UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFESIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

TESIS

"DESARROLLO DEL SISTEMA DE INFORMACIÓN MULTIUSUARIO PARA ADMISIÓN DEL HOSPITAL DE ILAVE – 2007"

PRESENTADA POR:

Bach. GLADYS MARLENY AUQUITIAS CONDORI

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ESTADÍSTICO E INFORMÁTICO

PUNO – PERÚ

2007

UNIVERSIDAD NACIONAL DEL ALTIPLANO – PUNO
FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERIA ESTADISTICA E INFORMÁTICA

"DESARROLLO DEL SISTEMA DE INFORMACIÓN MULTIUSUARIO PARA ADMISIÓN DEL HOSPITAL DE ILAVE – 2007"

TESIS

PRESENTADA POR:

Bach. GLADYS MARLENY AUQUITIAS CONDORI

A la Coordinación de la Facultad de Ingeniería Estadística e Informática de la Universidad Nacional del Altiplano, para optar el Título Profesional de: INGENIERO ESTADÍSTICO E INFORMÁTICO.

APROBADA POR:

PRESIDENTE:

Mg. AZANERO DE AGUIRRE EMMA O.

PRIMER MIEMBRO:

M.Sc. CARPIO VARGAS, EDGAR ELOY

SEGUNDO MIEMBRO:

M.Sc. VILLASANTE SARAVIA, FREDY HERIC

DIRECTOR:

M.Sc. QUISPE MAMANI, GODOFREDO

ASESOR:

M.Sc. HUATA PANCA, PERCY

Área : Informática
Tema : Sistemas de información
Fecha de Sustentación : 17/10/2007

DEDICATORIAS

A mis padres Alberto Auquitias Aquino

Y clara Condori Yucra porque creyeron y

Depositaron su entera confianza en cada reto

Que se me Presentaba sin dudar ni un solo

Momento de mí. Gracias por su comprensión y

Concejos.

A mi pareja Jhon, ya que siempre estuvo impulsándome en los momentos más difíciles con su valioso apoyo, sincero e incondicional.

Gracias por haber fomentado en mí el deseo de superación y el anhelo de triunfo en la vida.

AGRADECIMIENTOS

- A dios porque ha estado conmigo en cada paso que doy, cuidándome y dándome fortaleza para continuar.

- Un agradecimiento especial a mis padres, que nunca dejaron de creer en mí.

- Agradezco a los docentes, que marcaron con sus enseñanzas el futuro de todos nosotros. Gracias por prepararnos para un futuro competitivo no solo como los mejores profesionales sino también como mejores personas

ÍNDICE

CAPÍTULO I

PLAN DE INVESTIGACIÓN

CAPÍTULO II

MARCO TEÓRICO

ÍNDICE DE GRAFICOS

ÍNDICE DE TABLAS

ÍNDICE DE DIAGRAMAS

RESUMEN

El presente trabajo de investigación**,** se desarrolló en el Hospital de Ilave específicamente para el servicio de atención al usuario en Admisión. Es importante indicar que el área de admisión trabajaba anteriormente con una sola ventanilla lo cual demandaba mayor tiempo en la atención y por tanto se generaban colas extensas causando así malestar en el usuario. Razón por lo cual se ha desarrollado un Sistema de Información Multiusuario con el objetivo de optimizar la atención en el área de Admisión del hospital de Ilave. Para el desarrollo del sistema se utilizó el modelo de desarrollo espiral y el lenguaje de modelamiento unificado con lo cual se pudo generar los diagramas referentes al análisis y diseño del sistema. Ello nos llevó a la implementación y documentación del Sistema de Admisión, para la validación del sistema se hizo uso del ISO 9126 con lo cual se pudo verificar los requerimiento y realizar las pruebas de funcionalidad del sistema. Teniendo como resultado un sistema multiusuario con lo cual se generaron más puntos de atención en el área de Admisión. De esta manera se abordó uno de los principales problemas en la atención al usuario. Al culminar el presente trabajo de investigación se llegó a la conclusión de que el servicio de atención al paciente ha mejorado de esta manera que se pudo minimizar el tiempo de atención al usuario en el Hospital de Ilave.

Palabras clave: Base de Datos, Sistemas Multiusuario, Aplicaciones cliente Servidor y Sistemas de Información.

ABSTRACT

The present research work, was developed at the Hospital of Ilave. specifically for the service user at Admission Office. Is important indicate that this area previously worked with only one attention window which demanded more time and generate huge tails so causing discomfort users. The Reason for which we have developed a Multi Information System is optimizer the hospital admission area of Ilave. For system development we use spiral model development and unified modeling language which could generate diagrams concerning the analysis and system design was used. This led us to the implementation and documentation the System. The System Validation was done using the ISO 9126 thereby able to verify the requirements and testing of system functionality. Resulting in a multiuser system whereby it has more attention points were generated in the admission area. Thus addressed one of the main problems with customer service was solved. At last the present investigation was concluded that the service has improved patient so that could minimize the time of customer service at the Hospital of Ilave.

Key words: Database, Multi System, Client Server Applications and Information Systems.

INTRODUCCIÓN

Actualmente con el desarrollo de las tecnologías de la información y las Telecomunicaciones, los ordenadores pasaron a tomar un rol importante en el desempeño laboral del hombre. Específicamente en el manejo de la información digital mediante las automatizaciones cliente servidor. De esta manera se va ampliando las terminales de atención a los usuarios paralelizando la atención. el cual se aplica en distintas instituciones públicas y privadas que requieran múltiples puntos de atención.

El Hospital de Ilave - Redess Collao, es uno de ellos, específicamente el are de Admisión área en el cual se precisó de la actualización de su sistema a un sistema cliente servidor. Debido a que la población incrementa vertiginosamente el cual trae consigo la demanda de más usuarios al hospital de Ilave, esto implico contar con un Sistema Multiusuario para el Hospital de Ilave. Este trabajo de investigación aborda toda la problemática durante el desarrollo del Sistema. de acuerdo a los siguientes capítulos: Capitulo I el planteamiento del problema, justificación, objetivos y limitaciones, Capitulo II: se tomó como referencia de tesis trabajos anteriores con similares características. Capitulo III: se determinó el tamaño de la muestra, método de tratamiento de datos, se desarrolló el sistema de información multiusuario y se hizo la validación del software mediante el ISO 9126; Capitulo IV: finalmente se concluyó con los resultados obtenidos y conclusiones.

CAPÍTULO I

EL PROBLEMA DE INVESTIGACIÓN

1.1. PLANEAMIENTO DEL PROBLEMA

Debido que actualmente nos encontramos en la era del conocimiento, donde las computadoras pasaron a tomar mayor importancias en el desempeño laboral del hombre con el manejo de la información, mediante las automatizaciones de Base de Datos, En un principio Base de Datos de escritorio, ahora con el desarrollo de Sistemas de Bases de Datos Multiusuario, el procesamiento de los datos y administración de los mismos son más eficaces que los sistemas de escritorio, teniendo conocimiento de esto (Sistemas de Base de Datos Multiusuario) podemos decir que es necesario aplicarlo en distintas instituciones y lugares que requieran múltiples puntos de atención.

El hospital de Ilave-Redess Collao es una institución que está al servicio de la salud para la sociedad en general que tiene el objetivo de brindar el servicio de atención a los pacientes, por otro lado teniendo en cuenta que la población incrementa rápidamente el cual esto trae como consecuencia la demanda de pacientes en el hospital de Ilave y hace que más personas acudan al centro de salud.

Las personas que acuden al hospital de Ilave para poderse atender a cualquier servicio es necesario que tengan un numero de historia clínica y para ello, primero tienen que estar registrados en la Base de datos , para poderse registrarse pasan inicialmente por la ventanilla de admisión, entonces al registrarse es necesario que den sus datos personales si es su primera visita, su código de historia clínica en el caso que ya se hayan registraron anteriormente es decir no sea su primera visita, existen días en el que hay exceso de personas que acuden a los servicios de salud del hospital, en estos días se forman colas extensas y aglomeraciones, por tales motivos, esto nos da a conocer que se requiere en la unidad de servicio de admisión del hospital contar con más de una ventanilla de atención para reducir el tiempo de espera de los pacientes en el servicio de admisión, esto implica contar con un Sistema Multiusuario para el hospital de Ilave -Redess Collao.

Por otro lado, existe un sistema que se desempeña en la oficina de admisión denominado MINSA (tiene un interfaz en DOS) este sistema no es eficiente para la atención de múltiples ventanillas en la oficina de admisión porque su desempeño es monousuario y no cuenta con una capacidad de desempeño múltiple además cabe mencionar que no se desempeña en forma adecuada en el hospital de Ilave. Debido a que el MINSA se programó en forma genérica para los todos hospitales, también como consecuencia trae consigo que existan colas extensas aglomeraciones y como resultando hace que la atención a los pacientes sea más demoroso, por lo cual el hospital de Ilave requiere un sistema personalizado para la atención y adecuado a sus necesidades

Teniendo en cuenta estos puntos me he propuesto desarrollar un Sistema Multiusuario para la oficina de admisión del hospital de Ilave-Redess Collao, para que así la atención se realice con más de una ventanilla de atención, logrando ser más eficiente y reduciendo el tiempo de espera de los pacientes en el servicio de admisión.

FORMULACION Y DEFINICION DEL PROBLEMA

Teniendo en cuenta la necesidad anteriormente planteada desarrolle un sistema de información multiusuario para el hospital de Ilave, de tal manera que esto permite optimizar el tiempo de procesamiento de datos en la oficina de admisión mediante múltiples ventanillas (multiusuario) para facilitar el manejo de información, reducir el tiempo de espera de los pacientes y contar con un eficiente Sistema de Información.

Para lo cual me propuesto responder la siguiente pregunta:

¿De qué manera optimizar la atención en la oficina de admisión utilizando un Sistema de Información Multiusuario en el hospital de Ilave?

1.2. JUSTIFICACION

Los sistemas multiusuarios son de gran utilidad, actualmente la computadora es una herramienta importante que simplificando numerosos trabajos que tomaban mucho tiempo en realizarlos, además de hacer un manejo eficiente de la información el cual es un factor fundamental en el éxito y la competitividad de una empresa u organización, trayendo con ello innumerables tipos de sistemas de información para la realización de múltiples tareas.

Teniendo conocimiento que actualmente el hospital de Ilave cuenta con un sistema monousuario[1] y carece de operatividad en múltiples ventanillas no pudiendo así mantener una buena atención debido a las colas y aglomeraciones de los pacientes, se desarrolló un sistema multiusuario que apoye al desempeño en la oficina de admisión del hospital, este sistema estará distribuido en múltiples ventanillas logrando así reducir el tiempo de espera de los pacientes que acuden a este nosocomio.

1.3. OBJETIVOS

Para la ejecución del trabajo de investigación se ha propuesto los siguientes objetivos:

1.3.1. OBJETIVO GENERAL

Optimizar la atención de admisión utilizando un Sistema de Información Multiusuario en el hospital de Ilave.

1.3.2. OBJETIVOS ESPECÍFICOS

- Analizar y Diseñar un Sistema de Información Cliente Servidor para la oficina de admisión del hospital de Ilave.
- Implementar la base de datos en un servidor Interbase
- Prueba y test del Sistema de Información

[1] *http:// www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/c zap6-3.html*

1.4. HIPÓTESIS

El sistema de Información Optimiza la atención en admisión utilizando un Sistema de Información Multiusuario en el hospital de Ilave.

Prueba de Hipótesis:

Para la prueba de hipótesis se ha realizado una encuesta a los 10 trabajadores de la institución. **(Ver Anexo B)**

CAPÍTULO II

MARCO TEÓRICO

2.1. ANTECEDENTES DE LA INVESTIGACIÓN

Referente al tema de estudio y relacionados con el tema de investigación, existen trabajos realizados anteriormente en como tales son los siguientes:

Chique, A.(2003), "Desarrollo de Software para los Métodos de Líneas de Espera, Aplicada al Servicio de Admisión del Hospital Regional Manuel Núñez Butron " menciona que al Implantar un Servidor en el servicio de admisión (Ventanilla de atención), para así poder mejorar sus sistema de atención al cliente.

Condori, Q.(2005), "Desarrollo de Sistemas de Información Virtual para la Gestión Administrativa del CEGNE Sagrado Corazón de Jesús JULIACA-2003"indica Que la aplicación de Sistema de Información nos ayuda para tener una información Automatizada en el centro Educativo para tener una atención eficiente

Flores, V. (2004), Menciona que el desarrollo de un Sistema de Información mejora la atención que brindan a sus pacientes en el manejo de fichas clínicas, hoja de evolución, plan de trabajo y en administración los turnos en que deben ser atendidos los pacientes de "ESSALUD"

2.2. MARCO TEÓRICO

2.2.1. Definición de Sistemas de Información

"Todo sistema organizacional depende, en mayor o menor medida de una entidad abstracta denominada sistema de información."

Sistema de Información (SI):

Definición1:

Es un grupo de gente, una serie de procedimientos o equipo de procesamiento de datos que escoge, almacenan, procesan y recuperan datos para disminuir la incertidumbre en la toma de decisiones mediante el suministro de información a los niveles gerenciales para que sea utilizada eficientemente.

Definición2:

Es el medio por el cual los datos fluyen de una persona o departamento hacia otros y puede ser cualquier cosa, desde la comunicación interna entre los diferentes componentes de la organización y líneas telefónicas hasta sistemas de cómputo que generan reportes periódicos para varios usuarios.

Definición3:

Los sistemas de información es el medio por el cual se enlazan todos los componentes de un sistema para alcanzar el objetivo.

Sistemas de Información.-Un sistema de información es un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio.

El equipo computacional: el hardware necesario para que el sistema de información pueda operar.

El recurso humano que interactúa con el Sistema de Información, el cual está formado por las personas que utilizan el sistema.

Un sistema de información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información.

2.2.2. Características de los sistemas de información modernos:

- Sistemas sencillos sirviendo a funciones y niveles múltiples dentro de la empresa.
- Acceso inmediato en línea a grandes cantidades de información.
- Fuerte confiabilidad en la tecnología de telecomunicaciones.
- Mayor cantidad de inteligencia y conocimientos implícita en los sistemas.

La capacidad para combinar datos y gráficas.

2.2.3. Elementos de un sistema de información:

Un sistema de información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información.

a) Entrada de Información: Es el proceso[2] mediante el cual el Sistema de Información toma los datos que requiere para procesar la información. Las entradas pueden ser manuales o automáticas. Las manuales son aquellas que se proporcionan en forma directa por el usuario, mientras que las automáticas

[2] *< http://www.monografias.com/trabajos12/dispalm/dispalm.shtml>[Consulta:20 junio 2006]*

son datos o información que provienen o son tomados de otros sistemas o módulos. Esto último se denomina interfaces automáticas.

Las unidades típicas de entrada de datos a las computadoras son las terminales, las cintas magnéticas, las unidades de diskette, los códigos de barras, los escáner, la voz, los monitores sensibles al tacto, el teclado y el Mouse, entre otras.

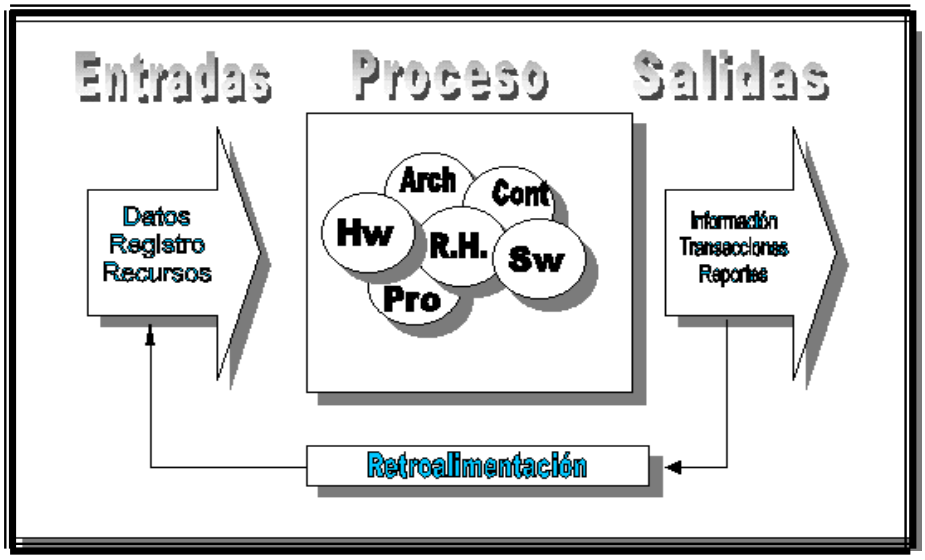
b) Almacenamiento de información.- El almacenamiento es una de las actividades o capacidades más importantes que tiene una computadora, ya que a través de esta propiedad el sistema puede recordar la información guardada en la sección o proceso anterior. Esta información suele ser almacenada en estructuras[3] de información denominadas archivos. La unidad típica de almacenamiento son los discos magnéticos o discos duros, los discos flexibles o diskettes y los discos compactos (CD-ROM).

c) Procesamiento de Información.- Es la capacidad del Sistema de Información para efectuar cálculos de acuerdo con una secuencia de operaciones preestablecida. Estos cálculos pueden efectuarse con datos introducidos recientemente en el sistema o bien con datos que están almacenados. Esta característica de los sistemas permite la transformación de datos fuente en información que puede ser utilizada para la toma de decisiones, lo que hace posible, entre otras cosas, que un tomador de decisiones genere una proyección financiera a partir de los datos que contiene un estado de resultados[4] o un balance general de un año base.

3< http://www.monografias.com/trabajos12/dispalm/dispalm.shtml> [Consulta:20 junio 2006]
4< http://www.monografias.com/trabajos5/estafinan/estafinan.shtml>[Consulta:20 junio 2006]

d) Salida de Información.- La salida es la capacidad de un Sistema de Información para sacar la información procesada o bien datos de entrada al exterior. Las unidades típicas de salida son las impresoras, terminales, diskettes, cintas magnéticas, la voz, los grafica dotes y los plotters, entre otros. Es importante aclarar que la salida de un Sistema de Información puede constituir la entrada a otro Sistema de Información o módulo. En este caso, también existe una interface automática de salida. Por ejemplo, el Sistema de Control de Clientes tiene una interface automática de salida con el Sistema de Contabilidad, ya que genera las pólizas contables de los movimientos procésales de los clientes

Grafico N° 01: Elementos de un sistema de Información



Fuente: **Referencia (2006)**

Un sistema de información es un conjunto de elementos que interactúan entre sí con el fin de apoyar las actividades de una empresa o negocio. Estos elementos son de Naturaleza diversa y normalmente incluyen:

- **El equipo computacional**, es decir, el hardware necesario para que el sistema de información pueda operar. Lo constituyen las computadoras y el equipo periférico que puede conectarse a ellas.
- **El recurso humano,** que interactúa con el Sistema de Información, el cual está formado por las personas que utilizan el sistema, alimentándolo con datos o utilizando los resultados que genere.
- **Los datos o información,** fuente que son introducidos en el sistema; son todas las entradas que necesita el sistema para generar como resultado la información que se desea.
- **Los programas,** que son procesados y producen diferentes tipos de resultados. Los programas son parte del software del sistema de información que hará que los datos de entrada introducidos sean procesados correctamente y generen los resultados que se esperan.

2.2.4. Clasificación de los sistemas de información:

A. Transaccionales (Sistemas transaccionales)

- Las principales características son:
- A través de éstos suelen lograrse ahorros significativos de mano de obra.
- Normalmente son el primer tipo de SI que se implanta en las organizaciones.
- Son intensivos en entrada y salida de información; sus cálculos y procesos suelen ser simples y poco sofisticados.
- Tienen la propiedad de ser recolectores de información.
- Son fáciles de justificar ante la dirección ya que sus beneficios son visibles y palpables.

B. Sistemas de Apoyo a las decisiones (Sistemas de Soporte a las Decisiones, Sistemas Gerenciales o Sistemas Ejecutivos, Sistema de Soporte para la Toma de Decisiones en Grupo.)

- Suelen introducirse después de haber implantado los sistemas transaccionales.
- Suelen ser intensivos en cálculos y escasos en entradas y salidas de información.
- La información que generan sirve de apoyo a los mandos intermedios y de alta administración en el proceso de la toma de decisiones.
- No suelen ahorrar mano de obra.
- La justificación económica para el desarrollo de estos sistemas es difícil.
- Suelen ser SI interactivos y amigables, con altos estándares de diseño gráfico y visual, ya que están dirigidos al usuario final.
- Apoyan la toma de decisiones que por su naturaleza son repetitivas.
- Pueden ser desarrollados directamente por el usuario final sin la participación operativa de los analistas.

C. Sistemas Estratégicos: (Sistemas Expertos, Sistemas Estratégicos)

- Su función principal no es apoyar a la automatización de procesos operativos ni proporcionar información para la toma de decisiones. Sin embargo, este tipo de sistemas puede llevar a cabo dichas funciones.
- Suelen desarrollarse “in mouse”.
- Típicamente su forma de desarrollo es a base de incrementos y a través de su evolución permanente dentro de la organización.

- Su función es lograr ventajas que los competidores no posean, tales como ventajas en costos y servicios diferenciados con clientes y proveedores.
- Apoyan el proceso de innovación dentro de la empresa.

Objetivos.

En la década de los noventas, los sistemas de información cumplirán dentro de las organizaciones tres objetivos básicos.

- ♦ Automatización de procesos operativos. (Sistemas transaccionales)
- ♦ Proporcionar información que sirva de apoyo al proceso de la toma de decisiones. (Sistemas de soporte a las decisiones)
- ♦ Lograr ventajas competitivas a través de su implementación y uso.(Sistemas estratégicos)

2.2.5. Objetivos generales de los SI

- La principal función de un SI es proporcionar a los encargados de la toma de decisiones, datos oportunos y exactos que les permitan tomar y aplicar las decisiones necesarias que mejoren al máximo la relación que existe entre los recursos de la empresa.
- Este sistema tiene el propósito general de ayudar a los gerentes en la planeación, control y toma de decisiones.
- Asegurar que la información exacta y confiable esté disponible cuando se necesite y que se le presente en forma fácilmente aprovechable.
- Incrementar la productividad operacional.
- Hacer que el proceso de información deje de ser información fragmentada, conjeturas inspiradas en la intuición y solución de problemas aislados.

En otras palabras, el objetivo primordial de un Sistema de Información Administrativo (SIA) es proporcionar a la empresa un mecanismo para el ejercicio de la administración.

Tendencias Futuras

El uso de la tecnología de información en las empresas se ha incrementado considerablemente y en un futuro será aún mayor. Las principales tendencias respecto a los Sistemas de Información son las siguientes:

- La tecnología de información se usará como parte de la estrategia corporativa, es decir, el uso de los Sistemas de Información que dan ventaja competitiva (sistemas estratégicos) se incrementará. Las empresas de más éxito serán manejadas por personas que sean capaces de desarrollar aplicaciones estratégicas de la tecnología de la Información de manera creativa.
- La tecnología será parte del trabajo en equipo en las empresas. Esta tecnología será usada para reducir el trabajo, mejorar la calidad, dar mejores servicios a los clientes o para cambiar la forma en que ser trabaja. Los trabajadores usarán las computadoras personales conectadas en red, y las fábricas usarán la tecnología para el diseño y control de producción.
- El uso de la tecnología transformará a la organización y cambiará su estructura. Como ejemplo de ello puede verse el uso del correo electrónico, el intercambio electrónico de datos y el acceso a información externa por medio de redes como internet.
- La tecnología facilitará la creación de las oficinas virtuales para las personas que requieren estar en diferentes localidades, permitiendo el uso del correo

electrónico y de conferencias por computadoras y de esta manera facilitar la comunicación global.

La tecnología de información apoyará de manera importante el rediseño de los procesos de negocios. Las técnicas de reingeniería de procesos continuarán apoyándose en los sistemas de información.

2.2.6. Definiciones de tecnología clientes servidor

IBM define al modelo cliente/servidor como la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas. El modelo soporta un medio ambiente distribuido en el cual los requerimientos de servicio hechos por estaciones de trabajo inteligentes o “clientes”, resultan en un trabajo realizado por otros computadores llamados servidores”.

Arquitectura.-Una arquitectura es un entramado de componentes funcionales que aprovechando diferentes estándares, convenciones, reglas y procesos, permite integrar una amplia gama de productos y servicios informáticos, de manera que pueden ser utilizados eficazmente dentro de la organización.

Debemos señalar que para seleccionar el modelo de una arquitectura, hay que partir del contexto tecnológico y organizativo del momento y, que la arquitectura Cliente/Servidor requiere una determinada especialización de cada uno de los diferentes componentes que la integran.

Cliente.- Es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

Servidor.- Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágenes, etc.

2.2.7. Ventajas y desventajas de la arquitectura cliente servidor

El esquema Cliente/Servidor posee las siguientes ventajas:

- Uno de los aspectos que más ha promovido el uso de sistemas Cliente/Servidor, es la existencia de plataformas de hardware cada vez más baratas. Esta constituye a su vez una de las más palpables ventajas de este esquema, la posibilidad de utilizar máquinas considerablemente más baratas que las requeridas por una solución centralizada, basada en sistemas grandes. Además, se pueden utilizar componentes, tanto de hardware como de software, de varios fabricantes, lo cual contribuye considerablemente a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones.

- El esquema Cliente/Servidor facilita la integración entre sistemas diferentes y comparte información permitiendo, por ejemplo que las máquinas ya existentes puedan ser utilizadas pero utilizando interfaces más amigables al

usuario. De esta manera, podemos integrar PCs con sistemas medianos y grandes, sin necesidad de que todos tengan que utilizar el mismo sistema operacional.

- Al favorecer el uso de interfaces gráficas interactivas, los sistemas construidos bajo este esquema tienen mayor interacción más intuitiva con el usuario. El uso de interfaces gráficas para el usuario, el esquema Cliente/Servidor presenta la ventaja, con respecto a uno centralizado, de que no es siempre necesario transmitir información gráfica por la red pues esta puede residir en el cliente, lo cual permite aprovechar mejor el ancho de banda de la red.

- Una ventaja adicional del uso del esquema Cliente/Servidor es que es más rápido el mantenimiento y el desarrollo de aplicaciones, pues se pueden emplear las herramientas existentes (por ejemplo los servidores de SQL o las herramientas de más bajo nivel como los sockets o el RPC).

- La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.

- El esquema Cliente/Servidor contribuye además, a proporcionar, a los diferentes departamentos de una organización, soluciones locales, pero permitiendo la integración de la información relevante a nivel global.

2.2.8. Sistema operativo multitarea

Es el modo de funcionamiento disponible en algunos sistemas operativos, mediante el cual una computadora procesa varias tareas al mismo tiempo.

Existen varios tipos de multitareas. La conmutación de contextos (context Switching)5 es un tipo muy simple de multitarea en el que dos o más aplicaciones se cargan al mismo tiempo, pero en el que solo se está procesando la aplicación que se encuentra en primer plano (la que ve el usuario). Para activar otra tarea que se encuentre en segundo plano, el usuario debe traer al primer plano la ventana o pantalla que contenga esa aplicación. En la multitarea cooperativa, la que se utiliza en el sistema operativo Macintosh, las tareas en segundo plano reciben tiempo de procesado durante los tiempos muertos de la tarea que se encuentra en primer plano (por ejemplo, cuando esta aplicación está esperando información del usuario), y siempre que esta aplicación lo permita. En los sistemas multitarea de tiempo compartido, como OS/2, cada tarea recibe la atención del microprocesador durante una fracción de segundo. Para mantener el sistema en orden, cada tarea recibe un nivel de prioridad o se procesa en orden secuencial. Dado que el sentido temporal del usuario es mucho más lento que la velocidad de procesamiento del ordenador, las operaciones de multitarea en tiempo compartido parecen ser simultáneas.

2.2.9. Sistemas de información multiusuario

Sistema Monousuario

Los sistemas monousuarios son aquellos que nada más puede atender a un solo usuario, gracias a las limitaciones creadas por el hardware, los programas o el tipo de aplicación que se esté ejecutando.

---

5 *Enciclopedia Guía del Estudiante (1996),” Nuevas Tecnologías” España 1996.*

Estos tipos de sistemas son muy simples, porque todos los dispositivos de entrada, salida y control dependen de la tarea que se está utilizando, esto quiere decir, que las instrucciones que se dan, son procesadas de inmediato; ya que existe un solo usuario. Y están orientados principalmente por los microcomputadores.

Sistema Multiusuario

Es todo lo contrario a monousuario; y en esta categoría se encuentran todos los sistemas que cumplen simultáneamente las necesidades de dos o más usuarios, que comparten mismos recursos. Este tipo de sistemas se emplean especialmente en redes.

En otras palabras consiste en el fraccionamiento del tiempo (timesharing).

2.2.10. Interbase

InterBase es un sistema administrador de bases de datos relacionales (Relational Data Base Management System - RDBMS), es decir, una aplicación encargada de administrar datos almacenados en bases de datos compuestas básicamente por tablas relacionadas entre sí.

2.2.11. Cliente - servidor interbase

Acceden a los datos por intermedio del RDBMS utilizando SQL.

Deben saber cómo conectarse al RDBMS y cómo utilizar SQL[6].

[6] Eduardo Ramírez *(2001),"SQL Server2000s" Editorial macro.*

El procesamiento de los datos lo hace el RDBMS. Por ejemplo, para mostrar un conjunto de datos filtrado, la aplicación debe solicitar el conjunto de datos filtrado.

El RDBMS aplica el filtro y envía el resultado.

Excelentes herramientas para mantener la integridad de los datos.

Excelentes herramientas para control de usuarios y seguridad.

Pensados para muchos usuario y muchos datos con alta concurrencia.

2.2.12. Conceptos de la metodología orientada a objetos

La metodología orientada a objetos ha derivado de las metodologías anteriores a éste. Así como los métodos de diseño estructurado realizados guían a los desarrolladores que tratan de construir sistemas complejos utilizando algoritmos como sus bloques fundamentales de construcción, similarmente los métodos de diseño orientado a objetos han evolucionado para ayudar a los desarrolladores a explotar el poder de los lenguajes de programación basados en objetos y orientados a objetos, utilizando las clases y objetos7 como bloques de construcción básicos.

Actualmente el modelo de objetos ha sido influenciado por un número de factores no sólo de la Programación Orientada a Objetos, POO (Object Oriented Programming, OOP por sus siglas en inglés). Además, el modelo de objetos ha probado ser un concepto uniforme en las ciencias de la computación, aplicable no sólo a los lenguajes de programación sino también al diseño de interfaces de usuario, bases de datos y arquitectura de

[7] < http://*www.adhocnet.cvaris/tutorial uml.htm*> *[Consulta:07 junio 2006]*

computadoras por completo. La razón de ello es, simplemente, que una orientación a objetos nos ayuda a hacer frente a la inherente complejidad de muchos tipos de sistemas.

Ventajas de la Metodología Orientada a Objetos

En síntesis, algunas ventajas que presenta son:

- **Reutilización.** Las clases están diseñadas para que se reutilicen en muchos sistemas. Para maximizar la reutilización, las clases se construyen de manera que se puedan adaptar a los otros sistemas. Un objetivo fundamental de las técnicas orientadas a objetos es lograr la reutilización masiva al construir el software.

- **Estabilidad.** Las clases diseñadas para una reutilización repetida se vuelven estables, de la misma manera que los microprocesadores y otros chips se hacen estables.

- **El diseñador piensa en términos del comportamiento de objetos y no en detalles de bajo nivel.** El encapsulamiento oculta los detalles y hace que las clases complejas sean fáciles de utilizar.

- Se construyen clases cada vez más complejas. Se construyen clases a partir de otras

- **clases, las cuales a su vez se integran mediante clases.** Esto permite construir componentes de software complejos, que a su vez se convierten en bloques de construcción de software más complejo.

- **Calidad.** Los diseños suelen tener mayor calidad, puesto que se integran a partir de componentes probados, que han sido verificados y pulidos varias veces.

- **Un diseño más rápido.** Las aplicaciones se crean a partir de componentes ya existentes. Muchos de los componentes están construidos de modo que se pueden adaptar para un diseño particular.

- **Integridad.** Las estructuras de datos (los objetos) sólo se pueden utilizar con métodos específicos. Esto tiene particular importancia en los sistemas cliente-servidor y los sistemas distribuidos, en los que usuarios desconocidos podrían intentar el acceso al sistema.

- **Mantenimiento más sencillo.** El programador encargado del mantenimiento cambia un método de clase a la vez. Cada clase efectúa sus funciones independientemente de las demás.

- **Una interfaz de pantalla sugestiva para el usuario.** Hay que utilizar una interfaz de usuario gráfica de modo que el usuario apunte a iconos o elementos de un menú desplegado, relacionados con los objetos. En determinadas ocasiones, el usuario puede ver un objeto en la pantalla. Ver y apuntar es más fácil que recordar y escribir.

- **Independencia del diseño.** Las clases están diseñadas para ser independientes del ambiente de plataformas, hardware y software. Utilizan solicitudes y respuestas con formato estándar. Esto les permite ser utilizadas en múltiples sistemas operativos, controladores de bases de datos, controladores de red, interfaces de usuario gráficas, etc. El creador del

software no tiene que preocuparse por el ambiente o esperar a que éste se especifique.

- **Interacción.** El software de varios proveedores puede funcionar como conjunto. Un proveedor utiliza clases de otros. Existe una forma estándar de localizar clases e interactuar con ellas. El software desarrollado de manera independiente en lugares ajenos debe poder funcionar en forma conjunta y aparecer como una sola unidad ante el usuario.
- **Computación Cliente-Servidor.** En los sistemas cliente-servidor, las clases en el software cliente deben enviar solicitudes a las clases en el software servidor y recibir respuestas. Una clase servidor puede ser utilizada por clientes diferentes. Estos clientes sólo pueden tener acceso a los datos del servidor a través de los métodos de la clase. Por lo tanto los datos están protegidos contra su corrupción.
- **Computación de distribución masiva.** Las redes a nivel mundial utilizarán directorios de software de objetos accesibles. El diseño orientado a objetos es la clave para la computación de distribución masiva. Las clases de una máquina interactúan con las de algún otro lugar sin saber donde residen tales clases. Ellas reciben y envían mensajes orientados a objetos en formato estándar.
- **Mayor nivel de automatización de las bases de datos.** Las estructuras de datos (los objetos) en las bases de datos orientadas a objetos están ligadas a métodos que llevan a cabo acciones automáticas. Una base de datos OO tiene integrada una inteligencia, en forma de métodos, en tanto que una base de datos relacional básica carece de ello.

- **Migración.** Las aplicaciones ya existentes, sean orientadas a objetos o no, pueden preservarse si se ajustan a un contenedor orientado a objetos, de modo que la comunicación con ella sea a través de mensajes estándar orientados a objetos.

- **Mejores herramientas CASE.** Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) utilizarán las técnicas gráficas para el diseño de las clases y de la interacción entre ellas, para el uso de los objetos existentes adaptados a nuevas aplicaciones. Las herramientas deben facilitar el modelado en términos de eventos, formas de activación, estados de objetos, etc. Las herramientas OO del CASE deben generar un código tan pronto se definan las clases y permitir al diseñador utilizar y probar los métodos recién creados. Las herramientas se deben diseñar de manera que apoyen el máximo de creatividad y una continua afinación del diseño durante la construcción.

2.2.13. Lenguaje de modelamiento unificado – UML

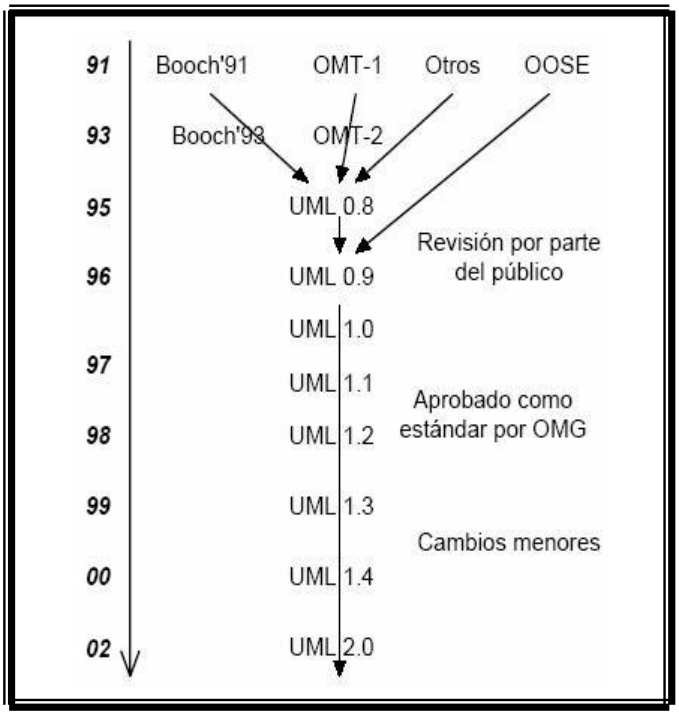
UML es el lenguaje de modelado de sistemas de software más conocido en la actualidad; es el estándar internacional aprobado por la OMG (Object Managment Group), consorcio creado en 1989 responsable de la creación, desarrollo y revisión de especificaciones para la industria del software.

Descripción de UML

El lenguaje UML comenzó a gestarse en octubre de 1994, cuando Rumbaugh se unió a la compañía Rational fundada por Booch (dos reputados investigadores en el área de metodología del software). El objetivo de ambos

era unificar dos métodos que habían desarrollado: el método Booch y el OMT (Object Modelling Tool). El primer borrador apareció en octubre de 1995. En esa misma época otro reputado investigador, Jacobson, se unió a Rational y se incluyeron ideas suyas. Estas tres personas son conocidas como los “tres amigos”. Además, este lenguaje se abrió a la colaboración de otras empresas para que aportaran sus ideas. Todas estas colaboraciones condujeron a la definición de la primera versión de UML.

Grafico N° 02: lenguaje de Modelado UML



Fuente: Referencia (2006)

Modelado: es el diseño de un software antes de su codificación, es la visualización de lo que se quiere construir.

Diagramas de UML

Diagrama de casos de uso

Se emplean para visualizar el comportamiento del sistema, una parte de él o de una sola clase. De forma que se pueda conocer cómo responde esa parte del sistema. El diagrama de uso es muy útil para definir cómo debería ser el comportamiento de una parte del sistema, ya que solo especifica cómo deben comportarse y no como están implementadas las partes que define. Por ello es un buen sistema de documentar partes del código que deban ser reutilizables por otros desarrolladores. El diagrama también puede ser utilizado para que los expertos de dominio se comuniquen con los informáticos sin llegar a niveles de complejidad. Un caso de uso especifica un requerimiento funcional, es decir indica esta parte debe hacer esto cuando pase esto.

En el diagrama nos encontramos con diferentes figuras que pueden mantener diversas relaciones entre ellas:

- **Casos de uso:** representado por una elipse, cada caso de uso contiene un nombre, que indique su funcionalidad. Los casos de uso pueden tener relaciones con otros casos de uso. Sus relaciones son:

  - ***Include*:** Representado por una flecha, en el diagrama de ejemplo podemos ver como un caso de uso, el de totalizar el coste incluye a dos casos de uso.
  - ***Extends*:** Una relación de una caso de Uso A hacia un caso de uso B indica que el caso de uso B implementa la funcionalidad del caso de uso A.
  - ***Generalización*:** Es la típica relación de herencia**.**

- **Actores:** se representan por un muñeco. Sus relaciones son:

  - ***Communicates*:** Comunica un actor con un caso de uso, o con otro actor.

- Parte del sistema (System boundary): Representado por un cuadro

Diagrama de clases

Forma parte de la vista estática del sistema. En el diagrama de clases como ya hemos comentado será donde definiremos las características de cada una de las clases, interfaces, colaboraciones y relaciones de dependencia y generalización. Es decir, es donde daremos rienda suelta a nuestros conocimientos de diseño orientado a objetos, definiendo las clases e implementando las ya típicas relaciones de herencia y agregación.

En el diagrama de clases debemos definir a estas y a sus relaciones.

La Clase

Una clase está representada por un rectángulo que dispone de tres apartados, el primero para indicar el nombre, el segundo para los atributos y el tercero para los métodos. Cada clase debe tener un nombre único, que las diferencie de las otras.

Un atributo representa alguna propiedad de la clase que se encuentra en todas las instancias de la clase. Los atributos pueden representarse solo mostrando su nombre, mostrando su nombre y su tipo, e incluso su valor por defecto.

Un método u operación es la implementación de un servicio de la clase, que muestra un comportamiento común a todos los objetos. En resumen es una función que le indica a las instancias de la clase que hagan algo.

Para separar las grandes listas de atributos y de métodos se pueden utilizar estereotipos.

Diagramas de objetos

Forma parte de la vista estática del sistema. En este diagrama se modelan las instancias de las clases del diagrama de clases. Muestra a los objetos y sus relaciones, pero en un momento concreto del sistema. Estos diagramas contienen objetos y enlaces. En los diagramas de objetos también se pueden incorporar clases, para mostrar la clase de la que es un objeto representado.

En este diagrama se muestra un estado del diagrama de eventos. Para realizar el diagrama de objetos primero se debe decidir qué situación queremos representar del sistema. Es decir si disponemos de un sistema de mensajería, deberemos decidir que representaremos el sistema con dos mensajes entrantes, los dos para diferentes departamentos, dejando un departamento inactivo. Para el siguiente diagrama de clases:

Diagrama de componentes

Se utilizan para modelar la vista estática de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. En el situaremos librerías, tablas archivos, ejecutables y documentos

que formen parte del sistema. Uno de los usos principales es que puede servir para ver que componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema.

Como ya hemos indicado antes todo objeto UML puede ser modificado mediante estereotipos, los standards que define UML son:

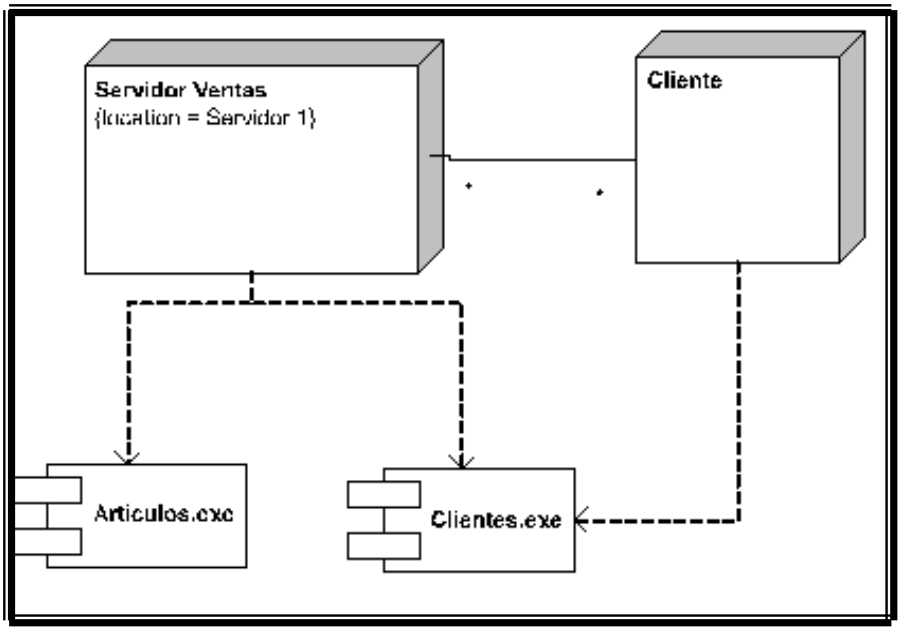
- Executable
- Library
- Table
- File
- Document

Diagramas de despliegue

En el diagrama de despliegue se indica la situación física de los componentes lógicos desarrollados. Es decir se sitúa el software en el hardware que lo contiene. Cada Hardware se representa como un nodo.

Un nodo se representa como un cubo, un nodo es un elemento donde se ejecutan los componentes, representan el despliegue físico de estos componentes.

Diagrama N° 01: Representación gráfica del Diagrama de Despliegue



Fuente: Referencia (2006)

Aquí tenemos dos nodos, el cliente y el servidor, cada uno de ellos contiene componentes. El componente del cliente utiliza un interface de uno de los componentes del servidor. Se muestra la relación existente entre los dos Nodos. Esta relación podríamos asociarle un estereotipo para indicar que tipo de conexión disponemos entre el cliente y el servidor, así como modificar su cardinalidad, para indicar que soportamos diversos clientes.

Como los componentes pueden residir en más de un nodo podemos situar el componente de forma independiente, sin que pertenezca a ningún nodo, y relacionarlo con los nodos en los que se sitúa.

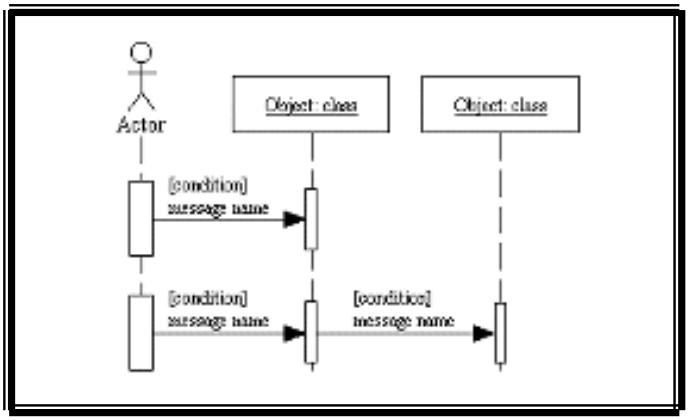
Diagramas secuencia

El diagrama de secuencia forma parte del modelado dinámico del sistema. Se modelan las llamadas entre clases desde un punto concreto del sistema. Es útil para observar la vida de los objetos en sistema, identificar llamadas a realizar o

posibles errores del modelado estático, que imposibiliten el flujo de información o de llamadas entre los componentes del sistema.

En el diagrama de secuencia se muestra el orden de las llamadas en el sistema. Se utiliza un diagrama para cada llamada a representar. Es imposible representar en un solo diagrama de secuencia todas las secuencias posibles del sistema, por ello se escoge un punto de partida. El diagrama se forma con los objetos que forman parte de la secuencia, estos se sitúan en la parte superior de la pantalla, normalmente en la izquierda se sitúa al que inicia la acción. De estos objetos sale una línea que indica su vida en el sistema. Esta línea simple se convierte en una línea gruesa cuando representa que el objeto tiene el foco del sistema, es decir cuando él está activo.

Diagrama: N° 02: Representación gráfica del diagrama de secuencia



Fuente: Referencia (2006)

En esta pantalla tenemos tres objetos, y un actor, situado a la izquierda que es el que inicia la acción. Como podemos ver el objeto de más a la derecha aparece más abajo que los otros existentes. Esto se debe a que recibe una llamada de creación. Es decir el objeto no existe en el sistema hasta que recibe la primera petición.

Relación de RUP Y UML

RUP y UML están estrechamente relacionados entre sí, pues mientras el primero establece las actividades y los criterios para conducir un sistema desde su máximo nivel de abstracción (la idea en la cabeza del cliente), hasta su nivel más concreto (un programa ejecutándose en las instalaciones del cliente), el segundo ofrece la notación gráfica necesaria para representar los sucesivos modelos que se obtienen en el proceso de refinamiento**.**

2.2.14. El Proceso Racional Unificado o Rup

(Rational Unified Process), es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. RUP es en realidad un refinamiento realizado por Rational Software del más genérico Proceso Unificado[8]Sus principales características son:

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo)
- Pretende implementar las mejores prácticas en Ingeniería de Software
- Desarrollo iterativo
- Administración de requisitos
- Uso de arquitectura basada en componentes
- Control de cambios

[8] *http://es.wikipedia.org/wiki/Rational_Unified_Process*

- Modelado visual del software
- Verificación de la calidad del software

El RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

El RUP divide el proceso de desarrollo en ciclos, teniendo un producto final al final de cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante:

A) inicio: se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos

B) elaboración: se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos

C) construcción: se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario

D) transición: se implementa el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los

Objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

En la metodología Rup se toma en consideración los Siguientes Puntos:

Ingeniería de Negocios: Entendiendo las necesidades del negocio.

Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.

Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.

Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.

Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo el solicitado está presente.

Disciplina de Soporte

Configuración y administración del cambio: Guardando todas las versiones del proyecto.

Administrando el proyecto: Administrando horarios y recursos.

Ambiente: Administrando el ambiente de desarrollo.

Distribución: Hacer todo lo necesario para la salida del proyecto

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierte luego en un entregable

al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Los elementos del RUP son:

- Actividades, Son los procesos que se llegan a determinar en cada iteración.
- Trabajadores, Vienen hacer las personas o entes involucrados en cada proceso.
- Artefactos, Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

2.3. MARCO CONCEPTUAL

2.3.1. Base de Datos

De forma sencilla podemos indicar que una base de datos No es más que un conjunto de información relacionada que se encuentra agrupada o estructurada.

Tipos de base de datos

Las bases de datos se pueden dividir en cuatro tipos básicos:

a) Bases de datos de fichero plano (o ficheros por bloques).

b) Bases de datos relacionales.

c) Bases de datos orientados a objetos.

d) Bases de datos híbridas.

a) Base de datos de fichero

Las bases de datos de fichero plano consisten en ficheros de texto divididos en filas y columnas. Estas bases de datos son las más primitivas y quizás ni tan siquiera merezcan considerarse como tales. Pueden ser útiles para aplicaciones muy simples, pero no para aplicaciones medianas o complejas, debido a sus grandes limitaciones.

b) Base de datos relacional

Las bases de datos relacionales son las más populares actualmente. Su nombre proviene de su gran ventaja sobre las bases de datos de fichero plano: la posibilidad de relacionar varias tablas de datos entre sí, compartiendo información y evitando la duplicidad y los problemas que ello conlleva (espacio de almacenamiento y redundancia). Existen numerosas bases de datos relacionales para distintas plataformas (Access, Paradox, Oracle, Sybase)9 y son ampliamente utilizadas. Sin embargo, tienen un punto débil: la mayoría de ellas no admite la incorporación de objetos multimedia tales como sonidos, imágenes o animaciones.

9< http://www.monografias.com/trabajos24/bases-de-datos/bases-de-datos.shtml > [*Consulta:20 Mayo 2006]*

c) Base de datos orientada a objeto

Las bases de datos orientadas a objetos incorporan el paradigma 10de la Orientación a Objetos (OO) a las bases de datos. La base de datos está constituida por objetos, que pueden ser de muy diversos tipos, y sobre los cuales se encuentran definidas unas operaciones. Las bases de datos orientadas a objetos pueden manejar información binaria (como objetos multimedia) de una forma eficiente. Su limitación suele residir en su especialización, ya que suelen estar diseñadas para un tipo particular de objetos (por ejemplo, una base de datos para un programa de CAD).

d) Base de datos híbrida

Las bases de datos híbridas combinan características de las bases de datos relacionales y las bases de datos orientadas a objetos. Manejan datos textuales y datos binarios, a los cuales se extienden las posibilidades de consulta. Es una tecnología reciente y aún existen pocas en el mercado.

Elementos de almacenamiento de una base datos.

***Campo*:** Es la unidad básica de una base de datos.

***Registro*:** Es el conjunto de información referida a una persona u objeto.

2.3.2. Clase

Una *clase* es una plantilla para objetos múltiples con características Similares. Las clases comprenden todas esas características de un conjunto particular de

[10] *< http://www.monografias.com/trabajos16/paradigmas/paradigmas.shtml#queson >[Consulta:15 Mayo 2006]*

objetos. Cuando se escribe un programa en lenguaje orientado a objetos, no se definen objetos verdaderos sino se definen clases de objetos.

2.3.3. Interfaz

Las interfaces básicas de usuario son aquellas que incluyen cosas como menús, ventanas, teclado, ratón, los “beeps” y algunos otros sonidos que la computadora hace, en general, todos aquellos canales por los cuales se permite la comunicación entre el hombre y la computadora.

La idea fundamental en el concepto de interfaz es el de mediación, entre hombre y máquina. La interfaz es lo que “media”, lo que facilita la comunicación, la interacción, entre dos sistemas de diferente naturaleza, típicamente el ser humano y una máquina como el computador. Esto implica, además, que se trata de un sistema de traducción, ya que los dos “hablan” lenguajes diferentes: verbo-icónico en el caso del hombre y binario en el caso del procesador electrónico.

2.3.4. Objeto

Se define a un objeto como “una entidad tangible que muestra alguna conducta bien definida”. **Un objeto** “es cualquier cosa, real o abstracta, acerca de la cual almacenamos datos y los métodos que controlan dichos datos”.

Los objetos tienen una cierta “integridad” la cual no deberá ser violada. En particular, un objeto puede solamente cambiar estado, conducta, ser manipulado o estar en relación con otros objetos de manera apropiada a este objeto.

2.3.5. Servidor

Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax[11], procesamiento de imágenes, etc.

2.3.6. Sistema

Conjunto de ordenadores s de normas y principios o conjuntos de Elementos o partes relacionales que coordinan entre sí, medio o manera para hace una cosa, conjunto de sorteares que persiguen un fin común.

[11] *< http://www.monografias.com/trabajos/modemyfax/modemyfax.shtml >[Consulta:15Mayo 2006]*

CAPÍTULO III

MATERIALES Y METODOS

3.1. POBLACIÓN DE LA INVESTIGACIÓN

Para la prueba del Software, la población estuvo constituida por todas las personas que hacen uso del sistema los cuales son 10 personas que trabajan en la atención de admisión en el Hospital de Ilave “REDESS COLLAO”.

3.2. MUESTRA

Se usó el muestreo no Probabilístico llamado también muestras de conveniencia o de juicio Este tipo de muestra se basa en el conocimiento y la opinión personal para identificar elementos de la población que van a incluirse en la muestra, una muestra a juicio se basa en el conocimiento de la población por parte de una persona generalmente es un experto en la materia.

3.3. DETERMINACIÓN DEL TAMAÑO DE LA MUESTRA

Para el caso de validación del Sistema de Admisión se consideró a los trabajadores del área de Admisión del Hospital de Ilave, que hacen un total de 10 personas, entre trabajadores nombrados, contratados y practicantes, los cuales participaron activamente interactuando con el sistema y mostrando las bondades y puntos débiles del sistema.

Tabla Nº 01: Muestra de la Investigación

Población del proyecto	Nº	%
Personal Nombrados	4	40
Contratados / Practicantes	6	60
Total	10	100

Fuente: Relación de Personal
Elaborado por: El ejecutor

MATERIAL INSTRUMENTAL

Los Recursos mínimos computacionales para la implementación del prototipo (Software) son:

HARDWARE

El hardware utilizado en la prueba del software tiene las siguientes características:

- Pentium II o superior.
- Micro procesar de 500 Mhz
- Memoria RAM de 512 Mb
- Disco duro de 60 Gb a más.
- Monitor SyncMaster color verdadero de 32 bits 800x600 píxeles a más.
- Unidad de CD-ROM de 32x a más.
- Cableado de Red

SOFTWARE

- S.O Windows XP
- Lenguaje de programación Delphi 7.0
- Servidor de Bases de Datos Interbase

- Cliente Interbase (Connective driver)

3.3. MÉTODOS

3.3.1. Tipo de Investigación

El tipo de investigación es semi-experimental ya que se implementa un entorno de software, el cual se experimenta en el personal que labora en el área de Admisión y se evalúa los resultados que se obtiene en cuanto al costo, tiempo y efectividad de las atenciones en el Hospital de Ilave.

3.3.2. Factores de Calidad del Software

Para verificar la calidad del software se hizo uso de las criterios de calidad de software propuestos por ISO 9126, el cual “ha sido desarrollado en un intento de identificar los atributos claves de calidad para el software, el estándar identifica seis atributos claves de calidad: La funcionalidad, la confiabilidad, la usabilidad, la eficiencia, la facilidad de mantenimiento y la portabilidad” (Pressman, R., 2002, 326). Estos criterios ubicados en una escala nos muestran el nivel de calidad del entorno como instrumento para la prueba del software:

ATRIBUTOS CLAVES DE CALIDAD

Tabla N° 02: Factores de Calidad de Software

INDICADORES	PUNTUACIÓN				
	1	2	3	4	5
1. Funcionalidad • Idoneidad • Corrección • Interoperabilidad • Conformidad • Seguridad					
2. Confiabilidad • Madurez • Tolerancia a fallos • Facilidad de recuperación					
3. Usabilidad • Facilidad de comprensión • Facilidad de aprendizaje • Operatividad					
4. Eficiencia • Tiempo de uso • Recursos utilizados					
5. Facilidad de mantenimiento • Facilidad de análisis • Facilidad de cambio • Estabilidad • Facilidad de prueba					
6. Portabilidad • Facilidad de instalación • Facilidad de ajuste • Facilidad de adaptación al cambio					
SUB TOTALES					

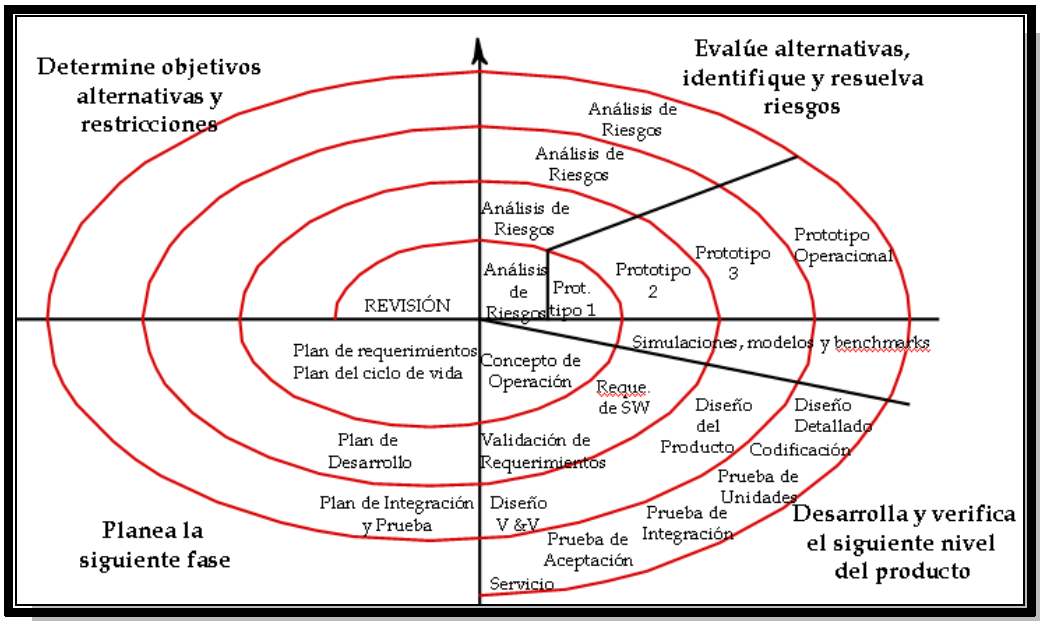
Fuente: Arquitectura Cliente servidor (2007)

3.4. MÉTODO DE TRATAMIENTO DE DATOS

3.4.1. Modelo Espiral

Para el desarrollo e implementación del Software de Admisión se hizo uso del modelo espiral, representándose como se muestra en la siguiente figura.

Grafico N° 03: Representación gráfica del Modelo Espiral



En el gráfico podemos observar que en el desarrollo del modelo se cuenta con cuatro etapas bien definidas las cuelas son:

a) Planteamiento de Objetivos

Donde se identifican los objetivos específicos para cada fase del Proyecto. Para el caso del proyecto, en el primer espiral se planteó los requisitos del sistema iniciales, los cuales se fueron refinando en las siguientes espirales de evolución, logrando finalmente conseguir los requisitos finales del sistema que son los que se consideran en el presente trabajo. El producto de este cuadrante de la metodología son los diagramas de casos de uso y los

diagramas de interacción que sirven de base para el proceso de diseño e implementación en un lenguaje de programación.

b) Identificación y reducción de riesgos

Los riesgos clave se identifican y analizan, y la información sirve para Minimizar los riesgos. En el trabajo se consideran riesgos a los elementos Que no permitan alcanzar el logro de los objetivos y cubrir las necesidades del sistema, de tal manera que no podamos lograr cubrir los requisitos mínimos enmarcados en la norma ISO 9126 que describe las características de calidad de software, como la usabilidad, funcionalidad, etc. Por lo tanto en cada espiral se realizó el análisis de estos riesgos, que en la última espiral no encontramos una amenaza grande al sistema y cumpliendo los requisitos del estándar de calidad.

c) Desarrollo y Validación

Se elige un modelo apropiado para la siguiente fase del desarrollo El primer modelo que funcionó del sistema es uno en papel, que tuvo como centro el diseño de la base de datos y de los módulos funcionales que conformarían el sistema, en base a las consultas realizadas en las tablas se pudo verificar que el modelo estaba listo para pasar a una segunda etapa en la cual se implementaría la base de datos en un sistema manejador de bases de datos, para luego realizar el primer prototipo de software que cubría los requisitos básicos que fueron refinándose en las siguientes espirales del modelo, hasta que finalmente se tuvo el modelo final que no se considera de ninguna manera terminado, pero que ya cubre todos los requisitos previstos por los usuarios.

d) Planeación

Se revisa el proyecto y se trazan planes para la siguiente ronda del espiral.

En este cuadrante se considera los diferentes planes considerados desde la espiral inicial hasta lograr la espiral final de desarrollo. En el primer espiral donde se concluyó las etapas de análisis y diseño iniciales con los requisitos del software y el primer modelo en papel que funciona se planificó el proceso para su implementación y verificación en una segunda espiral, donde se implementaría el modelo en software logrando el primer prototipo de software, este luego de las revisiones realizadas y la afinaciones en las tablas, los campos y las relaciones entre tablas para lograr las consultas que mostraran el producto del sistema se pasó a otros niveles en las espirales logrando el producto final que fue puesto a prueba y evaluado por el usuario final.

Los cuales podemos se disgregaron de la siguiente manera

- **Punto de Inicio**.- En esta sección se determinó los Objetivos (Desempeño Multiusuario, Reducción de Colas en ventanilla, etc.), alternativas (En cuanto a las herramientas como son: el Tipo de Software para su implementación, etc.) y restricciones (Compatibilidad de los Protocolos Ethernet, Conectividad, etc.) para el desarrollo del Sistema de Admisión.

- **Costo Acumulativo.-** Después de la Identificación de riesgos en la realización de la evaluación del software, este punto implicó realizar la solución de los mismos por Ejemplo (la re-codificación del sistema el cual demanda tiempo 'Horas - Hombre/Computador'), el cual generó costos.

- **Implementación.-** Esta sección describe la codificación, la realización de la prueba de integración (Unión de los Algoritmos en un solo programa), Generación de Resultados inmediatos.

- **Prototipo Operacional.-** El Software en su primera versión aun con errores de poca consideración que fueron resueltos en el proceso iterativo del modelo espiral, dando como resultado el software operativo, teniendo en cuenta que este requerirá de ajustes posteriores según el usuario.

3.4.2. Análisis y Diseño del Sistema de Admisión

El análisis del sistema está concentrado en los casos de uso que se describen más adelante (puesto que el implementar un caso de uso requiere de análisis – Concepto teórico de UML) y el diseño con el modelo de desarrollo espiral, sin dejar a lado el diseño ergonómico que ya está embebido en la aplicación 'Sistema de Admisión', aun a si se describe en forma teórica como se implementó el sistema.

3.4.3. Descripción de cómo se Implementó el Sistema:

- ❖ Se Ejecutó las tareas y funciones descritas en los casos de uso
- ❖ Satisface todos los requerimientos del usuario
- ❖ El sistema es Flexible a cambios posteriores
- ❖ El diseño se centra en la noción de arquitectura.

3.4.4. El Modelo de Diseño Consta de:

- Clases estructuradas (Esta sección esta detallada en los Diagramas de clases de UML que se pueden ver en las siguientes paginas dentro de la metodología de desarrollo Racional Unified Process).
- Diseños de subsistemas con interfaces definidas (componentes), como por ejemplo una Librería dinámica (DLL), dll con el que cuenta el sistema de Admisión.

3.5. OPERACIONALIZACIÓN DE VARIABLES

Tabla N° 03: Representación de Operacionalización de Variables

VARIABLE	DIMENSIÓN	INDICADOR	ÍNDICE
INDEPENDIENTE Implementación de Módulos del Sistema de Admisión.	Administración de usuarios	- Permite agregar usuarios - Selección del nivel de categoría(privilegio) - Selección de permisos - Control de acceso al banco de datos - Conclusiones	▪ Muy bueno ▪ Bueno ▪ Regular ▪ Deficiente
	Consulta Multiusuario	- Visualización multiusuario de un documento - Modificaciones en el documento Actualización	▪ Muy bueno ▪ Bueno ▪ Regular ▪ Deficiente
	Comunicación síncrona y asíncrona	- Consultas en tiempo real - Verificación de registros	▪ Muy bueno ▪ Bueno ▪ Regular ▪ Deficiente
DEPENDIENTE El Sistema de Admisión (revisiones del proyectos de investigación)	Reuniones de Los operadores del sistema	- Numero de accesos al sistema - Participaciones - Observaciones	▪ Alto ▪ Medio ▪ Bajo
	Costo	- Tiempo (horas hombre) - Dinero	▪ Alto ▪ Medio ▪ Bajo
	Calidad del trabajo	- Tema de actualidad - Estructura del trabajo - Originalidad - Utilidad	▪ Alto ▪ Medio ▪ Bajo

Elaborado por: El ejecutor
Fuente: Encuesta (Anexo B)

3.5.1. Tecnología a Usar:

Tecnología Cliente / Servidor

La Tecnología Cliente/Servidor**,** es un modelo que implica productos y servicios enmarcados en el uso de la tecnología de punta, y que permite la distribución.

Desde un punto de vista conceptual:

Es un modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información.

En términos de arquitectura:

Los distintos aspectos que caracterizan a una aplicación (proceso, almacenamiento, control y operaciones de entrada y salida de datos) en el sentido más amplio, están situados en más de un computador, los cuales se encuentran interconectados mediante una red de comunicaciones.

En este proyecto de tesis se aplica el modelamiento UML (lenguaje de modelado Unificado).

3.5.2. Metodología para el desarrollo del software

Metodología Rup

RUP es una guía de cómo usar UML de la forma más efectiva e eficiente. El objetivo de RUP es Asegurar la producción de software de calidad, dentro de

plazos y presupuestos predecibles. Dirigido por casos de uso, centrado en la arquitectura, iterativo (mini-proyectos) e incremental (versiones).

a) Ciclos y Fases de Rup:

RUP divide el proceso de desarrollo en ciclos, teniendo un producto al final de cada ciclo.

Cada ciclo se divide en cuatro Fases:

- Inicio
- Elaboración
- Construcción
- Transición

Cada fase concluye con un hito bien definido donde deben tomarse ciertas decisiones, a continuación se detalla cada una de las fases de la metodología Racional Unified Process.

FASE DE INICIO

En esta fase desarrollo los requisitos del producto desde la perspectiva del usuario, los cuales son establecidos en el artefacto Visión. Los principales casos de uso son identificados y se hace un refinamiento del Plan de Desarrollo del Proyecto (Esta etapa señaliza la implementación del 20% de artefactos y diagramas de UML (Casos de Uso, Clases)).

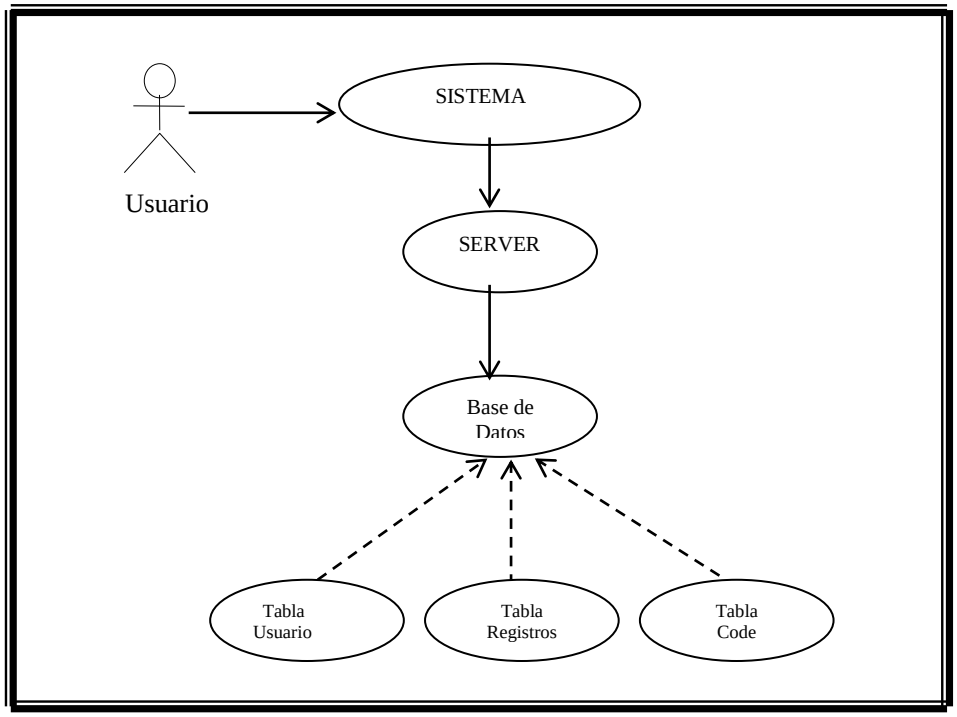
Descripción de acuerdo al sistema

Esta fase nos muestra como el sistema de admisión está construido, como funciona internamente y como se relaciona, El paciente al ingresar a la

ventanillas admisión entrega sus datos correspondientes para su registro, en caso que sea un paciente nuevo (primera visita al hospital de Ilave), en el otro caso sea un paciente entrega su código (número de historia clínica), EL Usuario está relacionado con el del sistema MULTIUSUARIO ADMISIÓN 2007 y este está asociado e implementado en base al servidor Interbase.

Al iniciar el sistema multiusuario, el usuario ingresa su nombre de usuario y password para acceder al sistema de admisión multiusuario, finalmente tiene acceso para consultar datos, agregar registros, modificar registros, Cambio de Password, Eliminación de registros, generar reportes, etc. De acuerdo a los permisos que tenga el usuario.

Diagrama N° 03: Diagrama de Casos de Uso 20%



DESCRIPCIÓN

Este Diagrama nos muestra la forma como se ha desarrollado el software: El administrador hace uso del sistema, el sistema está relacionado con un

servidor, el servidor está relacionado con una base de datos Interbase este a la ves depende de las tablas: usuario, registro, code.

USUARIO.- Es la persona que usa el sistema.

SISTEMA.- es el software desarrollado (sistema multiusuario) que manejara el administrador.

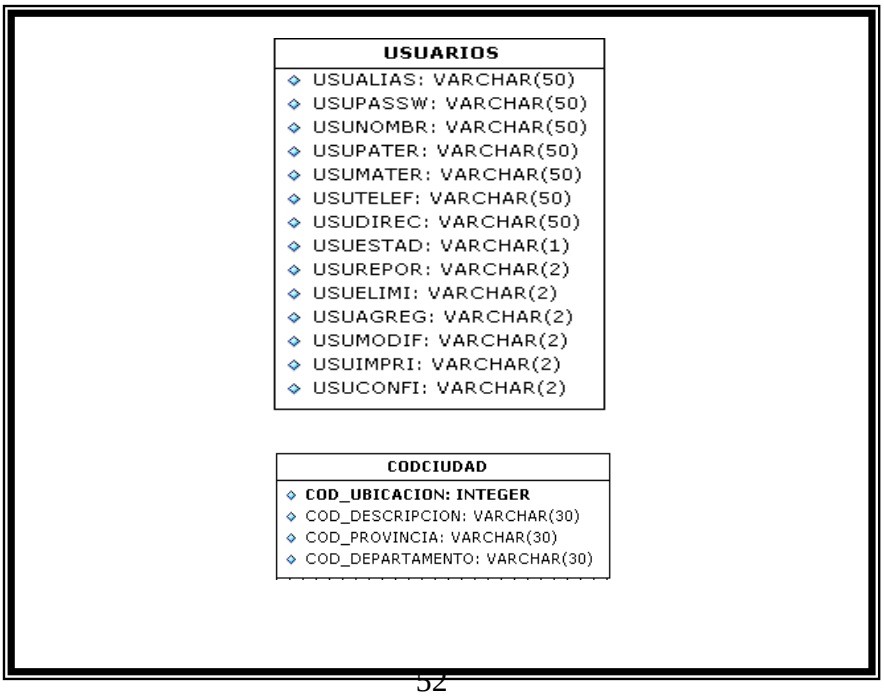
SERVER.- Es el servidor implantado a la oficina de admisión este a la ves administra la base de datos Interbase.

TABLA USUARIO.- En esta tabla se registra a las personas (trabajadores del hospital) con sus respectivas passward que harán uso del sistema.

TABLA REGISTRO.- En esta tabla se registran los datos de los pacientes con sus datos correspondientes.

TABLA CODE.- En esta tabla se registra los códigos de las ciudades para su enlace Correspondiente.

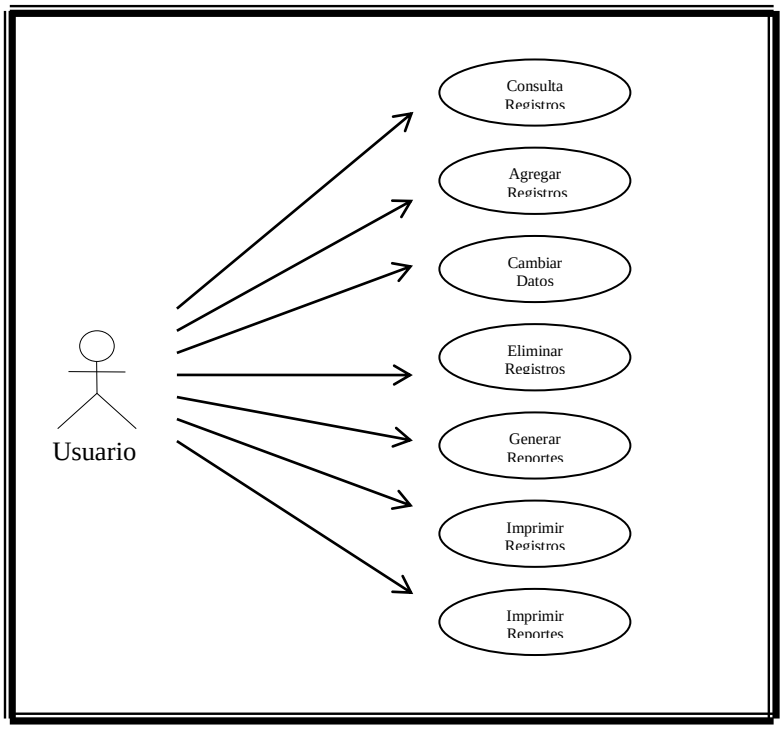
Diagrama N° 04: Diagrama de clases 20 %



FASE DE ELABORACIÓN

En esta fase se analizan los requisitos y se desarrolla un prototipo de arquitectura (incluyendo las partes más relevantes y/o críticas del sistema). Al final de esta fase, todos los casos de uso correspondientes a requisitos que serán implementados en el primer reléase de la fase de Construcción deben estar analizados y diseñados (en el Modelo de Análisis/Diseño). La revisión y aceptación del prototipo de la arquitectura del sistema marca el final de esta fase. En nuestro caso particular, por no incluirse las fases siguientes, la revisión y entrega de todos los artefactos hasta este punto de desarrollo también se incluye como hito. La primera iteración tendrá como objetivo la identificación y especificación de los principales casos de uso, así como su realización preliminar en el Modelo de Análisis/Diseño, también permitirá hacer una revisión general del estado de los artefactos hasta este punto y ajustar si es necesaria la planificación para asegurar el cumplimiento de los objetivos. (Esta etapa señaliza la implementación del 80% de artefactos y diagramas de UML).

Diagrama Nº 05: Diagrama de Casos de Uso 80%



DESCRIPCIÓN

El Usuario está relacionado con Consultar registros, Agregar registros, cambiar registros, eliminar registros, generar reportes, Imprimir reportes esto en una tabla registros.

Detalle de la Metodología.-En este grafico los demuestra un 80% del avance del sistema representados en casos diagramas de casos de uso y podemos apreciar que, el usuario es la persona que tiene asociación con el sistema.

Consultar Registros.- el administrador hace la búsqueda de información, como Número de Historia clínica, Nombres, Apellidos en la base de datos registrada en el servidor, esto en caso que existe una persona anterior mente ya registrada con sus datos correspondientes.

Agregar Registros.- El administrador agrega registros siempre que ya haya hecho una búsqueda para que no se repita la información agrega: Nro. De historia clínica, Nombres, Apellidos, edad, procedencia, ocupación, lugar y fecha de nacimiento, estado civil, nombre del padre nombre de la madre y en qué servicio se atenderá.

Cambia Datos.- El administrador cambia los datos de una persona en caso que si exista un error o fueron mal registrados (actualiza los datos).

Eliminar Registro.- El administrador elimina un registro si es repetido o no corresponda.

Generar Reporte.- El administrador genera reporte cada fin de mes o si la oficina los adquiere.

Imprimir Reporte.- Al generar un registro el administrador Imprime un reporte Correspondientes.

FASE DE CONSTRUCCIÓN

Durante la fase de construcción se terminan de analizar y diseñar todos los casos de uso, refinando el Modelo de Análisis/Diseño. El producto se construye en base a 4 iteraciones, cada una produciendo una reléase a la cual se le aplican las pruebas y se valida con el cliente/usuario. Se comienza la elaboración de material de apoyo al usuario. El hito que marca el fin de esta fase es la versión de la reléase, con toda la capacidad operacional del producto, lista para ser entregada a los usuarios para pruebas beta, En esta fase todos los componentes restantes se desarrollan e incorporan al producto,

❖ Todo es probado en profundidad. El énfasis está en la producción eficiente y no ya en la creación intelectual. El Producto de software integrado y corriendo en la plataforma adecuada. Manuales de Usuario. Una descripción del "reléase" actual.

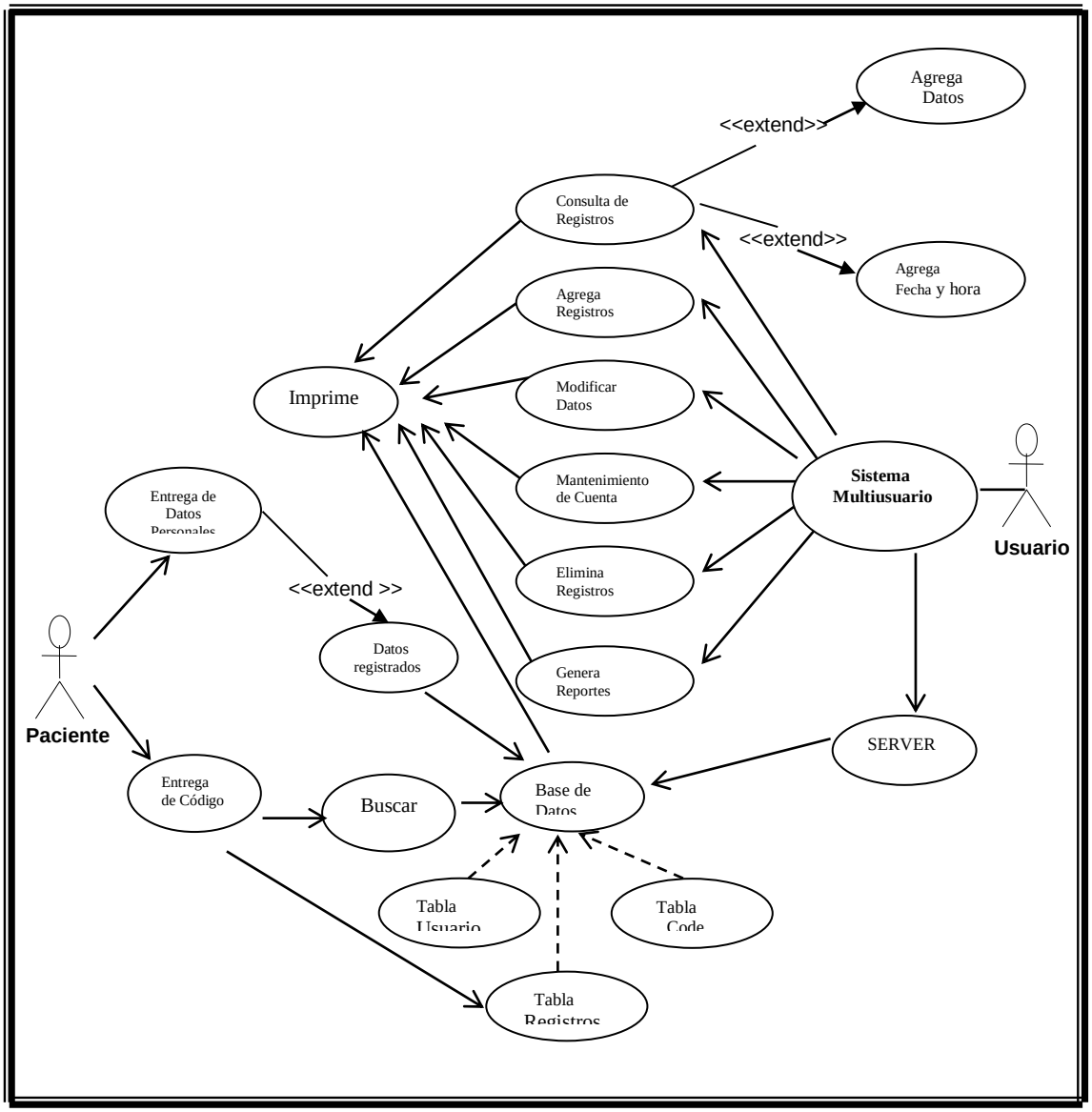
IMPLEMENTACIÓN

Propósito:

Definir la organización del código:

- Implementar clases y objetos en forma de componentes (fuente, ejecutables, Etc.)
- Probar las componentes desarrolladas
- Integrar las componentes en un sistema ejecutable

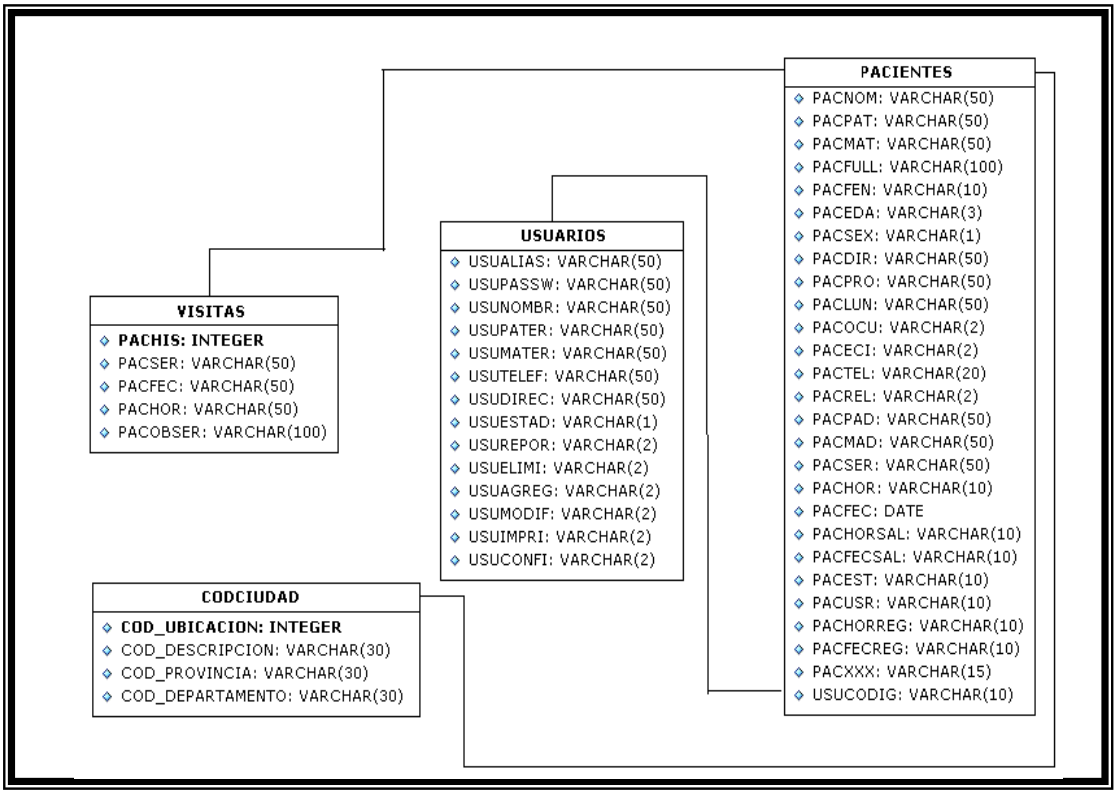
Diagrama Nº 06: Diagrama de casos de uso 100%



DESCRIPCIÓN

En este diagrama de casos de uso se puede ver claramente cómo se lleva a cabo el funcionamiento del sistema de admisión, además se puede apreciar los múltiples casos de uso que se lograron identificar durante el análisis del sistema.

Diagrama. Nº 07 Diagrama de clases 100%.



FASE DE TRANSICIÓN

En esta fase se prepararán dos raleases para distribución, asegurando una implantación y cambio del sistema previo de manera adecuada, incluyendo el entrenamiento de los usuarios. El hito que marca el fin de esta fase incluye, la entrega de toda la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario, la finalización del entrenamiento de los usuarios y el empaquetamiento del producto.

El objetivo es traspasar el software desarrollado a la comunidad de usuarios.

- ❖ Pruebas Beta para validar el producto con las expectativas del cliente
- ❖ Entrenamiento de usuarios
- ❖ Distribuir el producto

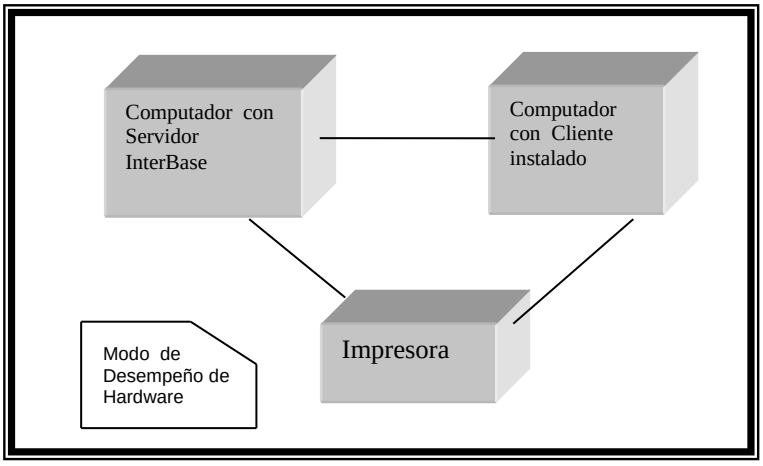
Propósito:

- Verificar la interacción entre los objetos
- Verificar la integración apropiada de componentes
- Verificar que se satisfacen los requerimientos
- Identificar los defectos y corregirlos antes de la instalación

DISTRIBUCIÓN

Producir un producto y hacerlo llegar a sus usuarios finales.

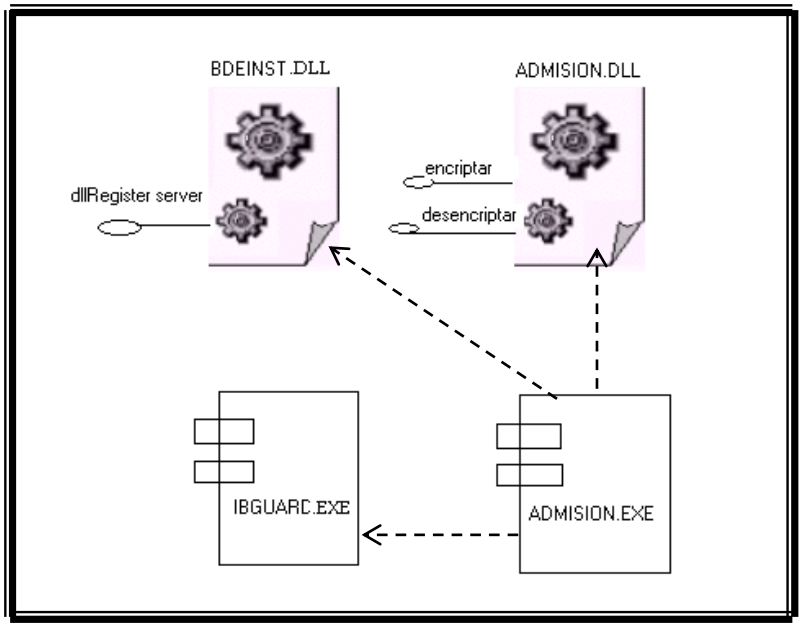
Diagrama. Nº 08: Diagrama de nodos del sistema de Admisión



DESCRIPCIÓN

Este diagrama nos representa la descripción física del desempeño de sistema DE ADMISIÓN, como podemos ver esta figura, tenemos dos computadoras, uno con servidor Interbase y el otro con un cliente instalado y estos a su vez asociados con una impresora .

Diagrama. Nº 09: Diagrama de componentes del Sistema de Admisión



DESCRIPCIÓN

Este diagrama nos representa como está relacionado el software internamente, el ejecutable ADMISION.exe depende de dos Librerías Dinámicas (DLLs) las cuales son ADMISIÓN.DLL, BDINST.dll. Del cual podemos detallar lo siguiente:

ADMISIÓN.EXE

Aplicación de admisión, se encarga de conectar al servidor y realizar operaciones con la base de datos conectados al servidor Interbase.

ADMISIÓN.DLL

Contiene funciones de Encriptado y Des encriptado de cadenas para el uso de Passwords

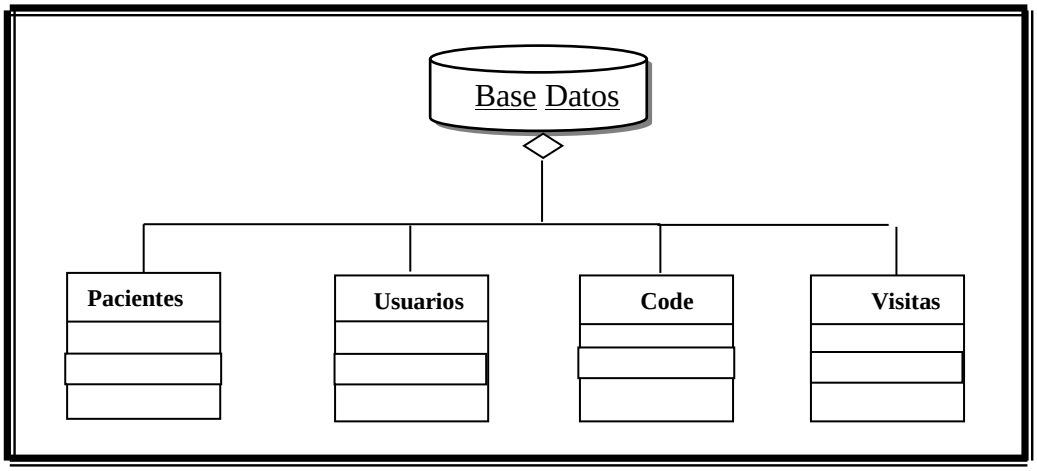
BDINST.DLL

Contiene librerías y archivos que usa Delphi para la ejecución de sus aplicaciones de base de datos y conexión con el servidor Interbase.

IBGUARD.EXE

Aplicación de propiedad de Interbase (Servidor), realiza la verificación de que el Servidor no sufra una caída de Servidor, en caso suceda ello **ibguard.exe** reiniciara Interbase (Servidor).

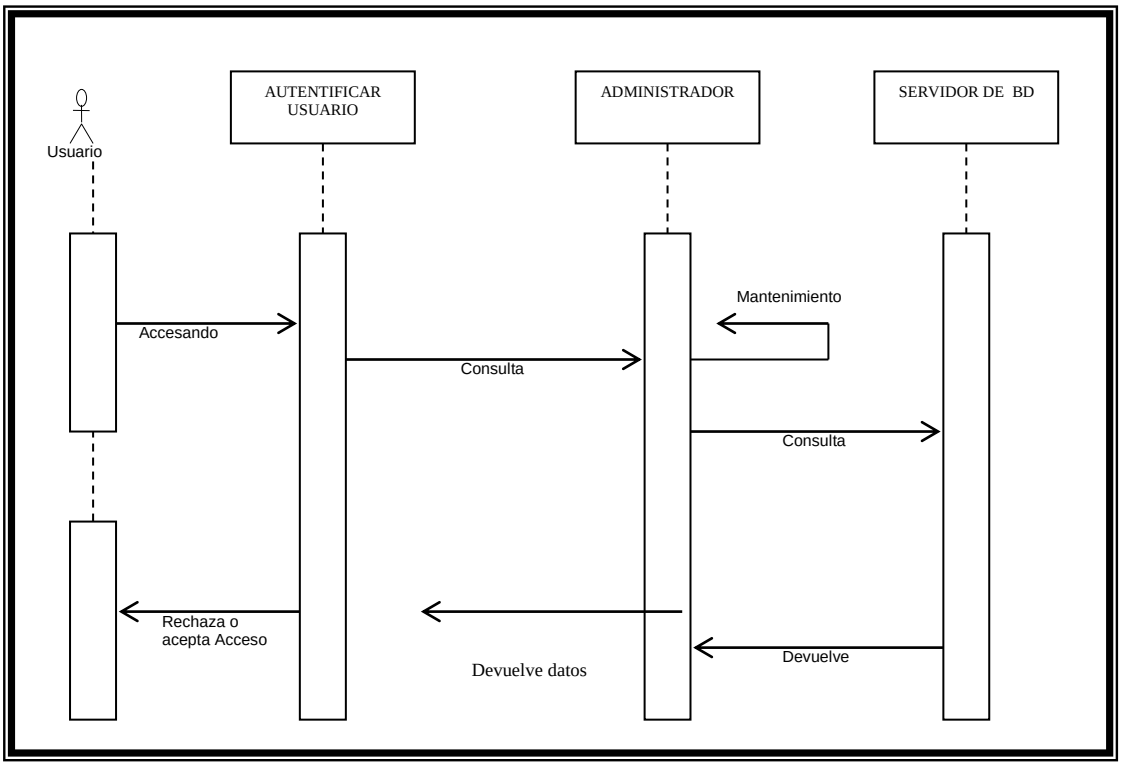
Diagrama. Nº 10: Diagrama de Componentes de la Base de Datos Física del Sistema de Admisión



DESCRIPCIÓN

Este diagrama nos representa los componentes físicos de la base de datos conectados a un servidor InterBase, Se puede decir que la Base de datos alojada en el Servidor InterBase, además en la base de datos se agregan las tablas: Pacientes, Usuarios, CodCiudad, Visitas. En notación UML.

Diagrama. Nro. 11: Diagrama de Secuencias:



DESCRIPCIÓN

Este diagrama nos da a conocer la secuencia de acciones con el usuario y el Sistema de Admisión. El Usuario accede mediante la autentificación de usuario una vez realizado esto con éxito, el usuario tiene acceso al sistema, para ello verifica el Sistema de Admisión verifica los datos (Password y usuario con los datos registrados en el Servidor), finalmente el servidor devuelve datos al usuario (acepta o rechaza el acceso).

3.5.3. Métricas de Software

Métricas Técnicas.- Se centran en las características de software, en base al sistema de admisión podemos resaltar que la aplicación es de tipo cliente servidor, la aplicación generada es Win32 (Compatible en el sistema operativo y con las librerías DLLs Standard), también se da especial importancia a la

reutilización de componentes en lo que se refiere a métricas técnicas, por lo tanto el Sistema de Admisión tiene un componente reutilizable, para llevar a cabo la encriptación y Des encriptación de cadenas, cumpliendo con los requerimientos de las Métricas Técnicas.

Métricas de Proyectos.- Las Métricas del Proyecto de Software son de carácter táctico. Esto es las métricas de software y los indicadores derivados de ellos, se utilizan un Gestor de proyectos y un equipo de software para adaptar el flujo del trabajo del proyecto y las actividades técnicas realizadas(Roger Pressman) .

Métricas Utilizadas para el Diseño.- Esta sección tiene cierto grado de relación con los aspectos **ergonométricos**, con los cuales se concluye que en el desarrollo de software se mediara su flexibilidad, deberá tener una interfaz amigable para su uso, un alto grado de simplificación para facilitar la Relación Usuario – Software.

3.6. VALIDACION DEL SOFTWARE

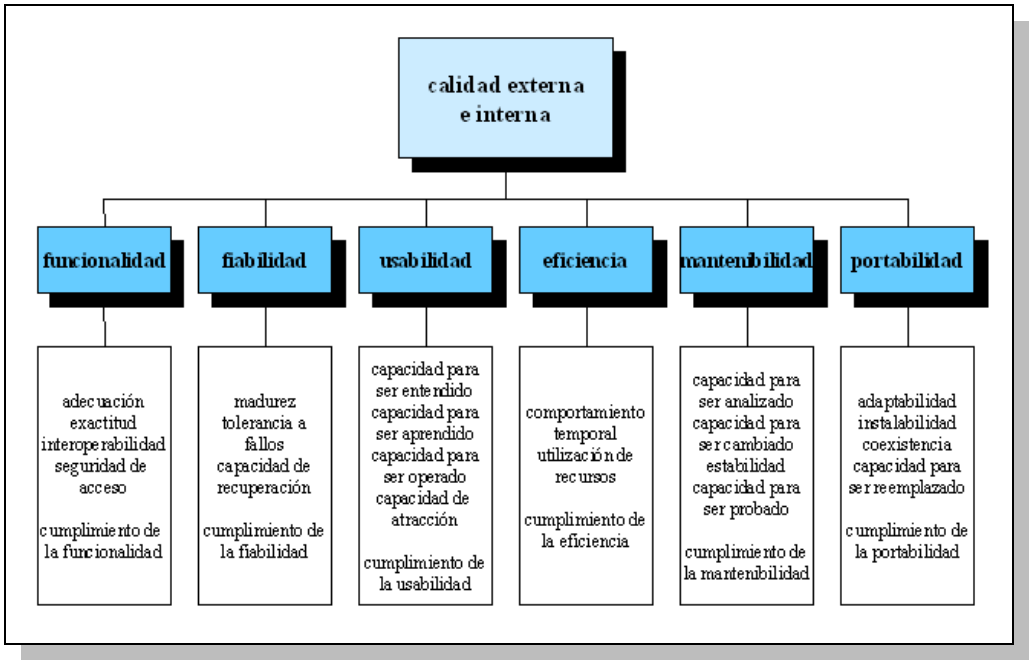
Evaluación del Nivel de Optimización

Para la evaluación del nivel de optimización del sistema se realizó:

3.6.1. Evaluación de la Calidad del Software (Estándar ISO – 9126)

Para la evaluación del nivel de calidad del Sistema de Admisión se aplicó los indicadores de calidad del Estándar ISO - 9126, para lo cual se hizo uso de la guía de evaluación de calidad de software (Ver anexo A), que consideras los factores que se muestra en el gráfico siguiente:

Gráfico Nº 04: Indicadores de calidad de Software según el Estándar ISO-9126



Haciendo uso de una escala de 1 a 5 puntos, según el baremo

Siguiente:

Tabla N° 04: Escala de Validación del Software

Baremo	Valor
Deficiente	1
Malo	2
Regular	3
Bueno	4
Muy bueno	5

Fuente: Indicador de calidad de Software
Elaborado por: El ejecutor

Además se tiene según el estándar, que realizando la sumatoria de los valores que se hayan colocado para cada indicador, se puede tener la siguiente clasificación:

Tabla. N° 05: Clasificación del Indicador de calidad de Software

Clasificación	Intervalo
Inaceptable	[27 - 54 >
Mínimamente aceptable	[54 – 81 >
Aceptable	[81 – 95 >
Cumple los requisitos	[95 – 122 >
Excede los requisitos	[122 – 135]

DESCRIPCIÓN

Para esta evaluación se consideró a los trabajadores nombrados, contratados y practicantes del área de admisión del Hospital de la Ciudad de Ilave, los cuales han tenido acceso al sistema y han realizado la calificación de acuerdo a los indicadores establecidos en la ficha de evaluación.

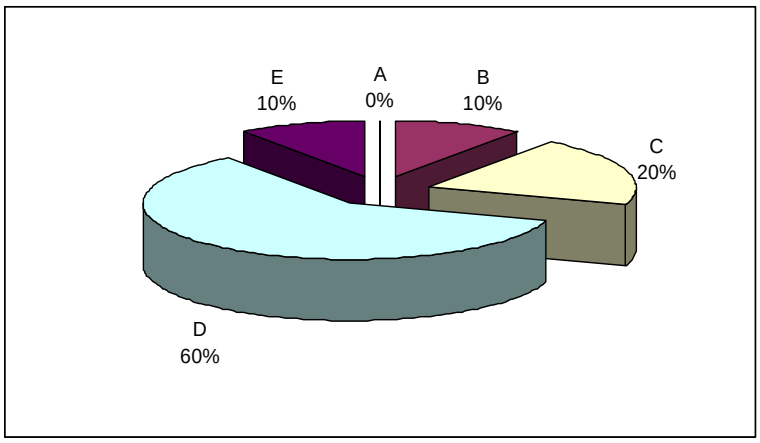
Los resultados obtenidos consolidados se muestran en el cuadro y gráfico siguiente:

Tabla. Nº 06: Medición de la Calidad de Software Estándar ISO-9126

Clasificación	Intervalo	Nº	%
A) Inaceptable	[27 - 54 >	0	00
B) Mínimamente aceptable	[54 – 81 >	1	10
C) Aceptable	[81 – 95 >	2	20
D) Cumple los requisitos	[95 – 122 >	6	60
E) Excede los requisitos	[122 – 135]	1	10
TOTAL		10	100

Fuente: Ficha de Evaluación
Elaboración: El ejecutor

Gráfico. Nº 05: Medición de la Calidad de Software Estándar ISO-9126



Fuente: Tabla Nº 06
Elaborado por: El ejecutor en base a la tabla N° 06

En la tabla N° 06 y el gráfico Nº 05 se presenta la medición de la calidad de software mediante el Estándar ISO -9126, que considera 6 factores principales, según el número de indicadores de calidad se puede obtener un puntaje entre 27 y 135 puntos, teniendo intervalos que ubican al producto en los intervalos de “Inaceptable”, “Mínimamente aceptable”, “Aceptable”, “Cumple los requisitos” y “Excede los requisitos”.

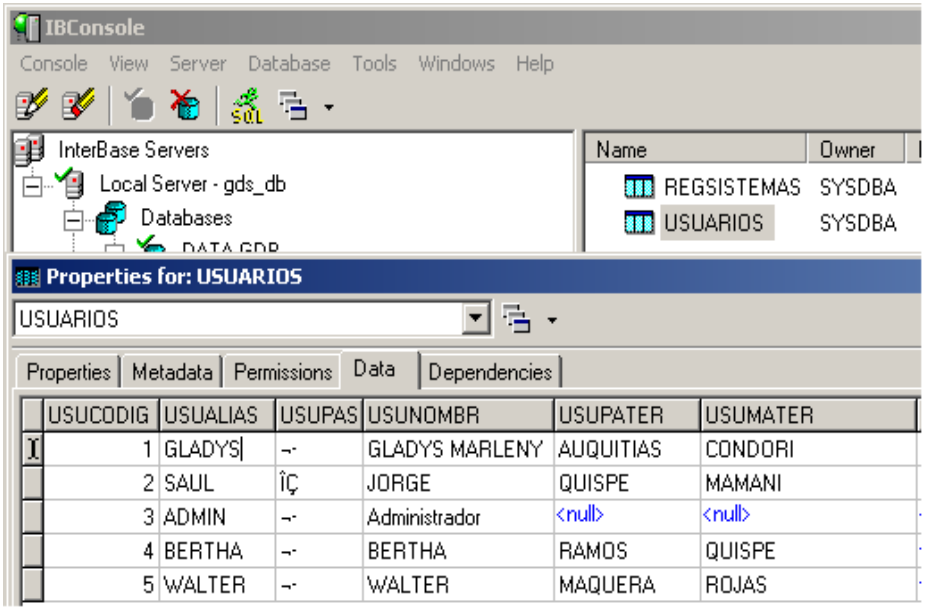
Se puede apreciar que en un 20% se tiene el calificativo de Aceptable, el cual indica que el software cumple con realizar las tareas básicas, pero que aún está faltando algunos otros criterios para ser considerado como óptimo. Se observa también que en un 60% obtiene el calificativo de “Cumple los requisitos”, esto nos indica que el sistema cumple a cabalidad todos los requisitos planteados por los usuarios y por lo tanto está en un nivel alto de calidad de software en cuanto a los factores e indicadores propuestos por el Estándar. Se observa también que en un 10% del total se obtiene el calificativo de “Excede los requisitos”, lo cual nos muestra que con el sistema se puede hallar proceso que mejoran y exceden los requisitos básicos del sistema.

Por tanto, en un 90% el sistema aprueba el Estándar ISO – 9126 de calidad de software, ubicándose mayoritariamente en la escala de “Cumple los requisitos”.

3.6.2. Evaluación de Funcionalidad

Según el ISO 9126, en este contexto se evalúa la Funcionalidad y seguridad de acceso del Software, Se ha implementado los accesos para cada usuario, en el servidor de base de datos Interbase 7.5.1.

Imagen. N° 01: Acceso al Servidor de la base de datos



3.6.3. Evaluación de Fiabilidad

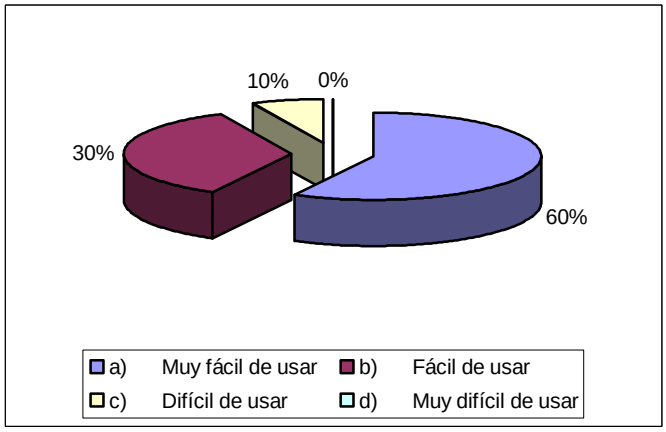
Según el ISO 9126, en este contexto se evalúo la tolerancia a fallos y la capacidad de recuperación. En este contexto se ha contemplado la implementación de procedimientos de control de flujo de datos en el Servidor (Trigres en la data), el cual gestiona la data independientemente asegurando la integridad de la información y evitando en todo momento las posibles fallas que pudieran generarse.

```
SET TERM ;
CREATE TRIGGER "BORRO_CLIENTE" FOR "PACIENTES"
ACTIVE AFTER DELETE POSITION 0
AS
BEGIN
  POST_EVENT 'DEL_CLIENTE';
END
COMMIT WORK
SET TERM ;

SET TERM  ;
CREATE TRIGGER "NUEVO_CLIENTE" FOR "PACIENTES"
ACTIVE AFTER INSERT POSITION 0
AS
BEGIN
  POST_EVENT 'NEW_CLIENTE';
END
COMMIT WORK
SET TERM ;
SET TERM ^ ;
CREATE TRIGGER "CLIENTE_BI" FOR "PACIENTES"
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
  NEW.PACHIS=GEN_ID(G_CLAVECLIENTE,1);
END
COMMIT WORK
SET TERM ;

SET TERM ^ ;
CREATE TRIGGER "CAMBIOS_EN_CLIENTE" FOR "PACIENTES"
ACTIVE AFTER UPDATE POSITION 0
AS
BEGIN
  POST_EVENT 'CAMBIO_CLIENTE';
END
COMMIT WORK
SET TERM ;
```

3.6.4. Evaluación de Usabilidad

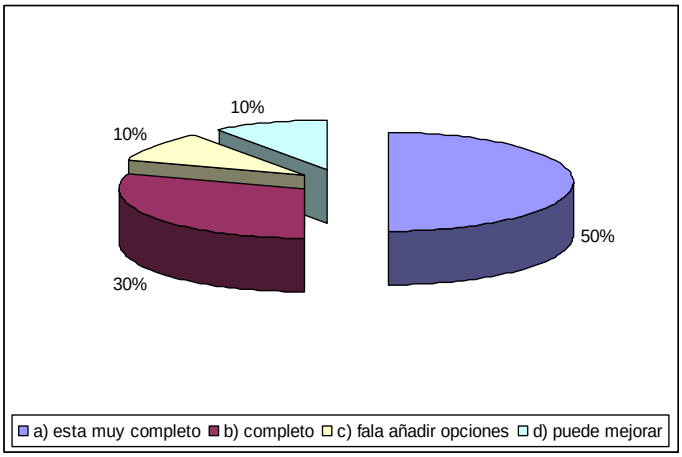
También se considera la apreciación de los operadores que evalúan el Sistema de Admisión en los factores de usabilidad, tiempo, costo y calidad del trabajo de investigación, lo cual mostramos a continuación:

Gráfico Nº 06: Apreciación sobre el uso de la Interfase del Sistema



Fuente: Tabla Nº 06
Elaborado por: El ejecutor en base a la tabla N°06

Gráfico Nº 07: Apreciación sobre si el sistema proporciona todos los medios para la interacción con los operadores del Sistema.



Fuente: Tabla Nº 06
Elaborado por: El ejecutor en base a la tabla N° 06

Interpretación:

En los gráficos que se observa podemos notar que con el uso del Sistema de Admisión se mejora significativamente en cuanto al tiempo que invierten en el proceso de aprobación que incluye desde la presentación del proyecto, las revisiones por parte del personal y la aprobación final. En este proceso indican mayoritariamente también, que el costo disminuye notablemente, pues ya no se hace uso de las impresiones, ni se tiene que gastar en transporte y otros

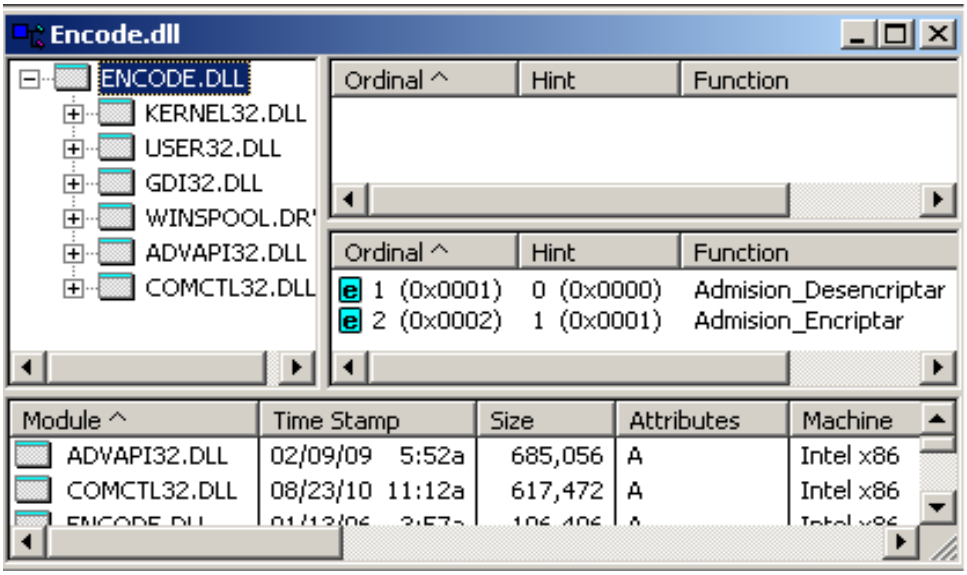
gastos anexos. Se nota también que la apreciación mayoritaria sobre la interfase del sistema es que es muy fácil de utilizar y estando al alcance de todos usuarios, se puede notar también que la interacción de los pacientes con los usuarios del sistema se mejora y se puede realizar las consultas directamente con el uso SQL Script.

Finalmente, los usuarios del sistema consideran que con el uso del sistema se mejora en forma y contenido el trabajo, demostrando que con el uso del sistema se optimiza el proceso de administración de trabajos de búsqueda, edición y consulta de historias clínicas del Hospital de Ilave - Puno.

3.6.5. Evaluación de Mantenibilidad

Según el ISO 9126, en este contexto se evalúo la capacidad de ser analizado, cambiado en forma total o parcial de cada uno de los componentes de software, para lo cual se desarrolló el Sistema de Admisión basado en componentes reutilizables de Software (Librerías Dinámicas Dlls), los cuales pueden ser cambiados o actualizados directamente o en línea, y para la mantenibilidad se trabajó con archivos de configuración (**config.ini**) cumpliendo así con la evaluación de Mantenibilidad del Software.

Imagen N° 02: Ilustración de Librerías Dinámicas Dlls



[SERVER]

Servidor=127.0.0.1

UserName=SYSDBA

Password=@W33#"/&?¡/%%$$"#

DataBase=C:\Sistema de Admision\Data\ADMISION.GDB

[CONFIGURACION]

Estado=Configurado

3.6.6. Evaluación de Portabilidad

Según el ISO 9126, en este contexto se evalúo la adaptabilidad, instabilidad, coexistencia y capacidad de ser reemplazada (cumplimiento de portabilidad). Para poder cumplir con esta regla propuesta por el ISO en referencia se desarrollaron los módulos de: aplicación y librerías dinámicas en lenguajes de programación que generen aplicaciones compatibles y multiplataforma (Win95, WinNT, WinXP, Win7), para lo cual se hizo uso del Microsoft Visual C++ y Delphi 7.0, los cuales trabajan independientemente, sin dependencia alguna para su operación

CAPÍTULO IV

RESULTADOS Y DISCUSIONES

De la Investigación realizada se obtuvo la reducción de colas generadas por la llegada de los pacientes al centro de salud, disminuyendo el tiempo de atención en la oficina de admisión del hospital (Oficina -Ingreso al Hospital), esta sección es evidente puesto que se implementó una Terminal más, llegando a ser ahora 2 terminales para la atención del público.

Desempeño de la Aplicación de modo Cliente Servidor, Utilizando el Servidor Interbase 7.5.1, de este modo dando uso a uno de los productos (Interbase) de **Borland** para dar solución a un problema, ahora situado en el hospital de Ilave

GENERACIÓN AUTOMÁTICA DE LA EDAD EN BASE A LA FECHA DE NACIMIENTO

ALGORITMO

función Edad(FechaNacimiento:string):integer;
var

iTemp,iTemp2,Nada:word;

```
    Fecha:      TDate;
        begin
    Fecha:      =StrToDate(FechaNacimiento);
        DecodeDate(Date,itemp,Nada,Nada);
        DecodeDate(Fecha,itemp2,Nada,Nada);
        if FormatDateTime('mmdd',Date) <
        FormatDateTime('mmdd',Fecha) then Result:=iTemp-iTemp2-1 else
Result:=iTemp-iTemp2;
        end;
```

VENTANA AGREGAR NUEVO REGISTRO

En esta ventana se genera automáticamente la edad del paciente en base a la fecha de nacimiento.

Imagen N° 03: Ventana para Agregar Nuevo registro

BUSQUEDA DE REGISTROS ALFABETICAMENTE RELACIONANDO APELLIDO PATERNO, APELLIDO MATERNO Y NOMBRES

ALGORITMO

```
Form2_password.IBQuery_Pacientes.Close;
Form2_password.IBQuery_Pacientes.SQL.Clear;
Form2_password.IBQuery_Pacientes.SQL.Text:='select * from pacientes
where
PACFULL like ' + quotedstr(edit5.text + '%');
Form2_password.IBQuery_Pacientes.Open;
```

VENTANA BÚSQUEDA DE REGISTROS ALFABÉTICAMENTE, EL CUAL FILTRA LOS REGISTROS.

En esta ventana se realiza la búsqueda de los registros de forma rápida y específica, logrando localizar el registro deseado.

Imagen N° 04: Ventana para la Búsqueda de registro

HISTORIA	APELLIDOS Y NOMBRES	PADRE	MADRE
2660	MAMANI MAMANI ALEX WILDER	JUAN	BENEDICTA
3021	MAMANI MAMANI ALFREDO	JUAN VICTOR	NICOLASA
3461	MAMANI MAMANI ABEL ROEL	CELSO	CRISTINA
5961	MAMANI MAMANI ASUNTA	QUINTIN	SIMONA
7440	MAMANI MAMANI AURELIANO	ANTONIA	LENADRO
8047	MAMANI MAMANI ANGEL ALBERTO	SABINO	ASUNTA
9507	MAMANI MAMANI AMELIA	FRANCISCO	FELIPA
11792	MAMANI MAMANI AMELIA	ANGEL	TEODOSIA
14640	MAMANI MAMANI ADDY BENILDA	HIPOLITO	VIMA MAXIMILIAN
16138	MAMANI MAMANI ALICIA	CIRILO	SERAFINA

Finalmente este trabajo de tesis se puede decir que la atención de admisión utilizando un Sistema de Información Multiusuario en el hospital de Ilave, es óptimo y pasan a resolver los siguientes puntos:

- Facilidad de trabajo en red, en base a un servidor Interbase.
- También se concluye que debido al incremento de ordenadores como clientes del servidor Internase (incremento de ventanillas de atención), disminuye el tiempo de espera de las personas que acuden al hospital de Ilave.
- Es imposible poder crear múltiples ventanillas sin un servidor de base de datos el cual demanda más tiempo en la atención a los pacientes y provoca extensas colas en la ventanilla de ADMISION del Hospital Redess Collao Ilave

CONCLUSIONES

- Con respecto a Objetivo general se logró optimizar la atención de admisión, utilizando el Sistema de Información Multiusuario (ADMISIÓN 2007) en el hospital de Ilave.

  La atención en el hospital de Ilave Redess Collao mejoro con la implementación del sistema de información multiusuario.

  El Sistema de Información Multiusuario nos ayudó a optimizar la atención de admisión en el Hospital de Ilave utilizando un servidor interbase.

- El software ADMISIÓN 2007 fue analizado, diseñado utilizando la tecnología de desarrollo cliente servidor y acompañada con la tecnología reutilización de componentes (dll), se utilizó RUP que es un proceso de desarrollo de software, junto con el Lenguaje Unificado de Modelado UML, para el desarrollo del software se utilizó el programa Delphi 7.0 y Visual C++.

- La implementación de la base de datos a un servidor interbase se realizó con el diseño de las tablas.

RECOMENDACIONES

- Se recomienda al director del hospital de Ilave Redess – Collao, incrementar las capacidades de Hardware en lo que se refiere a la Memoria Ram, Microprocedor del Servidor, todo ello para reducir el tiempo de respuesta del servidor a los PC- clientes, (ventanillas de atención) para así poder brindar una mejor atención en múltiples ventanillas, mejorando así la calidad de atención al Público.

- Se recomienda revisar el manual del usuario para poder dar el uso correcto y adecuado del Sistema de admisión.

- Se recomienda a las empresas privadas, como a las instituciones públicas actualizar sus sistemas de información monousuario a multiusuario para que a si logren mejorar la calidad de atención.

- También se recomienda que el personal que se desempeñe como usuario del Sistema de admisión tenga conocimientos básicos de computación, A nivel de usuario, todo ello para que no exista insatisfacción por parte de los pacientes, ya que estas deficiencias generaran quejas en contra de la institución.

- Se recomienda a los estudiantes de la Escuela Profesional de Ingeniería Estadística e Informática que tomen este trabajo como base para realizar otro trabajo similar o mejor, ya que este trabajo contiene información de carácter importante

BIBLIOGRAFÍA

CHIQUE, A. (2003) “Desarrollo de Software para los Métodos de Líneas de Espera, Aplicada al Servicio de Admisión del Hospital Regional “Manuel Núñez Butrón” Tesis: UNA –Puno.

CONDORI, Q. (2005) “ Desarrollo de Sistemas de Información Virtual para la Gestión Administrativa del CEGNE Sagrado Corazón de Jesús JULIACA-2003”, Tesis:UNA – Puno.

GILDARDO GUTIERREZ, JAMES (UNIMINUTO). (2005) “Tecnología en Informática. Bases de datos”edición bello – lima.

GUILLERMO VALLE, JOSE (2005). “Taller de Sistemas de Información 1” , Edición. InCo -Facultad de Ingeniería– lima.

RAMIREZ, EDUARDO (2001). “Aplicando SQL Server 2000”, Edición MACRO-centro de investigación y desarrollo.

SCHILDT, HERBERT(1990). “Guía para usuarios Expertos”, Edición IMPRESA- Viena fotocomposición.

SUCARI I. (2002). ”Optimización de Trayectorias Mediante Algoritmos Genéticos para Recorrido de un Robot Móvil en un Ambiente con Obstáculos”, Tesis:UNA – Puno.

WEBGRAFÍA

http://www.elrincondelvago.com, accesada [09/10/05]

http://www.ciber-tec.com/ads.htm, accesada [09/10/05]

http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/czap6-3.html, accesada [09/10/05]

http://ww.pcm.gobpe/portal_ongei/publicaciones/cultura/Lib5038, accesada [10/11/05]

http://www.monografias.com/trabajos24/bases-de-datos/bases-de-datos.shtml, accesada [10/11/05]

http://www.macine.epublish.cl/tesis/index-3_3_.html, accesada [10/11/05]

http://www.monografias.com/trabajos24/arquitectura-cliente-servidor/arquitectura-cliente-servidor.shtml, accesada 10/12/05]

http://www.ccdis.dis.ulpgc.es/ccdis/laboratorios/redes.html, accesada [11/12/05]

http://www.es.wikipedia.org/wiki/TCP/IP, accesada [11/12/05]

http://www.adhocnet.com/varis/glosario.htm, accesada [11/12/06]

http://www.linux-es.org/Html/FAQ_Linux_V2.0.2-146.html, accesada [12/05/05]

http://www.chenico.com/glosariotu.htm, accesada [12/05/06]

http://www.fbio.uh.cu/bioinfo/glosario.html, accesada [12/06/06]

ANEXO A

Ficha de Evaluación de la Calidad del Proyecto Estándar ISO - 9126

FICHA DE EVALUACIÓN DE LA CALIDAD DEL PRODUCTO

ESTANDAR ISO - 9126

Tabla N° 07: Ficha de evaluación de la calidad de software

INDICADORES	PUNTUACIÓN				
	1	2	3	4	5
1. FUNCIONALIDAD					
Adecuación: la capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas específicas y objetivos de los usuarios.					
Exactitud: la capacidad del producto software para proporcionar los resultados o efectos correctos y con el grado de precisión acordado.					
Interoperabilidad: la capacidad del producto software para interactuar con uno o más sistemas especificados					
Seguridad: referido a la capacidad del producto software para proteger la información y los datos					
Conformidad: la capacidad del producto software para adaptarse a los estándares, convenciones o regulaciones en leyes y prescripciones relativos a la funcionalidad					
2. FIABILIDAD					
Madurez: la capacidad del producto software para evitar fallos provocados por errores en el software.					
Tolerancia a fallos: la capacidad del producto software para mantener un nivel de rendimiento determinado en caso de defectos en el software o incumplimiento de su interfaz.					
Recuperabilidad: la capacidad del producto software para restablecer un determinado nivel de rendimiento y recuperar los datos afectados directamente en caso de ocurrir un fallo.					
Conformidad: la capacidad del producto software para adaptarse a estándares, convenciones y regulaciones referidas a la fiabilidad.					
3. USABILIDAD					
Comprensibilidad: la capacidad del producto software para permitir al usuario que entienda si el software es adecuado, y como debe utilizarse para determinadas tareas y bajo ciertas condiciones de uso.					
Facilidad de aprendizaje: la capacidad del producto software para permitir al usuario aprender su aplicación.					
Atracción: la capacidad del producto software para atraer al usuario.					
Conformidad: la capacidad del producto software para adaptarse a estándares, convenciones, guías de estilo y regulaciones relacionadas con la usabilidad.					
Operabilidad: la capacidad del producto software para permitir que el usuario lo opere y lo controle.					

I INDICADORES	PUNTUACIÓN				
	1	2	3	4	5
4. EFICIENCIA					
Comportamiento temporal: la capacidad del producto software para proporcionar tiempos de respuesta y de procesamiento apropiados cuando realiza sus funciones bajo condiciones determinadas.					
Utilización de recursos: la capacidad del producto software para utilizar cantidades y tipos de recursos apropiados cuando el software realiza su función bajo determinadas condiciones.					
Conformidad: la capacidad del producto software para adaptarse a estándares o convenciones relacionadas con la eficiencia.					
5. MANTENIBILIDAD					
Analizabilidad: Capacidad del producto software de diagnosticar sus deficiencias o causas de fallos, o de identificar las partes que deben ser modificadas.					
Cambiabilidad: Capacidad del producto software de permitir implementar una modificación especificada. La implementación incluye los cambios en el diseño, el código y la documentación.					
Estabilidad: Capacidad del producto software de evitar los efectos inesperados de las modificaciones.					
Facilidad de prueba: Capacidad del producto software de permitir validar las partes modificadas.					
Conformidad: Capacidad del producto software de cumplir los estándares o convenciones relativas a la mantenibilidad.					
6. PORTABILIDAD					
Adaptabilidad: la capacidad del producto software para ser adaptado para ambientes determinados sin realizar acciones o aplicar medios, más que los proporcionados para este propósito para el software considerado.					
Facilidad de instalación: la capacidad del producto software para ser instalado en un ambiente determinado.					
Coexistencia: la capacidad del producto software para coexistir con otro software independiente en un ambiente común compartiendo recursos.					
Reemplazabilidad: la capacidad del producto software para ser utilizado en lugar de otro producto de software para el mismo propósito en el mismo ambiente.					
Conformidad: la capacidad del producto software para adaptarse a estándares relacionados con la portabilidad.					
SUB TOTALES					
TOTAL					

ANEXO B

Encuesta

ENCUESTA

La presente encuesta es de carácter privado, por lo que le agradecemos de antemano su participación. Sus respuestas nos ayudarán a priorizar las áreas de mejora de nuestro *Sistema de Admisión Multiusuario* y adaptarlo mejor a las necesidades de su trabajo:

Lea las preguntas y marque con una (X)

1. ¿Cómo Considera Ud. el uso de la **interfaz** del sistema?

   a) Muy fácil de usar

   b) Fácil de usar

   c) Difícil de usar

   d) Muy difícil de usar

2. ¿La interfaz del sistema proporciona todos los medios para la **interacción** con los operadores del sistema?

   a) Esta muy completo

   b) Completo

   c) Falta añadir opciones

   d) Puede mejorar

3. Considera que las consultas operadores del sistema en cuanto a las observaciones realizadas en las revisiones, con el uso del sistema son:

   a) Más personalizadas y con mayor frecuencia

   b) No se pudo hacer consultas

   c) No hicieron mucho uso del sistema

   d) Faltó coordinación

4. La aprobación de su trabajo de investigación en función a la **usabilidad** del sistema de admisión, se realizó:

   e) Muy fácil de usar

   f) Fácil de usar

   g) Difícil de usar

   a) Muy difícil de usar

5. La aprobación de su trabajo de investigación en función al costo, con el uso el sistema, fue:

   a) Menos costoso

   b) Costo normal

   c) Costo mayor a lo normal

   d) Demasiado costoso

6. En las revisiones de su trabajo, considera que fueron:

   a) Muy interactivas y dinámicas

   b) Personalizadas y objetivas

   c) Normales sin mayor ventaja

   d) No se pudo participar muy activamente

ANEXO C

Plan de Capacitación

PLAN DE CAPACITACIÓN

Tabla N° 08: Cronograma de Capacitación

CRONOGRAMA DE CAPACITACIÓN			
FECHA	**HORA**	**TEMA**	**DESCRIPCIÓN**
08/03/2007	3:00 p.m. – 3:30 p.m.	Sistemas Cliente – Servidor	Conocimientos fundamentales de aplicaciones cliente servidor, trabajo en múltiples terminales
09/03/2007	3:00 p.m. – 3:30 p.m.	Instalación y desinstalación del software.	Descripción y adiestramiento de instalación del software, requerimientos de Hardware.
10/03/2007	3:00 p.m. – 3:30 p.m.	Configuración de Servidor Interbase	Detalles sobre la configuración del servidor Interbase.
11/03/2007	3:00 p.m. – 3:30 p.m.	Manejo y uso del Sistema de admisión	Reconocimiento de ventanas, manejo del administrador de usuarios, manejo y generación de reportes, manejo de editor de sentencias SQL, Otros

Sistemas Cliente- Servidor .- Se dio a conocer los conceptos fundamentales de aplicaciones cliente servidor, la forma como poder acceder a múltiples ventanillas mediante una red , configuración de la red para que así puedan acceder con muchas facilidad a múltiples terminales .

Instalación y Desinstalación del Software.-Se les describió detalladamente la forma de como poder instalar el software admisión 2007 , los requerimientos mínimos de hardware que debe tener una PC para que ellos puedan instálalo en cualquier otra Pc que cumplan estos requerimientos .

Manejo y Uso del Sistema de Admisión.- En este tema se les dio a conocer la forma adecuado como deben de hacer su manejo del software para que más adelante no tengan problemas en el almacenamiento de base de datos , generación de reportes , consultas con el editor de sentencias SQL.

ANEXO D

Manual del Usuario

UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO

FACULTAD DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

ESCUELA PROFECIONAL DE INGENIERÍA ESTADÍSTICA E INFORMÁTICA

MANUAL DE USUARIO

DESARROLLO DEL SISTEMA DE INFORMACIÓN MULTIUSUARIO PARA ADMISIÓN DEL HOSPITAL DE ILAVE – 2007

Elaborado por: Bach. Gladys Marleny AUQUITIAS CONDORI

ÍNDICE

MANUAL DEL USUARIO

El presente manual, tiene como objetivo, instruir al usuario, sobre el SISTEMA DE ADMISIÓN, presentando y explicando las principales pantallas de ingreso de datos y los reportes y generados con la información procesada.

1) VENTANA DE AUTENTIFICACION

En esta sección se requiere darse de alta en el sistema para poder acceder al sistema de Admisión y posteriormente dar uso al sistema, ingresando su nombre de usuario y su respectivo Password, finalmente presionamos **Aceptar** **,** en caso la contraseña sea correcta se lograra ingresar, en caso contrario no se podrá acceder.

Figura N°01: Ventana de Autentificación

2) VENTANA GENERAL

Este administrador nos permite navegar entre los registros de la base de datos mediante sus botones de navegación

Así como también nos permite agregar registros, eliminar, editarlos. Y en la parte superior contiene menús de acceso a otras ventanas como por ejemplo la ventana de generación de reportes.

En la parte inferior podemos visualizar el número de visitas que tuvo un paciente al centro de atención médica

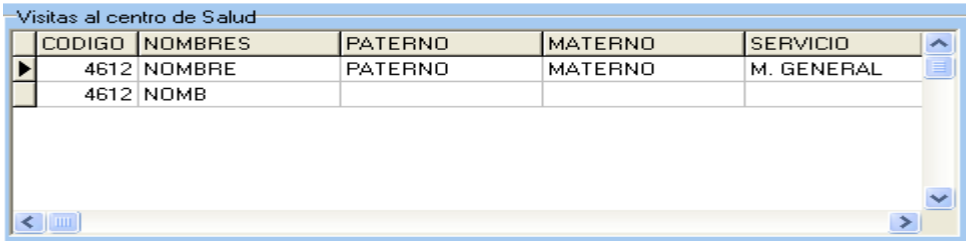


Figura N°02: Reporte de Visitas

Ahora podemos visualizar en la siguiente imagen la ventana Administrador

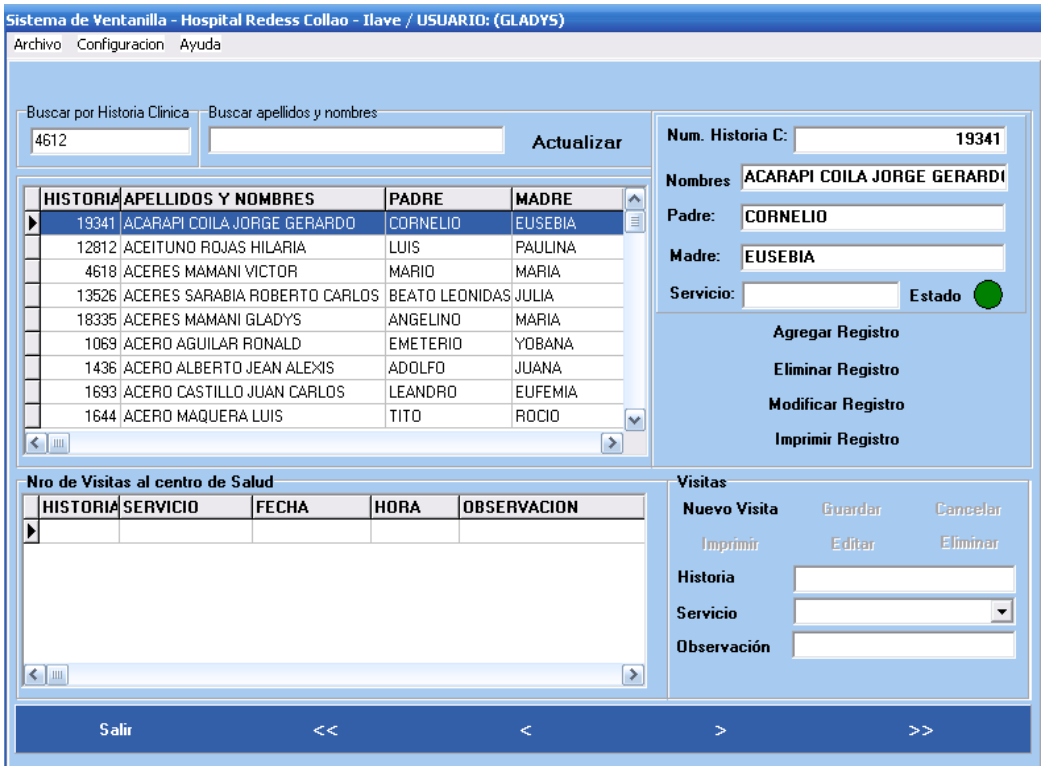


Figura N°03: Ventana de Búsqueda por: Nro. de Historia Clínica, Apellidos y Nombres

3) IMPRIMIR REGISTROS

En esta ventana podemos imprimir, reportes, registros.

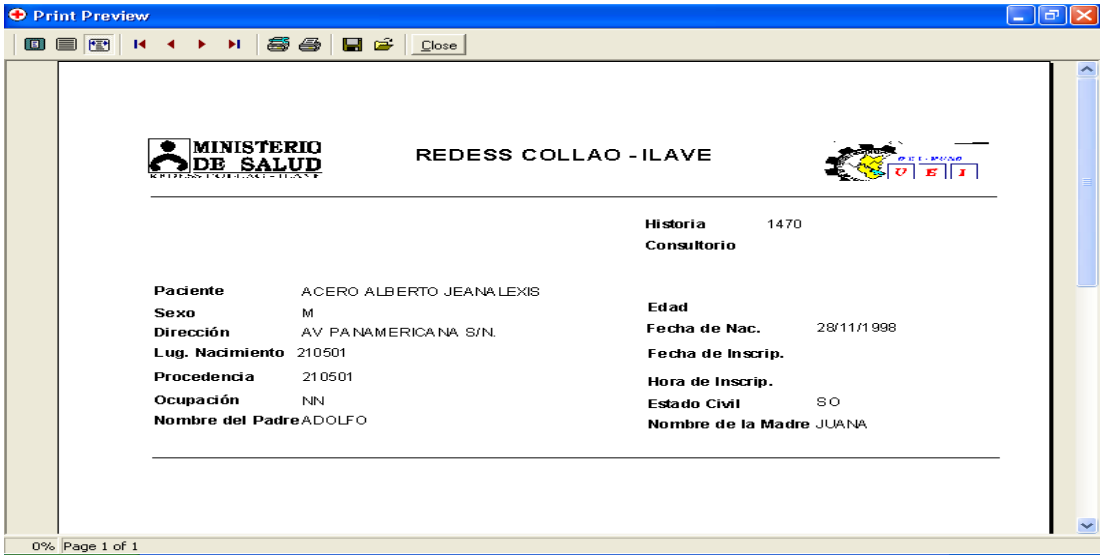


Figura N°04: Visualización de Modo de Impresión

4) VENTANA CAMBIAR PASSWORD

En esta se realiza el mantenimiento de cuentas de usuarios, en el cual podemos cambiar el Password y nombres de usuario.

Figura N°05: Ventana Para Cambiar Pasword

5) EDITOR DE CONSULTAS VIRTUALES

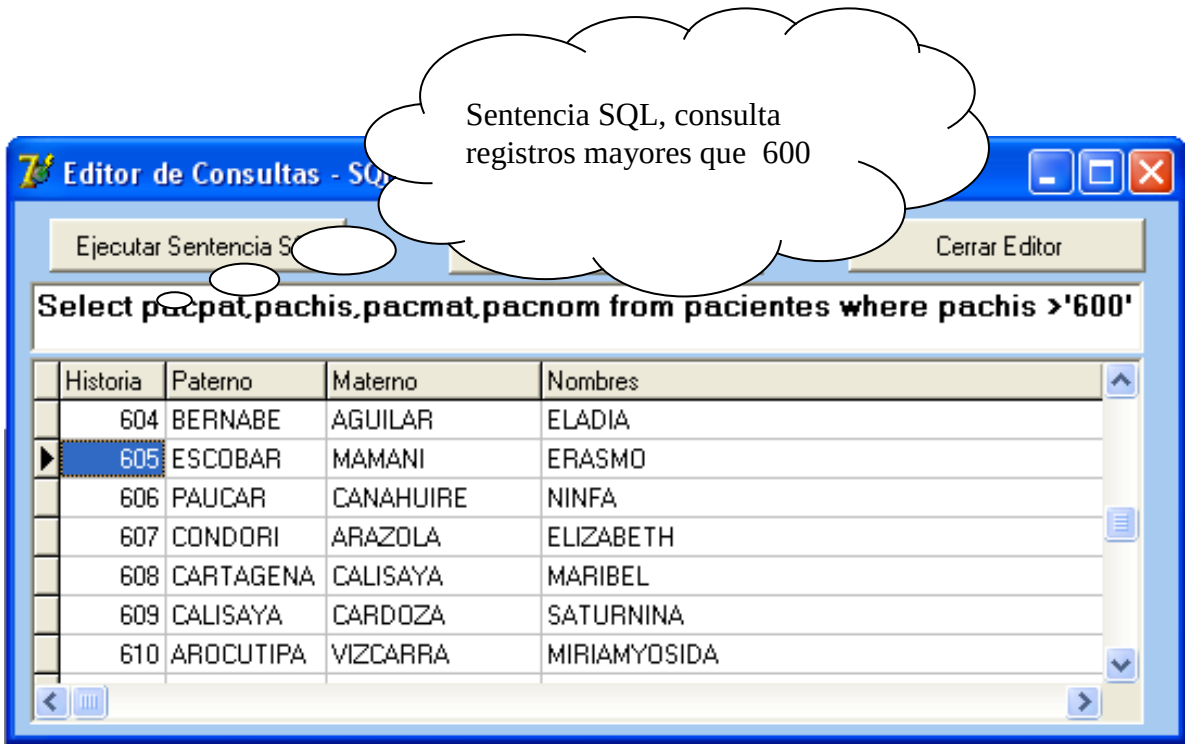


Figura N°06: Editor de Consultas

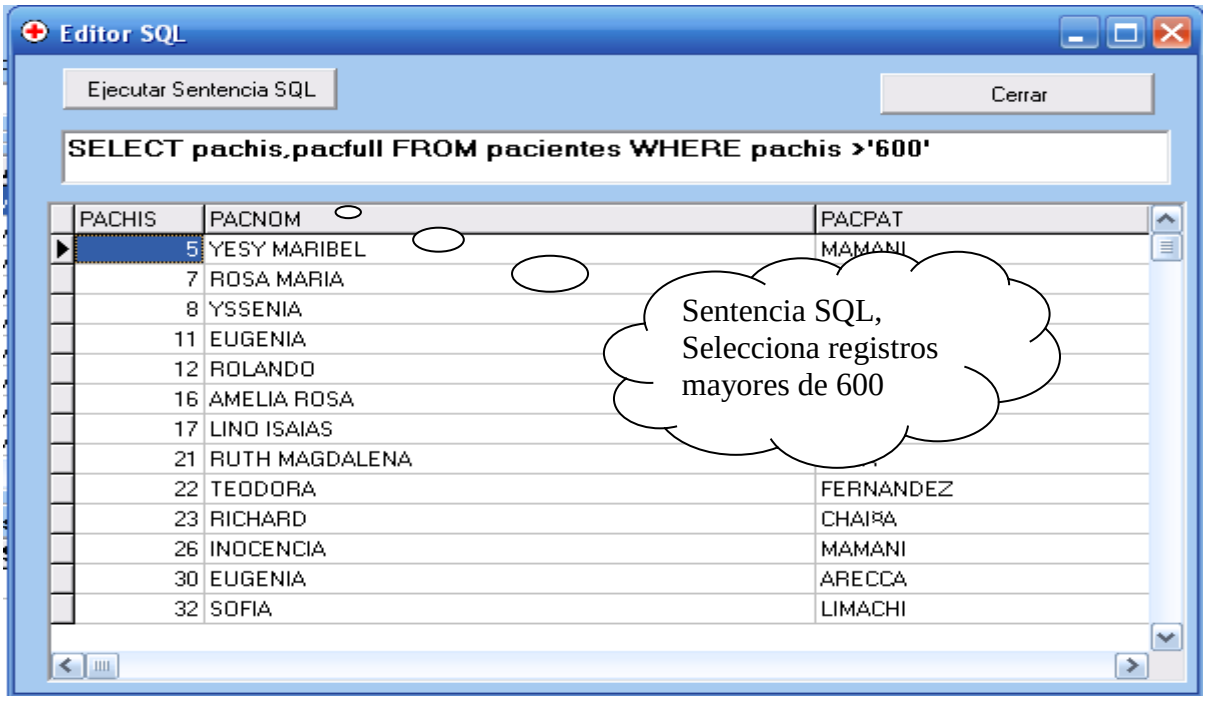


Figura N°07: Ventana Editor SQL (Selecciona registros)

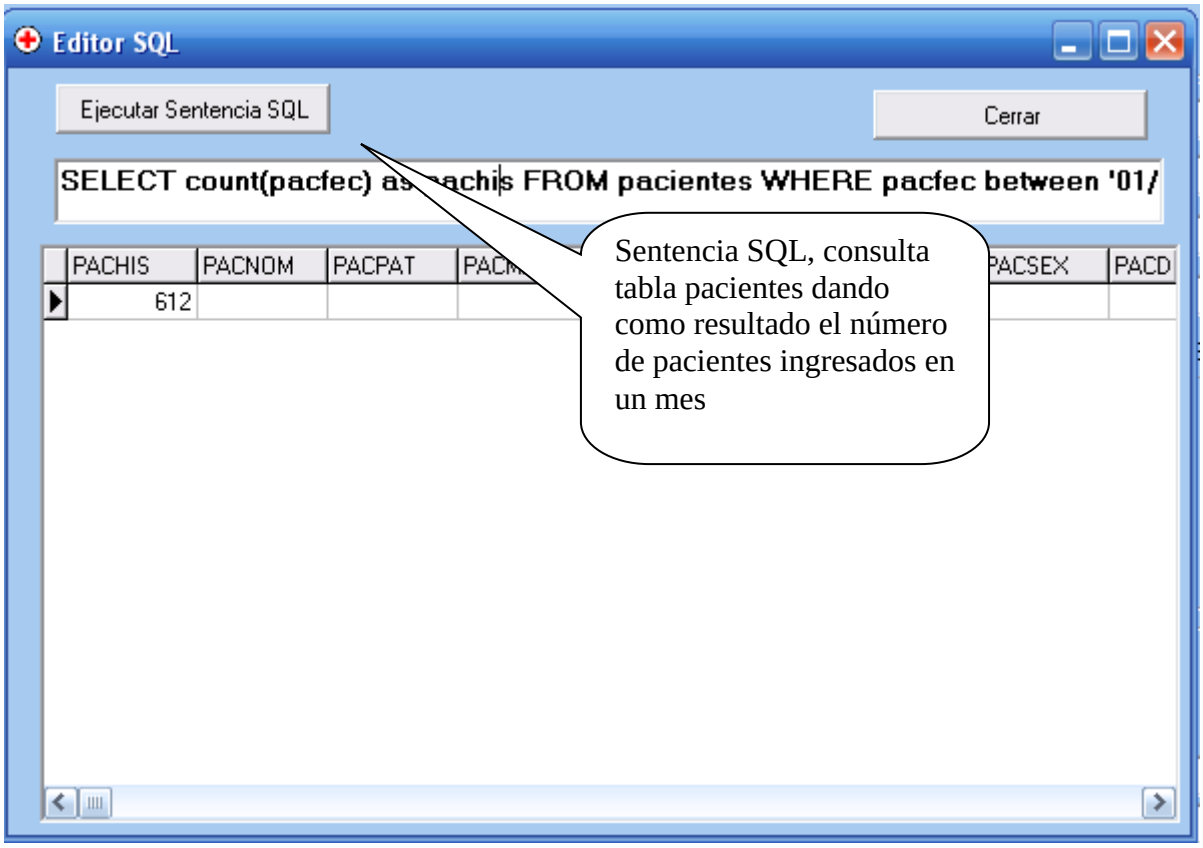


Figura N°08: Ventana Editor SQL (Consulta tabla)

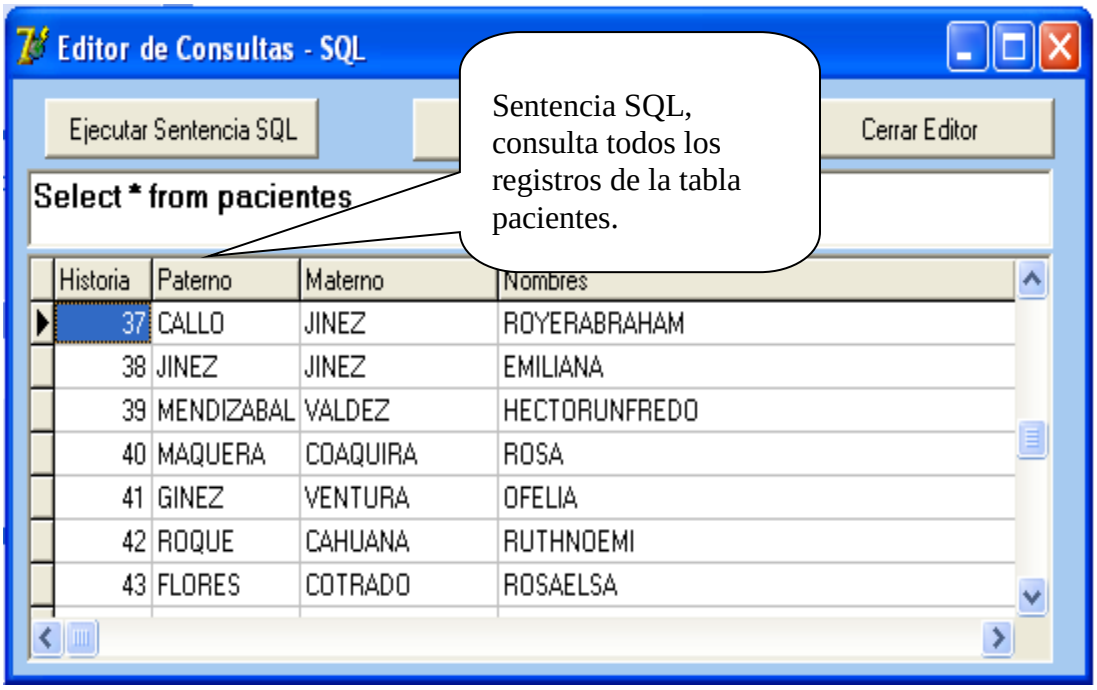


Figura N°08: Ventana Editor de Consulta SQL.

6) REPORTES GENERADOS

Ventana que genera reportes por fecha, el cual selecciona registros de acuerdo a la fecha de visita al centro de salud.

Figura N°09: Ventana para Generar Reportes Por Fecha

MINISTERIO DE SALUD
REDESS COLLAO - ILAVE

REDESS COLLAO - ILAVE

LISTA DE PACIENTES - INGRESADOS POR FECHA

Historia	Paciente	Padre	Madre	Edad	Fecha
635	CUEVA VELAZCO FELISA	CALIXTO	NICOLASA	65	02/01/2001
637	MAMANI ROBLES ERIKA	SABINO	ELENA	23	02/01/2001
638	JINEZ MAMANI SAMUEL	FELICIANO	MARIA	39	02/01/2001
639	VALDIVIA GORDILLO JOSEFA	VITALIANO	ALEJANDRINA	69	02/01/2001
640	CLEMENTE QUISPE HILDA	JUAN	ELENA	39	02/01/2001
641	COPA JALANOCA GENARA	CRUZ	VIVIANA	41	02/01/2001
642	LAYME LUPACA ALICIA	MARCELINO	ALICIA	81	02/01/2001
643	MAQUERA CERVANTES HERMINIA	BERNARDO	SOFIA	29	02/01/2001
644	LAYME CERVANTES SOFIA	PATRICIO	FAUSTA	51	02/01/2001
645	HUALLPARTUPA CCOSI RICHARD	JUSTO	VIVIANA	26	02/01/2001
646	GOMEZ CUTIPA NANCY GRACIELA	MARTIN	NICOLASA	30	02/01/2001

Figura N°10: Ventana Modo de Impresión - Reporte de Lista de Pacientes

REPORTE GENERADO

REPORTE DE VISITAS EN DETALLE

MINISTERIO DE SALUD
REDESS COLLAO - ILAVE

REDESS COLLAO - ILAVE

RESUMEN DE VISITAS AL CENTRO DE SALUD

Historia	Consultorio	Observación	Fecha	Hora
21797	DENTAL	GRAVE	09/12/2006	21:43:28
21797	MEDICINA	NEUMONIA	09/12/2006	21:46:44
21797	ORIENTACION	PLANIFICASION FAMILIAR	11/12/2006	7:30:08
21797	MEDICINA	NEUMONIA	09/12/2006	21:47:20
21797	CIRUGIA	HERIDO	09/12/2006	21:48:37
21797	CIRUGIA	GRAVE	10/12/2006	8:25:51
21797	TRAUMATOLOGIA	ACCIDENTE	10/12/2006	8:26:22
21797	SICOLOGIA	TRAUMA	10/12/2006	8:27:15
21797	MEDICINA GENERAL	ERIDAS LEVES	10/12/2006	8:35:39

7) REPORTE DE AGREGAR NUEVO REGISTRO

MINISTERIO DE SALUD
REDESS COLLAO - ILAVE

REDESS COLLAO - ILAVE

		Historia	21797
		Consultorio	DENTAL
Paciente	QUISPE TORRES MARITZA	**Edad**	26
Sexo	M	**Fecha de Nac.**	10/12/1980
Dirección	ARGENTINA 105	**Fecha de Inscrip.**	10/12/2006
Lug. Nacimiento	210501 ILAVE	**Hora de Inscrip.**	21:07:20
Procedencia	210501 ILAVE	**Estado Civil**	SO
Ocupación	ES	**Nombre de la Madre**	FLORENCIA
Nombre del Padre	JAIME	**Usuario**	

DATOS DEL PACIENTE

(Primera Visita)

		Historia	21797
		Consultorio	DENTAL
Paciente	QUISPE TORRES MARITZA		
Sexo	M	**Edad**	26
Dirección	ARGENTINA 105	**Fecha de Nac.**	10/12/1980
Lug. Nacimiento	210501 ILAVE	**Fecha de Inscrip.**	10/12/2006
Procedencia	210501 ILAVE	**Hora de Inscrip.**	21:07:20
Ocupación	ES	**Estado Civil**	SO
Nombre del Padre	JAIME	**Nombre de la Madre**	FLORENCIA

8) REPORTE GRAFICO

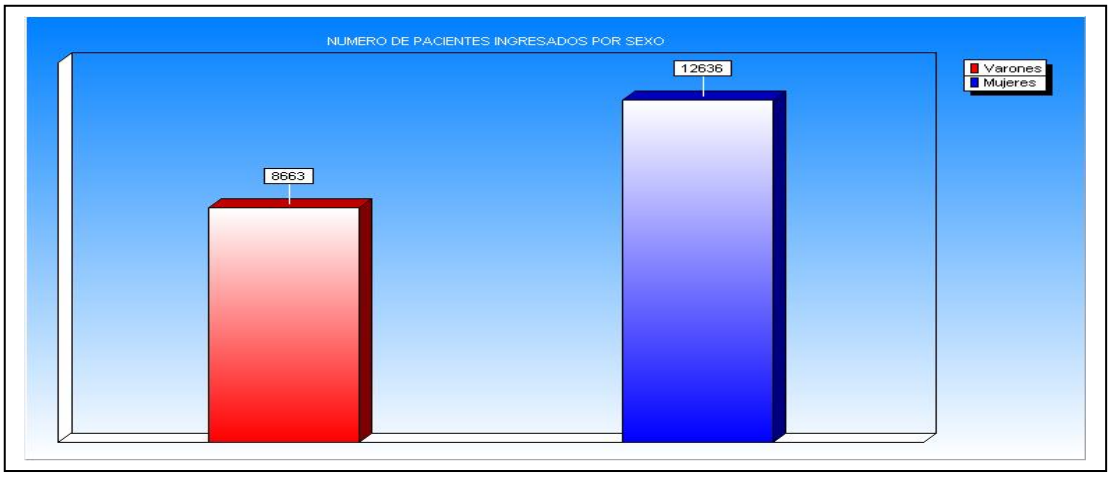
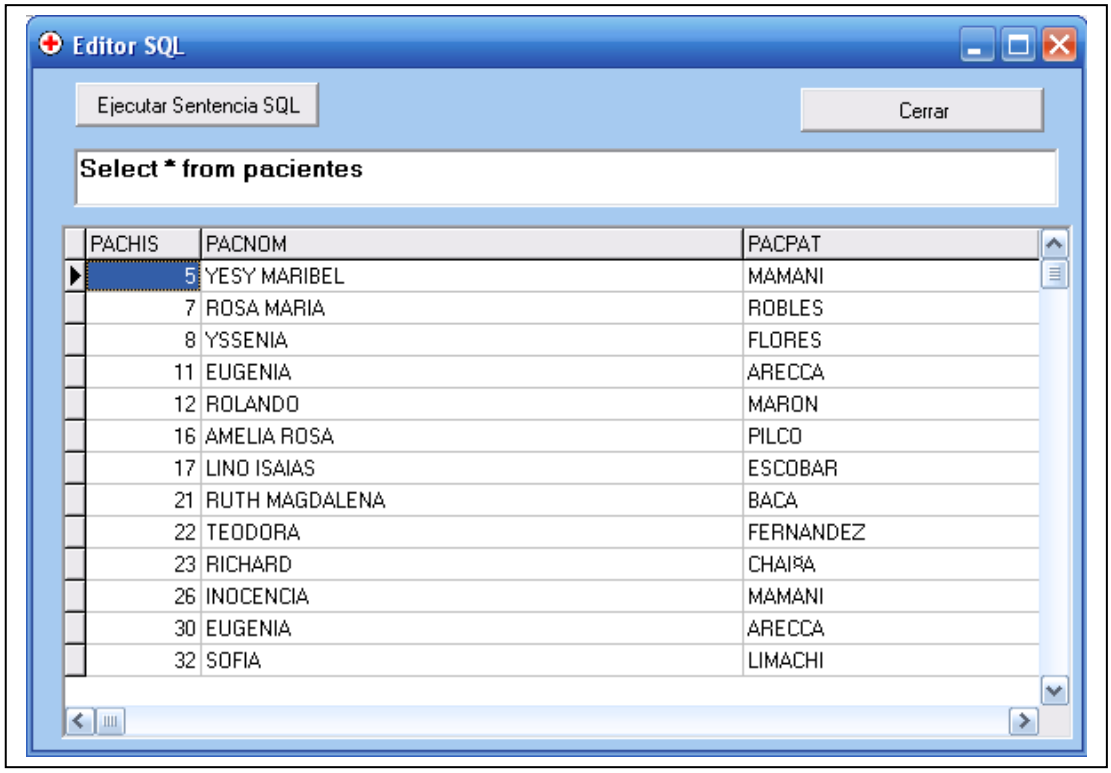
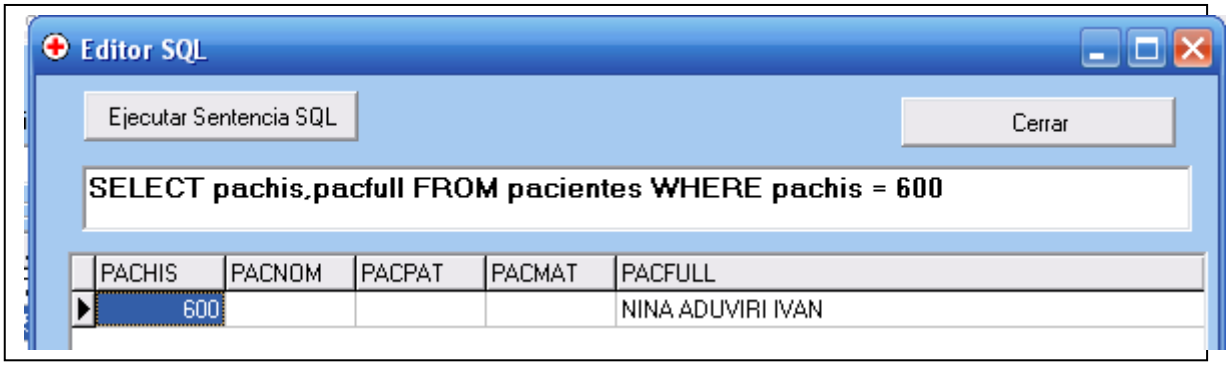


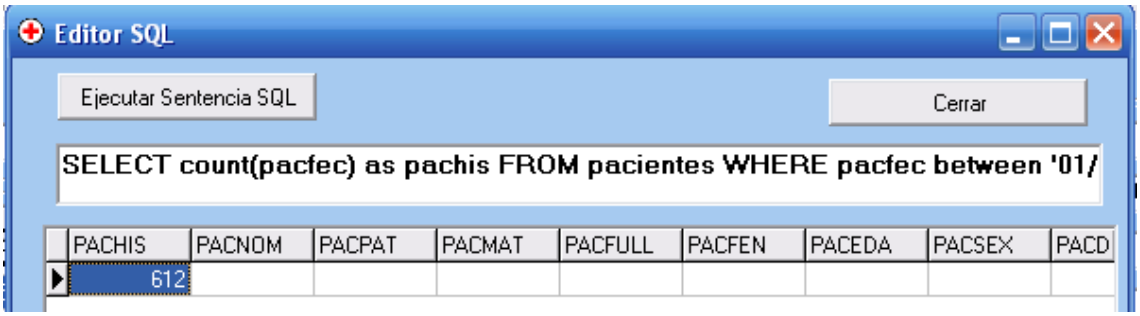
Grafico N°01: Número de Pacientes Ingresados por Sexo

9) REPORTES VIRTUALES



PACHIS	PACNOM	PACPAT
5	YESY MARIBEL	MAMANI
7	ROSA MARIA	ROBLES
8	YSSENIA	FLORES
11	EUGENIA	ARECCA
12	ROLANDO	MARON
16	AMELIA ROSA	PILCO
17	LINO ISAIAS	ESCOBAR
21	RUTH MAGDALENA	BACA
22	TEODORA	FERNANDEZ
23	RICHARD	CHAIÑA
26	INOCENCIA	MAMANI
30	EUGENIA	ARECCA
32	SOFIA	LIMACHI





ANEXO E

CODIGO FUENTE

CODIGO FUENTE

FORMULARIO INICIO.- Visualiza la primera pantalla de presentación en el sistema

```
procedure TForm1_inicio.Timer1Timer(Sender: TObject);
begin
Form2_Password.Show;
Form1_inicio.Hide;
Form1_inicio.TIMER1.Enabled :=FALSE;
end;

End.
```

FORMULARIO PASSWORD.- Nos visualiza la primera ventana para ingresar el passward del usuario, para tener acceso al sistema.

```
unit Password;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ExtCtrls, Buttons, StdCtrls, jpeg, DB, IBCustomDataSet,
IBDatabase, IBTable, IBQuery, Grids, DBGrids, Mask, DBCtrls, IBEvents;
const
NombreDLL       = 'Encode.dll';
type
Funcion_Encriptar=function(Cadena : String; Password : LongInt ): Pchar; stdcall;
Funcion_Desencriptar=function(Cadena : String; Password : LongInt ): Pchar; stdcall;
type
TForm2_password = class(TForm)

    Panel1: TPanel;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Image1: TImage;
    Panel2: TPanel;
    IBDatabase1: TIBDatabase;
    IBTransaction1: TIBTransaction;
    IBQuery_Password: TIBQuery;
    DataSource_Password: TDataSource;
    DBEdit3: TDBEdit;
    DataSource_Pacientes: TDataSource;
    DataSource_CodCiudad: TDataSource;
    DataSource_Registrado: TDataSource;
    IBQuery_Pacientes: TIBQuery;
    IBQuery_CodCiudad: TIBQuery;
    IBQuery_Registrado: TIBQuery;
    IBQuery_Agregar: TIBQuery;
    DataSource_Agregar: TDataSource;
    IBTable_Usuarios_Cambiar_Password: TIBTable;
    DataSource_Cambiar_Password: TDataSource;
    IBQuery_Print_Registro: TIBQuery;
    DataSource_Print_Registro: TDataSource;
    IBDataSet_Modificasiones: TIBDataSet;
    DataSource_Modificasiones: TDataSource;
    IBEvents_Modificasiones: TIBEvents;
    DBEdit2: TDBEdit;
```

```
    TxtUsuario: TEdit;
    TxtDecoded: TEdit;
    procedure SpeedButton2Click(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure IBEvents_ModificasionesEventAlert(Sender: TObject;
    EventName: String; EventCount: Integer; var CancelAlerts: Boolean);
    procedure Edit2KeyPress(Sender: TObject; var Key: Char);
    procedure FormDestroy(Sender: TObject);

private

  { Private declarations }

public

 HandleDLL               : THandle;

 Funcion_DLL_Encriptar     : Funcion_Encriptar;

Funcion_DLL_Desencriptar  : Funcion_Desencriptar;

  { Public declarations } end;

var
Form2_password: TForm2_password;
implementation
uses
Administrador,
Reporte,
TipoUsuario;

{$R *.dfm}

procedure TForm2_password.SpeedButton2Click(Sender: TObject);
begin
{IBQuery1.Active :=false;
IBTransaction1.Active :=false;
IBTransaction1.Active :=true;
IBQuery1.Active :=true;
DBGrid1.DataSource  := DataSource1;

}

application.Terminate;
end;



procedure TForm2_password.SpeedButton1Click(Sender: TObject);
begin

    TxtUsuario.Text:=UpperCase(TxtUsuario.Text); //Usuario -> Mayuscula
    EDIT2.Text:=UpperCase(edit2.Text); //Password -> Mayuscula

   //REALIZAMOS UNA CONSULTA BUSCANDO POR USUARIO WHERE Txt Usuario.Text

    IF Length(TxtUsuario.Text)>1 then begin
```

```
    IBQuery_Password.Close;
    IBQuery_Password.SQL.Clear;
    IBQuery_Password.SQL.Text:='select * from usuarios where usualias =' +
        quotedstr(TxtUsuario.text);
    IBQuery_Password.Open;
    END;

IF Length(DBEdit3.Text)>0 then begin

   TxtDecoded.Text:=Funcion_DLL_Desencriptar(DBEdit3.Text,990180);

if (TxtUsuario.Text=dbedit2.Text) then begin //Compara Usuario

   if edit2.Text = TxtDecoded.Text then begin //conparacion Password

    form2_password.Close;
    form3_administrador.Show;
    end
    else begin

        ShowMessage('Contraseña invalida ->[ '+ Edit2.Text + ' ] ' );

   end;

end
else begin

    ShowMessage('Este usuario no existe ->[ '+ TxtUsuario.Text + ' ] ' );

end;

end
else begin
ShowMessage('Este usuario no existe ->[ '+ TxtUsuario.Text + ' ] ' );
end;
end;
procedure TForm2_password.Edit1Change(Sender: TObject);
begin



IBQuery_Password.Close;
IBQuery_Password.SQL.Clear;

IBQuery_Password.SQL.Text:='select USUCODIG,USUALIAS,USUPASSW from USUARIOS
where USUALIAS like ' + quotedstr(DBEdit2.text + '%');

IBquery_Password.Open;
end;
procedure TForm2_password.Timer1Timer(Sender: TObject);
begin
//ACTUALIZA DATOS DE PRESENTACION

end;

procedure TForm2_password.FormCreate(Sender: TObject);
var KeyState: TKeyboardState;

// ACTIVA LAS TECLAS A MAYUSCULA
```

```
begin

HandleDLL:=LoadLibrary( PChar(ExtractFilePath(Application.Exename)+ NombreDLL )); if
HandleDLL = 0 then raise Exception.Create('No se pudo cargar la DLL');

@Funcion_DLL_Encriptar:=GetProcAddress(HandleDLL, 'Admision_Encriptar');

@Funcion_DLL_Desencriptar:=GetProcAddress(HandleDLL, 'Admision_Desencriptar');

IF not assigned(Funcion_DLL_Encriptar) OR not assigned(Funcion_DLL_Desencriptar) then

        raise Exception.Create('No se encontraron las funciones en la DLL'+#13+'Sistema
        Admision');

GetKeyboardState(KeyState);
if (KeyState[VK_CAPITAL]=0) then
KeyState[VK_CAPITAL]:=1
else KeyState[VK_CAPITAL]:=0;
SetKeyboardState(KeyState);

end;

procedure TForm2_password.IBEvents_ModificasionesEventAlert(
Sender: TObject; EventName: String; EventCount: Integer;
var CancelAlerts: Boolean);
begin



if EventName='NEW_CLIENTE' then

showmessage('Registro nuevo en la base de datos')//;
//ACTUALIZAR DATOS
{Form2_password.IBQuery_Agregar.Active:=false;
Form2_password.IBDataSet_Modificasiones.Active :=false;
Form2_password.IBQuery_Password.Active :=false;
Form2_password.IBQuery_Pacientes.Active:=false;
Form2_password.IBQuery_CodCiudad.Active :=false;
Form2_password.IBQuery_Registrado.Active :=false;
Form2_password.IBTable_Usuarios_Cambiar_Password.Active:=false;
Form2_password.IBQuery_Print_Registro.Active:=false;
Form2_password.IBTransaction1.Active :=false;
Form2_password.IBTransaction1.Active :=true;
Form2_password.IBQuery_Print_Registro.Active:=true;
Form2_password.IBTable_Usuarios_Cambiar_Password.Active:=true;
Form2_password.IBQuery_Registrado.Active :=true;
Form2_password.IBQuery_CodCiudad.Active :=true;
Form2_password.IBQuery_Pacientes.Active:=true;
Form2_password.IBQuery_Password.Active :=true;
Form2_password.IBDataSet_Modificasiones.Active :=true;
Form2_password.IBQuery_Agregar.Active :=true;
DBGrid1.DataSource  := Form2_password.DataSource_Agregar;
END

 }

else
if EventName='DEL_CLIENTE' then
showmessage('Se borró un registro de la tabla')
else
```

```
if EventName='CAMBIO_CLIENTE' then
showmessage('Se modificaron los datos del cliente');

end;

procedure TForm2_password.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
if (key = #13) then begin
speedbutton1.Click;
end;

end;

procedure TForm2_password.FormDestroy(Sender: TObject);
begin

   //DESCARGAMOS LAS Dlls DE MEMORIA

    if HandleDLL<>0 then
    FreeLibrary(HandleDLL);

end;

end.
```

FORMULARIO ADMINISTRADOR.- Este es el formulario principal del programa aquí podemos ingresar los datos del paciente que sean necesarios como: apellidos nombres, nombres de los padres, servicio que se atenderá si es su primera visita, estado en que se encuentra, lugar de procedencia,.

```
unit Administrador

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, Menus, ExtCtrls, IBDatabase, DB, IBCustomDataSet, IBQuery,
StdCtrls, Grids, DBGrids, Mask, DBCtrls, Buttons;
type
TForm3_administrador = class(TForm)

    Panel1: TPanel;
    MainMenu1: TMainMenu;
    Reportes1: TMenuItem;
    Reportedepacientes1: TMenuItem;
    Otros1: TMenuItem;
    Ayuda1: TMenuItem;
    AyudadeAdmision1: TMenuItem;
    AcercadeAdmision1: TMenuItem;
    N1: TMenuItem;
    Salir1: TMenuItem;
    CambiodeContraseas1: TMenuItem;
    CambiodeContraseas2: TMenuItem;
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    Edit4: TEdit;
    GroupBox3: TGroupBox;
    DBGrid1: TDBGrid;
    GroupBox4: TGroupBox;
    DBGrid2: TDBGrid;
```

```
Label4: TLabel;
Edit5: TEdit;
GroupBox5: TGroupBox;
Panel2: TPanel;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
DBEdit1: TDBEdit;
DBEdit2: TDBEdit;
DBEdit3: TDBEdit;
Label5: TLabel;
DBEdit4: TDBEdit;
GroupBox6: TGroupBox;
Label6: TLabel;
Label7: TLabel;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
Label11: TLabel;
Label12: TLabel;
Label13: TLabel;
DBEdit5: TDBEdit;
DBEdit6: TDBEdit;
DBEdit7: TDBEdit;
DBEdit8: TDBEdit;
DBEdit9: TDBEdit;
DBEdit10: TDBEdit;
DBEdit11: TDBEdit;
DBEdit12: TDBEdit;
SpeedButton1: TSpeedButton;
SpeedButton2: TSpeedButton;
SpeedButton3: TSpeedButton;
SpeedButton4: TSpeedButton;
Panel3: TPanel;
SpeedButton5: TSpeedButton;
SpeedButton6: TSpeedButton;
SpeedButton7: TSpeedButton;
SpeedButton8: TSpeedButton;
SpeedButton9: TSpeedButton;
SpeedButton10: TSpeedButton;
procedure Edit1Change(Sender: TObject);
procedure Edit2Change(Sender: TObject);
procedure Edit3Change(Sender: TObject);
procedure Salir1Click(Sender: TObject);
procedure Edit4Change(Sender: TObject);
procedure Edit5Change(Sender: TObject);
procedure SpeedButton5Click(Sender: TObject);
procedure DBEdit4Change(Sender: TObject);
procedure Edit5KeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
procedure Edit5KeyPress(Sender: TObject; var Key: Char);
procedure Edit4KeyPress(Sender: TObject; var Key: Char);
procedure Edit4KeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
procedure SpeedButton1Click(Sender: TObject);
procedure DBGrid1KeyPress(Sender: TObject; var Key: Char);
procedure DBGrid2KeyPress(Sender: TObject; var Key: Char);
procedure CambiodeContraseas2Click(Sender: TObject);
procedure SpeedButton6Click(Sender: TObject);
procedure SpeedButton7Click(Sender: TObject);
```

```
    procedure SpeedButton8Click(Sender: TObject);
    procedure SpeedButton9Click(Sender: TObject);
    procedure SpeedButton3Click(Sender: TObject);
    procedure SpeedButton10Click(Sender: TObject);

private

  { Private declarations }

public

  { Public declarations }

end;

var
Form3_administrador: TForm3_administrador;
implementation
uses
Agregar,
Password,
Cambiar_Password,
Reporte,
Lugar_De_Nacimiento_Procedencia,
TipoUsuario;

{$R *.dfm}

procedure TForm3_administrador.Edit1Change(Sender: TObject);
begin
end;
procedure TForm3_administrador.Edit2Change(Sender: TObject);
begin

PACIENTES WHERE pacpat LIKE ' + quotedstr(edit1.text + '%') + ' AND pacmat LIKE ' +
quotedstr(edit2.text + '%'));

end;

procedure TForm3_administrador.Edit3Change(Sender: TObject);
begin

PACIENTES WHERE pacpat LIKE ' + quotedstr(edit1.text + '%') + ' AND pacmat LIKE ' +
quotedstr(edit2.text + '%')+ ' and pacnom LIKE ' + quotedstr(edit3.text + '%'));

end;

procedure TForm3_administrador.Salir1Click(Sender: TObject);
begin
APPLICATION.Terminate;
end;
procedure TForm3_administrador.Edit4Change(Sender: TObject);
begin
EDIT5.Text:=";
EDIT5.Refresh;
IF Length(edit4.Text)>1 then begin

    Form2_password.IBQuery_Pacientes.Close;
    Form2_password.IBQuery_Pacientes.SQL.Clear;
```

```
   Form2_password.IBQuery_Pacientes.SQL.Text:='select
PACHIS,PACFULL,PACPAD,PACMAD from pacientes where pachis =' + quotedstr(edit4.text);

   Form2_password.IBQuery_Pacientes.Open;

END;

end;

procedure TForm3_administrador.Edit5Change(Sender: TObject);
begin
Form2_password.IBQuery_Pacientes.Close;
Form2_password.IBQuery_Pacientes.SQL.Clear;

Form2_password.IBQuery_Pacientes.SQL.Text:='select
PACHIS,PACFULL,PACPAD,PACMAD from pacientes where PACFULL like ' +
quotedstr(edit5.text + '%');

Form2_password.IBQuery_Pacientes.Open;
end;
procedure TForm3_administrador.SpeedButton5Click(Sender: TObject);
begin
application.Terminate;
end;

procedure TForm3_administrador.DBEdit4Change(Sender: TObject);

begin
if Length(DBedit4.Text)>1 then begin
Form2_password.IBQuery_Registrado.Close;
Form2_password.IBQuery_Registrado.SQL.Clear;

Form2_password.IBQuery_Registrado.SQL.Text:='select * from REGISTRADO where
pac_codigo=' + quotedstr(dbedit4.Text);

Form2_password.IBQuery_Registrado.Open;
end;

end;

procedure TForm3_administrador.Edit5KeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
begin
if (Key = VK_DOWN) then
begin

   DBGRID1.SetFocus;

end;



end;

procedure TForm3_administrador.Edit5KeyPress(Sender: TObject;
var Key: Char);
begin
if (key = #27) then begin
EDIT5.Text:='';
EDIT5.SetFocus;
```

```
end;

if (key=#13) then begin

//ACTUALIZAR DATOS

SpeedButton10.Refresh;
SpeedButton10.Click; // presiona el boton actualizar!
SpeedButton10.Repaint ;
end;

end;

procedure TForm3_administrador.Edit4KeyPress(Sender: TObject;
var Key: Char);
begin
if (key = #27) then begin
EDIT4.Text:=”;
EDIT5.SetFocus;
end;

end;

procedure TForm3_administrador.Edit4KeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
begin
if (Key = VK_DOWN) then
begin

   DBGRID1.SetFocus;

end;
end;
procedure TForm3_administrador.SpeedButton1Click(Sender: TObject);
begin
form_agregar.Show;
end;
procedure TForm3_administrador.DBGrid1KeyPress(Sender: TObject;
var Key: Char);
begin
if (key = #27) then begin

    EDIT4.Text:=”;
    EDIT5.Text:=”;
    EDIT5.SetFocus;

end;

if (key = #13) then begin

   DBGRID2.SetFocus;

end;

end;

procedure TForm3_administrador.DBGrid2KeyPress(Sender: TObject;
var Key: Char);
begin
if (key = #27) then begin
```

```
    EDIT4.Text:=”;
    EDIT5.Text:=”;
    EDIT5.SetFocus;

end;

end;

procedure TForm3_administrador.CambiodeContraseas2Click(Sender: TObject);
begin
FrmTipo_Usuario.Show;
end;
procedure TForm3_administrador.SpeedButton6Click(Sender: TObject);
begin
Form2_password.DataSource_Pacientes.DataSet.First;
end;
procedure TForm3_administrador.SpeedButton7Click(Sender: TObject);
begin
Form2_password.DataSource_Pacientes.DataSet.Prior;
end;
procedure TForm3_administrador.SpeedButton8Click(Sender: TObject);
begin
Form2_password.DataSource_Pacientes.DataSet.Next;
end;
procedure TForm3_administrador.SpeedButton9Click(Sender: TObject);
begin
Form2_password.DataSource_Pacientes.DataSet.Last;
end;
procedure TForm3_administrador.SpeedButton3Click(Sender: TObject);
begin
Edit4.Text :=”;
Edit5.Text :=”;
Edit4.Text :=DBEdit4.Text;
Edit4.Refresh;
Form2_password.IBQuery_Print_Registro.Close;
Form2_password.IBQuery_Print_Registro.SQL.Clear;

Form2_password.IBQuery_Print_Registro.SQL.Text:=’select * from pacientes where pachis =’ +
quotedstr(Edit4.text);

Form2_password.IBQuery_Print_Registro.Open;

// LISTO PARA LISTAR

Form_Reporte.QuickRep1.Preview;

end;

procedure TForm3_administrador.SpeedButton10Click(Sender: TObject);
begin

//ACTUALIZAMOS LOS DATOS

Form2_password.IBQuery_Agregar.Active :=false;
Form2_password.IBDataSet_Modificasiones.Active :=false;
Form2_password.IBQuery_Password.Active :=false;
Form2_password.IBQuery_Pacientes.Active:=false;
Form2_password.IBQuery_CodCiudad.Active :=false;
Form2_password.IBQuery_Registrado.Active :=false;
Form2_password.IBTable_Usuarios_Cambiar_Password.Active:=false;
```

```
Form2_password.IBQuery_Print_Registro.Active:=false;

    Form2_password.IBTransaction1.Active :=false;
    Form2_password.IBTransaction1.Active :=true;

Form2_password.IBQuery_Print_Registro.Active:=true;
Form2_password.IBTable_Usuarios_Cambiar_Password.Active:=true;
Form2_password.IBQuery_Registrado.Active :=true;
Form2_password.IBQuery_CodCiudad.Active :=true;
Form2_password.IBQuery_Pacientes.Active:=true;
Form2_password.IBQuery_Password.Active :=true;
Form2_password.IBDataSet_Modificasiones.Active :=true;
Form2_password.IBQuery_Agregar.Active :=true;
DBGrid1.DataSource  := Form2_password.DataSource_Pacientes;
DBGrid2.DataSource  := Form2_password.DataSource_Registrado;
end;

end.

program Admision;

uses
Forms,
Inicio in 'Inicio.pas' {Form1_inicio},
Password in 'Password.pas' {Form2_password},
Administrador in 'Administrador.pas' {Form3_administrador},
Agregar in 'Agregar.pas' {Form_agregar},
Lugar_De_Nacimiento_Procedencia in 'Lugar_De_Nacimiento_Procedencia.pas'
    {Form_Lugar_Nacim_Proced},
Cambiar_Password in 'Cambiar_Password.pas' {Form_Cambiar_Password},
Reporte in 'Reporte.pas' {Form_Reporte},
Estadistica in 'Estadistica.pas' {FrmEstadisticas},
TipoUsuario in 'TipoUsuario.pas' {FrmTipo_Usuario};

{$R *.res}

begin
Application.Initialize;
Application.Title := 'SISTEMA DE ADMISION - ILAVE';
Application.CreateForm(TForm1_inicio, Form1_inicio);
Application.CreateForm(TFrmTipo_Usuario, FrmTipo_Usuario);
Application.CreateForm(TForm2_password, Form2_password);
Application.CreateForm(TForm3_administrador, Form3_administrador);
Application.CreateForm(TForm_agregar, Form_agregar);
Application.CreateForm(TForm_Lugar_Nacim_Proced, Form_Lugar_Nacim_Proced);
Application.CreateForm(TForm_Cambiar_Password, Form_Cambiar_Password);
Application.CreateForm(TForm_Reporte, Form_Reporte);
Application.CreateForm(TFrmEstadisticas, FrmEstadisticas);
Application.Run;
end.
```

FORMULARIO REPORTE.- En este formulario nos genera los reportes como son: lista de pacientes, datos de un paciente, número de pacientes Ingresados por sexo para generar los reportes se ingresa los datos por fecha en el orden día /mes/año.

```
unit Reporte;

interface

uses
```

```
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ExtCtrls, QuickRpt, QRCtrls;
type
TForm_Reporte = class(TForm)

    QuickRep1: TQuickRep;
    TitleBand1: TQRBand;
    SummaryBand1: TQRBand;
    DetailBand1: TQRBand;
    QRLabel1: TQRLabel;
    QRDBText1: TQRDBText;
    QRLabel2: TQRLabel;
    QRImage1: TQRImage;
    QRImage2: TQRImage;
    QRLabel3: TQRLabel;
    QRLabel4: TQRLabel;
    QRDBText2: TQRDBText;
    QRDBText3: TQRDBText;

private

  { Private declarations }

public

   { Public declarations }

end;

var
Form_Reporte: TForm_Reporte;
implementation
uses
Password;

{$R *.dfm}

end.
```

FORMULARIO LUGAR DE PROCEDENCIA.- Nos visualiza los datos de lugar de procedencia con un código de ciudad , provincia, departamento.

```
unit Lugar_De_Nacimiento_Procedencia;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ExtCtrls, Mask, DBCtrls, StdCtrls, Grids, DBGrids;



type
TForm_Lugar_Nacim_Proced = class(TForm)

    GroupBox1: TGroupBox;
    Label1: TLabel;
    Edit1: TEdit;
    GroupBox2: TGroupBox;
    Label2: TLabel;
```

```
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Button1: TButton;
    Button2: TButton;
    GroupBox3: TGroupBox;
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    DBEdit4: TDBEdit;
    DBGrid1: TDBGrid;
    procedure Edit1Change(Sender: TObject);

private

  { Private declarations }

public

   { Public declarations }

end;

var
Form_Lugar_Nacim_Proced: TForm_Lugar_Nacim_Proced;
implementation
uses
Agregar,
Password,
Administrador;

{$R *.dfm}

procedure TForm_Lugar_Nacim_Proced.Edit1Change(Sender: TObject);
begin
Form2_password.IBQuery_CodCiudad.Close;
Form2_password.IBQuery_CodCiudad.SQL.Clear;

Form2_password.IBQuery_CodCiudad.SQL.Text:='select * from codciudad where
COD_DESCRIPCION like ' + quotedstr(edit1.text + '%');

Form2_password.IBQuery_CodCiudad.Open;
end;



end.
```

FORMULARIO CAMBIAR PASSWORD.- En esta ventana el administrador realiza el mantenimiento de cuente del usuario, puede cambiar las contraseñas de los usuarios, hacer las modificaciones de sus datos y actualización de sus datos.

```
unit Cambiar_Password;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, ExtCtrls, Grids, DBGrids, Buttons, StdCtrls, Mask, DBCtrls;
type
```

```
TForm_Cambiar_Password = class(TForm)

    Panel1: TPanel;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    DBGrid1: TDBGrid;
    DBEdit1: TDBEdit;
    DBEdit2: TDBEdit;
    DBEdit3: TDBEdit;
    DBEdit4: TDBEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    DBEdit5: TDBEdit;
    DBEdit6: TDBEdit;
    DBEdit7: TDBEdit;
    GroupBox1: TGroupBox;
    GroupBox2: TGroupBox;
    SpeedButton8: TSpeedButton;
    SpeedButton5: TSpeedButton;
    SpeedButton6: TSpeedButton;
    SpeedButton7: TSpeedButton;
    GroupBox3: TGroupBox;
    SpeedButton9: TSpeedButton;
    DBEdit8: TDBEdit;
    procedure SpeedButton5Click(Sender: TObject);
    procedure SpeedButton6Click(Sender: TObject);
    procedure SpeedButton8Click(Sender: TObject);
    procedure SpeedButton7Click(Sender: TObject);
    procedure SpeedButton9Click(Sender: TObject);

private

  { Private declarations }

public

   { Public declarations }

end;

var
Form_Cambiar_Password: TForm_Cambiar_Password;



implementation
uses
Password;

{$R *.dfm}

procedure TForm_Cambiar_Password.SpeedButton5Click(Sender: TObject);
```

```
begin
form2_password.DataSource_Cambiar_Password.DataSet.First;
end;
procedure TForm_Cambiar_Password.SpeedButton6Click(Sender: TObject);
begin
form2_password.DataSource_Cambiar_Password.DataSet.Prior;
end;
procedure TForm_Cambiar_Password.SpeedButton8Click(Sender: TObject);
begin
form2_password.DataSource_Cambiar_Password.DataSet.Next;
end;
procedure TForm_Cambiar_Password.SpeedButton7Click(Sender: TObject);
begin
form2_password.DataSource_Cambiar_Password.DataSet.Last;
end;
procedure TForm_Cambiar_Password.SpeedButton9Click(Sender: TObject);
begin
close;
end;

end.
```

FORMULARIO AGREGAR Este formulario agrega nuevos registros, en el momento de agregar registro nos genera la edad automáticamente , código de lugar de procedencia.

```
unit Agregar;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Grids, DBGrids, ExtCtrls, Mask, DBCtrls;
type
TForm_agregar = class(TForm)

    GroupBox1: TGroupBox;
    Panel2: TPanel;
    DBGrid1: TDBGrid;
    GroupBox2: TGroupBox;
    Label1: TLabel;
    Edit1: TEdit;
    GroupBox3: TGroupBox;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    GroupBox4: TGroupBox;
    GroupBox5: TGroupBox;
    Label2: TLabel;
    DBEdit1: TDBEdit;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    DBEdit9: TDBEdit;
    DBEdit8: TDBEdit;
    DBEdit7: TDBEdit;
```

```
    DBEdit6: TDBEdit;
    DBEdit5: TDBEdit;
    DBEdit4: TDBEdit;
    DBEdit3: TDBEdit;
    DBEdit2: TDBEdit;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label15: TLabel;
    Label17: TLabel;
    Label14: TLabel;
    Label18: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    DBText2: TDBText;
    DBText1: TDBText;
    DBComboBox1: TDBComboBox;
    DBEdit15: TDBEdit;
    DBEdit14: TDBEdit;
    DBEdit13: TDBEdit;
    DBEdit12: TDBEdit;
    DBEdit11: TDBEdit;
    DBEdit10: TDBEdit;
    GroupBox6: TGroupBox;
    Button6: TButton;
    Button7: TButton;
    Button8: TButton;
    Button9: TButton;
    GroupBox7: TGroupBox;
    Button4: TButton;
    Button10: TButton;
    Button11: TButton;
    Button12: TButton;
    Button13: TButton;
    GroupBox8: TGroupBox;
    Button5: TButton;
    DBEdit16: TDBEdit;
    Button14: TButton;
    procedure DBEdit8KeyPress(Sender: TObject; var Key: Char);
    procedure DBEdit7KeyPress(Sender: TObject; var Key: Char);
    procedure Button1Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button9Click(Sender: TObject);
    procedure Button10Click(Sender: TObject);
    procedure Button11Click(Sender: TObject);
    procedure Button12Click(Sender: TObject);
    procedure Button13Click(Sender: TObject);
    procedure Button14Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);

private
```

```
  { Private declarations }

public

  { Public declarations }

end;

var
Form_agregar: TForm_agregar;
implementation
USES
Administrador,
Password,
Lugar_De_Nacimiento_Procedencia;

{$R *.dfm}

procedure TForm_agregar.DBEdit8KeyPress(Sender: TObject; var Key: Char);
begin
if (key = #13) then begin
Form_Lugar_Nacim_Proced.Show;
end;

end;

procedure TForm_agregar.DBEdit7KeyPress(Sender: TObject; var Key: Char);
begin
if (key = #13) then begin
DBEDIT8.SetFocus ;
end;

end;

procedure TForm_agregar.Button1Click(Sender: TObject);
begin

// ACTIVA GRUPO DE CONTROLES

GroupBox5.Enabled :=true;

Form2_password.IBDataSet_Modificasiones.Insert;

Form2_password.IBDataSet_Modificasiones.FieldByName('PACHIS').asinteger:=1;

end;

procedure TForm_agregar.Button4Click(Sender: TObject);
begin
Form2_password.IBEvents_Modificasiones.UnRegisterEvents;
form_agregar.CLOSE;
end;
procedure TForm_agregar.FormShow(Sender: TObject);
begin
Form2_password.IBEvents_Modificasiones.Events.Add('NEW_CLIENTE');
Form2_password.IBEvents_Modificasiones.Events.Add('DEL_CLIENTE');
Form2_password.IBEvents_Modificasiones.Events.Add('CAMBIO_CLIENTE');
Form2_password.IBEvents_Modificasiones.RegisterEvents;
end;
procedure TForm_agregar.Button3Click(Sender: TObject);
```

```
begin

// ACTIVA GRUPO DE CONTROLES

GroupBox5.Enabled :=true;

Form2_password.IBDataSet_Modificasiones.Delete;
Form2_password.IBTransaction1.CommitRetaining;
end;
procedure TForm_agregar.Button2Click(Sender: TObject);
begin

// ACTIVA GRUPO DE CONTROLES

GroupBox5.Enabled :=true;
Form2_password.IBDataSet_Modificasiones.Edit;
end;

procedure TForm_agregar.Button5Click(Sender: TObject);

begin
DBedit16.Text := DBedit2.Text +' '+ DBedit3.Text +' '+ DBedit1.Text;
Form2_password.IBDataSet_Modificasiones.Post;
Form2_password.IBTransaction1.CommitRetaining;

//ACTUALIZAR DATOS

Form2_password.IBQuery_Agregar.Active :=false;
Form2_password.IBDataSet_Modificasiones.Active :=false;
Form2_password.IBQuery_Password.Active :=false;
Form2_password.IBQuery_Pacientes.Active:=false;
Form2_password.IBQuery_CodCiudad.Active :=false;
Form2_password.IBQuery_Registrado.Active :=false;
Form2_password.IBTable_Usuarios_Cambiar_Password.Active:=false;
Form2_password.IBQuery_Print_Registro.Active:=false;

    Form2_password.IBTransaction1.Active :=false;
    Form2_password.IBTransaction1.Active :=true;

Form2_password.IBQuery_Print_Registro.Active:=true;
Form2_password.IBTable_Usuarios_Cambiar_Password.Active:=true;
Form2_password.IBQuery_Registrado.Active :=true;
Form2_password.IBQuery_CodCiudad.Active :=true;
Form2_password.IBQuery_Pacientes.Active:=true;
Form2_password.IBQuery_Password.Active :=true;
Form2_password.IBDataSet_Modificasiones.Active :=true;
Form2_password.IBQuery_Agregar.Active :=true;

   DBGrid1.DataSource  := Form2_password.DataSource_Agregar;


// DESACTIVA GRUPO DE CONTROLES

GroupBox5.Enabled :=false;

end;

procedure TForm_agregar.Edit1Change(Sender: TObject);
begin
```

```
Form2_password.IBQuery_Agregar.Close;
Form2_password.IBQuery_Agregar.SQL.Clear;

Form2_password.IBQuery_Agregar.SQL.Text:='select
PACHIS,PACSEX,PACEDA,PACTEL,PACNOM,PACPAT,PACMAT,PACFULL,PACPAD,PACM
AD from pacientes where PACFULL like ' + quotedstr(edit1.text + '%');

Form2_password.IBQuery_Agregar.Open;
end;
procedure TForm_agregar.Button6Click(Sender: TObject);
begin
form2_password.DataSource_Agregar.DataSet.First;
end;
procedure TForm_agregar.Button7Click(Sender: TObject);
begin
form2_password.DataSource_Agregar.DataSet.Prior;
end;
procedure TForm_agregar.Button8Click(Sender: TObject);
begin
form2_password.DataSource_Agregar.DataSet.Next;
end;
procedure TForm_agregar.Button9Click(Sender: TObject);
begin
form2_password.DataSource_Agregar.DataSet.Last;
end;
procedure TForm_agregar.Button10Click(Sender: TObject);
begin
form2_password.DataSource_Modificasiones.DataSet.First;
end;
procedure TForm_agregar.Button11Click(Sender: TObject);
begin
form2_password.DataSource_Modificasiones.DataSet.Prior;
end;
procedure TForm_agregar.Button12Click(Sender: TObject);
begin
form2_password.DataSource_Modificasiones.DataSet.Next;
end;
procedure TForm_agregar.Button13Click(Sender: TObject);
begin
form2_password.DataSource_Modificasiones.DataSet.Last;
end;
procedure TForm_agregar.Button14Click(Sender: TObject);
begin
//ACTUALIZAR DATOS
Form2_password.IBQuery_Agregar.Active :=false;
Form2_password.IBDataSet_Modificasiones.Active :=false;
Form2_password.IBQuery_Password.Active :=false;
Form2_password.IBQuery_Pacientes.Active:=false;
Form2_password.IBQuery_CodCiudad.Active :=false;
Form2_password.IBQuery_Registrado.Active :=false;
Form2_password.IBTable_Usuarios_Cambiar_Password.Active:=false;
Form2_password.IBQuery_Print_Registro.Active:=false;

    Form2_password.IBTransaction1.Active :=false;
    Form2_password.IBTransaction1.Active :=true;

Form2_password.IBQuery_Print_Registro.Active:=true;
Form2_password.IBTable_Usuarios_Cambiar_Password.Active:=true;
Form2_password.IBQuery_Registrado.Active :=true;
Form2_password.IBQuery_CodCiudad.Active :=true;
```

```
Form2_password.IBQuery_Pacientes.Active:=true;
Form2_password.IBQuery_Password.Active :=true;
Form2_password.IBDataSet_Modificasiones.Active :=true;
Form2_password.IBQuery_Agregar.Active :=true;

   DBGrid1.DataSource  := Form2_password.DataSource_Agregar;

end;

procedure TForm_agregar.FormCreate(Sender: TObject);
begin

// GRUPO DE CONTROLES

GroupBox5.Enabled :=false;
end;

end.
```