos, dentro de una política de mejora continua, el Sistema Operador de los Servicios de Agua Potable y Alcantarillado del Municipio de Puebla (SOAPAP), necesita automatizar el proceso captación de posibles anomalías, mejoras y/o observaciones en la operatividad de su sistema comercial.

Cuando un usuario (por citar un ejemplo, de los muchos casos de incidencias) observa que en su estado de cuenta no es correcto el monto de su factura debido a que no tomaron la lectura del medidor correcto, por vivir en un condominio. Entonces este se inconforma con el encargado de las oficinas de atención a público y hace la observación, este a su vez lo comenta a su jefe. Entonces el jefe genera un memorándum a la subgerencia de informática, para que el subgerente lo delegue al jefe del departamento de sistemas y por último este al responsable de la modificación. Una vez modificado el dato se genera el memorándum de respuesta el cual sigue el mismo trayecto de regreso, hasta llegar al usuario que lo solicitó.

Este proceso además de ser tedioso, representa un gasto en papelería y tiempo; teniendo como resultado que el usuario no realice el pago en esa ocasión, siendo pérdidas monetarias para la empresa que pueden solucionarse con una automatización adecuada del proceso de quejas.

Por tales razones la empresa necesita de un software que permita la consulta vía Web. Esto con la finalidad de una fácil consulta para todos los usuarios sin requerir de la instalación en cada equipo de cómputo, que capte las incidencias, centralizadas en una mesa de ayuda, para que esta turne solicitudes a los jefes de los departamentos afectados y estos a su vez puedan girar órdenes de trabajo al personal a su cargo, para que solucionen las incidencia y haya una respuesta inmediata.

El SIGES (Sistema Gestor de Solicitudes), será un sistema capaz de administrar las órdenes de trabajo, llevando un control de las descripciones, fechas de creación, modificación, así como las personas a las que les fueron asignadas. Todo lo anterior para que escriban un resumen de la manera en que fueron resueltas dichas ordenes, calculando en automático el tiempo que transcurre desde que les fue asignada la tarea, hasta que les da una respuesta, todo esto con un histórico de todos los movimientos.

Este documento tiene como propósito mostrar la parte aplicativa de la Ingeniería de Software de un Sistema de Gestor de Solicitudes, el cual utiliza una base datos relacional con tareas asignadas a cada empleado de un departamento para su consulta precisa por computadora vía Web.

El objetivo general del trabajo es analizar, diseñar e implementar un Sistema Gestor de Solicitudes de anomalías para SOAPAP.

Los objetivos específicos del sistema son:

- Facilitar la captación de las incidencias, mejoras u observaciones del sistema comercial.
- Agilizar la asignación de órdenes de trabajo que resuelvan tales incidencias.
- Mostrar los tiempos de resolución para cada tarea y medir eficiencia.
- Facilitar la consulta de las órdenes de trabajo, tanto pendientes como las resueltas o las que están en proceso de.
- Facilitar la búsqueda de las órdenes creadas, por diferentes filtros.
- Controlar los permisos de consulta y modificación, según el usuario.
- Llevar un historial de las modificaciones realizadas a cada órden de trabajo.
- Ahorrar tiempo de respuesta.
- Proporcionar al usuario un sistema amigable y confortable de consulta bajo un ambiente Web.

Este documento se divide en cuatro capítulos. En el Capítulo 1 se presenta el Marco Teórico. El Capítulo 2 se ocupa del Análisis del Sistema. El Diseño es tratado en el Capítulo 3. En el Capítulo 4 se detalla la Implementación del Sistema. Finalmente se presentan Conclusiones y Perspectivas seguidas de la Bibliografía consultada para realizar este trabajo.


1
MARCO TEORICO

1.1 Ingeniería de Software

En las primeras décadas el desafío principal en el mundo de la computación era desarrollado del Hardware de las computadoras, para que se redujera el costo de procesamiento y almacenamiento. La potencia de grandes computadoras de las décadas pasadas está hoy disponible en una computadora personal. Las enormes capacidades de procesamiento y almacenamiento del Hardware moderno presentan un gran potencial de cálculo. El Software es el que nos facilita utilizar y explotar este potencial, incrementando o incluso disminuyendo sus precios, junto con una fuerte tendencia a la estandarización y una gran diversidad de marcas y modelos.[5]

Esto hace que, a la hora de plantearnos la adquisición de un sistema informático completo, ya sea para gestionar una en empresa, controlar un proceso industrial, o para uso domestico, el software marca la diferencia. Entre varios productos de características de Hardware similares, nos decidiremos por una determinada compañía vendedora basándonos en las prestaciones, calidad y facilidad de uso de su software. Por otra parte, el desarrollo de Software no es una tarea fácil. La complejidad actual de los sistemas informáticos hace necesarios el desarrollo de proyectos de Software de decenas de miles de líneas de código. Esto no puede ser abordado directamente, empezando a programar sin más. Es necesario analizar que es lo que tenemos que hacer, como lo vamos a hacer, como se van a coordinar todas las personas que van intervenir en el proyecto y como vamos a controlar el desarrollo del mismo de forma que al final obtengamos los resultados esperados.[5]

Como vemos, el Software dentro de cualquier sistema basado en el uso de computadoras, es el componente cuyo desarrollo presenta mayores problemas: es más difícil de planificar, el que tiene mayor probabilidad de fracaso, y el que tiene menos posibilidades de que se cumplan las estimaciones de costos inícíales. Por otra parte, la demanda de Software aumenta continuamente, lo que aumenta la magnitud de estos problemas.

En un nivel técnico, la ingeniería de sistemas de información 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.[5] Se han desarrollado muchos métodos para hacer modelos de sistemas de información. Sin embargo, los métodos orientados a objetos van a llegar a ser el estándar. Para ciertos sistemas de información crítico, se han desarrollado métodos formales 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. El uso de métodos formales es recomendado cuando sea posible en un ciclo vital del software.

La disciplina que se encarga de establecer un orden en el desarrollo de Sistemas de Software es la Ingeniería de Software.[5]

La Ingeniería de Software es la aplicación práctica del conocimiento científico en el diseño y construcción de programas para computadoras y la documentación asociada requerida para desarrollarlos, operarlos y mantenerlos[5]. A diferencia de los métodos tradicionales, la tecnología de componente ensambla para formar una solución de software. Actualmente, hay dos estándares de componentes en competencia: JavaBeans por Sun y DCOM por Microsoft. Los componentes de software son materiales reusables para construir sistemas de software. La tecnología Component-Base (Basada en Componentes) es un método poderoso por la empresa de la ingeniería de sistemas de información porque es una tecnología que reduce el conflicto entre sistemas de alta complejidad y de la búsqueda para la alta calidad y la productividad.

La Ingeniería de Software es una disciplina que todavía se esta desarrollando. Podemos esperar en el futuro su crecimiento y madurez en los próximos años.

1.1.1 Evolución de la Industria del Software

Durante la primera era del desarrollo de las computadoras el Hardware sufrió continuos cambios, se buscó un mejor rendimiento, una reducción del tamaño y un costo más bajo para dar lugar a sistemas informáticos más sofisticados. Se pasó de los procesadores con válvulas de vacío a los dispositivos micro electrónicos que son capaces de procesar 200 millones de instrucciones por segundo, mientras que el software se contemplaba simplemente como un añadido. La programación de computadoras era un arte de "andar por casa" y el desarrollo del software se realizaba virtualmente sin ninguna planificación. Durante este periodo se utilizaba en la mayoría de los sistemas una orientación por lotes.

La mayoría del software se desarrollaba y era utilizado por la misma persona u organización. La misma persona lo escribía, lo ejecutaba, y si fallaba, lo depuraba, debido a que la movilidad en el trabajo era baja, los ejecutivos estaban seguros de que esa persona estaría allí cuando se encontrara algún error, por otra parte el diseño era un proceso implícito realizado en la mente de alguien, y la documentación normalmente no existía.

En la Segunda era el Software se llegó a establecer como producto y aparecieron nuevas "casas de Software". Las casas desarrollaban proyectos en los que se producían programas de decenas de miles de sentencias fuente. En algunos casos había que comprar Software al exterior para implementar el programa y esto añadía miles de sentencias más al proyecto, un nuevo problema apareció cuando había que corregir todas aquellas sentencias, cuando se detectaban fallos modificados cuando cambiaban las necesidades de los usuarios o cuando adaptaban un nuevo dispositivo de hardware que se hubiera adquirido. Estas actividades se llamaron mantenimiento del software el cual comenzó a absorber recursos en una medida alarmante.

Aún peor, la naturaleza personalizada de muchos programas los hacia imposibles de mantener, había comenzado la llamada "crisis del software".

Durante la tercera era aparecen múltiples computadoras cada una ejecutando funciones concurrentes y comunicándose con alguna otra. Las redes de área local y de área global, las comunicaciones digitales de alto ancho de banda y la creciente demanda de acceso "instantáneo" a los datos, supusieron una fuerte presión sobre los desarrolladores del software. Esta era también se caracteriza por la llegada y el amplio uso de los microprocesadores y las computadoras personales.

La cuarta era está empezando ahora, las tecnologías orientadas a los objetos están desplazando rápidamente a enfoques de desarrollo de software más convencionales en muchas áreas de aplicación, los sistemas expertos y el software de inteligencia artificial se ha trasladado del laboratorio a las aplicaciones prácticas, para un amplio rango de problemas del mundo real. El software de redes neuronales artificiales ha abierto excitantes posibilidades para el reconocimiento de formas y habilidades de procesamiento de información al estilo de cómo lo hacen los humanos.

Como respuesta a la crisis del software, muchas industrias están adoptando prácticas de Ingeniera de Software.

1.1.2 Características del Software

Definición: "Software: Instrucciones de computadora que cuando se ejecutan proporcionan la función y el comportamiento deseado. Estructuras de datos que facilitan a los programas manipular adecuadamente la información, y documentos que describen la operación y el uso de los programas". [5]

El Software se diferencia de otras cosas que los hombres pueden construir, por ejemplo cuando se construye Hardware, el proceso creativo humano (análisis, diseño, construcción y prueba) se traduce en una forma física, en cambio el software es un elemento del sistema que es lógico, en lugar de físico, por tanto estas son algunas de sus características:

- El software se desarrolla, no se fabrica en sentido clásico.
- El Software no se estropea.
- La mayoría del Software se construye a medida, en vez de ensamblar componentes existentes.

El Hardware se construye mediante un esquema de circuitería digital, se hace algún análisis para asegurar que realiza la función adecuada, se selecciona cada componente de un catálogo y se realiza la compra.[6]

Los diseñadores de Software no realizan las mismas actividades, no existen catálogos de componentes de Software, se puede realizar la compra de un Software ya desarrollado, pero completo, no como un componente que se pueda ensamblar y formar un nuevo programa.

1.1.3 Aplicaciones del Software

El Software puede aplicarse en situaciones del mundo real donde se haya establecido un conjunto específico de pasos procedimentales (es decir, un algoritmo). Para determinar la naturaleza de una aplicación se deben considerar dos factores importantes[5]:

- El contenido que se refiere al significado y a la forma de la información de entrada y de salida.
- El determinismo de la información que se refiere a que tan predecible es el orden y del tiempo de llegada de los datos.

No obstante es difícil establecer categorías genéricas para las aplicaciones del Software que sean significativas. Conforme aumente la complejidad del software, es más difícil establecer compartimientos nítidamente separados. Su aplicación se extiende en las siguientes áreas.

Software de sistemas: es un conjunto de programas que han sido diseñados para servir a otros programas. Por ejemplo los compiladores, editores y editores de gestión de archivos procesan estructuras de informaciones complejas pero determinadas. Otras aplicaciones de sistemas son por ejemplo ciertos componentes del Sistema Operativo,

utilidades de manejo de periféricos, procesadores de telecomunicaciones que procesan datos indeterminados.[5]

Características:

- Fuerte interacción con el Hardware de la computadora.
- Utilización por múltiples usuarios.
- Operación concurrente que requiere una planificación.
- Compartición de recursos.
- Sofisticada gestión de procesos.
- Estructuras de datos complejas.
- Múltiples interfaces externas.

Software de tiempo real: Es el software que mide, analiza y controla sucesos del mundo real como o ocurre.[5]

Elementos:

- Un componente de adquisición de datos que recolecta y da formato a la información recibida del exterior.
- Un componente de análisis que transforma la información según se requiera.
- Un componente de control. Salida que responda al entorno externo.
- Un componente de monitorización que coordina todos los demás componentes de forma que pueda mantenerse la respuesta en tiempo real utilizando un rango de un milisegundo a un minuto.

El término tiempo real tiene un significado diferente de "interactivo" o "tiempo compartido".

Un sistema de tiempo real debe responder dentro de unas ligaduras estrictas de tiempo. El tiempo de respuesta de un sistema interactivo (o de tiempo compartido) puede ser sobrepasado sin producir ningún desastre.

Software de gestión: El procesamiento de información de gestión constituye la mayor de las áreas de aplicación del software. Los sistemas discretos como nóminas, cuentas de débitos, inventarios, etc., han evolucionado hacia el software de sistemas de información de gestión (SIG), su principal tarea es reestructurar los datos existentes en orden para facilitar las operaciones comerciales o gestionar la toma de decisiones. Además de las tareas convencionales de procesamientos de datos y calculo interactivo, por ejemplo el procesamiento de transacciones en puntos de ventas.[5]

Software de ingeniería y científico: Se caracteriza por los algoritmos de "manejo de números", se aplica desde la astronomía a la vulcanología, desde el análisis de la presión de los automotores a la dinámica orbital, etc.

Sin embargo el campo del software científico y de ingeniería se ha visto ampliado con el diseño asistido por computadora (del Ingles CAD), los simuladores gráficos y otras aplicaciones interactivas que lo acercan más al software de tiempo real e incluso al software de sistemas.[5]

Software empotrado: Este software reside en memoria de sólo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo. El software empotrado se puede encontrar en productos inteligentes ejecutando funciones muy limitadas y curiosas por ejemplo el control de teclas de un horno de micro ondas o suministrar una función con capacidad de control por ejemplo funciones digitales de un automóvil, tales como control de gasolina, sistema de frenado etc.[5]

Software de inteligencia artificial: El software de inteligencia artificial (IA) hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados, el cálculo o análisis directo. Las áreas de su aplicación son las siguientes[5]:

- Sistemas expertos llamados Sistemas basados en el conocimiento.
- Reconocimiento de patrones (imágenes y voz).
- Prueba de teoremas.
- Juegos.
- Y en los últimos años la nueva rama llamada redes neuronales.

Software de computadoras personales: Debido al uso de las computadoras personales el software se ha generalizado en la década pasada. Algunas de sus aplicaciones son las siguientes: procesamiento de textos, las hojas de cálculo, los gráficos por computadora, entretenimientos, gestión de bases de datos, aplicaciones financieras de negocios y personales, y redes de acceso a bases de datos externas[5].

Este software representa uno de los diseños más innovadores en el campo del Software.

1.1.4 Problemas del Software

Los problemas que afligen al desarrollo del Software son causados por las propias características del Software y por los errores cometidos por el productor.

Los desarrolladores se centran sobre los siguientes aspectos. La planificación y estimación de costes son muy imprecisas. La "Productividad" de la comunidad del software no se corresponde con la demanda de sus servicios. La calidad del software no llega a ser a veces ni aceptable.[5]

Otras dificultades del software:

- No existe suficiente tiempo de recoger los datos sobre el proceso de desarrollo del Software, sin estos datos no podemos evaluar la eficacia de las nuevas herramientas, técnicas o estándares.
- La insatisfacción del cliente con el sistema "terminado" es muy frecuente, debido a la vaga indicación de requisitos del cliente y a la falta de comunicación entre el cliente y el programador.
- La calidad del Software es cuestionable. Es por ello que se ha empezado a comprender la importancia de la prueba sistemática y completa, también han empezado a surgir conceptos como la fiabilidad y la garantía de calidad.

El software existente puede ser muy difícil de mantener. El mantenimiento se lleva la mayor parte de los recursos invertidos en el software. Aunque aún no se ha considerado un criterio importante en la aceptación del mismo.

1.1.5 Paradigmas de la Ingeniería de Software

No existe un único enfoque para solucionar el problema del software. Sin embargo, mediante la combinación de métodos completos para todas las fases del desarrollo del software, mejores herramientas para automatizar estos métodos, bloques de construcción mas potentes para la implementación del software, mejores técnicas para la garantía de calidad del Software y una filosofía predominante para la coordinación, control, y administración, podemos conseguir una disciplina para el desarrollo del software, una disciplina llamada Ingeniería de Software[5].

La Ingeniería de Software surge de la Ingeniería de Sistemas y de Hardware. Abarca un conjunto de tres elementos clave (métodos, herramientas y procedimientos) que facilitan al gestor controlar el proceso del desarrollo del Software y suministrar a los que practiquen dicha Ingeniería las bases para construir Software de alta calidad de una forma productiva. Los métodos indican como construir técnicamente el Software. Abarcan un conjunto amplio de tareas: planificación y estimación de proyectos, análisis de requerimientos del sistema y del software, diseño de estructura de datos, arquitectura de programas y procedimientos algorítmicos, codificación, prueba y mantenimiento. Los

métodos introducen una notación especial orientada a un lenguaje o gráfica y un conjunto de criterios para la calidad del software[5].

Las herramientas suministran un soporte automático o semiautomático para los métodos. Existen actualmente herramientas que soportan cada uno de los métodos mencionados anteriormente. Es decir, hablamos de CASE (Ingeniería de software asistida por computadora).

Los procedimientos son el pegamento que junta los métodos y las herramientas, facilita un desarrollo racional y oportuno del Software de computadora. Estos definen la secuencia en la que se aplican los métodos y las entregas (documentos, informes, formas, etc.) que se requieren, los controles que ayudan a asegurar la calidad y coordinar los cambios y las directrices que ayudan a los administradores del software a evaluar el progreso.

La Ingeniería de software está compuesta por una serie de pasos que abarcan los métodos, las herramientas y los procedimientos. Estos pasos se denominan frecuentemente paradigmas de la Ingeniería de Software. La elección de un paradigma se lleva de acuerdo a la naturaleza y de la aplicación, los métodos y herramientas a usar y los controles y entregas requeridos. Entre los paradigmas tenemos los siguientes[5].

El ciclo de vida clásico

También llamado "modelo en cascada", el paradigma del ciclo de vida exige un enfoque sistemático y secuencia! del desarrollo del software que comienza en el nivel del sistema y progresa a través del análisis, diseño, codificación, prueba y mantenimiento.[5]

El modelo esta formado por las siguientes etapas.

Ingeniería y análisis del sistema: Se establecen los servicios del sistema, sus restricciones y logros consultando a los usuarios del sistema. Estos se definen entonces de una manera clara y entendible por los usuarios y el equipo de programadores.

Análisis de requisitos del Software: En este el proceso de recopilación de los requisitos se centra e intensifica especialmente para el Software. Los requisitos, tanto del sistema como del software, se documentan y se revisan con el cliente.

Diseño: Se enfoca sobre cuatro atributos distintos del programa: la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz.

Codificación: La codificación traduce en una forma legible el diseño para la máquina.

Prueba: Se centra en la lógica interna del software, asegurando que todas las sentencias se han probado, y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

Mantenimiento: El software sufre cambios después de que se entregue al cliente. Los cambios ocurrirán debido a que se hayan encontrado errores, a que el software deba adaptarse a cambios del entorno externo, o debido a que el cliente requiera ampliaciones funcionales o del rendimiento.

Construcción de Prototipos: La construcción de prototipos es un proceso que facilita al programador la creación de un modelo del software. El modelo tomará una de las siguientes formas:

- Un prototipo en papel o un modelo basado en computadora que describa la interacción humano-computadora, esto facilitará al usuario la comprensión de cómo se producirá tal interacción.
- Un prototipo que implementa algunos subconjuntos de la función requerida del programa deseado.
- Un programa existente que ejecute parte o toda la función deseada, pero que tenga otras características que deban ser mejoradas en el nuevo trabajo de desarrollo.

El modelo en espiral

El modelo ha sido desarrollado para cubrir las mejores características de los modelos anteriores, añadiendo el análisis de riesgo. El modelo define cuatro actividades principales[5]:

- Planificación: determinación de objetivos, alternativas y restricciones.
- Análisis de riesgo: análisis de alternativas e identificación / resolución de riesgo.
- Ingeniería: desarrollo del producto de "siguiente nivel".
- Evaluación del Cliente: Valoración de los resultados de la Ingeniería.

Técnicas de Cuarta generación

Este término abarca un amplio espectro de herramientas de software que facilitan al desarrollador del software la especificación de algunas características del software de alto nivel. El Paradigma T4G para la ingeniería de software se orienta hacia la posibilidad de

especificar el software a un nivel más próximo al lenguaje natural o en una notación que proporcione funciones significativas [5].

1.1.6 Visión Genérica de la Ingeniería de Software

El proceso de desarrollo del software contiene tres fases genéricas, independientemente del paradigma elegido. Las tres fases, definición, desarrollo y mantenimiento, se encuentran en todos los desarrollos del software [5].

La **fase de definición** se centra sobre el ¿Qué?, esto es, durante la definición el desarrollador intenta identificar que información ha de ser procesada, que función y rendimiento se desea, que interfaces han de establecerse, que restricciones de diseño existen y que criterios de validación se necesitan para definir un sistema correcto. Se producirán tres pasos específicos:

1. El análisis del sistema: el análisis define el papel de cada elemento de un sistema.
2. Planificación del proyecto de software. Una vez establecido el ámbito del software, se analizan los riesgos, se asignan los recursos, se estiman los costos, se definen las tareas y se planifica el trabajo.
3. Análisis de requerimientos: el ámbito proporciona la dirección a seguir, pero antes de iniciar el trabajo, es necesario disponer de la información de manera más detallada del ámbito de información y de función del Software.

La fase de desarrollo se centra en el ¿Cómo? Durante esta fase el desarrollador intenta descubrir como han de diseñarse las estructuras de datos y la arquitectura del software, como han de implementarse los detalles procedurales, como ha de traducirse el diseño a un lenguaje de programación y como ha de realizarse la prueba. Se producirán tres pasos específicos:

- Diseño del Software: Este traduce los requerimientos del software a un conjunto de representaciones que describen la estructura de los datos, la arquitectura, el procedimiento algorítmico y las características de la interfaz.
- Codificación: las representaciones del diseño deben ser traducidas a un lenguaje de programación, dando como resultado unas instrucciones ejecutables por la computadora. El paso de la codificación es que lleva a cabo esa traducción.
- Prueba del Software: Una vez que se ha implementado el software en una forma ejecutable por la máquina, debe ser probado para descubrir los defectos que puedan existir en la función, en la lógica y en la implementación.

La fase de mantenimiento se centra en el cambio. Esta fase vuelve a aplicar los pasos de las fases de definición, planteamiento y desarrollo, pero en el contexto del Software ya existen. Durante esta fase se encuentran tres tipos de cambios:

- Corrección: Incluso, siendo aplicadas las mejores actividades de garantía de calidad, es muy probable que el cliente descubra defectos en el software. El mantenimiento correctivo cambia el software para corregir los defectos.
- Adaptación: con el paso del tiempo es probable que cambie el entorno original. El mantenimiento adaptativo consiste en modificar el software para acomodarlo a los cambios de su entorno externo.
- Conforme se utiliza el Software, el cliente/usuario puede descubrir funciones adicionales que podría interesar que estuvieran incorporadas. El mantenimiento perfectivo amplía el software más allá de sus requerimientos funcionales originales.

1.1.7 Diagramas de Flujo de Datos

La información que viaja a través del Software es modificada por una serie de transacciones. "El diagrama de flujo de datos es una técnica gráfica que representa el flujo de la información y las transacciones que se aplican a los datos al moverse desde la entrada hasta la salida"[6]. También conocido como grato de flujo de datos o como diagrama de burbujas.

Las reglas siguientes pueden ayudar a elaborar un DFD óptimo.

- Elegir nombres con significado para los procesos, flujos, almacenes y terminadores.
- Numerar los procesos.
- Re-dibujar el DFD tantas veces como sea necesario.
- Evitar los DFD's demasiado complejos.
- Asegurarse de que el DFD sea internamente consistente y que también lo sea con cualquier DFD relacionado con el.
- Elegir nombres con significado para los procesos, flujos, almacenes y terminadores.

Componentes de los diagramas de flujo de datos

Representación gráfica de los procesos que componen el sistema y las interfaces entre ellos [6]:

Entidad Externa (Rectángulo): Indica las fuentes o destinos de los datos

Proceso (Círculos): Transforman los datos. Los flujos de entrada son diferentes a los de salida.

Flujo de datos (Flecha): Flujo de datos.

Almacén (Rectángulo abierto): El medio físico no es especificado; incluye medios manuales de almacenamiento. No incluye almacenamientos temporales.

1.1.8 Diccionario de Datos

Es otra herramienta importante en el modelado de proyectos, sin ésta el modelo de los requerimientos del usuario no puede considerarse completo, lo que se tendría sería un borrador y el análisis de sistemas se extraviaría por tanto el usuario no podrá entender los detalles de la aplicación.

El Diccionario de datos es un listado organizado de todos los datos pertinentes al sistema, con definiciones precisas y rigurosas para que el usuario y el analista tengan un entendimiento común de todas las entradas, salidas, componentes de almacén y cálculos intermedios [6].

El diccionario de datos define los datos haciendo lo siguiente:

- Define el significado de los flujos y almacenes que se muestran en el DFD.
- Describe la composición de agregados de paquetes de datos que se muestran a lo largo de los flujos, paquetes complejos (ej. Domicilio de un cliente), que puede descomponerse en unidades mas elementales (como ciudad, estado y código postal)

- Describe la composición de los datos en los almacenes.
- Especifica los valores y unidades relevantes de piezas elementales de información en los flujos de datos y en los almacenes de datos.
- Describe los detalles de las relaciones entre almacenes que se enfatizan en un diagrama de entidad-relación.

El diccionario de datos lo crea el analista durante el desarrollo del modelo del sistema, pero el usuario debe ser capaz de leerlo y entenderlo para poder verificar el modelo. Podemos verificar si el diccionario de datos es correcto analizándolo en conjunto con el DFD y el Diagrama de Entidad-Relación.

1.2 Las Bases de Datos

Las bases de datos y su tecnología están teniendo un impacto decisivo sobre el creciente uso de los computadores. No es exagerado decir que las bases de datos desempeñan un papel crucial en casi todas las áreas de aplicación de los computadores, como los negocios, la ingeniería, la medicina, el derecho, la educación y la biblioteconomía, por mencionar solo unas cuantas.

El termino dase de datos se refiere a un conjunto de datos relacionados entre sí. Por datos entendemos hechos conocidos que pueden registrarse y que tienen un significado implícito. En otras palabras, una base de datos tiene una fuente de la cual se derivan los datos, cierto grado de interacción con los acontecimientos del mundo real y un público que esta activamente interesado en el contenido de la base [11].

Las bases de datos pueden se de cualquier tamaño y tener diversos grados de complejidad. La enorme cantidad de información debe organizarse y controlarse para que los usuarios puedan buscar, obtener y actualizar los datos cuando sea necesario.

De forma sencilla podemos indicar que una base de datos no es más que un conjunto de datos que pertenecen al mismo contexto almacenados sistemáticamente para su uso posterior. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta [7].

En la actualidad, y gracias al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos tienen formato electrónico, que ofrece un amplio rango de soluciones al problema de almacenar datos.

En informática existen los sistemas gestores de bases de datos (SGBD en ingles database management system DBMS), que son un conjunto de programas que permiten a los usuarios crear y mantener las bases de datos con un acceso a los datos de forma rápida y estructurada [11]. Las propiedades de los sistemas gestores de bases de datos se estudian en informática.

Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas. También son ampliamente utilizadas en entornos científicos con el objeto de almacenar la información experimental.

El 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. Las bases de datos manuales, pueden ser difíciles de gestionar y modificar. Por ejemplo, en una guía de teléfonos no es posible encontrar el número de un individuo si no sabemos su apellido, aunque conozcamos su domicilio.

Del mismo modo, en un archivo de pacientes en el que la información esté desordenada por el nombre de los mismos, será una tarea bastante engorrosa encontrar todos los pacientes que viven en una zona determinada. Los problemas expuestos anteriormente se pueden resolver creando una base de datos informatizada.

Desde un punto de vista 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 manipulan ese conjunto de datos.

Desde el punto de vista más formal, podríamos definir una base de datos como un conjunto de datos estructurados, fiables y homogéneos, organizados independientemente en máquina, accesibles a tiempo real, compartibles por usuarios concurrentes que tienen necesidades de información diferente y no predecible en el tiempo.

La idea general es que estamos tratando con una colección de datos que cumplen las siguientes propiedades:

- Están estructurados independientemente de las aplicaciones y del soporte de almacenamiento que los contiene.
- Presentan la menor redundancia posible.
- Son compartidos por varios usuarios y/o aplicaciones.

1.2.1. Componentes de una Base de Datos

Los principales componentes de una base de datos son:

Datos: Una base de datos puede estar en distintas máquinas. Si se encuentra en una máquina grande se tiende a multiusuario. Si se tiene una máquina pequeña se atiende a un único usuario [7]. Los datos pueden ser:

- Integrados: La base de datos es la unificación de varios archivos que otro modo serían distintos datos con redundancia.
- Compartidos: Piezas de datos entre diferentes usuarios con fines distintos.

Si la base de datos no es compartida se conoce como personal o específica.

Hardware: Consta de:

- Almacenamiento secundario.
- Controladores.
- Procesadores.
- Memoria.

Software: Entre la base de datos física y los usuarios hay una capa de software conocida como administrador (gestor) de base de datos o el servidor de base de datos o sistema de administración de base de datos (DBMS Data Base Manageament System) donde todas las solicitudes son manejadas por el DBMS y su función es ocultar a los usuarios de la base de datos los detalles físicos [7].

Usuarios: Existen tres tipos de usuarios:

Usuario final: interactúa con la base de datos desde una terminal en línea a través de un lenguaje de consulta, o a través de un programa (interfaz). Mediante el lenguaje de consulta el usuario queda libre para poder hacer cualquier operación y mediante un programa el usuario queda restringido a lo que en éste, se ha establecido [7].

Programador de Aplicaciones: Desarrolla los sistemas necesarios para permitir la posibilidad de comunicación o extensión de información desde la Base de Datos [7].

Administrador: Mantiene en forma óptima y eficiente la base de datos, controlando procedimientos, instalaciones, procesos, etc. Realiza funciones de auditoría, maneja la seguridad de la base de datos, además de crear usuarios y accesos permitidos [7].

Cuando se utiliza una base de datos se obtienen grandes beneficios ya que estas son menos laboriosas, rápidas, el control de la información es centralizado y la información

siempre se mantiene actualizada. Como consecuencia del control centralizado tenemos la seguridad, compartición de datos, cumplimiento de normas, disminución de redundancia, eliminación de inconsistencias y se mantiene la integridad de la información.

Ventajas de las Bases de Datos [7].

Referidas a	Ventajas
Los datos	• Independencia de estos respecto de los tratamientos y viceversa. • Mejor disponibilidad de los mismos. • Mayor eficacia en la recogida, codificación y entrada.
Los resultados	• Mayor coherencia. • Mayor valor informativo. • Mejor y más normalizada la documentación de la información.
Los usuarios	• Acceso más rápido y sencillo de los usuarios finales. • Más factibilidades para compartir los datos por el conjunto de usuarios. • Mayor flexibilidad para atender demandas cambiantes.

Desventajas de las Bases de Datos [7].

Referidas a	Desventajas
La implantación	• Costosa en equipos (lógicos y físicos). • Ausencia de estándares. • Larga y difícil puesta en marcha. • Rentabilidad a medio plazo.
Los usuarios	• Personal especializado. • Desfase entre teoría y práctica.

1.1.2 Sistema de Gestión de Base de Datos

Los Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos 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. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y DataBase Management System, su expresión inglesa [7].

Hoy en día, son muchas las aplicaciones que requieren acceder a datos. Bien sea un sencillo programa doméstico, bien una suite para la gestión empresarial. Estos datos se deben almacenar en algún soporte permanente, y las aplicaciones deben disponer de un medio para acceder a ellos. Normalmente, la forma en que un programa accede a un archivo es a través del Sistema operativo. Este provee de funciones como abrir archivo, leer información del archivo, guardar información, etc. No obstante, este procedimiento de acceso a archivos es altamente ineficaz cuando se trata con un volumen elevado de información. Es aquí donde aparecen los Sistemas Gestores de Bases de Datos: proporcionan un interfaz entre aplicaciones y sistema operativo, consiguiendo, entre otras cosas, que el acceso a los datos se realice de una forma más eficiente, más fácil de implementar y, sobre todo, más segura [7].

Objetivos de los SGBD

Existen distintos objetivos que deben cumplir los SGBD:

- Abstracción de la información. Los usuarios de 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 asegurada 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 seguridad 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.

1.3 Delegación

Hago mención a este tema, puesto que la delegación de actividades, es la parte fundamental de la existencia del SIGES (Sistema Gestor de Solicitudes). De manera general este sistema llevará la administración de la delegación de incidencias, hasta la persona responsable o que tenga el conocimiento suficiente para que estas puedan ser resueltas de una manera ordenada, pronta y con mayor control.

Ahora más que nunca hay que romper los mitos de la delegación tradicional. Las empresas actuales necesitan delegar más, la cantidad de personas que reportan a cada manager ha crecido, los tiempos disponibles se han acortado.

La información que se aporta a la gente debe ser fundamentalmente la necesaria. Si se brinda esa información, el gerente ejercerá más poder porque puede consagrar su tiempo y sus energías a actividades y resoluciones más importantes. La productividad y la eficacia de su área serán más elevadas porque será mayor la cantidad de personas que adopta decisiones y la gente estará más estimulada. La moral será más elevada, la productividad será alta como así también la eficiencia.

Es este tipo de información, en la cual se basará el gerente para una delegación más propia y tomar decisiones importantes del desempeño de su área, cito la importancia del **Sistema Gestor de Solicitudes (SIGES)**, para cuantificar en desempeño del personal y del monitoreo de las incidencias.


2
ANALISIS

2.1 Descripción General

Antecedentes

Este proyecto será realizado para la dependencia SOAPAP, la cual necesita automatizar el proceso de llevar el control de la cantidad de incidencias presentadas, contra las resueltas, en el monitoreo del Sistema Comercial.

Cuando se requiere la corrección en algún apartado del sistema de cobranza de la dependencia, es necesario crear un oficio con la solicitud, esperar a que sea firmado y aprobado, para después enviar la correspondencia. Una vez que el documento llega al área correspondiente, este es turnado a la persona responsable de hacer los cambios, pudiendo pasar varios días desde que la incidencia fue detectada.

Por tales razones dicha dependencia necesita de un software que permita la consulta vía Web de solicitudes hechas, para que los responsables de las áreas afectadas, creen órdenes de trabajo al personal a su cargo y así, resolver las incidencia de una manera más rápida, llevando un control y un histórico de estas.

Ámbito del software

El SIGES (Sistema de Gestor de Solicitudes), podrá llevar un control de las incidencias detectadas dentro de un proyecto, como y que tan eficaz fueron atendidas, las personas involucradas y el tiempo que implicó, pudiendo así reorganizar o replantear las actividades realizadas en un área.

2.2 Alcance del Proyecto

El SIGES (Sistema de Gestor de Solicitudes), será capaz de mostrar a cada usuario según su perfil, las solicitudes y/o órdenes de trabajo en espera de ser atendidas o ya resueltas, así como un histórico de los movimientos realizados a cada una de estas, para llevar el control de las incidencias captadas y cuantificar la eficiencia.

2.3 Objetivos

El objetivo general de SIGES es desarrollar un sistema que utilice una base datos relacional (dependiendo de la tarea asignada a cada empleado de un departamento) para su consulta precisa por computadora vía Web.

Los objetivos específicos del sistema son:

- Facilitar la captación de las incidencias, mejoras u observaciones del sistema comercial.
- Agilizar la asignación de órdenes de trabajo que resuelvan tales incidencias.
- Mostrar los tiempos de resolución para cada tarea y medir eficiencia.
- Facilitar la consulta de las órdenes de trabajo, tanto pendientes como o las resueltas o las que están en proceso.
- Facilitar la búsqueda de las órdenes creadas, por diferentes filtros.
- Controlar en los permisos de consulta y modificación, según el usuario.
- Llevar un historial de las modificaciones realizadas a cada orden de trabajo.
- Ahorro en tiempo de respuesta.
- Proporcionar al usuario un sistema amigable y confortable de consulta bajo un ambiente Web.

2.4 Restricciones del Sistema

- El usuario administrador, solo puede modificar los catálogos, no puede crear o modificar los estados de las solicitudes y/o órdenes de trabajo.
- Un usuario con permiso para generar solicitudes, no podrá cerrarla hasta que todas las órdenes de trabajo, derivadas de esta hayan sido resueltas.
- Un usuario con permisos normales, no podrá cerrar una solicitud, solo podrá responder una orden de trabajo generada por su coordinador.
- Un usuario con permiso de consulta, no puede realizar ningún movimiento.
- Cada coordinador de área partir de una solicitud, solo podrá asignar órdenes de trabajo al personal a su cargo o a otro coordinador de área.
- Solo se podrá cerrar una solicitud, si y solo si, todos las órdenes de trabajo derivadas de esta, hayan sido atendidas.

2.5 Descripción de Interfaces con Otros Sistemas

El sistema será desarrollado con una programación por capas, entiéndase por capas a un estilo de programación en la que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario.

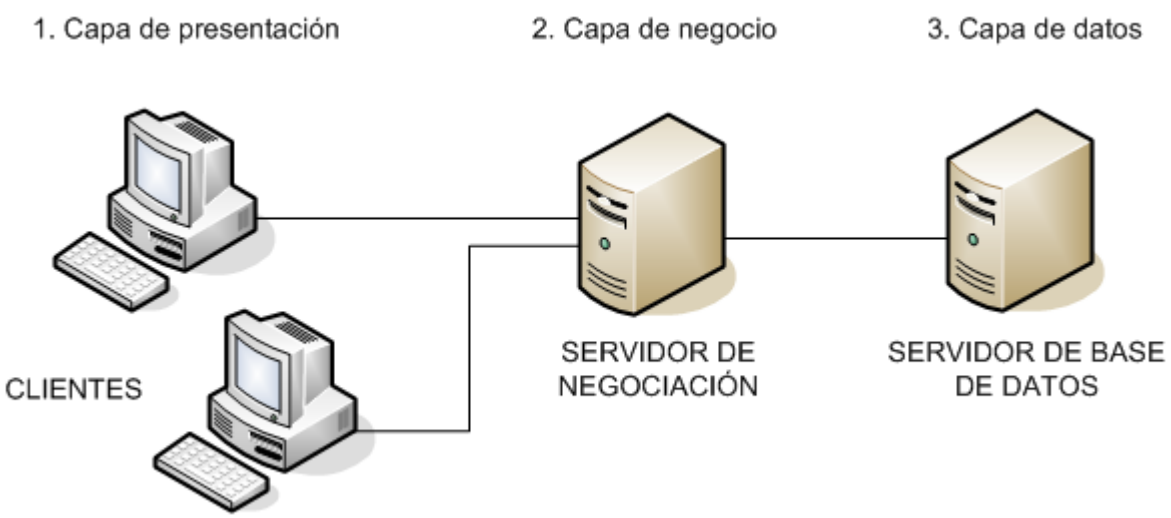


Figura 2.1 Programación a tres capas.

1.- Capa de presentación: es la que ve el usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no haya errores de formato). Esta capa se comunica únicamente con la capa de negocio [9]. Esta capa es realizada en Visual Studio 2005 con Microsoft ASP.NET 2.0.

2.- Capa de negocio: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos para almacenar o recuperar datos de él [9]. Esta capa es realizada en Visual C#.Net 2005.

3.- Capa de datos: es donde residen los datos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio [9]. Esta capa es realizada en Microsoft SQL Server Express 2005.

2.6 Descripción de la Información

La Ingeniería de Software está compuesta por una serie de pasos que abarcan los métodos, las herramientas y procedimientos. Estos pasos se denominan frecuentemente paradigmas de la ingeniería del software. La elección de un paradigma se lleva a cabo de acuerdo a la naturaleza de la aplicación, los métodos y herramientas a usar y controles y entregas requeridos. Para el desarrollo de este sistema se seleccionó el paradigma de Ciclo de vida clásico o secuencial, el cual nos lleva a desarrollar las siguientes etapas [10]:

Análisis:

- Descripción general
- Requerimientos del sistema
- Alcances del proyecto
- Objetivos
- Restricciones del proyecto
- Descripción de interfaces con otros sistemas
- Diagramas de flujos de datos
- Miniespecificaciones
- Especificaciones de control

Diseño:

- Diagrama de arquitectura
- Estructura de datos
- Diseño detallado de módulos
- Diagramas de eventos
- Diagramas de estados
- Algoritmos

Implementación y Pruebas:

- Aplicación del lenguaje de codificación
- Módulos relevantes codificados
- Prueba de la caja negra
- Prueba alfa
- Prueba beta

2.6.1 Diagramas de Flujo de Datos (DFD's)

La idea de analizar el flujo de datos es estudiar el empleo de datos en cada proceso al interior de una institución. Un Diagrama de Flujo representa la esquematización gráfica de un algoritmo, el cual muestra gráficamente los pasos o procesos a seguir para alcanzar la solución de un problema. Su correcta construcción es sumamente importante porque, a partir del mismo se escribe un programa en algún Lenguaje de Programación. Si el Diagrama de Flujo está completo y correcto, el paso del mismo a un Lenguaje de Programación es relativamente simple y directo.

Es importante resaltar que el Diagrama de Flujo muestra el sistema como una red de procesos funcionales conectados entre sí por " Tuberías " y "Depósitos" de datos que permite describir el movimiento de los datos a través del Sistema. Este describirá: lugares de origen y destino de los datos, transformaciones a las que son sometidos los datos, lugares en los que se almacenan los datos dentro del sistema, los canales por donde circulan los datos. Además de esto podemos decir que este es una representación reticular de un Sistema, el cual lo contempla en términos de sus componentes indicando el enlace entre los mismos.

Este análisis permite:

- Tener una visión de lo general a lo particular
- Comprender las relaciones entre sistemas y subsistemas
- Especificar, en forma rigurosa, procesos o transformaciones que ocurren en cada sistema
- Mejor comunicación entre los desarrolladores y los usuarios
- Separación lógica y física del flujo de información

Para desarrollar un diagrama de flujo de datos hay que seguir los siguientes pasos:

1. Hacer lista de actividades para determinar:
   a. Entidades externas
   b. Flujos de datos
   c. Procesos
   d. Almacenes de datos
2. Diagrama de contexto (Nivel 0)
3. Crear diagrama hijo por cada proceso de nivel 0
4. Revisar el modelo
5. Desarrollar DFD físico a partir del DFD lógico

A continuación se presenta una técnica gráfica (Diagrama de Flujo de Datos) para representar el flujo de información y las transformaciones que se aplican a los datos al moverse desde la entrada a la salida [10].

DFD [Nivel 0]

Figura 2.2 Diagrama de Flujo de Datos que muestra el Nivel 0.

DFD [Nivel 1]

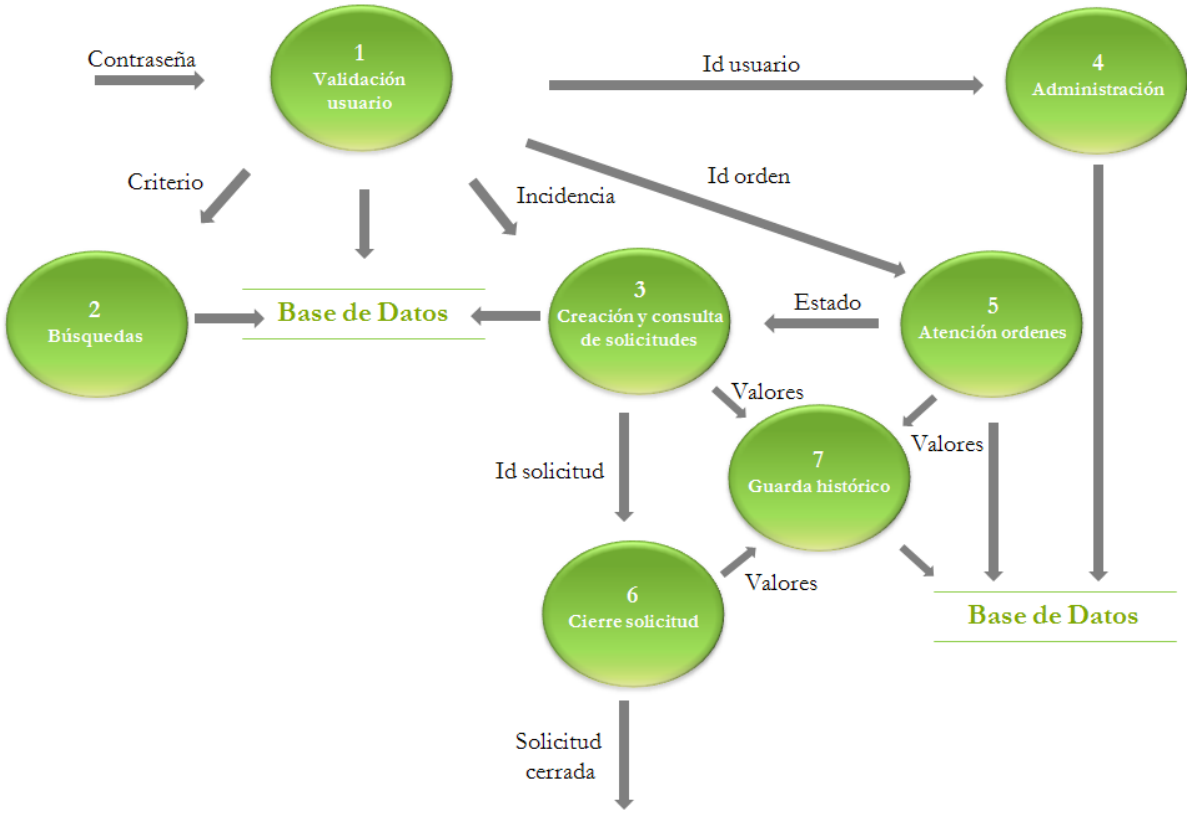


Figura 2.3 Diagrama de Flujo de Datos que muestra el Nivel 1.

DFD [Nivel 2]

1 Validación usuario

Figura 2.4 Diagrama de Flujo de Datos que muestra el Nivel 2, validación de usuario.

2 Búsqueda

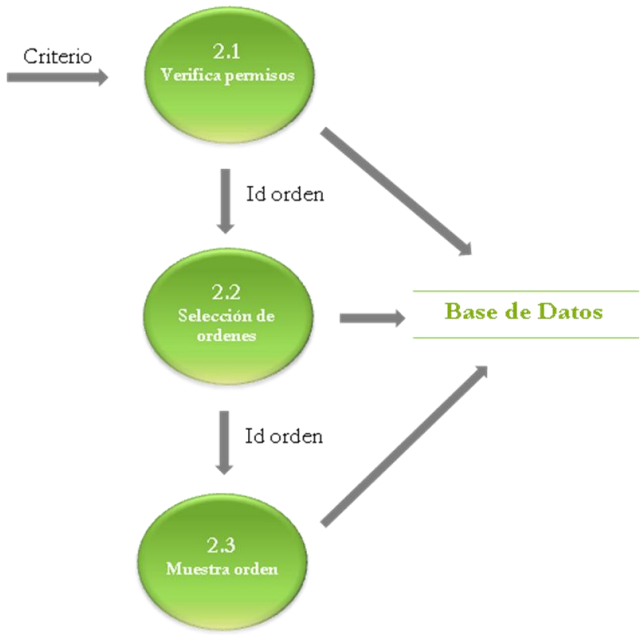


Figura 2.5 Diagrama de Flujo de Datos que muestra el Nivel 2, búsqueda.

3 Creación y consulta solicitudes

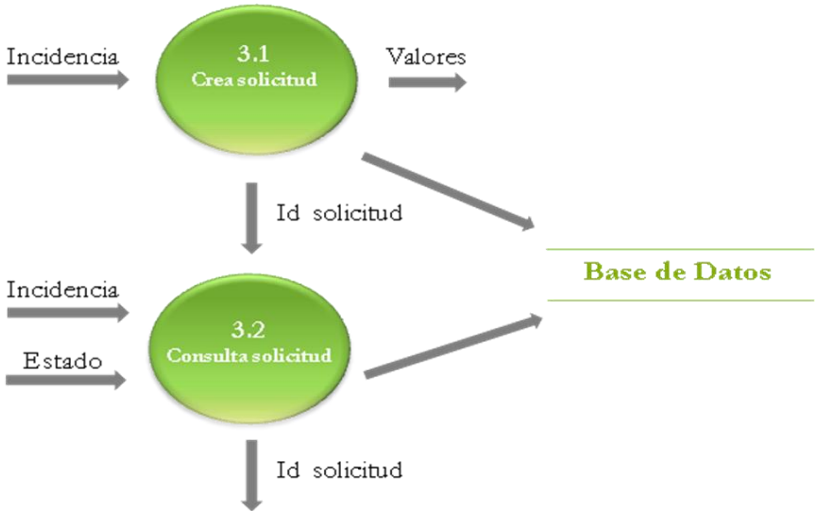


Figura 2.6 Diagrama de Flujo de Datos que muestra el Nivel 2, creación y consulta de solicitudes.

4 Administración

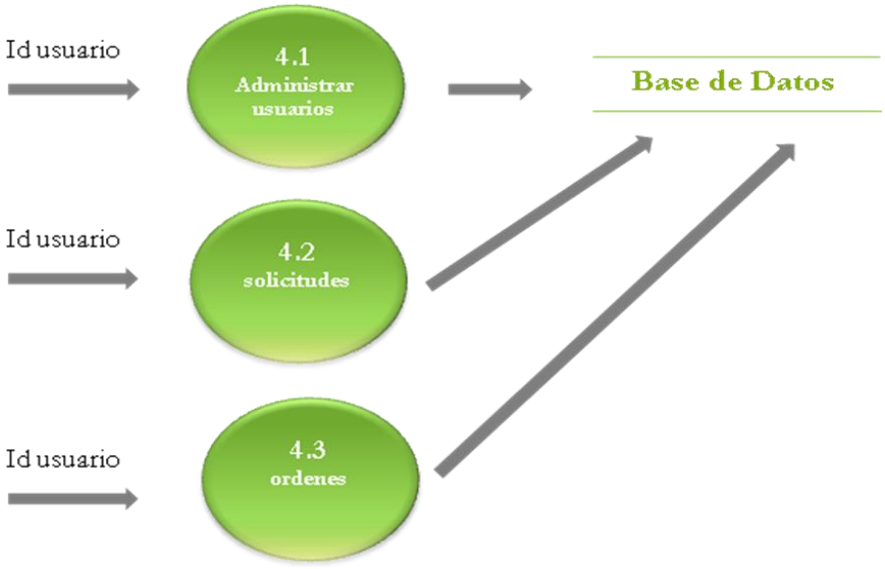


Figura 2.7 Diagrama de Flujo de Datos que muestra el Nivel 2, Administración.

5 Atención de órdenes

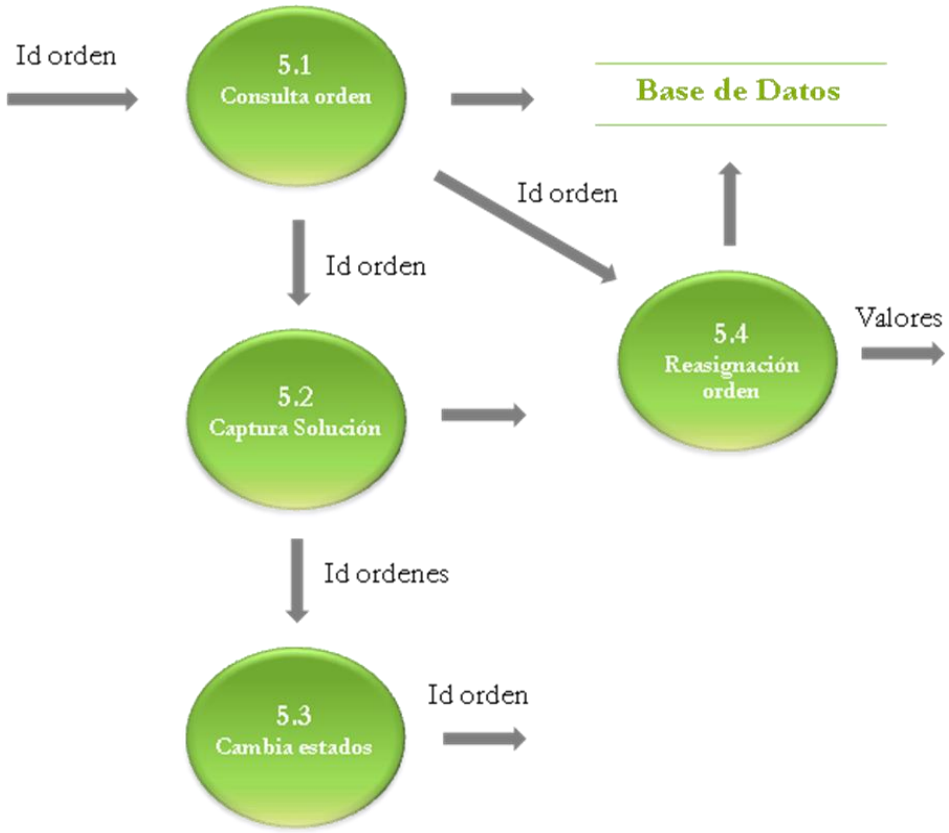


Figura 2.8 Diagrama de Flujo de Datos que muestra el Nivel 2, atención de órdenes.

6 Cierre solicitud

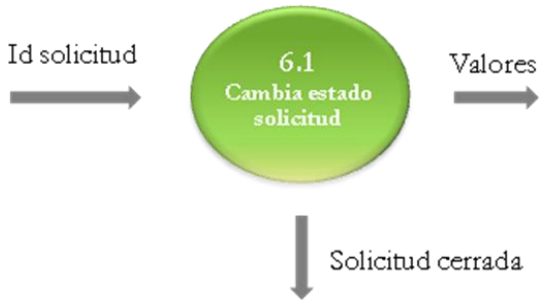


Figura 2.9 Diagrama de Flujo de Datos que muestra el Nivel 2, Cierre solicitud.

7 Guarda histórico

Figura 2.10 Diagrama de Flujo de Datos que muestra el Nivel 2, Guarda histórico.

DFD [Nivel 3]

4.1 Administrar usuarios

Figura 2.11 Diagrama de Flujo de Datos que muestra el Nivel 3, Administrar uaurios.

4.2 Solicitudes

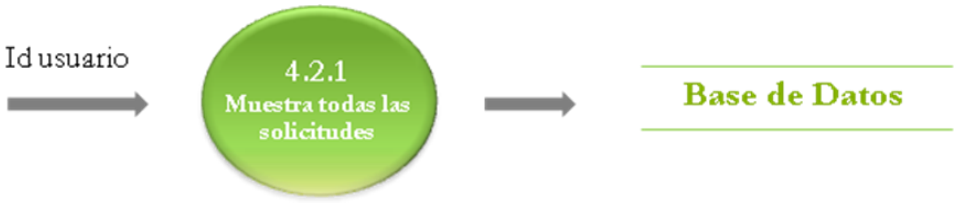


Figura 2.12 Diagrama de Flujo de Datos que muestra el Nivel 3, Solicitudes

4.3 Ordenes

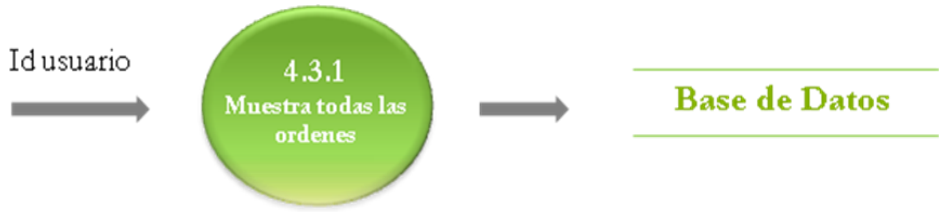


Figura 2.13 Diagrama de Flujo de Datos que muestra el Nivel 3, Ordenes.

5.3 Cambia estados

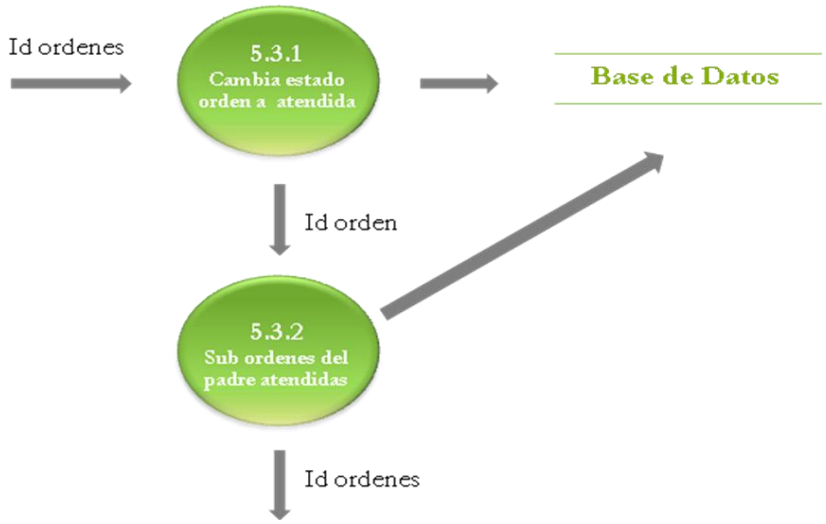


Figura 2.14 Diagrama de Flujo de Datos que muestra el Nivel 3, cambia estados.

5.4 Reasignación orden

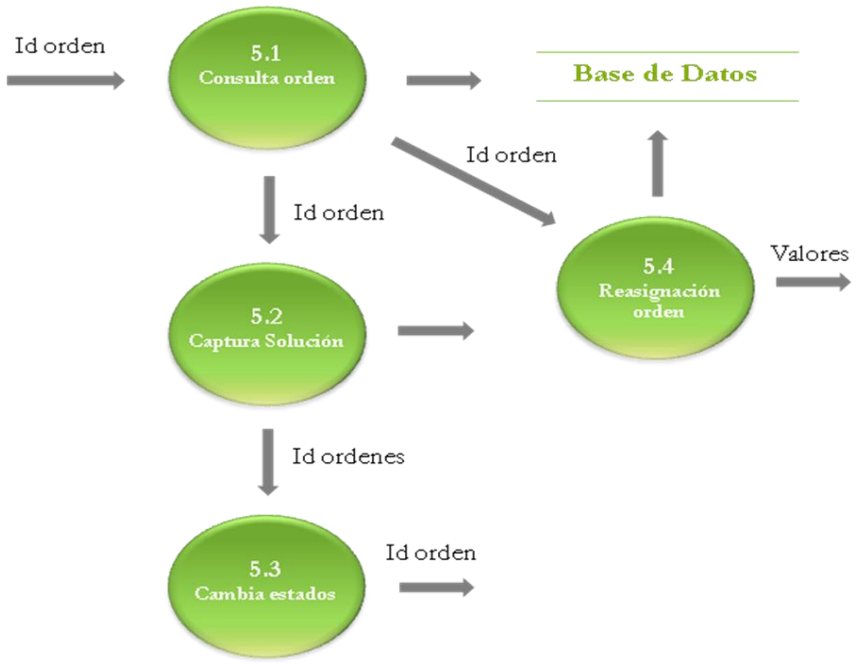


Figura 2.15 Diagrama de Flujo de Datos que muestra el Nivel 3, reasignación ordenes.

2.6.2 Diccionario de Datos

```
contraseña = {carcter_permitido}
    caracter_permitido = [A-Z | a-z | 0-9 |.| - |]


id_usuario = {numero_usuario}
    numero_usuario = [0-9]


criterio = {filtros_busqueda}
    filtros_busqueda = [nombre_usuario | no_empleado | fecha | tipo |estado |]
        nombre_usuario = [apellido_paterno | apellido_materno | nombre]
            apellido_paterno= {letras}
            apellido_materno = {letras}
            nombre = {letras}
            letras = [A-Z | a-z]
```

```
        no_empleado = {numero_nomina}
            numero_nomina = [0-9]
        fecha = dia + mes + año
            día = [1-31]
            mes = [1-12]
            año = [0-9]
        tipo = {numero_catalogo_tipo}
            numero_catalo_tipo = [1-9]
        estado = {numero_catalogo_estado}
            numero_catalo_estado = [1-9]


incidencia = *descripción de la problemática*
                {descripcion}
    descripción = [A-Z | a-z | Á | É | Í | Ó | Ú | á | é | í | ó | ú | , | . | : | ; | ¿ | ? |
                ¡ | ! | ( | ) | 0-9 |]


valores = {numero_historico} + {id_orden}
    numero_historico = [0-9]
    id_orden = {numero_orden}
        numero_orden = [0-9]


solicitud_cerrada = {id_solicitud}
    id_solicitud = {numero_solicitud}
        numero_solicitud = [0-9]



id_sesion = {numero_sesion}
    numero_sesion = [A-Z | 0-9 | # |]
```

2.6.3 Mini-Especificaciones

1. Validación usuarios: Verifica que el usuario esté dado de alta en la base de datos, y que la contraseña sea la correcta.
    1.1. Usuario valido: Verifica que el usuario este dado de alta en la base de datos, que la contraseña sea la correcta y su estado sea activo, para así asignarle una sesión.
    1.2. Asignación de permisos: Una vez validado el usuario y se inició una sesión, se cargará la configuración inicial, con los permisos correspondientes al usuario, los cuales, fueron ingresados a la alta del usuario.

2. Búsqueda: Muestra un listado de las órdenes ó solicitudes (según sean los permisos asignados al usuario) que le hayan sido asignadas.
    2.1. Verifica permisos: Verifica los permisos de cada usuario en la base de datos. Estos dependen del departamento al que son asignados.
    2.2. Selección de órdenes: Mostrará un listado de las órdenes o solicitudes para cada usuario.
    2.3. Muestra órdenes: Este mostrará una breve descripción, y datos de la orden ó solicitud.

3. Creación y consulta solicitudes: Esta es el inicio de toda orden. En este módulo solo podrán crear y consultar el avance de las solicitudes.
    3.1. Crea solicitud: Solo los usuarios con permisos para este módulo podrán crear solicitudes de incidencias.
    3.2. Consulta solicitudes: Mostrará el detalle de la solicitud.

4. Administración: En este módulo permitirá modificar los usuarios, mostrar las solicitudes y órdenes.
    4.1. Administración usuarios: Permite administrar los usuarios.
        4.1.1. Altas, Bajas y Modificaciones usuarios: Como su nombre lo dice, permite hacer altas, bajas y modificaciones de los permisos, nombre y contraseña de todos los usuarios que tienen acceso al sistema.
    4.2. Solicitudes: Muestra las solicitudes.
        4.2.1. Muestra todas las solicitudes: Muestra todas las solicitudes sin restricción alguna de las mismas.
    4.3. Ordenes: Muestra las órdenes

4.3.1. Muestra todas las órdenes: Muestra todas las órdenes sin restricción alguna de las solicitudes.

5. Atención de órdenes: Se podrá consultar, cambiar el estado ó reasignar una orden.
   5.1. Consulta orden: Se mostrará el contenido de las órdenes asignadas al usuario.
   5.2. Captura solución: Una vez que el problema fue solucionado, se capturará la solución de esta, para que el jefe de oficina evalúe si cierra esta orden o la vuelve a asignar.
   5.3. Cambia estados: Según el nivel de asignación se irán modificando los estados de una orden.
      5.3.1. Cambia estado orden atendida: Si una orden ya fue atendida al usuario se le activa la opción de orden atendida, la cual permite liberarla al usuario padre de esta orden.
      5.3.2. Subórdenes del padre atendidas: Al cambiarle el estado a esta orden verifica el estado de las órdenes padre de esta, por si fuera la última, cambia el estado de la suya y de su padre.
   5.4. Reasignación orden: Si se requiriera de la participación de alguien mas del departamento que dependa la solución del problema, esta orden se podrá reasignar a tantos usuarios como o permisos se tengan, para la solución de esta.
      5.4.1. Crea orden: Se crea una orden hijo de otra orden, por si se requiriera a más de un usuario involucrado para una misma solución.
      5.4.2. Atención de órdenes: Se vuelve cíclico, pues esta orden puede ser consultada, atendida y reasignada, si fuera el caso. Para un mejor entendimiento ver punto 5.

6. Cierre solicitud: Una vez que todas las orden asignadas a una solicitud, estén atendidas, se podrá cerrar una solicitud.
   6.1. Cambia estados: Según el nivel de asignación se irán modificando los estados de una solicitud: Para un mejor entendimiento ver punto 5.3

7. Guarda Histórico: Cada vez que se modifique el estado de una orden, así como o la reasignación de esta, será almacenada en esta tabla.
   7.1. Inserta valores: Inserción de los registros por modificación.


3
DISEÑO

3.1 Modelo Conceptual

Los diagramas E-R se utilizan como herramienta tanto en técnicas de análisis como de diseño en el proceso de construcción de software. Es una herramienta habitual en el paradigma funcional o tradicional de la Ingeniería del Software [10].
Formalmente, los diagramas E-R son un lenguaje gráfico para describir conceptos. Informalmente, son simples dibujos o gráficos que (si se saben interpretar) describen la información que trata un sistema y el software que lo automatiza.

El modelo Entidad Relación es un instrumento que permite modelar el mundo real en el proceso de diseño de las bases de datos. En él se pueden observar las entidades presentes en nuestro problema y los atributos que los caracterizan, así como las relaciones entre ellas.

Las entidades [10] aparecen representadas por un rectángulo, estas entidades representan los objetos de la base de datos, como son: Usuarios, Ordenes, Aéreas, Dependencias, Tipos, Estados, Histórico, etc.

Las relaciones [10] se representan a través de rombos donde se denota la conexión que existe entre entidades, como pueden ser entre usuarios y órdenes por una creación. Así como una orden es asociada a uno o varias áreas.

Los atributos [10] de los campos almacenan la información o propiedades de un objeto y se representan por una elipse, estos determinan las características de la entidad, por ejemplo, DN_NoEmpleado, DS_Nombre, DS_Login, DS_Password, etc.

Entidades **fuertes y débiles** [10]. Cuando una entidad participa en una relación puede adquirir un papel fuerte o débil. Una entidad débil es aquella que no puede existir sin participar en la relación, es decir, aquella que no puede ser unívocamente identificada solamente por sus atributos. Una entidad fuerte es aquella que sí puede ser identificada unívocamente. En los casos en que se requiera, se puede dar que una entidad fuerte "preste" algunos de sus atributos a una entidad débil para que, esta última, se pueda identificar.

Las entidades débiles se representan mediante un doble rectángulo, es decir, un rectángulo con doble línea.

La Cardinalidad [10] de las relaciones. El tipo de cardinalidad se representa mediante una etiqueta en el exterior de la relación, respectivamente: "1:1", "1:N" y "N:M", aunque la notación depende del lenguaje utilizado, la que más se usa actualmente es el unificado.

3.1.1 Diagrama Entidad Relación

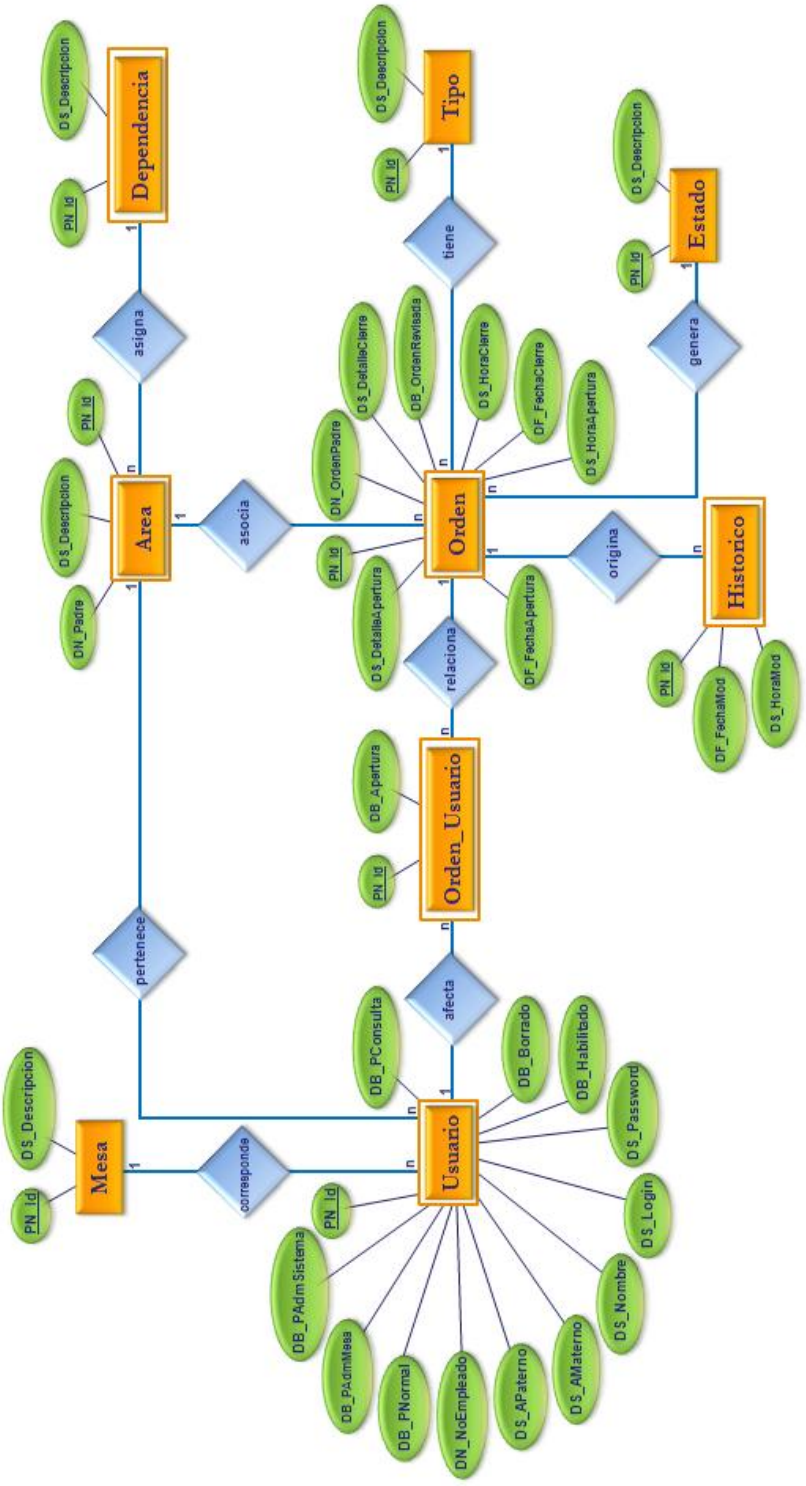


Figura 3.1 Diagrama Entidad Relación del SIGES usando notación de Peter Chen.

3.2 Modelo Relacional

El objetivo del modelo relacional es crear un "esquema" (schema), lo cual como se mencionará posteriormente consiste de un conjunto de "tablas" que representan "relaciones", relaciones entre los datos.

Estas tablas, pueden ser construidas de diversas maneras:

- Creando un conjunto de tablas iniciales y aplicar operaciones de normalización hasta conseguir el esquema más óptimo. Las técnicas de normalización se explican más adelante en este capítulo.
- Convertir el diagrama e-r a tablas y posteriormente aplicar también operaciones de normalización hasta conseguir el esquema óptimo.

La primera técnica fue de las primeras en existir y, como es de suponerse, la segunda al ser más reciente es mucho más conveniente en varios aspectos[12]:

- El partir de un diagrama visual es muy útil para apreciar los detalles, de ahí que se llame modelo conceptual.
- El crear las tablas iniciales es mucho más simple a través de las reglas de conversión.
- Se podría pensar que es lo mismo porque finalmente hay que "normalizar" las tablas de todas formas, pero la ventaja de partir del modelo e-r es que la "normalización" es mínima por lo general.
- Lo anterior tiene otra ventaja, aún cuando se normalice de manera deficiente, se garantiza un esquema aceptable, en la primera técnica no es así.

La Conversión de un modelo e-r a tablas, pude generar dos posibilidades[12]:

1.
    1. crear una tabla para el conjunto de entidades A de mayor nivel
        - columnas(A) = atributos(A)
    2. para cada conjunto de entidades B de menor nivel, crear una tabla tal que:
        - columnas(B) = atributos(B) U llave primaria(A)
2.
    1. si A es un conjunto de entidades de mayor nivel para cada conjunto de entidades B de menor nivel, crear una tabla tal que:
        - columnas(B) = atributos(B) U atributos(A)

Es importante mencionar que a pesar de que existen 2 métodos para convertir una generalización a tablas, no hay una regla exacta de cual usar en determinado caso. A continuación se mencionan algunos consejos útiles para la determinación de cual método emplear[12]:

- Si la entidad de nivel superior está relacionada con otra(s) entidades puede sugerirse emplear el método (1) ya que de esa manera la tabla (A) será la única involucrada en la relación, de otra forma se tendrían tres tablas (B1,B2 y B3) formando parte de la relación
- Es importante tomar en cuenta la pertenencia de instancias, si se considera que hablamos de una generalización disjunta, donde no se puede pertenecer a varias entidades de nivel inferior, quizás sea recomendable el método (1), en otro caso se podría pensar en el método (2).
- También es importante analizar ambos casos con respecto a las "consultas" que se deseen realizar ya que esto también determina en muchos casos el método a emplear.

Aplicando los términos que definen la estructura de datos relacionales, tomando como base el modelo entidad relación anterior, en los siguientes apartados, se muestra como se conforman cada una de las tablas.

Estado

PN_Id	DS_Descripcion
PK	

PN_Id: Campo que identifica cada estado.

DS_Descripcion: Nombre descriptivo del estado.

Tipo

PN_Id	DS_Descripcion
PK	

PN_Id: Campo que identifica cada tipo.

DS_Descripcion: Nombre descriptivo del tipo.

Dependencia

PN_Id	DS_Descripcion
PK	

PN_Id: Campo que identifica cada dependencia.

DS_Descripcion: Nombre de la dependencia.

Area

PN_Id	DS_Descripcion	DN_Padre	FN_IdDependencia
PK			FK

PN_Id: Campo que identifica cada área de una dependencia.

DS_Descripcion: Nombre del área.

DN_Padre: Campo que permitirá organizar áreas según su jerarquía.

Mesa

PN_Id	DS_Descripcion
PK	

PN_Id: Campo que identifica cada área de la mesa de ayuda.

DS_Descripcion: Nombre del área de la mesa de ayuda.

Usuario

PN_Id	DN_NoEmpleado	DS_APaterno	DS_AMaterno	DS_Nombre	DS_Login
PK					

DS_Password	DB_Habilitado	DB_Borrado	DB_PAdmSistema	DB_PAdmMesa

DB_PNormal	DB_PConsulta	FN_IdArea	FN_IdMesa
		FK	FK

PN_Id: Campo que identifica a cada usuario.

DN_NoEmpleado: Número de empleado asignado por nómina según su dependencia.

DS_APaterno: Campo reservado para el apellido paterno del usuario.

DS_AMaterno: Campo reservado para el apellido materno del usuario.

DS_Nombre: Campo reservado para el nombre o nombres del usuario.

DS_Login: Campo reservado para el nombre corto del usuario.

DS_Password: Campo reservado para un conjunto de caracteres seleccionado por el usuario.

DB_Habilitado: Campo booleano para validar si el usuario está habilitado.

DB_Borrado: Campo booleano para validar si el usuario fue borrado.

DB_PAdmSistema: Campo booleano para validar si el usuario tiene permiso de administrdor.

DB_PAdmMesa: Campo booleano para validar si el usuario tiene permiso de mesa de ayuda, los cuales pueden crear solicitudes.

DB_PNormal: Campo booleano para validar si el usuario tiene permiso de recibir y resolver órdenes de trabajo.

DB_PConsulta: Campo booleano para validar si el usuario tiene permiso sólo para consultar órdenes.

Orden

PN_Id	DN_OrdenPadre	DS_DetalleApertura	DS_DetalleCierre	DF_FechaApertura
PK				

DS_HoraApertura	DF_FechaCierre	DS_HoraCierre	DB_OrdenRevisada

FN_IdTipos	FN_IdEstados	FN_AreaCierre
FK	FK	FK

PN_Id: Campo que identifica a cada orden.

DN_OrdenPadre: Numero de orden que origina a otra.

DS_DetalleApertura: Breve descripción que origina la orden.

DS_DetalleCierre: Breve descripción de la solución de la orden.

DF_FechaApertura: Fecha en la que se creó la orden.

DS_HoraApertura: Campo reservado para la hora en que se creó la orden.

DF_FechaCierre: Fecha en la que se cerró la orden.

DS_HoraCierre: Campo reservado para la hora en que se cerró la orden.

DB_OrdenRevisada: Campo bandera que servirá para mostrar si una orden creada ya ha sido vista por el usuario a la que fue asignada.

Orden_Usuario

PN_Id	BD_Apertura	FN_IdOrdenes	FN_IdUsuario
PK		FK	FK

PN_Id: Campo que identifica cada estado.

BD_Apertura: Campo para validar si es un usuario de apertura o no.

Historico

PN_Id	DF_FechaMod	DS_HoraMod	FN_UsuMod	FN_IdEstados
PK				

FN_IdTipo	FN_IdOrden
	FK

PN_Id: Campo que identifica cada movimiento de una orden.

DF_FechaMod: Campo reservado para la fecha de modificación.

DS_HoraMod: Campo reservado para la hora de modificación.

3.2.1 Normalización

Su propósito es analizar los atributos de las entidades para confirmar que ellos están ubicados en la entidad correcta y que no son necesarias entidades adicionales, proveyendo una base para el diseño de la base de datos lógica que minimice los accesos a la base de datos. Para que un modelo de datos no tenga redundancia, y para prevenir anomalías de actualización, debe ser normalizado, esto es, que cada atributo de una entidad sea completamente dependiente de su identificador primario. Es el proceso de búsqueda de la forma en la cual los datos pueden ser trabajados independientemente, pero manteniendo sus relaciones. Permite optimizar el modelo conceptual. En otras palabras, es una representación de los datos en una forma más clara, sencilla, visual, y fácil de implementar. Una representación que contiene nada más que la información necesaria [10].

Primera Forma Normal (1FN) Remover Grupos Repetitivos

Poner el modelo en 1FN significa sacar de las entidades los atributos repetitivos (o grupos repetitivos) como entidades separadas. Por cada atributo que no es identificador, pregunte: ¿Hay más de un valor para este atributo en cualquier ocurrencia de la entidad? Identifique todos los atributos en la lista para los cuales la respuesta es "si". Busque el conjunto de tales atributos que siempre tienen el mismo número de valores dentro de cualquier ocurrencia individual de la entidad. Por cada conjunto de estos atributos, cree una nueva lista de atributos, y copie en esta lista el identificador primario. Para cada lista de este tipo que surja de este trabajo, seleccione su identificador primario. Este identificador primario incluirá al menos uno de los atributos de la lista, además del identificador primario de la entidad original (entidad "padre"). Repita este proceso hasta que no queden grupos de atributos respectivos en la entidad padre [10].

Por lo general la mayoría de las relaciones cumplen con estas características, así que podemos decir que la mayoría de las relaciones se encuentran en la primera forma normal.

Para ejemplificar la siguiente imagen (Figura 3.2) muestra una tabla en primera forma normal.

Area

PN_Id	DS_Descripcion	DN_Padre	FN_IdDependencia
PK			FK

Figura 3.2 Ejemplo de tabla en 1FN.

Como esta relación maneja valores atómicos, es decir un solo valor por cada uno de los campos que conforman a los atributos de las entidades, ya se encuentra en primera forma normal, gráficamente así representamos a las relaciones en 1FN

Segunda Forma Normal (2FN) - Remover las dependencias parciales

Es necesario normalizar en 2FN cuando se tiene una relación de M:N, la cual debe transformarse en dos relaciones de 1:N. Considere la lista de cada entidad que contiene identificadores primarios compuestos, y para cada atributo que no es identificador en la lista, pregunte: ¿El valor de este atributo depende completamente de los identificadores parciales? Si la respuesta es "no", saque el atributo de lista y determine los identificadores parciales de los cuales éste depende. Haga una lista separada para todos los atributos que dependen del (los) mismo (s) identificador (res) parcial (es). Cada lista que se crea forma la base de una nueva entidad, en la que debe copiarse el identificador parcial relevante de la entidad original. De un nombre a cada nueva entidad creada, documéntelas en el estándar definido, y seleccione un identificador primario. Elimine los calificativos heredados de la entidad original [10].

A manera de resumen puedo mencionar que 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. De acuerdo con está definición, cada tabla que tiene un atributo único como clave, esta en segunda forma normal.

La segunda forma normal se representa por dependencias funcionales como (figura3.3):

Orden

PN_Id	DN_OrdenPadre	DS_DetalleApertura	DS_DetalleCierre	DF_FechaApertura
PK				

DS_HoraApertura	DF_FechaCierre	DS_HoraCierre	DB_OrdenRevisada

FN_IdTipos	FN_IdEstados	FN_AreaCierre
FK	FK	FK

Figura 3.3 Ejemplo tabla en 2FN.

Tercera Forma Normal - Eliminar las Dependencias Transitivas

Para tener el modelo en 3FN deben resolverse las transitividades en los atributos (datos) repetidos, sacándolos como entidades independientes. Para todos los atributos que no son identificador, pregunte: ¿Los valores de este atributo dependen del valor de otro atributo cualquiera distinto del identificador primario? Si la respuesta es "si", el atributo dependiente debe ser trasladado a otra entidad, con el atributo determinante como identificador primario. Separe los atributos afectados y llévelos a una nueva entidad, la que deberá tener otro nombre, una por cada grupo de atributos con el mismo nuevo identificador primario. Las entidades resultantes estarán en tercera forma normal.

Sólo aquellos atributos que serán importantes (objeto de consultas) y aquellos más importantes por el usuario deben sacarse. La diferencia con respecto a la 1FN es que en este caso los datos repetidos salen como "padre", mientras que en la 3FN los datos repetitivos salen como "hijos" de la entidad original [10].

Si a caso esto parece complicado de identificar, este es un resumen en donde consiste en eliminar la dependencia transitiva que queda en una segunda forma normal. Entonces, una relación esta en tercera forma normal si está en segunda forma normal y no existen dependencias transitivas entre los atributos, nos referimos a dependencias transitivas cuando existe más de una forma de llegar a referencias a un atributo de una relación.

Por ejemplo, consideremos el siguiente caso (figura 3.4):

Orden_Usuario

PN_Id	BD_Apertura	FN_IdOrdenes	FN_IdUsuario
PK		FK	FK

Figura 3.4 Ejemplo de tabla en 3FN.

Con referencia a la información anterior puedo concluir que las tablas del sistema SIGES (Sistema Gestor de Solicitudes) están en Tercera Forma Normal (3FN). Como se muestra a continuación:

Estado

PN_Id	DS_Descripcion
PK	

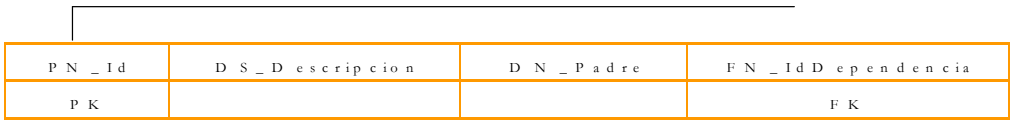
Tipo

PN_Id	DS_Descripcion
PK	

Dependencia

PN_Id	DS_Descripcion
PK	

Area

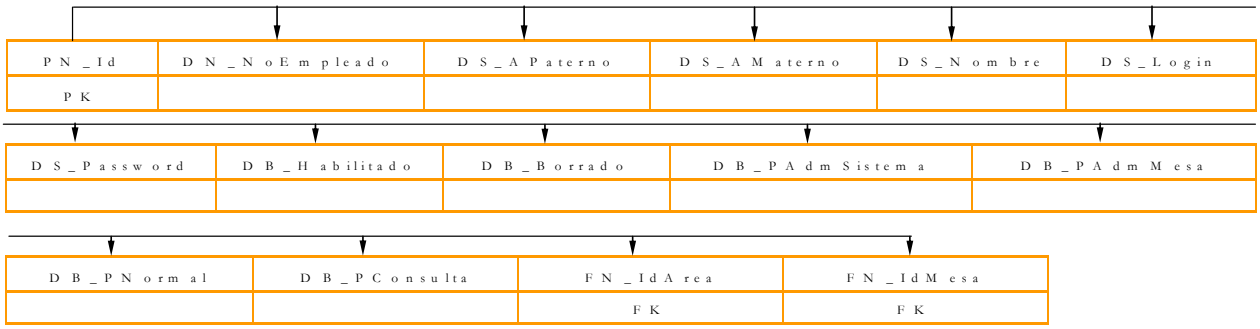


PN_Id	DS_Descripcion	DN_Padre	FN_IdDependencia
PK			FK

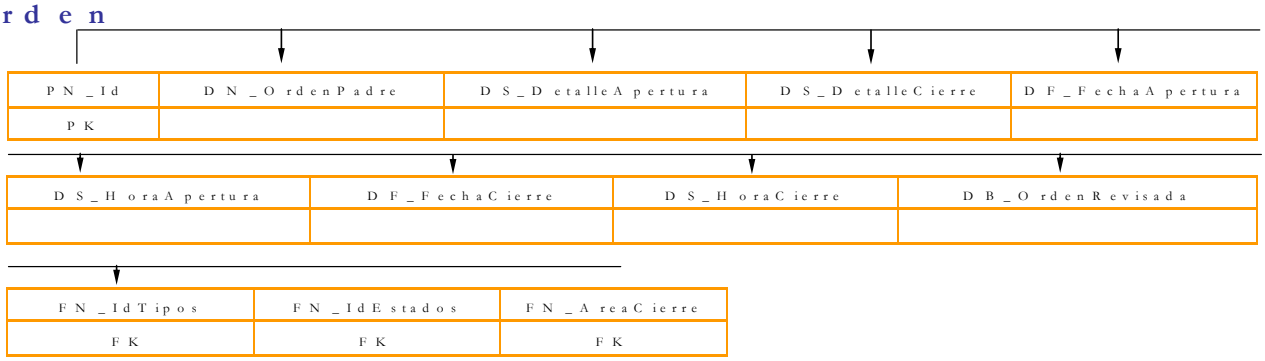
Mesa

PN_Id	DS_Descripcion
PK	

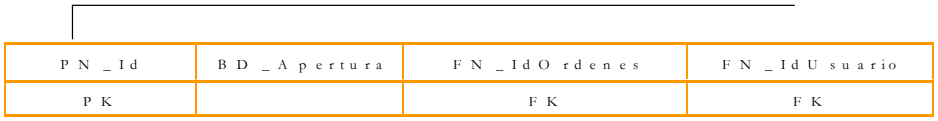
Usuario



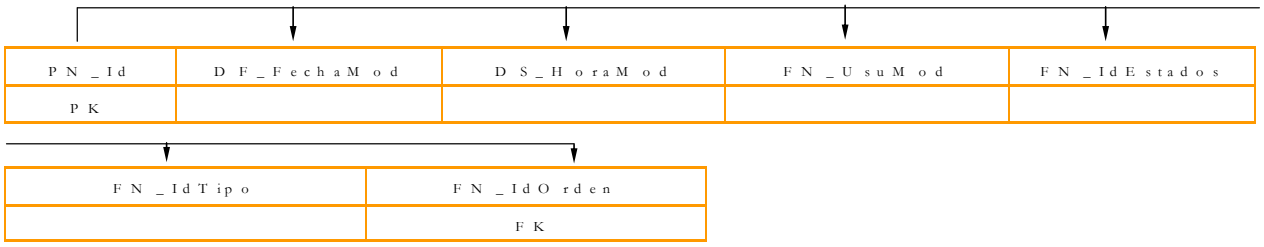
Orden



Orden_Usuario



Historico




4
IMPLEMENTACION

4.1 Implementación del Sistema

El Sistema Gestor de Solicitudes (SIGES) tiene como finalidad crear y resolver solicitudes que se convierten en órdenes de trabajo. Las solicitudes serán creadas por usuarios adscritos a la mesa de ayuda, los cuales son los únicos que podrán cerrar dicha solicitud ó reasignarla cuantas veces sean necesarias, para resolver la incidencia del sistema comercial.

Estas solicitudes abiertas, son asignadas a los jefes de áreas, las cuales son: DESARROLLO, FORMACION, SOPORTE y SOLICITUDES; cada una de estas áreas tienen actividades especificas que permitirán resolver diferentes tipos de incidencias.

Cada jefe de área podrá consultar todas sus solicitudes, las cuales se convierten en órdenes de trabajo, que podrá reasignar al personal a su cargo, según su criterio. A su vez cada uno de los miembros de cada área, si necesitara del apoyo ó contribución de alguien más para dar solución a dicha orden de trabajo, podrá reasignarla a otro compañero de la misma área.

Cada usuario podrá visualizar:

- Las órdenes que le fueron asignadas.
- Las ordenes que asignó.
- El usuario que le generó dicha orden.
- Fecha y hora de creación.
- Tipo de orden.
- Breve descripción de la orden o incidencia que la genera.
- Fecha y hora de cierre.
- Cuando asigna una orden, mostrará el estado de revisada cuando al usuario al que se le asignó ya la consultó.
- Un contador del tiempo que lleva asignada una orden sin resolver, el cual se detendrá cuando sea atendida dicha orden.

Una solicitud puede ser cerrada si y solo si, todas las órdenes de trabajo generadas de ella están resueltas. Es decir, las órdenes de trabajo asignadas a un miembro de cierta área, tiene que resolver su orden de trabajo para que el estado de la orden del jefe de área cambie a resuelta, para que la solicitud de la mesa de ayuda cambie a atendida y se pueda cerrar la solicitud.

El SIGES, cuenta con el usuario administrador, el cual puede consultar todas las solicitudes y órdenes de trabajo creadas a cualquier usuario. Así como realizar Altas, Bajas y Modificaciones a cualquier tipo de usuario.

El sistema utiliza la siguiente infraestructura:

Equipo de consulta:

Software:

- Navegador de Internet

Hardware:

- Conexión a Internet

Equipo de desarrollo:

Software:

- Microsoft Visual Studio 2005 Professional:
  - Visual C#
  - Visual Web Developer
  - Web Application Projects
- Microsoft .Net Framework 2.0
- Infragistics NetAdvantage 2006 Vol.1 for CLR 2.0
- Microsoft SQL Server 2005.
- Microsoft SQL Server Management Studio Express.
- Macromedia Fireworks 8.
- Microsoft Windows XP Professional SP3.

Hardware:

- Procesador: AMD a 1.7 Ghz.
- RAM: 768 MB.
- Disco Duro: de 20 GB (partición).

Equipo de publicación:

Software:

- Microsoft .Net Framework 2.0
- Infragistics NetAdvantage 2006 Vol.1 for CLR 2.0
- Microsoft SQL Server 2005.
- Microsoft Windows 2003 Server.
- IIS (Internet Information Server) versión 6.

Hardware:

- Procesador: Inter Xeon a 3.0 Ghz
- RAM: de 3.8 G.
- Disco Duro de 26 G.

4.2 Diseño de Interfaz y Ejemplos de Códigos

La primera pantalla (Figura 4.1) que es mostrada al usuario ingresará su usuario y contraseña. Aquí se verifica que sea un usuario válido, es decir, que esté dado de alta en la base de datos, que la contraseña sea correcta y esté habilitado para ingresar, tecleando caracteres válidos.

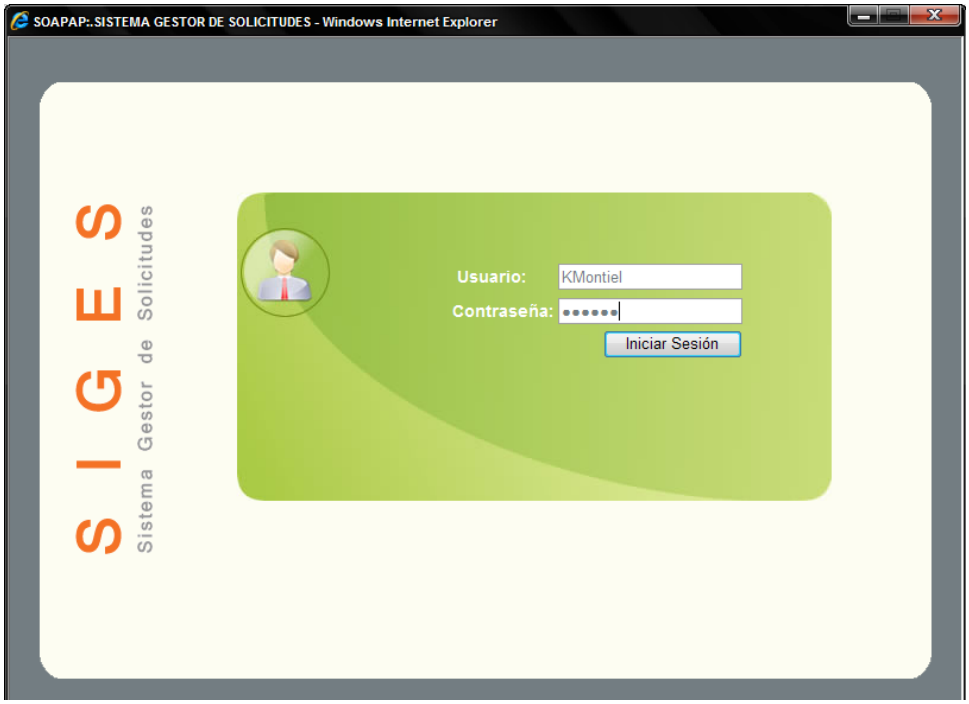


Figura 4.1 Pantalla principal.

Parte del código de esta ventana:

```
protected void BtnInicioSesion_Click(object sender, EventArgs e)
    {
        misUsuarios.BuscarTodos();
        String errorCnx = misUsuarios.testConexion();
        if (errorCnx == null)
            {
                if (validarEntrada())
                    {
                        stUsuarioFirmado st = new stUsuarioFirmado();
                        st.Id = misUsuarios.Seleccionado.Id;
```

```
                    st.AdminMesa = misUsuarios.Seleccionado.AdminMesa;
                    st.AdminSistema = misUsuarios.Seleccionado.AdminSistema;
                    st.ApellidoMaterno = misUsuarios.Seleccionado.ApellidoMaterno;
                    st.ApellidoPaterno = misUsuarios.Seleccionado.ApellidoPaterno;
                    st.Area = misUsuarios.Seleccionado.Area;
                    st.Borrado = misUsuarios.Seleccionado.Borrado;
                    st.Consulta = misUsuarios.Seleccionado.Consulta;
                    st.Habilitado = misUsuarios.Seleccionado.Habilitado;
                    st.Login = misUsuarios.Seleccionado.Login;
                    st.Mesa = misUsuarios.Seleccionado.Mesa;
                    st.NoEmpleado = misUsuarios.Seleccionado.NoEmpleado;
                    st.Nombre = misUsuarios.Seleccionado.Nombre;
                    st.Normal = misUsuarios.Seleccionado.Normal;
                    st.Password = misUsuarios.Seleccionado.Password;
                    st.UsuarioValido = misUsuarios.Seleccionado.UsuarioValido;
                    usuarioFirmado = st;
                    Response.Redirect("wfrMenu.aspx");
                }
            }
        else
                {
                 MuestraError(true);
                 lblError.Text = errorCnx;
                }
    }
#endregion
```

En esta pantalla (Figura 4.2) el usuario cumplió con los requisitos para iniciar sesión y a su vez, fueron habilitadas las opciones de menú, según su perfil. Este ejemplo es de una pantalla del menú de un usuario del área de mesa de ayuda, el cual tiene la opción de crear, consultar y cerrar solicitudes, las cuales generan órdenes de trabajo.

Figura 4.2 Pantalla de un usuario del área de mesa de ayuda en sesión.

Parte del código de esta ventana:

```
#region Propiedades

    protected usuarios misUsuarios
      {
        Get
          {
             if (Session["misUsuarios"] == null)
               {
                 Session["misUsuarios"] = new usuarios(
                 (AdmonTablasV1_0_1.SqlConectorBD)Application["cnxSEGURIDAD"]);
                 return (usuarios)Session["misUsuarios"];
                 }
             else
                  return (usuarios)Session["misUsuarios"];
          }
        set
        {  Session["misUsuarios"] = value;
        }
      }

    protected stUsuarioFirmado usuarioFirmado
      {
         get
           {
            if (Session["usuarioFirmado"] == null)
              {
                 Session["usuarioFirmado"] = new stUsuarioFirmado();
                 return (stUsuarioFirmado)Session["usuarioFirmado"];
              }
            else
                 return (stUsuarioFirmado)Session["usuarioFirmado"];
           }
           set { Session["usuarioFirmado"] = value; }
       }

#endregion

.
.
.

    protected void lnkAdmSol_Click(object sender, EventArgs e)
      {
         Response.Redirect("wfrCatSolicitudes.aspx");
      }

.
.
.
```

```
protected void LnkModPassword_Click(object sender, EventArgs e)
    {
        Response.Redirect("wfrCambiar_Clave.aspx");
    }
```

En esta pantalla (Figura 4.3) el usuario (con este permiso), podrá consultar todas las solicitudes por fecha de apertura, fecha de cierre, tipo ó estado, según sea el criterio deseado. También podrá crear y cerrar estas, si y solo si todas las órdenes de trabajo derivadas de esta se encuentren cerradas ó atendidas.

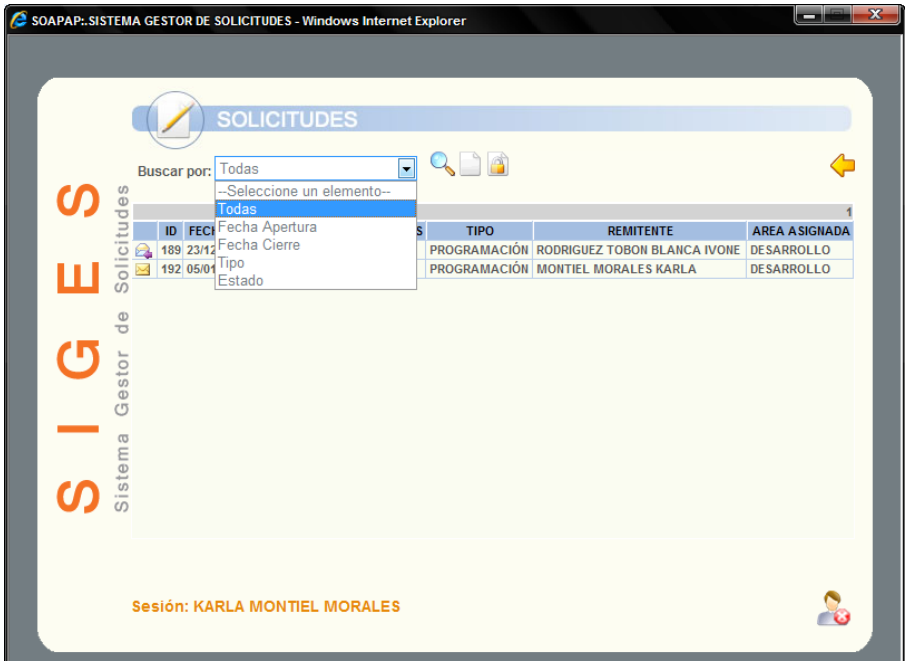


Figura 4.3 Pantalla de consulta de solicitudes.

Parte del código de esta ventana:

```
protected void buscar()
    {
        if (ddlFiltro.SelectedIndex != 0)
            {
                if (ddlFiltro.SelectedIndex == 1)
                  {
                    misOrdenes.BuscarTodosSolicitudes();
                    llenarUWG();
                  }
                if (ddlFiltro.SelectedIndex == 2||ddlFiltro.SelectedIndex == 3)
                  {
                    if (txtBuscar.Text != "" &
                          misOrdenes.esCadenaValida(txtBuscar.Text))
                      {
                        misOrdenes.BuscarPorFiltroSolicitudes(ddlFiltro.SelectedValue +
                        " LIKE ''" + txtBuscar.Text + "%''");
                        llenarUWG();
```

```
                }
            else
                {
                    uwgSolicitudes.Clear();
                    MuestraError(true);
                      lblError.Text = "Es necesario ingresar el criterio de búsqueda
                      valido";
                }
            }
        if (ddlFiltro.SelectedIndex == 4)
            {
              misOrdenes.BuscarPorFiltroSolicitudes(ddlFiltro.SelectedValue +
              " LIKE ''" + ddlSubFiltro2.SelectedValue + "%''");
              llenarUWG();
            }
          if (ddlFiltro.SelectedIndex == 5)
            {
              misOrdenes.BuscarPorFiltroSolicitudes(ddlFiltro.SelectedValue +
              " LIKE ''" + ddlSubFiltro.SelectedValue + "%''");
              llenarUWG()
            }
    }
  else
    {
      MuestraError(true);
      lblError.Text = "Es necesario ingresar un criterio de búsqueda";
        }
}
```

Este es un ejemplo de la creación de una solicitud (Figura 4.4). Como podemos ver muestra la fecha actual, el área a la que se desea asignar, una breve descripción de la incidencia que genera la solicitud y el tipo de solicitud.

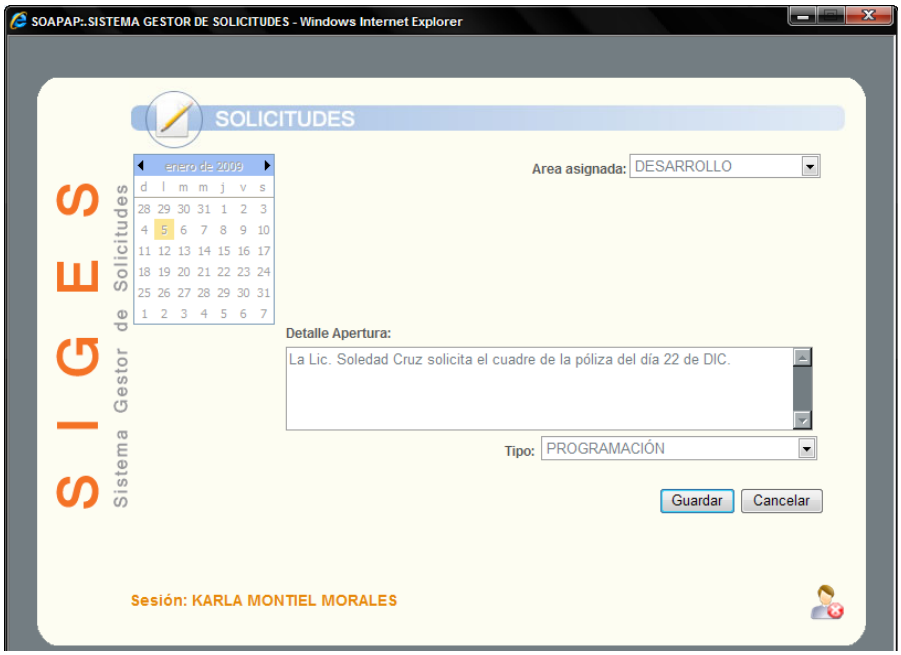


Figura 4.4 Pantalla de la creación de una solicitud.

Parte del código de esta ventana:

```
protected void btnGuardar_Click1(object sender, EventArgs e)
  {
     if (validacion())
     {
        if (Nuevo)
        {
           miOrden = new orden(misOrdenes.datosConector);
           actualizarValores(miOrden);
           try
             {
                if (usuarioFirmado.Id != 0)
                  {
                    if (!miOrden.Insertar(usuarioFirmado.Id))
                      {
                         MuestraError(true);
                         lblError.Text = "Error al insertar la solicitud";
                      }
                    aceptar();
                  }
                else
                  {
                     Response.Redirect("paginaError");
                  }
                  }
             catch (Exception)
               {
                  Response.Redirect("paginaError");
                  throw;
               }
             }
           else
             {
               if (misOrdenes.Seleccionado.UsuApertura == usuarioFirmado.Id)
                 {
                    if (misOrdenes.Seleccionado.Estado == 4 ||
                          misOrdenes.Seleccionado.Estado == 5)
                      {
                          actualizarValores(misOrdenes.Seleccionado);
                          if (!misOrdenes.Seleccionado.Actualizar
                                  (usuarioFirmado.Id,false))
                             {
                                MuestraError(true);
                                lblError.Text = "Error al cerrar la solicitud";
                             }
                           aceptar();
                      }
                 }
             }
        }
    }
```

.
.
.

```
private void actualizarValores(orden pOrden)
    {
        if (Nuevo)
            {
                pOrden.Estado = 1;
                pOrden.OrdenPadre = 0;
                pOrden.DetalleApertura = txtDetApertura.Text;
                pOrden.FechaApertura = DateTime.Now.ToShortDateString();
                pOrden.HoraApertura = DateTime.Now.Hour.ToString();
                pOrden.Tipo = Convert.ToInt32(drpTipoAp.SelectedValue);
                pOrden.UsuApertura = usuarioFirmado.Id;
                pOrden.AreaCierre = Convert.ToInt32(ddlsMesa.SelectedValue);
            }
        else
            {
                pOrden.Estado = 6;
                pOrden.FechaCierre = DateTime.Now.ToShortDateString();
                pOrden.HoraCierre = DateTime.Now.Hour.ToString();
            }
    }
```

Cualquier usuario puede cambiar su contraseña, en esta pantalla (Figura 4.5) es necesario ingresar la contraseña anterior correctamente, para poder ingresar una actual.

Figura 4.5 Pantalla de cambio de contraseña.

Parte del código de esta ventana:

```
protected void btnAceptar_Click(object sender, EventArgs e)
            {
                lblError.Text = "";
                if (validacion())
                {
                    miUsuario.Password = txtClaveNue.Text;
                    if (miUsuario.Actualizar(true))
```

```
                {
                    misUsuarios.BuscarPorFiltro("PN_Id=" + miUsuario.Id);
                    miUsuario = misUsuarios.Seleccionado;

                    stUsuarioFirmado st = new stUsuarioFirmado();
                    st.AdminMesa = miUsuario.AdminMesa;
                    st.AdminSistema = miUsuario.AdminSistema;
                    st.ApellidoMaterno = miUsuario.ApellidoMaterno;
                    st.ApellidoPaterno = miUsuario.ApellidoPaterno;
                    st.Area = miUsuario.Area;
                    st.Borrado = miUsuario.Borrado;
                    st.Consulta = miUsuario.Consulta;
                    st.Habilitado = miUsuario.Habilitado;
                    st.Id = miUsuario.Id;
                    st.Login = miUsuario.Login;
                    st.Mesa = miUsuario.Mesa;
                    st.NoEmpleado = miUsuario.NoEmpleado;
                    st.Nombre = miUsuario.Nombre;
                    st.Normal = miUsuario.Normal;
                    st.Password = miUsuario.Password;
                    usuarioFirmado = st;
                    lblError.ForeColor = System.Drawing.Color.Green;
                    lblError.Text = "¡La clave se cambio correctamente!";
                }
                else
                    lblError.Text = "Error al insertar";
            }
            else
                lblError.ForeColor = System.Drawing.Color.Red;
        }

.
.
.

protected void btnCancelar_Click(object sender, EventArgs e)
        {
            Response.Redirect("wfrMenu.aspx");
        }

.
.
.
```

En la siguiente pantalla (Figura 4.6) se muestra como una solicitud al asignarse a un usuario de vuelve una orden de trabajo. En este ejemplo el usuario Montiel Morales Karla generó una solicitud al usuario Victor Manuel Beristain Ocaña, el cual asignará una orden de trabajo al usuario Arturo López García, especificando una breve descripción de esta y el tipo. Tanto el usuario Arturo como Victor tienen que cerrar sus órdenes, las cuales tienen estructura de árbol, esto quiere decir que la orden de Arturo es hija de la orden de Victor, para poder cerrar la solicitud de Karla.

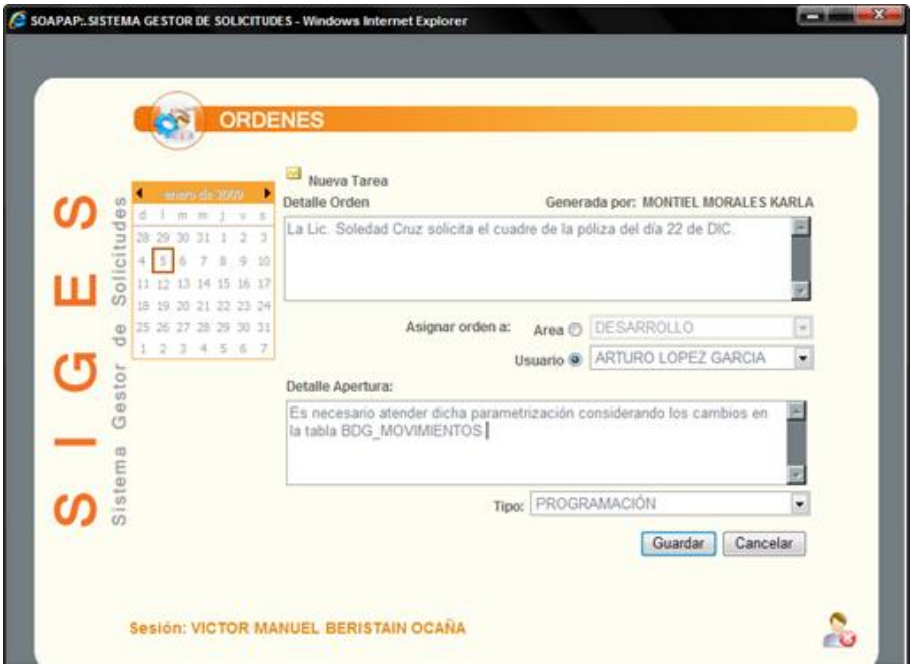


Figura 4.6 Asignación de una orden de trabajo a un usuario.

Parte del código de esta ventana:

```
protected void btnGuardar_Click(object sender, EventArgs e)
 {
    if (validacion())
     {
        miOrden = new orden(misOrdenes.datosConector);
        misOrdenes_padre = new ordenes(misOrdenes.datosConector);
        // Si es nuevo agrego
        if (Nuevo==1)
          {
            if (actualizarValores(miOrden, misOrdenes_padre))
               {
                  if (!miOrden.Insertar(usuarioFirmado.Id ))
                    {
                        MuestraError(true);
                        lblError.Text = "Error al insertar";
                    }

                    aceptar();
               }
          }
```

```
            else
                {
                    misOrdeneses_padre.BuscarTodosOrdeneses();
                    misOrdeneses_padre.seleccionar("PN_ID="  +
                    misOrdeneses.Seleccionado.OrdenPadre);
                    if (actualizarValores(misOrdeneses.Seleccionado, misOrdeneses_padre))
                      {
                          aceptar();
                      }
                }
            }
        }
```

.
.
.

```
private Boolean actualizarValores(orden pOrden, ordenes pOrdenPadre)
  {
    ordenes misOrdeneses_tmp;
    if (Nuevo==1)//Para agregar una orden
      {
        pOrden.Estado = 1;
        misOrdeneses.Seleccionado.Estado = 2;
        misOrdeneses.Seleccionado.Actualizar(usuarioFirmado.Id, false);

        pOrden.UsuApertura = usuarioFirmado.Id;
        pOrden.OrdenPadre = misOrdeneses.Seleccionado.Id;
        pOrden.DetalleApertura = txtDetApertura.Text;
        pOrden.FechaApertura = wcalFecha.SelectedDate.ToShortDateString();
        pOrden.HoraApertura = DateTime.Now.Hour.ToString() + ':' +
        DateTime.Now.Minute.ToString();

        if (rdoUsuario.Checked)
          {
            pOrden.UsuCierre = Convert.ToInt32(ddlstUsuario.SelectedValue);
          }
          if (rdoArea.Checked)
          {
            pOrden.AreaCierre = Convert.ToInt32(ddlsMesa.SelectedValue);
          }
          pOrden.Tipo = Convert.ToInt32(drpTipoAp.SelectedValue);
      }
      if (Nuevo==3)//Caso Para cerrar una orden
      {
        if (pOrden.Estado == 1 || pOrden.Estado == 3)// esta abierta o no hay
              hijos con estado pendiente
          {
              if(rbtnRealizada.Checked)
                    pOrden.Estado = 4;
              if (rbtnNoRealizada.Checked)
                    pOrden.Estado = 5;
              pOrden.DetalleCierre = txtDetCierre.Text;
              pOrden.FechaCierre = DateTime.Now.ToShortDateString();
              pOrden.HoraCierre = DateTime.Now.Hour.ToString() + ':' +
              DateTime.Now.Minute.ToString();
```

```
            if (!pOrden.Actualizar(usuarioFirmado.Id, false))
            {
                MuestraError(true);
                lblError.Text = "Error al actualizar la orden seleccionada";
                return false;
            }
            if (pOrden.OrdenPadre != 0)
            {
                misOrdenes_tmp = new ordenes(misOrdenes.datosConector);
                misOrdenes_tmp.BuscarPorFiltroOrdenes("DN_OrdenPadre = " +
                pOrden.OrdenPadre + " and FN_IdEstados in(1,2)");
                if (misOrdenes_tmp.totalregistros == 0)
                    pOrdenPadre.Seleccionado.Estado = 3;
                else
                    pOrdenPadre.Seleccionado.Estado = 2;
                pOrdenPadre.Seleccionado.Actualizar(usuarioFirmado.Id, false);
            }
        }
        else
            {
                MuestraError(true);
                lblError.Text = "No puede cerrar esta orden, aun tiene
                sub ordenes pendientes";
                return false;
            }
        }
        return true;
    }
```

Esta ventana (Figura 4.7), exclusiva del administrador del sistema. Realiza búsquedas, agrega nuevos usuarios, modifica o elimina usuarios.

Figura 4.7 Pantalla de administración de usuarios, en búsqueda de usuarios.

Parte del código de esta ventana:

```
protected void buscar()
    {
        if (ddlFiltro.SelectedIndex != 0)
            {
                if (ddlFiltro.SelectedIndex == 1)
                    {
                      misUsuarios.BuscarTodos();
                    }
                else
                    {
                      misUsuarios.BuscarPorFiltro(ddlFiltro.SelectedValue + " LIKE ''" +
                      txtUsuario.Text + "%''");
                    }
                uwgUsuarios.DataSource = misUsuarios.get_vistaPorDefault();
                uwgUsuarios.DataBind();
            }
            else
              {
                MuestraError(true);
                lblError.Text = "Debe seleccionar un criterio de busqueda";
              }
    }
```

En esta pantalla (Figura 4.8) se muestra la modificación de los datos de un usuario. Esta misma ventana y campos son los mismos para dar de alta un nuevo usuario.

Figura 4.8 Pantalla de modificación o alta de un usuario.

Parte del código de esta ventana:

```
protected void btnGuardar_Click(object sender, EventArgs e)
        {
            if (validacion())
            {
                // Si es un usuario nuevo agrego el nuevo registro
                if (Nuevo)
                {
                    miUsuario = new usuario(misUsuarios.datosConector);
                    actualizarValores(miUsuario);
                    if (!miUsuario.Insertar())
                    {
                        MuestraError(true);
                        lblError.Text = "Error al insertar";
                    }
                    else
                        aceptar();
                }
                else
                {
                    actualizarValores(misUsuarios.Seleccionado);
                    if (txtPassword.Text.Length > 0)
                    {
                        if (!misUsuarios.Seleccionado.Actualizar(true))
                        {
                            MuestraError(true);
                            lblError.Text = "Error al actualizar";
                        }
                        else
                            aceptar();
                    }
                    else
                    {
                        if (!misUsuarios.Seleccionado.Actualizar(false))
                        {
                            MuestraError(true);
                            lblError.Text = "Error al actualizar";
                        }
                        else
                            aceptar();
                    }
                }
            }
        }
```


5
CONCLUSIONES

5.1 CONCLUSIONES

El presente proyecto de tesis cumplió con los objetivos generales y particulares planteados. Se concluye que SIGES (Sistema Gestor de Solicitudes) es un sistema que como lo requiere SOAPAP (Sistema Operador de los Servicios de Agua Potable y Alcantarillado), cuenta con un ambiente de fácil acceso, esto gracias a que está hecho bajo una plataforma web, lo que facilita su utilización dentro de toda la organización por estar en la Intranet. Con solo teclear la dirección en el navegador se accede a este y se permite el ingreso del usuario y la contraseña. La actualización es inmediata en todos los clientes con solo actualizar la versión en el servidor, permitiendo también que la información de las incidencias pueda ser vista rápidamente. Se aprovechó la infraestructura actual de la dependencia, con esto no fue necesario comprar hardware ni para el servidor ni para cada uno de los equipos que accedan al sistema. De esta forma se obtuvo un ahorro en tiempo de respuesta, pues una incidencia es atendida en menor tiempo y canalizada con quien corresponda, documentando la causa que la originó, a quien le fue asignada y el tiempo que se tardó en resolver la misma.

Se decidió utilizar Visual Studio.Net - C#, ya que es la plataforma con que cuenta SOAPAP, además de ser un lenguaje seguro, por su estricta comprobación de tipos, destrucción de objetos y liberación de memoria por la existencia automática del recolector de basura, por la generación de su código MSIL (Microsoft Intermediate Language), no se tiene que preocupar por el sistema operativo o procesador en el que se va a ejecutar, ya que solo se genera código nativo y a medida que se va ejecutando se compila, ahorrando tiempo y memoria. Por otra parte se manejaron tres capas en el modelo para poder hacer mejores actualizaciones sin necesidad de manipular todo el proyecto. Otra razón es que el hardware y software que tiene actualmente el servidor (Windows 2003 Server e IIS (Internet Information Server) versión 6), es óptimo para esta plataforma de desarrollo.

5.2 PERSPECTIVAS

Este sistema se desarrolló de acuerdo a los requerimientos de la dependencia, tanto para el manejo de los catálogos como para la delegación de las órdenes de trabajo; pero tiene la opción de crecer según lo demanden los requerimientos de la dependencia.

Para poder hablar de la migración del sistema, se tendrían que analizar cada una de las capas del proyecto:

1.- Capa de presentación: es la que ve el usuario. En esta capa se utilizaron controles infragistics V6, por lo que sería necesario sustituirlos por los controles propios del lenguaje a migrar, por ejemplo Php.
2.- Capa de negocio: se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Esta capa se puede migrar totalmente para lenguajes de programación orientados a objetos, por ejemplo Java.

3.- Capa de datos: es donde residen los datos. Por la utilización del un estándar de Sql, es posible la migración de la base de datos tanto para software libre (MySQL), o como para los motores de bases de datos fuertes como Oracle.

Por lo anteriormente mencionado, cada una de las capas se puede migrar a software libre sin modificaciones considerables, con la restricción de que el sistema operativo sea Windows.

SIGES puede crecer con un módulo o proyecto adicional, en donde el usuario instale una aplicación local que constantemente esté verificando si tiene órdenes de trabajo pendientes de realizar, avisando con alertas.

BIBLIOGRAFIA

[1] Ruiz-Olalla, C. (2001): "Gestión de la calidad del servicio", [en línea] 5campus.com, Control de Gestión.

[2] Vásquez R.; Díaz A. (2001) "El conocimiento de las expectativas de los clientes: Una pieza clave de la calidad de servicio en el Turismo."

[3] Olga Ganser. Sistema de Gestión de Calidad. Articulo Wikilerning. Comunidades de Wikis libres para aprender.

[4] Administración de un departamento de sistemas.
http://www.monografias.com/trabajos/departservi/departservi.shtml

[5] Dr. Abraham Sánchez López., Ingeniería de Software Diplomado en computación, Facultad de Ciencias de la computación. Marzo 2004. Pué. Méx.

[6] Roger S. Pressman. Ingeniería del Software Un enfoque práctico, Tercera Edición, Mc Graw Hill, 19.

[7] MC. Alma Delia Ambrosio Vázquez. Introducción a los Sistemas de Bases de Datos. Diplomado en Bases de Datos, Facultas de Ciencias de la Computación, Octubre 2005, Puebla, México.

[8] Visión y Poder. Una fórmula eficaz para ganar en la crisis de las Pymes. Por el Lic. Raúl Miranda. Exclusivo para Holistica2000.

[9] http://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas

[10] Dra. María Josefa Somodevilla García., Análisis y Diseño de Base de Datos. Diplomado en computación, Facultad de Ciencias de las computación. Otoño 2004. Pué. Méx.

[11] "Sistemas de Bases de datos" Conceptos fundamentales, Elmasri / Navathe. Segunda Edición.

[12] "Introducción a las Bases de Datos" El Modelo Relacional. Olga Pons Capote, Nicolás Marín Ruiz, Juan Miguel Medina Rodríguez, Silvia Acid Carrillo, Mª Amparo.