ESCUELA POLITECNICA DEL EJÉRCITO

SEDE LATACUNGA

FACULTAD DE INGENIERIA DE SISTEMAS E INFORMATICA

PROPUESTA METODOLOGICA PARA GARANYIZAR LA CALIDAD DEL SOFTWARE UTILIZANDO ESTANDARES DE PRUEBAS ISO, IEEE. CASO PRACTICO SISTEMA DE ADMINISTRACION PARA LA GRANJA AVICOLA REGALO DE DIOS

PROYECTO PREVIO A LA OBTENCIÓN DEL TITULO DE INGENIERO EN SISTEMAS E INFORMATICA

MAYRA GABRIELA VILLACÍS MORA

Latacunga-Ecuador

2007

AGRADECIMIENTO

Agradezco a Dios nuestro señor, que llena mi vida de dicha y bendiciones, brindándome la sabiduría para culminar el presente proyecto.

A mis padres de todo corazón por su amor, cariño y comprensión y sacrificio reflejado con dedicación y esfuerzo durante los años de mi vida estudiantil.

A la ESPEL Carrera de Ingeniería en Sistemas e Informática por haberme impartido los conocimientos teóricos y prácticos de la especialidad.

Deseo destacar el apoyo incondicional y oportuno del Ing. Edison Espinosa en calidad de Director y de la Ing. Nancy Jacho como Codirectora quienes me guiaron en el transcurso del presente trabajo.

Mayra

DEDICATORIA

Mi tesis la dedico con todo mi amor y cariño.

A ti Dios que me diste la oportunidad de vivir y regalarme una familia maravillosa.

Con mucho cariño principalmente a mis padres que me dieron la vida y han estado conmigo en todo momento. Gracias por todo Papi y Mami por darme una carrera para mi futuro y por creer en mi aunque hemos pasado momentos difíciles siempre han estado apoyándome y brindándome todo su amor.

A mis hermanos Edwin, Ximena e Ivonne por estar conmigo y apoyarme siempre, los quiero mucho.

A ti pequeño sobrino Mateito que con tu risueña existencia has sabido brindarme tu angelical compañía durante las horas de arduo trabajo

Mayra

CAPITULO I

EVOLUCIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE

I. I.- INTRODUCCIÓN

A partir del siglo XVI es muy importante la Evolución y avance del Proceso de desarrollo de Software que afecta a los métodos de investigación y a los criterios de verificación de las teorías. Surgen así nuevos métodos de investigación más adecuados a los estudios científicos de la época, que constituyen un cambio radical hasta la actualidad, experimentado avances importantes entre los que se cuenta con el desarrollo de nuevas disciplinas como la **Ingeniería del software,** que "se ocupa de la creación de nuevas técnicas, normas, estándares y métodos teniendo similitudes importantes con la aplicación que se ocupa de la creación de productos software".[1]

El Proceso de desarrollo de software ha evolucionado de manera tal que se hace necesario hoy en día, el uso de herramientas y técnicas que se aplican en las ramas de la Ingeniería para lograr productos competitivos y confiables, este conjunto de herramientas y técnicas aplicadas permiten reducir la complejidad de los procesos y el diseño de los mismos.

Debido a este entorno personalizado del Proceso de Desarrollo de Software, el diseño era un proceso implícito, realizado en la mente de alguien y la documentación normalmente no existía, para ello, tuvieron que dedicar grandes esfuerzos a la investigación y aplicaron métodos de ingeniería Informática al análisis, diseño, fabricación y control de calidad del producto, como consecuencia de esto surgieron entonces las primeras Metodologías, Técnicas, Normas y Estándares en el mercado ampliándose considerablemente

[1] PRESSMAN Roger,Ingenieria de Software, 4ta edición, New York,1997

Al no utilizarse metodología alguna en el proceso de desarrollo del software, los programas contenían numerosos errores e inconsistencias, lo que obligaba a una depuración continua, incluso mucho después de haber sido entregados al cliente.

Estas continuas modificaciones no hacían sino aumentar la inconsistencia de los programas, que se alejaban cada vez más de la corrección dando comienzo a la denominada crisis del software. Hoy en día, el proceso de desarrollo de software ha cambiado drásticamente, reafirmando que son necesarias las técnicas y tecnología eficientes de Ingeniería de Software para resolver los múltiples problemas que se derivan de las aplicaciones en donde se desarrollan sistemas de software de gran tamaño.

Este Proyecto tiene como objetivo presentar las técnicas de Ingeniería de Software que pueden ser aplicadas en el proceso de desarrollo de software prácticos, a través de la evolución la Ingeniería de Software que tiene como principal objetivo servir como base para la producción de software de calidad.

1.1.1 INGENIERIA DE SOFTWARE

Desde hace cuatro décadas, la ingeniería del software se ha venido consolidando como una rama importante dentro del campo de la informática en busca de métodos de desarrollo y técnicas que permitan producir software de gran calidad con recursos limitados, es así que podemos tener varias definiciones de Ingeniería de Software..

- La Ingeniería de Software "es la rama de la computación que tiene relación con los métodos, técnicas, metodologías y estándares usados en el desarrollo de software con el fin de lograr productos de calidad, siendo su aplicación un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación, y mantenimiento del software, es decir la aplicación de la ingeniería al Software tiene como finalidad producir software de calidad en un proceso controlado y predecible".[2]

2 PRESSMAN Roger,Ingenieria de Software, 4ta edición, New York,1997

- Es una disciplina que "concierne a todos los aspectos de la producción de software, estableciendo el uso de principios robustos de la ingeniería a fin de obtener económicamente software que sea fiable y que funcione eficientemente sobre máquinas reales".[3]
- Es la ciencia del desarrollo y mantenimiento de sistemas computacionales que se comportan de manera confiable, eficiente capaz de desarrollar y mantener los sistemas.
- Para el desarrollo de este proyecto se determina entonces que la Ingeniería de Software es, la disciplina que permite realizar un análisis completo al desarrollo de software, estructurando e indagando algunos principios generales de producción, para diseñar los métodos, técnicas, herramientas y estándares propios de cada etapa, obteniendo un desarrollo integro y confiable que garantiza la calidad del producto.

1.1.2. CARACTERISTICAS DE LA INGENIERIA DE SOFTWARE

- Resaltan el poder evaluar y aplicar las técnicas de computación existentes, de manera rentable y fácil de usar, para generar y mantener sistemas de software dentro de las restricciones de tiempo, funcionalidad y costos acordados.
- Desarrolla metodologías y herramientas que colaboran a la productividad en el diseño e instrumentación de sistemas computacionales, así como de criterios para la verificación y la evaluación de la calidad del software y la eficiencia del mismo.
- Permite producir "productos de calidad de acuerdo con la planificación establecida, permitiendo en una forma mas organizada, planificar su trabajo, desarrollar el trabajo de acuerdo al plan establecido, esforzarse a través de la aplicación de sus herramientas, técnicas y estándares a producir productos de máxima calidad".[4]

[3] PRESSMAN Roger,Ingenieria de Software, 4ta edición, New York,1996
[4] http://congreso.hispalinux.es/congreso2001/actividades/ponencias/robles.pdf ,2006

- Define la secuencia en que se aplican los métodos, los documentos que se requieren, los controles que permiten asegurar la calidad y las directrices que permiten a los gestores evaluar los procesos.
- Proporciona el fortalecimiento de los procesos a través de técnicas de desarrollo, tecnologías de información, como soporte al desarrollo y arquitectura de software para que puedan construir aplicaciones confiables, mantenibles, eficientes y bajo las restricciones de costos y recursos.
- Permite analizar qué es lo que tenemos que hacer, cómo lo vamos a hacer, cómo se van a coordinar todas las personas que van a intervenir en el proyecto y cómo vamos a controlar el desarrollo del mismo de forma que al final obtengamos los resultados esperados.

1.2 PROCESO SOFTWARE

Proceso "es un conjunto de procedimientos o actividades enlazadas que, en conjunto cumplen los objetivos o políticas de una empresa, normalmente lo hacen dentro del contexto de una estructura organizacional que define roles funcionales".[5]

El proceso software se define como el conjunto de actividades ordenadas, necesarias para producir un software de calidad, ejecutado por un conjunto de recursos humanos organizados según una estructura concreta con un soporte de herramientas y técnicas.

Requiere un conjunto de conceptos, metodologías y lenguaje propio, a este proceso también se le llama ciclo de vida del software que comprende cuatro grandes fases: **concepción, elaboración, construcción y transición**. La concepción define el alcance del proyecto y desarrolla un

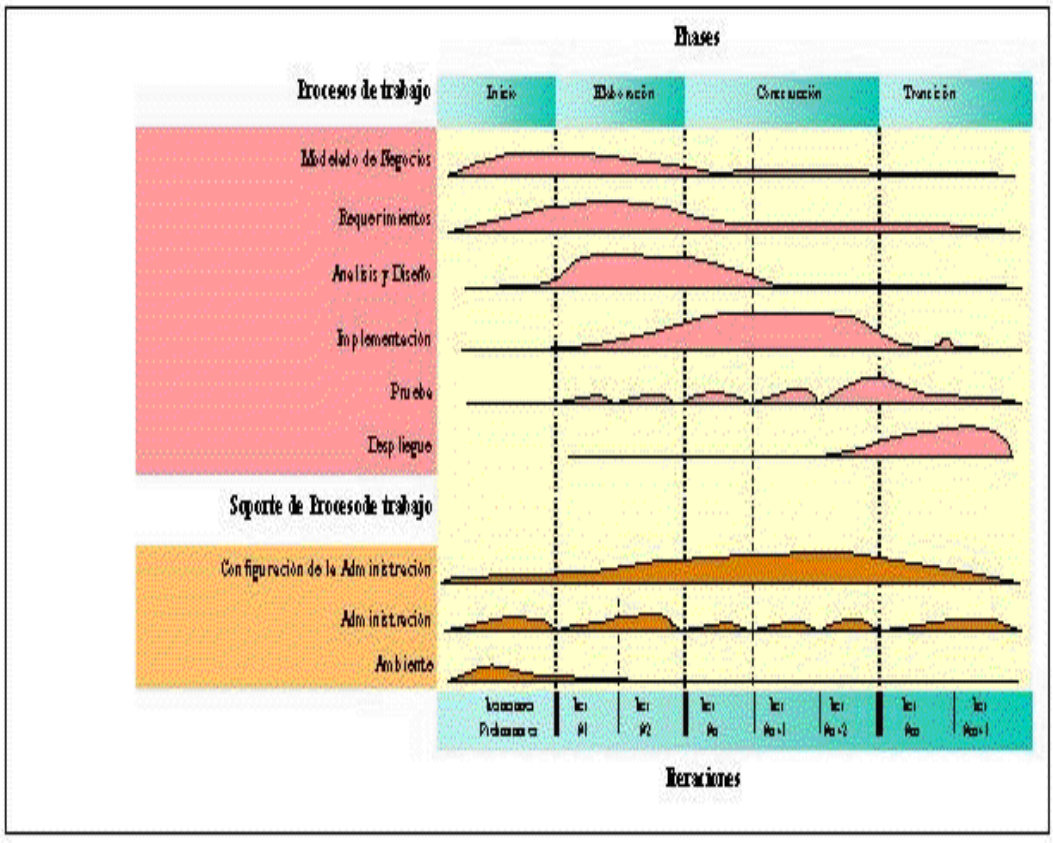
[5] PRESSMAN Roger,Ingenieria de Software, 4ta edición, New York,1997

caso de negocio. La elaboración define un plan del proyecto, especifica las características y fundamenta la arquitectura. La construcción crea el producto y la transición transfiere el producto a los usuarios.

Es un factor crítico en la entrega de software de calidad, ya que tiene por finalidad gestionar y transformar las necesidades del usuario, sometiéndole a una evaluación que permite así determinar los puntos fuertes y los puntos débiles del proceso de software, con el fin de mejorar la calidad de dicho proceso y en último término, la calidad del producto software desarrollado, a partir de las principales iniciativas de evaluación para entregar un producto integro y fiable.

.

FASES DEL PROCESO DE DESARROLLO DE SOFTWARE



1.2.1 CARACTERISTICAS DEL SOFTWARE

El software "es un término amplio, que incluye los elementos para identificación y análisis de un problema a ser resueltos, es un componente esencial de toda actividad basado en el uso de la informática, teniendo en cuenta, que la informática, es en la actualidad de interés colectivo, razón por la cual las normas de protección deben ser claras y específicas".[6]

Se centran en las necesidades básicas e identificadas del análisis y diseño de un producto para cubrir las necesidades, brindando confianza en el producto final.

Cuya finalidad es servir al desarrollo, funcionamiento de software permitiendo un mayor esfuerzo en su diseño y optimización, pero también les obliga a ser muy fiables, cumpliendo estrictamente las características, especificaciones para las que fueron creados.

El Software posee características las mismas que permiten identificar un Software de alta calidad y son las siguientes:

- ***Utilidad*** .- Determina si el sistema hace lo que sus usuarios esperan que haga para ser entendido, permite al usuario entender si el software es adecuado y cómo puede ser operado, controlado para adherirse a normas, convenciones, guías de estilo o regulaciones relacionadas con la usabilidad**.**
- ***Confiabilidad***.- Capacidad del producto software para proporcionar tiempos de respuesta íntegros, fiables, bajo condiciones determinadas proporcionando resultados eficientes que garantizan calidad**.**
- ***Mantenibilidad*****.** - Es la capacidad de analizar, diagnosticar deficiencias o causas de los fallos en el software, que han de ser modificadas, permitiendo que una determinada modificación sea implementada para evitar efectos inesperados.

[6] PRESSMAN Roger,Ingenieria de Software, 4ta edición, New York,1997

- ***Eficiencia*** .- Medida en relación a los recursos del computador bajo el tiempo de ejecución, usando recursos adecuados a la velocidad de proceso, dando lugar a la satisfacción por parte del usuario
- ***Facilidad de prueba*** .-Constituye una característica importante dentro de los proyectos de desarrollo de software, hace referencia a la facilidad que ofrece un producto para realizar inspecciones técnicas formales en todos los pasos del proceso de desarrollo del software
- ***Rápida implantación***.- Programación orientada a objetos, con una interfase muy intuitivo de fácil manejo y de muy corto tiempo de aprendizaje, especialmente diseñado para que no sólo pueda manejarse con el ratón, si no que también resulte cómodo mediante el teclado.
- ***Multiusuario***.- Posibilidad de trabajar en red sin límite de usuarios. permite a mas de un solo usuario acceder al sistema. Los usuarios acceden al sistema mediante su nombre (login o cuenta) y como medida de seguridad deben introducir su contraseña de acceso**.**
- ***Control de Accesos***.- Completísima gestión de accesos por usuario independientemente del sistema operativo de la red. Permite definir a que partes del programa pueden acceder los usuarios y con que derechos de trabajo.
- ***Exportación de Datos***.- Toda la información puede ser impresa o exportada hacia otras aplicaciones de bases de datos, procesadores de texto, lo que permite el intercambio de informaciones con otros software.
- ***Modular*****.**- Compuesto por diferentes módulos perfectamente integrados entre sí, permitiendo que los datos introducidos en un módulo sean válidos para toda la aplicación sin necesidad de introducirlos de nuevo.
- ***Ayuda en línea***.- Completo sistema de ayuda al usuario desde cualquier punto del programa donde experimente dificultades o dudas, mediante un simple click del ratón sobre el botón Ayuda.

La importancia relativa de las características depende en el tipo de producto y en el ambiente en el que será utilizado, con la capacidad que se desarrollan Software para la industria, es recomendable generar productos que cumplan cabalmente con las características de software de calidad y con las expectativas del cliente.

1.2.2 PRUEBAS

Las Pruebas " se definen como una herramienta que se utiliza para asegurar el correcto desarrollo y funcionamiento del producto Software, previniendo los posibles fallos que se puedan generar en el proceso de desarrollo".[7]

Está ampliamente demostrado que una temprana inclusión de las actividades de prueba en el proceso de desarrollo de software, detecta, previene y permite solucionar los errores de una forma rápida, eficaz y con el menor coste posible, abarcando las pruebas de verificación, revisiones y auditoria e incluye las pruebas de validación que se realizan durante el ciclo de vida del software para asegurarse de que el producto sea integro y fiable.

En la actualidad, son muchas las pruebas que existen en el desarrollo de software, introduciendo y popularizado una serie de estándares para medir y certificar la calidad, tanto del sistema a desarrollar, como del proceso de desarrollo en sí.

Un número creciente de herramientas automatizadas como son las Pruebas han surgido para ayudar a definir y aplicar un proceso de desarrollo de software efectivo. Cumple un papel primordial en el proceso de producción de software, ya que enfoca una área fundamental y principal que consiste en la generación de especificaciones correctas que describen con claridad, y en forma consistente, Compacta, el comportamiento del software; de esta manera, se pretende minimizar los errores relacionados al desarrollo de software.

Estudios realizados muestran que más del 53% de los proyectos de software fracasan por no realizar un estudio previo de pruebas en cada una de las fases del proceso de desarrollo de software. Otros factores como falta de participación

[7] http://sunsite.uakom.sk/sunworldonline/swol-08-1996

del usuario, requerimientos incompletos y el cambio a los requerimientos, también ocupan sitiales altos en los motivos de fracasos.

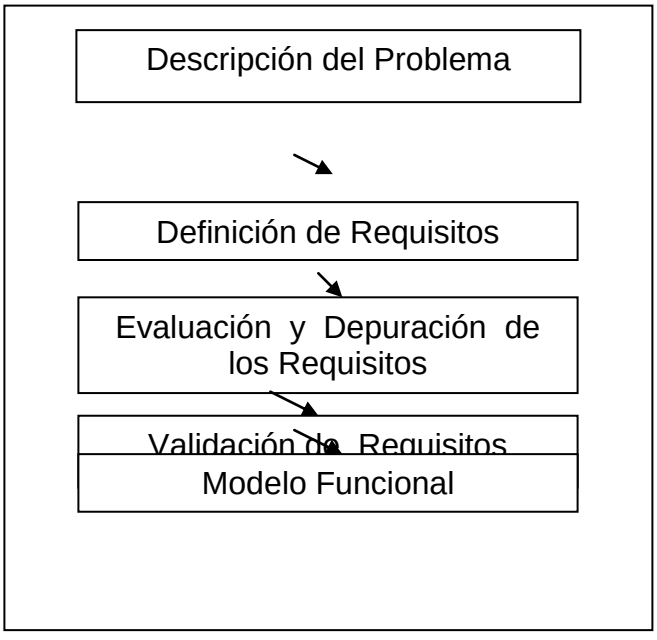
Las tareas que abarca son de verificación, revisión y auditoria e incluye validación que se realizan durante el ciclo de vida del software para asegurar la satisfacción con los requisitos.

Con el trabajo de aplicar pruebas se pretende alcanzar las siguientes ventajas:

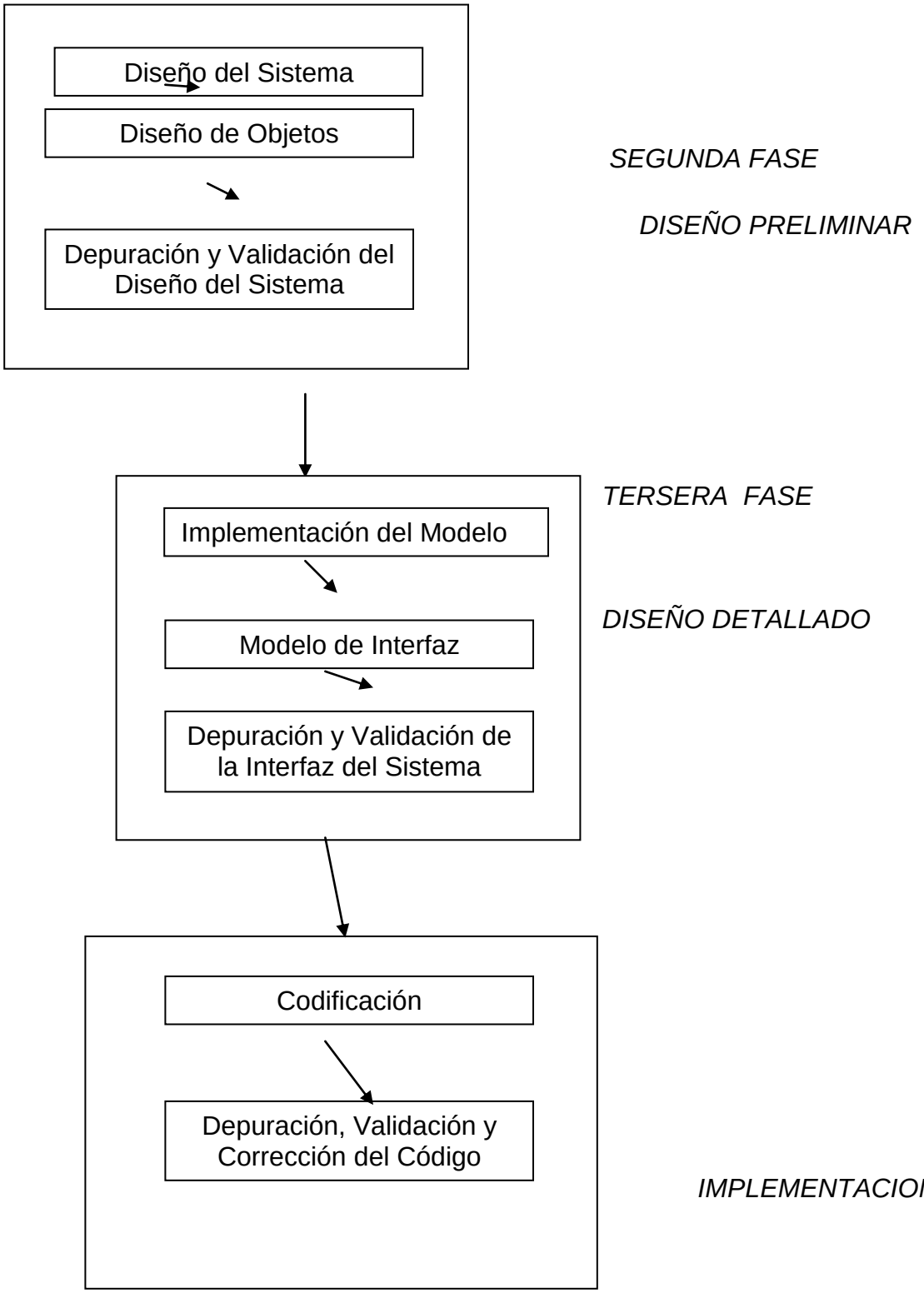
- Ψ Resaltar la importancia que tiene la Ingeniería de Software dentro del ciclo de desarrollo.
- Ψ Dar a conocer las diferentes alternativas que existen para identificar requerimientos.
- Ψ Ayudar a comprender la diferencia que existe entre las diferentes técnicas utilizadas.
- Ψ Minimizar las dudas que se tiene sobre los casos de uso. y utilizar herramientas dentro de la administración de requisitos.

Las Pruebas para el desarrollo de nuestro proyecto se las define como la herramienta que permite planificar, analizar, ejecutar las tareas de verificación y validación, en las etapas del proceso de desarrollo software,..orientadas a la detección de defectos en el desarrollo del producto.

DIAGRAMA ESTRUCTURAL DE LA APLICACIÓN DE PRUEBAS EN CADA FASE DEL PROCESO DE DESARROLLO DE SOFTWARE



ANALISIS PRELIMINAR


Diseño del Sistema
Diseño de Objetos
Depuración y Validación del Diseño del Sistema
SEGUNDA FASE
DISEÑO PRELIMINAR
Implementación del Modelo
Modelo de Interfaz
Depuración y Validación de la Interfaz del Sistema
TERSERA FASE
DISEÑO DETALLADO
Codificación
Depuración, Validación y Corrección del Código
IMPLEMENTACION

1.3 ESTANDAR DE PRUEBAS

Proporciona a los desarrolladores y creadores de software, un conjunto de procedimientos y técnicas para el buen desarrollo, implementación o incluso mantenimiento de software, es así como podemos obtener diferentes definiciones de Estándares de Pruebas.

- "Adoptan metodologías basadas en los conceptos, definiciones, clasificaciones y procedimientos de calidad, a través de estrategias que consisten en utilizar técnicas de modelado de procesos para capturar, evaluar y rediseñar el proceso de desarrollo de software".[8]
- Determinan y Analizan el estado actual del proceso de desarrollo de software desde la perspectiva de cada etapa y propone mejoras a las tareas relacionadas con las fases de Concepción, Elaboración, Construcción y Transición.
- Son las directrices basadas en los resultados consolidados de la ciencia, la tecnología y la experiencia, orientadas a la promoción del óptimo beneficio en el proceso de desarrollo de software, el propósito de los estándares de pruebas es el de alcanzar el grado óptimo de orden en un contexto dado .
- Para el desarrollo de este proyecto se determina que los Estándares de Pruebas permiten mejorar la calidad de los productos desarrollados y del proceso de desarrollo, incrementando la productividad, dentro de las restricciones de tiempo, funcionalidad y costos acordados en el desarrollo de Software**.**

1.3.1 IMPORTANCIA DE LOS ESTANDARES DE PRUEBAS

Si bien las metodologías y estándares de Pruebas tienen su importancia fundamental ya que están relacionados con el desarrollo y construcción de software, buscando mantener altos niveles de confiabilidad y control de la

[8] http://www.shu.ac.uk/tfre/web.links.html ,2004

solución informática, la seguridad informática y sus principios de diseño seguro. La aplicación de los estándares de Pruebas buscan elementos comunes que permitan orientar a los Ingenieros en Sistemas, Analistas y programadores en estrategias y principios que disminuyan potenciales problemas de calidad y seguridad asociados con el desarrollo del software.

La industria del software en su creciente y decidida penetración, se ha visto en la necesidad de generar mejores soluciones sistematizadas a través de los estándares de Pruebas, que aumentan la efectividad de los procesos, y obtienen un producto software de óptima calidad que permita a la organización adelantar con oportunidad y alto contenido estratégico sus directrices de negocio.

Se ha convertido en la única alternativa posible para abordar con éxito los retos que plantea el nuevo mercado global de la informática, en cuanto consideramos la creciente presión en todas las empresas de desarrollo de software para producir aplicaciones cada vez más complejas, de mayor calidad, y en periodos de tiempo más corto.

Permiten la implantación de procesos bien definidos, así como métodos cuantitativos y controles de calidad, que permitan evaluar la calidad y madurez de sus procesos y productos apoyándose en componentes software ya desarrollados, que son combinados adecuadamente para satisfacer los requisitos del sistema, ofreciendo soluciones robustas para muchos de los problemas que se plantean en la construcción de grandes sistemas de software.

1.3.2 CARACTERISTICAS DE LOS ESTANDARES DE PRUEBAS APLICADOS EN EL DESARROLLO DE SOFTWARE

Las Características de los Estándares de Pruebas en el desarrollo de software a lo largo de los años ha adquirido un grado de profesionalismo digno y característico de la ingeniería. Gran parte de este profesionalismo se debe en gran medida a la madurez adquirida por la ingeniería de software y la creciente complejidad de los sistemas de software que se vieron involucrados como elementos activos en tareas críticas de la industria. Pero paralelamente al crecimiento en la complejidad del software, dio origen a un conjunto de medidas y estándares internacionales

que trabajando desde las bases del proceso de desarrollo del software, tienen como objetivo definir y asegurar un conjunto de estándares conocidos que se focalizan en el elevar la madurez del proceso de desarrollo de software para así asegurar su calidad y mejorar la calidad del producto obtenido.

1.3.2.1 IEEE

Los estándares de Pruebas IEEE "especifican el formato y contenido de los planes para la gestión de desarrollo de Software, basado en un conjunto de prácticas y procedimientos para proporcionar una guía detallada para la preparación y actualización de los planes de gestión de los proyectos software".[9]

Estas prácticas detalladas y procedimientos deberían tener en cuenta los factores del entorno, organizacionales y políticos que pueden influenciar en la aplicación de este estándar. Este estándar de Pruebas IEEE prescribe un Plan para la Gestión de desarrollo de Software, es el documento de control para gestionar un proyecto de desarrollo de software desde el inicio hasta obtener el producto. Este estándar está limitado por el tamaño, o complejidad del producto software, es aplicable todas las clases de software, incluyendo a cualquiera de los segmentos del ciclo de vida de un producto software.

Los estándares de Pruebas IEEE se caracterizan por ser "el conjunto mínimo de elementos integrales que describe un proceso iterativo para la gestión y ejecución de las actividades de Mantenimiento del Software, estableciendo normas y herramientas, técnicas aplicables en el proceso practico de desarrollo de software".[10]

Los criterios establecidos se aplican tanto a la planificación del Mantenimiento del Software mientras este en desarrollo, como a la planificación y ejecución de las actividades de desarrollo para productos software. De esta forma se describe el alcance del estándar de pruebas IEEE aunque se basa específicamente en las

[9] http://sunset.usc.edu/publications/TECHRPTS/2000/usccse2000-501/usccse2000-501.pdf ,2004

[10] http://ltsc.**ieee**.org/ 2004
http://www.internettime.com 2002

fases de desarrollo y de producción de un producto software, las fases que se describen cubren todo el ciclo de vida de un software, cualquiera que sea su tamaño o complejidad. Las fases del ciclo de vida mediante las que se dirige el estándar IEEE son:

- Identificación del Problema
- Análisis
- Diseño
- Implementación
- Pruebas del Sistema
- Pruebas de Aceptación
- Implementación

La evolución de los Estándares de Pruebas IEEE es quizás el ejemplo mas claro en la Gestión de desarrollo de Software y posteriormente en el mantenimiento y las modificaciones que se realizan, de este modo, un equipo de desarrollo puede trabajar en paralelo, compartiendo las normas periódicamente según se van creando o modificando las necesidades que conforman el proyecto.
Las especificaciones de requisitos, los documentos de análisis y de diseño, el Código fuente y ejecutable, los procedimientos y datos de prueba pueden ser sometidos a control de Configuración. Con el control riguroso de los estándares de Pruebas IEEE, es posible entonces mantener el registro del estado de todos estos elementos, lo que facilita la introducción de cambios si se tiene registro de las dependencias entre ellos.
Un cambio implica generalmente la actualización tanto del Código fuente, como de los documentos de especificación de requisitos, análisis y diseño, casos de prueba y manuales. Por lo tanto, en el escenario de la aplicación de los estándares de pruebas IEEE, resulta de utilidad mantener un registro de las dependencias entre los elementos de Configuración.

1.3.2.2 ISO

Los estándares de pruebas internacionales ISO con la finalidad de implementar un sistema de gestión y aseguramiento, constituyen un instrumento importante en la Ingeniería de Software en el desarrollo de sus procesos. Cabe resaltar que los Estándares de Pruebas ISO " se caracterizan por garantizar la calidad, reducción de costos, mayor rentabilidad, mejoras en la productividad, motivación y con un mejor posicionamiento en el mercado, es decir constituye una importante herramienta para el desarrollo de software".[11]

Los beneficios que consiguen al implementar las normas ISO son considerables, pues permiten obtener una mayor satisfacción a los Ingenieros, Analistas y Usuarios por la confianza en el desarrollo de Software brindando fundamentalmente la reducción de costos y obteniendo un alto grado de eficiencia en el proceso de desarrollo de software. Los Estándares de Pruebas ISO aportan grandes beneficios en el desarrollo de software de calidad a las empresas desarrolladoras del producto software, "es más una herramienta necesaria que otorga ventajas competitivas, siendo el propósito inicial de los Estándares ISO",[12] por lo que el interés esta centrado más que en el mejoramiento, en la Implementación de un sistema de calidad basados en un proceso de facilitación y de concientización hacia el cambio en cada una de las etapas del proceso de desarrollo, el cual aportará elementos y reglas que creen un ambiente favorable para el nuevo software a desarrollarse.

Los Estándares de Pruebas ISO se idearon originalmente para el mejoramiento de la calidad en busca de ofrecer mejores productos y servicios a sus clientes, requiriendo de un ambiente propicio para su aplicación permitiendo obtener los siguientes beneficios:

- Orientación hacia lo estratégico y mayor delegación de lo operativo.
- Innovación permanente en métodos, y en cada una de las etapas del proceso de desarrollo de Software.

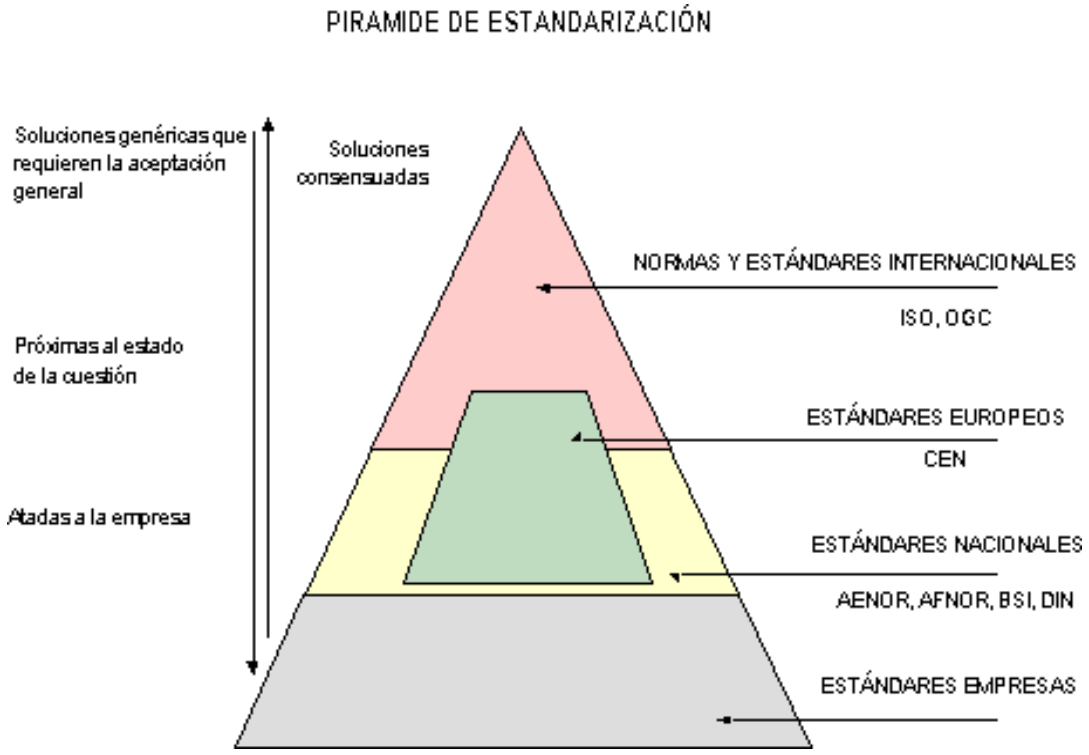
---

[11] http://www.digitalearth.gov/ 2006

[12] . http://geogratis.ISO. cgdi.gc.ca/ 2006

- Menos costos
- Menos tiempo para decidir
- Más utilización de la tecnología .

En actualidad "a nivel mundial los Estándares de Pruebas ISO son requeridos, debido a que garantizan la calidad de un producto mediante la implementación de controles exhaustivos, asegurándose de que todos los procesos que han intervenido en su fabricación operan dentro de las características previstas. La estandarización es el punto de partida en la estrategia de la calidad".[13]



Estos Estándares fueron escritos con el espíritu de que la calidad de un producto no nace de controles eficientes, si no de un proceso productivo y de soportes que

[13] . PIATTINI Mario.G,Calidad en el desarrollo y mantenimiento del Software,1era edición,Mexico,2002 http://clearinghouse.cnr.gob.sv/metadatos/princ.htm ,2006

operan adecuadamente, su implementación asegurándole al Ingeniero de Software que la calidad del producto que él esta desarrollando se mantendrá en el tiempo

1.3.2.3 ANSI

Los Estándares de Pruebas ANSI promueven y facilitan las normas que definen los requisitos, es lógico que se preocupe y se sienta involucrado en las actividades relacionadas para gobernar, desarrollar y llevar a cabo la política sobre la evaluación del proceso de desarrollo de software.

El desarrollo de software basado en Estándares de Pruebas ANSI se ha convertido actualmente en uno de los mecanismos más efectivos para la construcción de grandes sistemas y aplicaciones de software centrándose en los aspectos extra-funcionales y de calidad, como un paso hacia una verdadera ingeniería. La creciente necesidad de realizar sistemas complejos en cortos periodos de tiempo, a la vez que con menores esfuerzos tanto humanos como económicos, está favoreciendo el avance de lo que se conoce como Desarrollo de Software basado en Estándares de Pruebas ANSI.

Una de las principales características del desarrollo de software basado en Estándares de Pruebas ANSI es que "son aplicados, diseñados y desarrollados con el objetivo de poder ser reutilizados, reduciendo el tiempo de desarrollo, mejorando la fiabilidad del producto final y siendo más competitivos en costes, definiéndolos como una clase especial de herramientas técnicas para el proceso de desarrollo de software".[14]

Dentro de la Ingeniería del Software los Estándares de Pruebas ANSI adoptan la definición de calidad, que es un objetivo importante para cualquier producto, por tanto, el principal objetivo de los Estándares ANSI para el desarrollo de un

[14] http://www.google.com/search?q=cache:CTycFYk8c5UJ:adala.org/encuentros/jasl2/ponencias/008.pdf+estandar+ANSI&hl=es&ie=UTF-8, 2006

producto software es satisfacer una necesidad o varias necesidades de un usuario, señalando que esto implica que la calidad de un producto software no se puede referir únicamente a obtener un producto sin errores.

Los principales esfuerzos de la comunidad de software en estos temas se han basado hasta ahora en los aspectos funcionales de los Estándares ANSI, es decir, en la funcionalidad que ofrecen. Sin embargo, por lo general se han venido obviando muchos de los aspectos de calidad que intervienen a la hora de seleccionar estándares, normas, componentes y ensamblarlos para construir aplicaciones que satisfagan los requisitos del cliente. Este tipo de aspectos, que llamaremos “extra-funcionales”, cada vez acapara más la atención de los analistas e ingenieros de software.

El propósito de los estándares de Pruebas ANSI es hacer las cosas más fáciles, definiendo características del sistema que se utilizan cotidianamente permitiendo desarrollar un software más fácil y seguro, estableciendo requisitos mínimos de fabricación, eliminando inconsistencias y variaciones innecesarias en las interfaces. Gracias a la aplicación de los Estándares ANSI se obtiene beneficios que permite a los diseñadores discutir los mismos conceptos y hacer valoraciones comparativas, en cuanto al mantenimiento y la evolución tienen la misma estructura y el mismo estilo lo que hace que todos los sistemas sean fáciles de reconocer y garantizan la calidad.

Los Estándares de Pruebas ANSI "provee los servicios de análisis, concretamente en las áreas de desarrollo del producto software, reconociendo la competencia de ciertas entidades para llevar a cabo, las normas de los estándares de pruebas ANSI, son acordes con cada una de las etapas de desarrollo de software a través de análisis, pruebas de revisión minuciosa".[15]

[15] http://www.lcc.uma.es/~av/Publicaciones/02/CalidadDSBC.pdf, 2005

Los Estándares de Pruebas ANSI proporcionan un proceso que da respuestas a los problemas de Análisis, Diseño, Codificación e Implementación, a la vez que protege los intereses de cada una de las etapas de desarrollo de Software, en esencia, los Estándares de Pruebas ANSI aceleran la aceptación del desarrollo del producto software garantizando la calidad y mejorando la seguridad del Sistema a Implementarse en la Empresa.

1.4 CALIDAD

La Calidad se define como el conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia, "es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad de un producto, proceso o servicio que le confieren su aptitud para satisfacer las necesidades establecidas o implícitas".[16]

El modelo de Calidad postula que la calidad es responsabilidad de todos, en especial de los niveles de dirección, ya que debe ser construida en todas las fases y procesos, empezando por el análisis, diseño; la empresa debe garantizar la calidad en las fases de uso, consumo o posesión del producto o servicio, generándose en todas las áreas del desarrollo de un producto software.

La calidad es aplicable, medible y varía de un sistema a otro o de un programa a otro, puede medirse después de elaborado el producto, esto puede resultar muy costoso si se detectan problemas, por lo que es imprescindible tener en cuenta tanto la obtención de la calidad como su control durante todas las etapas del ciclo de vida del software.

La calidad especialmente en el desarrollo de software crece de forma continua, a medida que los clientes se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que realmente no dan respuesta a sus necesidades.

---

[16] PIATTINI Mario.G,Calidad en el desarrollo y mantenimiento del Software,1era edición,Mexico,2002

Es un tema que está cobrando una importancia cada vez mayor en el desarrollo de software, destacándose como atributo fundamental para el éxito de un producto software.

Dentro de la Ingeniería de software, la calidad se obtiene mejorando día a día el proceso de producción, mantenimiento y gestión del software, para optimizar la calidad de los productos o servicios es preciso conocer al cliente y sus necesidades, conocer a la competencia y poseer un modelo de calidad, esto permitirá incrementar la fiabilidad, reducir el mantenimiento, aumentar la satisfacción del cliente, mejorar la dirección del proyecto, detectar errores rápido e incrementar el beneficio

1.4.1 METODOLOGIA

La Metodología dentro de la Ingeniería del Software desempeña un rol fundamental, ya que enfocan una área substancial para lo que se desea producir, la metodología es la piedra angular para la implementación de software en una empresa y tiene como objetivo relacionar unidades, técnicas y herramientas entre sí y proveerles de un instrumento para analizar, desarrollar y diseñar sistemas, así como aumentar la productividad informática en general, obteniendo la información requerida para que su aplicación se ha fácil y rentable.

La metodología "es una rama de la Filosofía que trata pues, de los métodos y técnicas de la adquisición de conocimientos, investigación, elaboración de resultados, y busca depurar los conocimientos “válidos”. [17]

A la Metodología se la define como la herramienta que " tienen un único fin producir software de gran calidad, desarrollando el análisis y seguimiento de cada uno de los elementos que constituyen un proyecto, la gestión de los mismos se realiza manteniendo un exhaustivo control documental que permite validar constantemente el cumplimiento de los requerimientos presentados y solicitados por el cliente, guiando los procesos de filosofía, ciencias, métodos prácticos, así como de los procesos de desarrollo y configuración de Software".[18]

---

[17] PRESSMAN Roger,Ingenieria de Software, 4ta edición, New York,1997

[18] http://www.fi.uba.ar/materias/7500/schenone-tesisdegradoingenieriainformatica.pdf,2006

Para aplicar el sistema de calidad al ciclo de vida es necesario la utilización de Metodologías adecuadas que permitan medir la calidad del proyecto, en realidad, comparamos los parámetros de calidad de éste con estimaciones realizadas mediante el uso de estándares o datos que aporta la experiencia.

Las metodologías se utilizan para evaluar y controlar el proceso de desarrollo del software, de forma que permitan:

- Indicar la calidad del producto.
- Evaluar la productividad de los desarrolladores.
- Evaluar los beneficios (en cuanto a calidad y productividad)
- Derivados del uso de nuevos métodos y herramientas de ingeniería del software.
- Establecer una línea base para la estimación.
- Justificar el uso de nuevas herramientas o de formación adicional.

La única forma de hacer bien las cosas es actuando oportunamente en cada etapa del proyecto para superar las expectativas y asegurar la calidad del ciclo completo de desarrollo de un producto software.

1.4.2 FACTORES DE CALIDAD DEL SOFTWARE

La implantación de los Factores de Calidad del Software, en una organización ayuda a mejorar la gestión del desarrollo de software, y esto traerá como consecuencia una disminución de los problemas y errores, favoreciendo las relaciones y comunicación entre las personas y grupos de organización, y de éstos con los clientes.

Los Factores de Calidad del Software, aplicados a lo largo de todo el proceso de ingeniería del software, engloban a los métodos y herramientas de análisis, diseño, codificación y pruebas, al control de la documentación y de los cambios, a los procedimientos para asegurar el ajuste a los estándares, y a los mecanismos de medida. Hoy en día nos encontramos con que muchas de nuestras actividades

dependen actualmente del estudio del futuro Ingeniero, ya que se preocupa en incrementar la calidad del software en la que se compromete a llevar acabo dentro de un grupo de trabajo, pues asumiendo este rol, la ingeniería del software es la producción de software de calidad, todos deseamos que nuestros sistemas de software sean rápidos, fiables, fáciles de usar, legibles, modulares y estructurados .Por una parte, se consideran cualidades tales como la velocidad o la facilidad de uso, cuya presencia o ausencia en un producto de software puede ser detectada por sus usuarios permitiendo distinguirse dos tipos de factores de calidad:

Los factores de calidad externos son aquellos que son perceptibles por los usuarios

- Corrección
- Robustez
- Extensibilidad
- Reutilización
- Compatibilidad
- Eficiencia
- Portabilidad
- Facilidad de uso
- Funcionalidad

Los factores de calidad internos son los perceptibles por los profesionales en computación

- Modularidad
- Legibilidad

Corrección

La corrección es la cualidad principal, más fácil de decir que de lograr, posee la capacidad de realizar con exactitud sus tareas, tal y como se definen en la especificación.

Incluso el primer paso hacia la corrección es difícil: debemos ser capaces de especificar los requisitos del sistema de una forma precisa, lo que es en sí una ardua tarea. Los métodos que aseguran la corrección son usualmente condicionales. Un sistema de software importante, incluso uno pequeño según los estándares de hoy, implica a tantas áreas que sería imposible garantizar su corrección manejando todos los componentes y propiedades en un solo nivel.

En la solución condicional de la corrección, sólo hay que preocuparse en garantizar que cada nivel sea correcto bajo el supuesto de que los niveles inferiores son correctos.

Robustez

Es la capacidad del software de reaccionar apropiadamente ante condiciones excepcionales.

La robustez complementa la corrección, tiene que ver con el comportamiento de un sistema en los casos previstos por su especificación; la robustez caracteriza lo que sucede fuera de tal especificación.

La robustez es por naturaleza una noción más difusa que la corrección tiene que ver aquí con casos no previstos por la especificación, no es posible decir, como con la corrección, el sistema debería “realizar sus tareas” en tal caso; donde las tareas son conocidas, el caso excepcional formaría parte de la especificación y regresaríamos al terreno de la corrección produciendo mensajes de error apropiados para terminar su ejecución limpiamente en lo posible.

Extensibilidad

Es la facilidad de adaptar el producto software a los cambios de especificación. El problema de extensibilidad es un problema de escala para programas pequeños, realizar cambios no es normalmente una tarea difícil; pero a medida que el software crece comienza a ser cada vez más difícil de adaptar.

El cambio es omnipresente en el desarrollo del software: cambios en los requisitos, algoritmos, representación de los datos, de las técnicas de implementación, aunque muchas de las técnicas que mejoran la extensibilidad se pueden aplicar con pequeños ejemplos, su relevancia sólo se ve con claridad en

los grandes proyectos. Hay dos principios esenciales para mejorar la extensibilidad:

- Simplicidad del diseño: una arquitectura simple siempre será más fácil de adaptar a los cambios que una compleja.
- Descentralización: cuanto más autónomos sean los módulos, más alta es la probabilidad de que un cambio afecte a un solo módulo, o a un número pequeño de módulos, en lugar de provocar una reacción en cadena de cambios en el sistema completo.

Reutilización

Es la capacidad de los elementos de software de servir para la construcción de aplicaciones diferentes.

La necesidad de la reutilización surge de la observación de que los sistemas software a menudo debería ser posible explotar y evitar reinventar soluciones a problemas que ya han sido encontradas con anterioridad, capturando un elemento de software reutilizable para aplicar en muchos desarrollos diferentes.

La reutilización tiene una influencia sobre todos los demás aspectos de la calidad del software, ya que al resolver el problema de la reutilización se tendrá que escribir menos software y en consecuencia se podrán dedicar entonces mayores esfuerzos a mejorar los otros factores, tales como la corrección y la robustez.

Compatibilidad

La compatibilidad es importante debido a que los sistemas software no se desarrollan en el vacío, necesitan interactuar con otros. Pero con mucha frecuencia los sistemas tienen dificultades para interactuar porque hacen suposiciones contradictorias sobre el resto del mundo. Un ejemplo es la amplia variedad de formatos de archivos soportados por muchos sistemas operativos. Un programa puede usar directamente como entrada los resultados de otro sólo si los formatos de archivos son compatibles la clave de la compatibilidad recae en la homogeneidad del diseño y en acordar convenciones estándares para la comunicación entre programas. Los enfoques incluyen:

- Formatos de archivos estándares, como en el sistema Unix, donde cualquier archivo de texto es simplemente una secuencia de caracteres.
- Estructuras de datos estándares como en los sistemas, donde tanto los datos como los programas, se representan mediante árboles binarios.
- Interfaces de usuario estándares, como en las diferentes versiones de Windows donde todas las herramientas utilizan un solo paradigma para la comunicación con el usuario, basado en componentes estándares tales como ventanas, íconos, menús, etc.

Eficiencia

Es la capacidad de un sistema software para exigir la menor cantidad posible de recursos hardware, tales como tiempo del procesador, espacio ocupado de memoria interna y externa o ancho de banda utilizado en los dispositivos de comunicación.

Casi sinónimo de eficiencia es la palabra “rendimiento”. La comunidad del software muestra dos tipos de actitud con relación a la eficiencia:

- Algunos desarrolladores tienen una obsesión con las cuestiones de rendimiento y le dedican gran cantidad de esfuerzos a presuntas optimizaciones.
- Por otro lado, existe la tendencia de las cuestiones de eficiencia, como se evidencia en las frases de la industria “hágalo correcto antes de hacerlo rápido” y “de todos modos los modelos de computadoras del año que viene van a ser un 50% más rápidos”.

De manera más general, la preocupación por la eficiencia tiene relacion con objetivos tales como la extensibilidad y la reutilización; optimizaciones extremas que pueden hacer al software tan especializado que limite el cambio y la reutilización.

Portabilidad

Es la facilidad de transferir los productos software a diferentes entornos hardware y software.

La portabilidad tiene que ver con, la Facilidad de transferir productos a diferentes plataformas que realmente programamos y que incluye el sistema operativo, el sistema de ventanas y otras herramientas fundamentales. Muchas de las incompatibilidades existentes entre las plataformas son injustificadas, y convierte a la portabilidad en un asunto primordial tanto para los que desarrollan como para los que usan el software.

Facilidad de uso

Es la facilidad con la cual personas con diferentes formaciones y aptitudes pueden aprender a usar los productos software y aplicarlos a la resolución de problemas cubriendo la facilidad de instalación, de operación y de supervisión.

La definición insiste en los diferentes niveles de experiencia de los posibles usuarios. Este requisito plantea uno de los mayores retos de los diseñadores de software preocupados por la facilidad de uso: cómo proporcionar explicaciones y guías detalladas a los usuarios novatos sin fastidiar a los usuarios expertos que quieren ir directo al grano.

Una de las claves de la facilidad de uso es la simplicidad estructural, un sistema bien diseñado, construido de acuerdo a una estructura clara y bien pensada, tiende a ser más fácil de aprender y usar que uno confuso.

Funcionalidad

Funcionalidad es la extensión o cantidad de posibilidades proveídas por un sistema, impidiendo la pérdida de consistencia por el aumento inmoderado de funciones y propiedades, encargadas de trabajar una y otra vez en la consistencia del sistema en general, tratando de encajar cada cosa en un molde global. Un buen sistema de software esta basado en un conjunto reducido de ideas, que deben explicarse como consecuencia de los conceptos básicos de Ingeniería de Software.

Oportunidad

Es la capacidad de un sistema de ser lanzado cuando los usuarios lo requieren o antes. Un gran producto software que aparece demasiado tarde puede no

alcanzar su objetivo. Esto es cierto en otras industrias también, pero pocas evolucionan tan rápidamente como el software.

Otros factores

Además de las cualidades analizadas, existen otras que afectan a los usuarios de sistemas software y a la gente que compra estos sistemas o encarga su desarrollo en particular:

- **Verificabilidad:** es la facilidad para preparar procedimientos de aceptación, especialmente datos de prueba y procedimientos para detectar fallos y localizar errores durante las fases de validación y operación.
- **Integridad:** es la capacidad de los sistemas software de proteger sus diversos componentes (programas, datos, etc.) contra modificaciones y accesos no autorizados.
- **Reparabilidad**: es la capacidad para facilitar la reparación de los defectos.
- **Economía:** junto con la oportunidad, es la capacidad que un sistema tiene de completarse con el presupuesto asignado o por debajo del mismo.

La gestión de los Factores de Calidad de Software va tomando cada día mayor importancia en el ámbito del desarrollo de software, de esta forma los esfuerzos encaminados a integrar la gestión de la calidad dentro de la gestión de los proyectos deben provocar también un aumento de la productividad.

Este esquema permitiría la evaluación de la calidad del proyecto sin que se vea afectada por los cambios, puesto que permite reconsiderar los factores de calidad y a la vez evoluciona el proyecto.

CAPITULO II

ESTUDIO Y ANALISIS DE LOS ESTANDARES DE PRUEBAS IEEE, ISO, ANSI PARA LOS REQUISITOS,

ANALISIS, DISEÑO, CODIFICACION E IMPLEMENTACION

El objetivo final de la Ingeniería de Software es producir software de calidad abarcando el proceso de desarrollo de Software, que puede describirse y controlarse desde diversas perspectivas como es el grado de aceptación del software, la aprobación del software con la especificación y características que lo hacen funcional para el usuario. Para garantizar la calidad en el proceso de desarrollo de software es necesario aplicar estándares de pruebas, revisiones y así detectar defectos en la lógica, función o implementación del mismo.

Durante los años 80, aparecen estándares que miden la calidad de software IEEE, ISO y ANSI herramientas para ejecución de pruebas y evaluación de software considerando que los problemas a resolver hoy son más complejos y críticos, por lo que se requiere la aplicación de métodos, herramientas efectivas para la implementación de pruebas.

2.1 TIPOS DE ESTÁNDARES DE PRUEBAS

La complejidad en el desarrollo de Software actual, ha llevado a buscar la evolución tecnológica basada en el Estudio y Análisis de los Estándares de Pruebas IEEE, ISO y ANSI para asegurar la calidad de producto, obteniendo un mejoramiento continuo de todas las fases relacionadas con el desarrollo de Software, conllevando a beneficios como mejorar la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión.

En términos generales, provee una guía de cómo obtener el control del proceso de desarrollo y mantenimiento de software, hacia una cultura de ingeniería de Software, describiendo un juego de pruebas básico, estandarizado y proporcionando un marco común de referencia que puede servir como el control de integridad para el proceso de comprobación asociado. Que son las descripciones siguientes:

IEEE Constituyen un papel importante en el proceso de desarrollo de Software. "Estableciendo procedimientos de calidad personalizados para cada Fase, su desarrollo es trivial por lo que requiere de planificación, aplicación y esfuerzo llevando a cabo un proceso de comprobación con éxito". [19]

ISO **"**Establecen una definición general de lo que es la calidad del producto software en general, tanto desde el punto de vista del usuario final como del que lo construye, de ahí que existe la visión externa como la interna de establecer Modelos de Calidad". [20] Permitiendo realizar mediciones de calidad previas antes de dar por culminado el desarrollo del producto, sirviéndonos de apoyo para desarrollar un producto integro y fiable.

ANSI "Abordan la calidad en el proceso de desarrollo de software definiendo los principales términos y resumiendo los pasos a realizar para llevar a cabo las tareas de verificación, validación de las etapas del ciclo de vida presentando los principales modelos y marcos de calidad" [21] Presentan las principales técnicas de detección de errores dado que en este campo se realizan análisis de las habilidades que mejoran notablemente el proceso manual de verificación y validación en el transcurso del proceso de desarrollo Software

2.1.1 EXPONER LOS ESTÁNDARES DE PRUEBAS IEEE, ISO, ANSI

Los estándares de pruebas, prescriben el alcance, acercamiento, recursos, de las etapas del proceso de desarrollo de software identificando los artículos, rasgos a ser probados en cada fase a ser desarrollada. Revisa el esquema que se establece para el proceso de desarrollo de software, implantando las acciones a ejecutarse para alcanzar un nivel mayor de madurez y calidad.

Los errores promueven la definición y aplicación de un proceso de pruebas minuciosas y bien planificadas. Permitiendo validar y verificar el software, entendiendo como **validación** el proceso que determina si el software satisface

[19] http://window.to/concepcion.com.do, 2000

[20] http://www.mvp-access.com/rubenvigon/pdf/testing_software.pdf.,2003.

[21] http://www.lsi.us.es/docs/doctorado/memorias/Memoria_ProyectoInvestigador.pdf,2005

los requisitos, y **verificación** como el proceso que determina si los productos de una fase satisfacen sus condiciones.

Dentro de la Ingeniería de Software están otros Estándares de Pruebas IEEE y el uso de ellos es totalmente voluntario, se sujeta cada cinco años por lo menos para revisión o reafirmación de intereses garantizando calidad internamente y comprende el.

- IEEE STD 829 Año 1983 “Documentación de Pruebas”
- IEEE STD 1008 Año 1987 “Pruebas Unitarias de Software”

2.1.1.1 IEEE STD 829

“El propósito de esta norma es describir un juego de pruebas de software básicos, estandarizando la definición para un plan de pruebas que sirve como un control de integridad para el proceso de comprobación asociado con el ciclo de vida del Software”.[22]

Especifica el formulario de documentos de pruebas individuales que son asociadas con los aspectos dinámicos del software probando la ejecución de procedimientos, código, definiendo el propósito, contorno, y volumen de cada fase aplicables a la comprobación, este estándar no se restringe por el tamaño, ni complejidad del software.

Comprende técnicas exactas usadas para tratar el proceso de desarrollo de software como una guía detallada que satisface los requisitos, brindando precaución para garantizar la calidad.

2.1.1.2 IEEE STD 1008

Esta norma define un acercamiento sistemático y un documento integrado para la comprobación a través de pruebas unitarias en el proceso de Desarrollo de Software.

[22] http://www.procuno.com/users/taller/Presentaciones/PresentacionISO.ppt.,2005

"Describe un proceso de pruebas compuesto jerárquicamente por tareas y actividades, que definen un juego de validación para cada etapa del proceso de desarrollo de software, con resultados que están disponibles a ser verificados".[23]
Este Estándar puede aplicarse a cualquier software, no especifica las unidades que deben ser probadas, es utilizado para el desarrollo, comprobación y modificación de las etapas consideradas parte de la unidad que prueba el proceso de desarrollo, identificando necesidades de análisis de fracaso y corrección de faltas en el desarrollo, utilizando medios específicos y herramientas que prueban la documentación, dirección de la configuración, convicción de calidad, y dirección del proceso de comprobación.

"La Ingeniería de Software es cada vez más pendiente de los requerimientos de calidad en el proceso de desarrollo de software por lo que se basa en los Estándares de Pruebas ISO que actúan con mayor predicción por conseguir la habilidad de producir software eficiente y con calidad".[24]
Por consiguiente dentro de los estándares de Pruebas ISO que influyen la validación y verificación del proceso de desarrollo de software encontramos los siguientes:

- ISO/IEC 9126 : Estándar del Modelo de Calidad del Software
- ISO/IEC 14598 : Estándar para el Proceso de Evaluación del Producto Software

2.1.1.3 ISO/IEC 9126

"Este estándar define la usabilidad como una contribución relativamente independiente a la calidad del software, asociada con el diseño y la evaluación de la interfaz de usuario y la interacción".[25]

[23] PRESSMAN Roger, Ingeniería de Software, 4ta edición, New York,1997.
[24] PRESSMAN Roger, Ingeniería de Software, 4ta edición, New York,1997.
[25] HAUG Michael,OLSEN Eric,Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

La meta es promover el proceso de desarrollo de software a través de métodos, herramientas y técnicas para lograr niveles altos de eficiencia y calidad en el producto Software, su estrategia consiste en demostrar lo que puede hacerse para mejorar las prácticas de desarrollo a través de las técnicas que crean una meta común mejorando en si el proceso.

Los procesos de pruebas en esta norma comprende la metodología, métodos de apoyo, técnicas y la correcta habilidad para construir un equipo de desarrollo, convirtiéndose en una meta difícil para establecer el proceso de control bien definido en el ciclo de vida del producto software.

Asegura un ambiente indispensable y apropiado para el acercamiento de la calidad total de los problemas en el desarrollo del software comprometiéndose a la implementación y mejora en cada una de las fases en que se procesa el Software. "En concreto, la norma ISO-9126 define un modelo general de calidad basado en seis características principales funcionalidad, fiabilidad, facilidad de uso, eficiencia, mantenibilidad y portabilidad".[26]

2.1.1.4 ISO/IEC 14598

Comprende un proceso a seguir para evaluar el desarrollo de Software siendo una disciplina que constituye una alternativa valida para mejorar el proceso, en un intento de superar la complejidad del software. "Su objetivo se centra en ofrecer propuestas que comprenden los aspectos de calidad dentro de las fases del ciclo de vida del software, con vistas a ofrecer guías y modelos de calidad para satisfacer las necesidades declaradas o implícitas. La importancia de este estándar implica que la calidad de un producto software no se puede referir únicamente a obtener un producto sin errores si no a obtener mejores resultados en cuanto al proceso de desarrollo".[27] Un Proceso de evaluación de un producto software se define como un conjunto de características y sub características, dependiendo del tipo de producto a evaluarse, permitiendo medir el progreso en el

[26] http://www.lcc.uma.es/~av/misConfs/Calidad%20de%20Componentes%20CR%20Junio%202004.ppt,2004
[27] PRESSMAN Roger, Ingeniería de Software, 4ta edición, New York,1997.

transcurso del desarrollo considerando la evaluación aplicable en todas las fases de desarrollo.

.La gestión de pruebas de software es una de las primeras actividades a ser completadas en un proyecto software, a medida que el proyecto evoluciona debe ser validado periódicamente para reflejar las situaciones desarrolladas, así cada dificultad debe ser remplazada dentro del desarrollo de Software.

La Terminología de la Ingeniería de Software sigue la Guía de Estándares de Pruebas ANSI, identificando herramientas, técnicas y métodos opcionales, que incluyen las actividades de iniciación y finalización dentro del proyecto

Para la gestión de proyectos software en cuanto a la validación y verificación de Pruebas que garantizan la calidad en el desarrollo de Software, abarca dos tipos de Estándares de Pruebas y son los siguientes:

- ANSI 829 (1983) Estándar para la Documentación de Pruebas de Software
- ANSI 1012 (1986) Plan de Validación y Verificación de Software

2.1.1.5 ANSI/IEEE 829

Contiene directamente o por referencia los planes para las funciones de soporte para el proyecto software. "Su objetivo es vigilar la gestión de desarrollo, configuración, aseguramiento de la calidad del software, verificación y validación por lo que cuentan con un soporte detallado de las funciones a realizarse en cada fase del desarrollo de Software, especificando las responsabilidades, requerimientos de recursos y herramientas de soporte al proyecto". [28]Un buen proceso de Pruebas permite obtener una mejor idea de la calidad del software.

Genera un buen plan de pruebas que facilita enormemente el trabajo de un producto software dependiendo de las acciones combinadas entre los tres tipos de disciplinas:

- Desarrollo
- Gestión

[28] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

- Control

Cuyo objetivo es asegurar un nivel de calidad en el producto software desarrollado pruebas que descubren los defectos en el software por consiguiente, esta norma prueba el esfuerzo de las actividades que se desarrollan para obtener un producto fiable.

2.1.1.6 ANSI/IEEE 1012

Determina las metas a establecer para la calidad del proceso de desarrollo, ya que la calidad del producto va a estar en función de la calidad del proceso de desarrollo, "sin un buen proceso de desarrollo es casi imposible obtener un buen producto".[29]

El objetivo de sus actividades es comprobar si un producto posee o no una determinada característica de calidad caso contrario procede a determinar que existen defectos en el producto y hay que corregirlos.

El proceso de prueba conlleva la realización de un conjunto de tareas a lo largo del ciclo de vida del sistema, de acuerdo con el estándar ANSI 1012 el conjunto de pruebas que se deben realizar son:

- Prueba modular, prueba unitaria o prueba de componentes
- Prueba de integración
- Prueba del sistema
- Prueba de aceptación

Establece los mecanismos de control de calidad para medir y evaluar las metas a alcanzar llegando a satisfacer los requisitos establecidos por el Usuario.

2.2 ANALISIS Y ESPECIFICACION DE LOS ESTANDARES DE PRUEBAS

Son la prevención de errores independientemente en el ciclo de desarrollo del Software, al detectar un error inmediatamente será corregido. Su objetivo es la prevención de errores, lo importante no es la función de probar, sino la función de

[29] http://www.yahoo.com/Computers_and_Internet/Software/Testing,2000

diseñar pruebas como el mejor medio de prevenir errores antes de que el sistema pase a la fase de Producción.

"Las Estrategias de los Estándares de Pruebas permiten una correcta construcción del desarrollo de Software, para el control de Calidad". [30]

Incorporando pruebas planificadas para el diseño, ejecución, agrupación y evaluación del software desde su especificación hasta lo abstracto.

Para la Ejecución y Evaluación de los Estándares se permitirá definir y especificar para su respectivo Análisis cada una de las características que perciben.

ESTÁNDARES DE PRUEBAS IEEE

2.2.1 IEEE STD 829-1998

Aprobado el 16 de Septiembre del año 1998 es el Estándar para la Documentación de Pruebas de Software, esta norma especifica el formulario de pruebas individuales, casos de prueba, plan de prueba, la especificación de procedimiento de prueba y el informe sumario de las Pruebas.

El plan de pruebas prescribe el alcance, acercamiento, recursos, y horario de las actividades de comprobación, que identifica los artículos, rasgos, tareas a ser probados y los riesgos asociados con el plan.

2.2.1.1 Alcance

"Esta norma describe un documento de pruebas básicas que son asociadas con los aspectos dinámicos de software, que prueba la ejecución de procedimientos y código definiendo el propósito, contorno, y volumen de cada documento básico con un enfoque normal en la comprobación dinámica",[31]

[30] http://quercusseg.unex.es/jhernandez/dsoa03/papers/proceedings.pdf.

[31] The Institute of Electrical and Electronics Engineers, Inc.345 East 47th Street, New York, NY 10017-2394, USA

Para identificar los rasgos a ser probados un plan de pruebas tendrá la siguiente estructura:

- **Propósito:** El Plan de Pruebas es un documento que especifica una sucesión de acciones para la ejecución de pruebas, describiendo el alcance, acercamiento, recursos, y horario de actividades a ser probadas.

- **Introducción Plan de Pruebas:** Es un documento que especifica los detalles e informa en cualquier evento lo que ocurre durante el proceso de comprobación describiendo las características y referencias de los siguientes documentos:
    - El plan del Proyecto
    - El plan de convicción de Calidad;
    - El plan de gestión de Configuración;
    - Las políticas y Normas Pertinentes;
- **Características a ser probadas :** Identifica las características que incluyen los requisitos lógicos o físicos proporcionando referencias a la documentación de pruebas que contiene:
    - La especificación de Requisitos;
    - La especificación del Plan;
    - Guía del Usuario
    - Guía del Funcionamiento
    - Guía de la Instalación
    - Determina específicamente las características y combinaciones relacionadas con las etapas de desarrollo de software a ser probadas y el plan de pruebas asociado.
    - Describe el acercamiento global a probar, especifica las actividades, técnicas, y herramientas que se usan para probar, describiendo el detalle suficiente para permitir la identificación de las tareas de comprobación y estimación del tiempo
    - Identifica las técnicas que se usarán para rastrear los requisitos.

- **Características a no ser probadas:** Esta sección Identifica todas las características y combinaciones significantes que no se probarán y las razones del por que no se llevara a efecto pruebas.
- **Criterio de la Suspensión y requisitos de Reanudación:** Especifica el criterio a ser usado para determinar si cada ítem que es probado ha pasado o ha faltado la comprobación.

Posee el criterio de suspender todo o una parte de la actividad de comprobación del ítem de prueba asociados con este plan. Detalla las actividades de comprobación que deben repetirse, cuando la prueba es resumida.

- **Pruebas a Entregar :** Esta Sección se encarga de Identificar los documentos siguientes que deben ser incluidos:
  - El plan de la Prueba
  - Especificaciones de Diseño de Prueba
  - Especificaciones de caso de Prueba
  - Especificaciones de procedimiento de Prueba;
  - Informe de la transmisión de Pruebas
  - Registro de Prueba
  - Informe de los incidentes de Prueba
  - Informe de Pruebas

  Dentro de las pruebas deben identificarse los datos y herramientas de rendimiento aplicables en el desarrollo del producto Software.
- **Tareas de Pruebas:** Identifica y prepara el juego de tareas internas necesarias con la habilidad requerida para realizar la comprobación.
- **Las Necesidades del entorno:** Especifica los requisitos, propiedades del ambiente de pruebas conteniendo las características físicas y de comunicaciones del Software, el modo de uso o suministros necesarios que sirven de apoyo a las pruebas. Determina el nivel de seguridad que debe mantenerse, los componentes, datos, herramientas necesarias que están disponibles en las pruebas del desarrollo de Software.

- **Las Responsabilidades:** Identifica los grupos responsables para el manejo, diseño, preparación y ejecución de las pruebas. Estos grupos pueden incluir a los diseñadores, verificadores, representantes del usuario, personal de soporte técnico, personal de administración de datos, y personal de apoyo de calidad para revisar las pruebas.

- **Necesidades del personal y Capacitación:** Especifica y provee las pruebas del personal por el nivel de habilidad e Identifica las opciones de capacitación de las habilidades necesarias.
- **Los Riesgos y contingencias:** Identifica las teorías de alto riesgo del plan de prueba especificando los planes de contingencia, que exigen la planificación de un horario para que se cumpla con la fecha de entrega.

2.2.1.2 Especificación De Plan De Prueba

Especifica las actividades de las pruebas e identifica los rasgos a ser probados y sus pruebas asociadas. La especificación del plan de prueba tendrá la estructura siguiente:

a) **Identificador del Plan de Pruebas:** Proporciona referencias estrictas sobre las Pruebas asociadas para el Proceso de Desarrollo del Producto Software.
b) **Rasgos a ser probados:** El objetivo de esta especificación es describir los rasgos, referencias y combinaciones necesarias para los requisitos asociados
c) **Perfecciones del Acercamiento:** Especifica el acercamiento de técnicas descritas en el plan de pruebas a ser usadas y permite identificar los resultados de las pruebas aplicadas.
d) **Identificación de las Pruebas:** Es la descripción breve de cada caso de prueba asociado con el plan. Un caso de la prueba puede ser identificado en cada una de las especificaciones del plan y en cada procedimiento asociado con el Plan de Pruebas.

2.2.1.3 Especificación De Caso De Prueba

Para definir y especificar un caso de prueba debe ser identificado en el plan de prueba. La especificación de los casos de prueba tendrá la estructura siguiente:

a) Identificador de Casos Específicos de Pruebas: Es el que permite la identificación, asignación y especificación de caso de prueba.

b) Apartados de prueba : Identifica y describe los caso de prueba para cada etapa, considerando las referencias de la documentación de pruebas que comprenden:

- Especificación de Requisitos;
- Especificación del Plan;
- Guía de Usuarios
- Guía de Funcionamiento
- Guía de Instalación

c) Especificación de Entrada: Especifica cada entrada en la que exige ejecutar el caso de prueba, algunas de las entradas se especificarán por el valor mientras otras por las constantes y transacciones. Además identifica las bases de datos apropiada para archivos, mensajes terminales y memoria del sistema operativo.

d) Especificaciones de Rendimiento: Especifica el rendimiento exacto en el que se da a conocer los resultados de las pruebas, destinado para cada fase de rendimiento.

e) Necesidades medioambientales: En esta sección dentro de las necesidades ambientales tenemos:

- Hardware

  Especifica la característica y configuración del hardware que se exige para ejecutar los casos de Prueba.

- Software

  Especifica el sistema y el software de la aplicación en el que se ejecuta los casos de prueba, incluyendo el Sistema operativo, recopiladores, simuladores, y herramientas de prueba. Además, las

pruebas pueden actuar recíprocamente con el software de la aplicación.

f) Requisitos Especiales a ser Procesados: Describe cualquier imposición especial de requisitos en los procedimientos que ejecutan los casos de prueba, involucrando la intervención del operador y procedimientos de determinación de rendimiento.

g) Dependencias de los Casos de Prueba: Son los que identifican los casos de prueba que deben ejecutarse anterior al caso de prueba ya ejecutado, creando dependencia.

2.2.2 IEEE STD 1008-1987

Comprende las Pruebas Unitarias de Software, fue aceptada el 11 de Diciembre de 1987 es el verificador y supervisor de las pruebas de Unidad. "Su principal objetivo es especificar un acercamiento a las pruebas de unidad del Software, utilizadas como una base practica del legítimo diseño del Software, describiendo los conceptos y diseños del Software a ser probado".[32]

Proporciona una guía para ayudar con la aplicación y uso de las Pruebas de Unidad garantizando un producto integro y fiable.

Esta norma fue desarrollada para ayudar aquéllos que proporcionan la entrada, dirigen, supervisan y evalúan la comprobación de la unidad como son:

- Plan de pruebas
- Perfeccionamiento del Plan de pruebas
- Pruebas de Unidad
- Ejecución de los procedimientos de pruebas
- Control para la terminación de pruebas
- Evaluación del esfuerzo de las pruebas de Unidad
- Pruebas de Unidad del Software que Prueban las Fases

En el proceso de Desarrollo de Software cada fase es asociada por tareas que especifican el rendimiento para cada actividad, conteniendo la información

[32] http://www.fing.edu.uy/inco/cursos/ingsoft/iis/files/examenes/ex_030226.doc,2002

necesaria expresando los parámetros de un problema y la solución de acuerdo con las características del software.

2.2.2.1 Plan De Pruebas Unitarias

Uno de los objetivos es identificar los procesos del software a ser medidos bajo las condiciones específicas de la conducta requerida en la documentación del software. Los casos de Prueba directamente reflejan los objetivos de la prueba.

2.2.2.2 Pruebas de Unidad

Programan los módulos junto con los datos asociados operando y satisfaciendo las siguientes condiciones**:**

- Todos los módulos son de un solo programa de computadora
- Por lo menos uno de los módulos en el juego no ha completado las pruebas de unidad
- Es un juego de módulos junto con sus datos asociados y procedimientos, su objeto es la comprobación del procedimiento.

La documentación describe los requisitos funcionales, interfaz, actuación del plan para las Pruebas de Unidad.

Actividades de las Pruebas de Unidad

Se especifican las entradas, tareas y rendimientos involucrados en las pruebas de unidad

Durante el proceso de comprobación, el plan de pruebas debe especificar y determinar las actividades que se realizan, para determinar el informe sumario en el que conste todas las actividades ejecutadas siendo las siguientes:

2.2.2.3 Fase De Planificación Del Plan De Pruebas

2.2.2.3.1 Plan General Recursos y Horarios

Se planifica las pruebas de unidad general que ocurren durante la planificación de la prueba global y debe grabarse en el documento de la planificación correspondiente**.**

Tareas del Plan

- Identifica las áreas de riesgo que describe los requisitos del software para la aplicación del plan de pruebas.
- Especifica los requisitos identificando los procedimientos, estados, funciones, y características de los datos durante el desarrollo de software, cada tabla debe cubrirse por un caso de prueba unitaria.
- Especifica los requisitos para la satisfacción y terminación normal de las pruebas de Unidad en el proceso de Software.
- Determina los recursos requeridos para ejecutar las pruebas unitarias considerando el hardware, tiempo de acceso, comunicaciones, software, herramientas, archivos, formularios o otros suministros.
- Especifica un horario disponible para las actividades de las pruebas unitarias.

2.2.2.4 Determinan los Aspectos a ser Probados

2.2.2.4.1 Determinan las Entradas

- Documentación de requisitos de unidad
- Documentación del Diseño de arquitectura del Software

2.2.2.4.2 Determinan las Tareas

- Describe los Requisitos Funcionales de las Pruebas de unidad.
- Identifica los Requisitos y procedimientos adicionales que pueden probarse eficazmente al nivel de las Pruebas de unidad.

- Identifica el Estados de las Pruebas de Unidad que se detallan en el documento especificando el estado inactivo o de entrada.
- Identifica las características de los datos de entrada, salida y el rendimiento.
- Determina los datos validos y los datos inválidos a ser probados en las Pruebas de Unidad.

2.2.2.4.3 Definición Del Plan General

- **Desarrollo de Tareas**
  - Identifica los casos de prueba existentes y procedimientos a ser considerados para el uso.
  - Identifica los recursos para ejecutar las Pruebas Unitarias especificadas en el plan.
  - Especifica un horario para las Pruebas especificación detallada en el Plan de Pruebas.

- **Desarrollo de Salidas**
  - Información especifica del Plan de Pruebas Unitarias
  - Especificación de los recursos para las pruebas

2.2.2.5 Fase Que Determina Las Pruebas

2.2.2.5.1 Plan de Pruebas

- **Diseño de Entradas**
  - Documentación de los requerimientos
  - Lista de elementos incluidos en la comprobación de Pruebas
  - Información del Plan de Pruebas Unitarias.
  - Documentación del plan de unidad
  - Especificación de las pruebas disponibles antes del proceso de comprobación.
- **Diseño de Tareas**

- Determina la Arquitectura para las Pruebas basándose en las características a ser probadas, su objetivo es identificar los casos de Pruebas asociados y especificados en el Plan de Pruebas.
- Documentación de los requisitos, que especifican los procedimientos de las pruebas de unidad en el plan.
- Especifica las características y técnicas de los casos de prueba basados en la información del Plan.

➢ **Diseño de Salidas**

- Especificación del Diseño del Plan de Pruebas Unitarias.
- Especificación de los procedimientos de pruebas separadas
- Especificación de los Casos de prueba separados
- Perfeccionamiento del plan de Pruebas de unidad

2.2.2.5.2 Diseño Plan Determinado

➢ **Ejecutar las Entradas**

- Información exacta del Plan de Pruebas Unitarias
- Documentación que especifica las características técnicas de los casos de Pruebas Unitarias.
- Estructura y descripción de los datos del software.
- Recursos de apoyo para las pruebas unitarias
- Actividades y Herramientas disponibles para pruebas antes de la comprobación

➢ **Ejecutar Tareas**

- Obtener y Verificar los Datos de Pruebas a ser utilizados en la modificación.
- Obtener los recursos especificados que sirven de apoyo a las pruebas.

- Obtener las características de las Pruebas .que contienen los procedimientos, asegurando la planificación y ejecución de las Pruebas de Unidad.

2.2.2.6 Fase Que Establece Las Pruebas Unitarias

2.2.2.6.1 Ejecución de los Procedimientos de Prueba

- **Ejecución de las Entradas**

  - Verificación de los datos de prueba
  - Recursos de apoyo de prueba
  - Configuración de las características de prueba
  - Especificaciones de los Casos de Prueba

- **Ejecución de las Tareas**

  - Prepara el ambiente para la ejecución de las pruebas de Unidad.
  - Determina los resultados para cada caso de prueba, determinando si la unidad pasó o falló basado en los resultados obtenidos.

- **Ejecución del Rendimiento de las Salidas**

  - Describe el Informe de la ejecución de Pruebas, en su contenido se especifica desde el inicio de la prueba, análisis y los resultados del análisis de fracaso de las pruebas.
  - Analiza las especificaciones de la pruebas
  - Examina los datos de la prueba

2.2.2.6.2 Confirmación para la Terminación

- **Confirmación de las Entradas**

  - Restricción de los requisitos

- Investigación de la ejecución
- Detalles de las pruebas
- Descripción de la estructura de los datos del software.

➢ **Confirmación de las Tareas**

- Determina las necesidades de los requisitos en el proceso de desarrollo de las pruebas y se procede a evaluar el esfuerzo.
- Información del proceso de comprobación. determinando si las pruebas satisfacen su evaluación.
- Suplementa alguna prueba por pruebas adicionales como complemento, siguiendo las especificaciones. de procedimiento de prueba de acuerdo con:
    - La Actualización de los Datos de prueba adicionales
    - Permite modificar las especificaciones de los procedimientos de prueba
    - Ejecuta las pruebas adicionales.

➢ **Confirmación de las Salidas**

- Informe sumario de las Pruebas hasta la culminación del Proceso de aplicación.
- Revisión de la especificación de los Datos y Pruebas adicionales.

2.2.2.6.3 Evaluación del Esfuerzo de las Pruebas de Unitarias

➢ **Evalúan las Entradas**

- Especificación del Plan de Pruebas de Unidad.
- Información de la ejecución.
- Verificación de la información
- Especificación de los Casos de Prueba

- **Evalúan las Tareas**

  - Describe el estado de las pruebas especificando la razón para cada variante, e identificando las pruebas que no son resueltas, determinando las causas del por que no existe soluciones.
  - Evalúa el plan de las Pruebas Unitarias basado en los resultados.
  - Certifique que los productos incluyan la especificación del plan de prueba, casos de prueba, procedimientos y datos de prueba.

- **Evalúan las Salidas de Rendimiento**

  - Completa el informe sumario de las Pruebas el que consta de un apéndice informativo que contribuye con la Aplicación y Pautas del Uso de las Pruebas Unitarias
  - Confirma el uso de la norma como una base para la comparación, como una fuente de ideas para modificar y como un reemplazo para las prácticas actuales.
  - Detallan las comprobaciones adicionales especificando el número y tipo de aprobación para cada proyecto.
  - La documentación posee la información de las pruebas adicionales consideradas en el Plan de Pruebas.

Este Estándar seguirá un modelo genérico demostrando la efectividad, beneficios y técnicas para mejorar el proceso de desarrollo de software.

2.2.3 ISO 9126 - 1991

"Es el Estándar que comprende un Modelo de Calidad para el proceso de desarrollo, el cual consta de características y sub características garantizando la calidad, eficiencia de un producto software". [33]

Las características que permiten describir la calidad de un producto aplicando el Estándar ISO 9126 son:

Funcionalidad.- es la capacidad del software de proveer las funciones que cumplen con las necesidades implícitas y explícitas cuando es utilizado bajo ciertas condiciones.

Fiabilidad.- Permite mantener un nivel específico de rendimiento bajo determinadas condiciones de uso.

Usabilidad.- Es la capacidad del producto software de ser entendido, aprendido, usado y atractivo al usuario, cuando se usa bajo ciertas condiciones.

Eficiencia.- Ofrece el rendimiento apropiado con respecto a la cantidad de recursos utilizados, bajo condiciones prefijadas.

Mantenibilidad.- Capacidad del producto de ser modificado. Dichas modificaciones pueden incluir correcciones, mejoras o adaptaciones a cambios en el entorno y en los requisitos y especificaciones funcionales.

Portabilidad.- Determina la capacidad del software de ser trasladado de un entorno informático a otro

2.2.3.1 Propósito

Expresamente se encarga de examinar todos los dominios para la aprobación, comprobación y dirección de la Calidad considerada a lo largo del proceso de desarrollo de Software, obteniendo una guía en el mejoramiento, validación y verificación del ciclo de vida del producto que se procesa.

Está claro que el mejoramiento y verificación del producto procesado eficazmente es un problema central que debe irse entendiendo directamente con las características que pueden ser medidas a través de la comprobación de los siguientes pasos:

[33] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

- Validar la definición de requisitos
- Identificar requisitos software
- Identificar objetivos para el diseño software
- Identificar requisitos para las pruebas del software
- Identificar requisitos para el aseguramiento de la calidad
- Identificar criterios de aceptación para un producto software terminado
- Integridad de la Documentación.
- Mantenimiento del producto

Las actividades realizadas durante el proceso de desarrollo y mantenimiento aseguran la aplicación de una norma de calidad definida, resolviendo los problemas siguientes:

- Determinan los puntos específicos de calidad a ser logrados.
- Especificación del Cliente
- Recursos disponibles
- Habilidades disponibles
- Niveles de riesgo que pueden aceptarse

"La aplicación del proceso de comprobación basado en el Modelo de Calidad consiste en evaluar los problemas, adoptando estrategias, métodos, técnicas y herramientas que pueden aplicarse para realizar la Validación y Verificación minuciosa".[34]

2.2.3.2 Comprobación y Aprobación en el Proceso de Desarrollo de Software

El software debe ser analizado en el contexto del Proceso de Desarrollo, detallando los pasos que se llevan a cabo basados en la Validación y Verificación de las Fases del Proceso de Desarrollo del Producto, incluyendo todos los aspectos del plan de Pruebas asociados durante el ciclo de desarrollo incrementando la eficiencia.

2.2.3.3 Principales Beneficios del Proceso de Comprobación

[34] http://www.lisi.usb.ve/publicaciones/02%20calidad%20sistemica/calidad_29.pdf, 2003

La influencia competitiva en la calidad del software exige aumentar la Calidad y productividad en todas las fases del ciclo de vida del Software ganando ventajas competitivas y reduciendo el tiempo

Permite la búsqueda sistemática para los defectos o riesgos en todo el proyecto basados en el hecho factible de los requisitos teóricos, que ejerce la presión para entregar un software de calidad.

Ejecución de la prueba "Las actividades de Prueba o comprobación se ejecutan adecuadamente para ayudar a mejorar el descubrimiento de problemas y asegurar la fiabilidad del software".[35]

Cada caso de prueba se ejecuta en el ambiente específico, ya que las pruebas tienen que ser reproducidas durante todo el ciclo de vida del Software para descubrir errores en caso de no ser encontrados se puede volver a ejecutar la prueba.

2.2.3.3.1 Documentación

Esta documentación es fácilmente reutilizable para las actividades de comprobación de futuro en este archivo se describe los casos de prueba. A menudo las opciones de comprobación permanecen alertas durante un eslabón claro e identificable entre las especificaciones del producto, módulos del software, casos de prueba, datos de prueba y archivos de calidad para el proyecto de desarrollo disponible.

2.2.3.3.2 Medida

El Modelo de calidad del producto a menudo se basa en los indicadores relacionados a las características de calidad de software como la Fiabilidad, Funcionalidad, Eficiencia, Usabilidad, Mantenibilidad y Portabilidad permitiendo obtener Software con un buen nivel de calidad y confiabilidad.

2.2.3.3.3 Recursos para el Modelo de Calidad

[35] http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion-Evaluacion-6.pdf, 2004

Las características del Modelo de Calidad requieren de algunos recursos que afectan a las actividades del proceso de desarrollo de software, exigiendo que los requisitos funcionales sean bien determinados y específicos los cuales serán revisados por los casos de pruebas, estableciendo el tiempo suficiente para probar.

2.2.4 ISO 14598 – 1997

"Es el Estándar autorizado para el Proceso de Evaluación del Software definiendo la calidad Externa de un producto que satisface las necesidades establecidas".[36]
La evaluación prepara un procedimiento a seguir para las características de calidad, incluyendo otros aspectos como el tiempo y el coste para la estimación de la calidad de un producto.
Constituye el plan de evaluación que describe los métodos de evaluación y el programa de acciones del evaluador con el siguiente soporte lógico:

- Visión General
- Planificación y Gestión
- Proceso para Evaluadores

2.2.4.1 Plan de evaluación

El plan de evaluación describe los métodos de evaluación y el programa de acciones del evaluador que debe ser consistente con el plan de mediciones.

Proceso de Evaluación

Se encarga de:

2.2.4.1.1 Establecer requisitos de evaluación.

- Establecer el propósito de la evaluación.
- Identificar los tipos de producto.
- Especificar el modelo de calidad.

[36] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

- Características de Calidad
- Especificar evaluación.
- Seleccionar métricas.
- Establecer niveles para las métricas.

2.2.4.1.2 Establecer criterios de evaluación

- Diseñar evaluación
- Producir plan de evaluación

2.2.4.1.3 Ejecutar la evaluación

- Tomar medidas
- Comparar con criterios
- Valorar resultados

2.2.4.2 Módulos de Evaluación

El propósito de Evaluación del Software se puede definir como el modo práctico de asegurar que el software desarrollado alcanza un cierto nivel de calidad y está basada en la evaluación mediante pruebas.

"El objetivo a perseguir es obtener un sistema con características de Usabilidad y accesibilidad siendo necesario incorporar los siguientes criterios durante el ciclo de vida del desarrollo de Software desde las etapas más tempranas de recopilación de requisitos hasta las etapas finales de implementación, pruebas y mantenimiento".[37]

- **Productos intermedios:**

  - Decidir sobre la aceptación de un producto intermedio de un subcontratista;

[37] http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion-Evaluacion-6.pdf, 2005

- Decidir cuando un proceso está completo y cuando remitir los productos al siguiente proceso;
- Predecir o estimar la calidad del producto final;
- Recoger información con objeto de controlar y gestionar el proceso.

➢ **Producto final:**

- Decidir sobre la aceptación del producto;
- Decidir cuando publicar el producto;
- Comparar el producto con otros productos competitivos;
- Seleccionar un producto entre productos alternativos;
- Valorar tanto el aspecto positivo como negativo cuando está en uso;
- Decidir cuando mejorar o reemplazar un producto

Para obtener calidad en el software es necesario identificar los factores de calidad que son esenciales para la aplicación, integridad, seguridad, exactitud, casos de uso, mantenimiento y portabilidad.

2.2.4.3 Establecer requisitos de evaluación

Realiza un estudio previo que permite no solo basarse en el estudio de las características o habilidades individuales sino que también es necesario conocer los requisitos que se necesitan para la evaluación (test de usabilidad) para incorporar mejoramientos.
Las pruebas incluyen efectividad, eficiencia y satisfacción estos factores ayudan en la definición de los requisitos.

2.2.4.4 Establecer criterios de Evaluación

Para alcanzar niveles de calidad se implanta criterios de evaluación utilizando métodos de diseño iterativo que permitan realizar pruebas tempranas que justifican la necesidad de incorporar criterios de evaluación durante todo el ciclo de vida del Proceso de Desarrollo de Software.

Describe las actividades que se deben llevar a cabo para alcanzar un objetivo concreto dentro del análisis, diseño y evaluación de todos los detalles de iteración en el proceso de desarrollo del Sistema.

2.2.4.5 Ejecutar la evaluación

Las técnicas de evaluación son implementadas durante la fase operativa, su objetivo es orientar y dirigir la evolución de Pruebas en función al desarrollo del Producto alcanzando un nivel alto de calidad

En el ámbito de la Ingeniería de Software tanto para el desarrollo y diseño de un producto la tecnología proporciona un ambiente responsable para actividades complejas de informática, en efecto nos permitiremos describir los Estándares de Pruebas ANSI proporcionando procesos bien definidos y las pautas especificas para el proceso del Ciclo de vida del Software.

2.2.5 ANSI/IEEE 829 – 1983

"Es el Estándar para la Documentación de Pruebas de Software, que provee beneficios múltiples, registrando básicamente criterios para la aceptación de un producto, proceso o servicio, originando la creación de procesos técnicos".[38]

La información que contiene el documento es generada por los resultados de las diferentes pruebas del software que establecen una guía de estilo, que detecta un defecto no descubierto.

2.2.5.1 Alcance

Este estándar prescribe el formato y contenido del Documento para la gestión de Pruebas del producto Software. Un documento para la Gestión de Pruebas Software es el documento de control para gestionar el desarrollo de un proyecto software; definiendo los procesos técnicos y de gestión necesarios para satisfacer los requisitos del proyecto.

[38] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

Este estándar puede ser aplicado a todos los tipos de proyectos software. no está limitado por el tamaño o complejidad puede ser aplicado a cualquiera de los segmentos del ciclo de vida de un producto software.

2.2.5.2 Pasos del proceso de pruebas

Se considera que los problemas a resolver hoy son más complejos y críticos, por lo que se debe de emplear métodos y herramientas efectivas, para la implementación de Pruebas para lo cual se debe considerar el siguiente proceso:

- **Planificación** Al comenzar el desarrollo se establecen guías, métodos, estimaciones de recursos requeridos y estándares involucrados.
- **Identificación** Se realiza una estimación más detallada de los recursos requeridos.
- **Especificación** Es la descripción de las pruebas a nivel funcional y a nivel detallado (ejecución paso a paso).
- **Ejecución** Comprende el desarrollo de las pruebas tanto automatizadas como manuales, registrando los resultados.
- **Análisis de defectos** Se identifican los defectos y sus probables causas, también se planean acciones correctivas.
- **Consumación** Se registra la documentación y se prepara el equipo y los casos de prueba para uso posterior

2.2.5.3 Paquetes de trabajo

Son las tareas y actividades que deben establecerse en orden para satisfacer los acuerdos del proyecto identificando la estructura y esquema de las actividades y sub actividades.

2.2.5.4 Funciones de Soporte

Permiten dar soporte al proyecto de software gestionando la configuración, aseguramiento de la calidad del software, verificación y validación.

2.2.5.5 Calendario o Agenda

Esta subsección ofrece una agenda para que las distintas pruebas, actividades y tareas del proyecto fuesen desarrolladlas en un determinado orden y tiempo

2.2.6 ANSI/IEEE 1012-1986

"Es la norma para la verificación y Validación del Software definiendo los métodos y técnicas rentables para construir un Producto integro y fiable basado en componentes".[39]

El propósito mayor es producir software de aplicación basados en pruebas para que ellos puedan llevar paso a paso un control de cada una de las etapas del proceso de desarrollo, logrando un nivel competitivo, reduciendo el costo y tiempo. La generación de Pruebas tiene su propio ciclo que comenzara en la fase de requisitos de Software desde allí ira en paralelo con el proceso de desarrollo de Software que comprende la estructura del Diseño, Objetivos de Diseño, Interfaces de Usuario, Requerimientos de Usuarios, ejecutando el código y Mantenimiento.

Verificación y Validación "Es el proceso que determina si los requisitos para un sistema o componente son completos y correctos, los productos de cada fase de desarrollo satisfacen los requisitos o condiciones impuestas por la fase previa y si el sistema o componente final es acorde con los requisitos especificados abordan tres aspectos fundamentales:

- La calidad de los requisitos,
- La calidad de los productos intermedios
- Las pruebas de aceptación". [40]

2.2.6.1 Alcance

"El Objetivo primordial de este Estándar es producir Software con calidad, basado en perspectivas tales como componentes, usuarios o del propio producto".[41] El

[39] http://www.lsi.us.es/docs/doctorado/memorias/1147448819Memoria_ProyectoInvestigador.pdf,2004

[40] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

[41] http://www.yahoo.com/Computers_and_Internet/Software/Testing,2004

proceso de Verificación y Validación del Software permite determinar si se esta construyendo el producto correctamente y si satisface los requerimientos impuestos previamente en el desarrollo del Software.

Los objetivos primarios que determina esta norma para el desarrollo de pruebas son:

- Implementar pruebas de unidad para encontrar los posibles errores garantizando calidad en la fase que se analiza.
- Determinar las funciones y rentabilidad de los requisitos dados y plan
- Definir el modelo de prueba mas adecuada de acuerdo a su criterio.

2.2.6.2 Generación de Pruebas

Para realizar esta tarea, se encuentran numerosas técnicas y pautas que evitan muchos errores, posteriormente es necesario analizar en profundidad dicha especificación de pruebas para identificar y corregir posibles problemas, conflictos o irregularidades. Es de vital importancia realizar el análisis, verificación y validación, de las etapas del proceso de desarrollo de Software durante él se pueden descubrir conflictos en los requisitos, análisis, diseño permitiendo conocer mejor el problema**.**

2.2.6.3 Componente de Software

Es un elemento que puede desplegarse independientemente según la fase de desarrollo en la que se encuentre basado en operaciones que tecnológicamente aseguran la calidad y son:

- Diseña, analiza, revisa el plan y la aplicación según el modelo del componente dado, manejando, coordinando las actividades que realizan para aprobar y evaluar la actuación de los componentes en los argumentos específicos que opera.
- Define el plan de prueba, proceso de mando, normas, y métrica de evaluación de calidad para los componentes del software. controlando el proceso de desarrollo de software para asegurar la calidad.

2.2.6.4 Propiedades de los Componentes de Software

Consta de las siguientes propiedades básicas:

- **Identidad**: Cada componente debe ser identificable en su desarrollo y ambiente de trabajo.
- **Orden del Modelo** Los componentes del software participan en un sistema, enfocado en el modulo del sistema .para realizar una tarea específica.
- **Entrega Independiente**. Juegan un papel funcional independiente en un sistema, apoyando una tarea específica, pudiendo ser reemplazados bajo ciertas condiciones
- **Interfaz** Es un contrato que especifica lo que el cliente debe hacer para usar la interfaz, accediendo a una función de servicio definiendo qué tipo de servicios y aplicaciones tiene.

Poseen un limitado alcance porque incluyen la especificación para el análisis, diseño, plan, modelos del plan, código de fuente e implementación.

2.3 ESTUDIO COMPARATIVO DE LOS ESTÁNDARES DE PRUEBAS APLICADOS EN LAS ETAPAS DEL DESARROLLO DE SOFTWARE

Hoy en día, debido a la complejidad los Estándares de Pruebas se han convertido en una tarea vital en el proceso de desarrollo de Software, por lo que la Ingeniería de Software ha introducido y popularizado una serie de estándares de Pruebas para medir y certificar la calidad, tanto del sistema a desarrollar, como del proceso de desarrollo en sí.

Para que el proceso de los Estándares de Pruebas sea eficaz debe estar integrado dentro del propio proceso de desarrollo de manera sistemática, minimizando el factor experiencia o intuición

“El Objetivo principal del Estudio de los Estándares de Pruebas es el control de calidad ajustándose a una especificación concreta a llevarse a cabo en el

transcurso del proceso de desarrollo de Software".[42] De acuerdo a las especificaciones que plantean se procederá a realizar un estudio crucial comparativo de la secuencia de sus actividades que hacen referencia a las etapas del ciclo de vida del Software de forma clara e independiente.

Cuando hablamos del Estudio Comparativo nos referimos al análisis, verificación, validación y aceptación dinámica del comportamiento de una norma, basada en la observación de un conjunto seleccionado de ejecuciones controladas permitiendo establecer el nivel beneficioso que se obtiene con su aplicación.

Como primer punto describiremos el estándar IEEE/STD 829 generado para la documentación de Pruebas de Software aplicando el **Plan de Pruebas** que detallan las actividades requeridas para dirigir las pruebas, procurando tener un proceso de desarrollo bien definido y estable.

2.3.1 IEEE STD 829

El proceso del Plan de Pruebas, dentro del ciclo de vida de desarrollo software consiste en evaluar el sistema, para verificar que satisface los requisitos esperados, o para identificar diferencias entre los resultados esperados y los obtenidos. El proceso de pruebas asegura el correcto funcionamiento e integración del sistema.

"El objetivo es mejorar el proceso de desarrollo de software para encontrar los fallos más importantes, lo más rápido, determinando las metas y el alcance de la evaluación".[43]

Identifica las tareas de comprobación usadas para perseguir los requisitos documentados en el sistema, preparando todo el plan de prueba, casos de prueba, y especificaciones del procedimiento. Este acercamiento verificará la exactitud, comprensión de la información en las áreas cubiertas por las pruebas que representan el uso de la producción del sistema comprendiendo las siguientes etapas.

- Comienzan las pruebas a nivel de módulo

[42] http://www.softwaretechnews.com/stn5-4/inspections.html,2000
[43] http://www.iie.org.mx/bolAT02/aplica1.pdf, 2000

- ❖ Continuar hacia la integración del sistema completo y a su instalación
- ❖ Culminar con la aceptación del producto por parte del cliente

La aplicación de este estándar comienza en la prueba de cada módulo, que normalmente la realiza el propio personal de desarrollo en su entorno

2.4 ETAPAS DEL PROCESO DE DESARROLLO DE SOFTWARE BASADO EN EL ESTANDAR DE PRUEBAS IEEE STD 829

2.4.1 Fase Preliminar

"Es una técnica que se usa para capturar los requerimientos del sistema que no son si no las necesidades y expectativas del Cliente o usuario, declarados en las normas de desarrollo proporcionando las entradas de uso de datos" [44]

El proceso de verificación de los requerimientos comienza con el:

- ➢ Análisis, inspección en la cual se busca evaluar la consistencia, completitud y factibilidad de los requerimientos, tanto individualmente como juntos.
- ➢ Deben ser revisados y validados por los distintos actores involucrados con el sistema acción que debe aclarar los compromisos al respecto, tanto en el sentido de prioridades y balance entre requerimientos.

2.4.1.1 Identificar los Casos de Uso

Son considerados como una característica de la relación de dependencia de los requerimientos para reaccionar a los cambios de sus elementos, siendo la fuente principal para determinar las tareas que posteriormente desarrollara el Sistema.

Específicamente esta norma establece que el control de verificación de los casos de uso concluyen a que es indispensable mantener un control al momento de detallarlos, debido a que existen casos que carecen de la precisión necesaria para elaborar y determinar las tareas que debe ser capaz de ejecutar el Sistema.

[44] http://blogs.msdn.com/askburton/archive/2004/09/20/232065.aspx, 2005

2.4.1.2 Detalle de los Casos de Uso

Es la facilidad que el software debe proveer a sus usuarios constituyendo la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo incluyendo el diseño y implementación.

Guía el desarrollo de la arquitectura permitiendo conceptualizar, gestionar y desarrollar adecuadamente el Sistema.

Para detallar los casos de Uso que determinan la funcionalidad total del sistema se debe considerar la siguiente pregunta:

- ¿Qué debe hacer el sistema … para cada usuario?
- ¿Cuál es la Guía para el Proceso de desarrollo?

2.4.1.3 Definir la Interfaz Inicial del Sistema

Captura los requisitos funcionales basados en los caso de usos, describiendo de qué forma el usuario va a utilizar el sistema, denotando la navegabilidad, extensión, conectividad entre funciones. Es el módulo que específica los vínculos, formatos, guiones para la semántica de la navegación, asegurándose que el diseño de la interfaz sea apropiado: presentando un escenario que determine facilidad de uso, interactividad, legibilidad, estética, características de despliegue, personalización, accesibilidad.

- Los defectos y errores en las interfaces se deben detectar antes, de las uniones entre los módulos.
- Se examina con mayor detalle cada interfaz poco a poco al ir desarrollándola.
- La depuración es mucho más fácil, al encontrar un error ejercitando los mecanismos de interacción.

2.4.1.4 Desarrollo del Diagrama de Clases

La calidad de la estructura del Sistema se define en términos del diagrama de clases desde el punto de vista de la complejidad, considerando aspectos como

complejidad de atributos, complejidad de operaciones y la unión jerarquía de clases.

Muestra gráficamente las entidades del sistema y sus relaciones, estableciendo la realización de los casos de uso en clases ya que "es el proceso que permite el paso de una representación en términos de análisis hacia el diseño incluyendo la orientación hacia el entorno de implementación".[45]

El Diagrama de Clases consta de dos niveles de atributos:

- Completitud Nivel Conceptual: define en forma clara y concisa todas las tareas referentes al Sistema.

- Agregación Nivel Especificación: define de forma clara y concisa todas las propiedades (atributos) y métodos de la clase.

2.4.2 Fase de Diseño

Es el proceso de aplicar técnicas y principios con el propósito de definir la estructura de los datos de entrada, salida para ordenarlas de forma detallada, determinando el modelo de dominio de la información para implementar el Software, describe como se comunica el Software consigo mismo y con los operadores y usuarios que lo emplean. Para ensamblar el Diseño del Sistema se aplica la Prueba de Integración que no es más que un conjunto de pasos que permiten al diseñador describir todos los aspectos del Sistema a construir evaluando la calidad del desarrollo del proyecto, para evaluar el proceso de diseño se establece criterios técnicos a seguir

- Debe presentar una organización jerárquica a través del control de los componentes del software.
- Debe ser modular, es decir realizar funciones y subfunciones especificas realizando una partición lógica del Software.

---

[45] http://msdn.microsoft.com/architecture/journ/default.aspx?pull=/library/enus/dnmaj/html/aj3softfac.asp

- Contener abstracciones de datos y procedimientos.
- Producir módulos que presenten características de funcionamiento independiente.
- Conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno exterior.
- Utilizar métodos que puedan repetirse según la información obtenida durante el análisis de requisitos de Software.

Estos criterios permitirán en la Fase de Diseño del Sistema garantizar la calidad a través de la aplicación de principios fundamentales

2.4.2.1 Identificar la Arquitectura

Comprende la organización global del sistema desarrollando todas las funciones y subfunciones de un Sistema centralizándose en lo que debe hacer incluyendo interfaces bien definidas, notación gráfica, información y comportamiento del Sistema. Proporciona el argumento que se debe considerar al determinar la arquitectura de un sistema a desarrollarse y son:

- Organizar el sistema en subsistemas
- Identificar la concurrencia inherente al problema
- Asignar los subsistemas a los procesos y tareas
- Seleccionar la administración de datos
- Manejar el acceso a recursos globales
- Seleccionar la implementación de control en el software
- Manejar las condiciones de contorno
- Establecer las prioridades

2.4.2.2 Modelo de Análisis y Diseño

Determina los objetos que deben estar activados exclusivamente para el estilo de control en el sistema a desarrollarse, considerando todos los requisitos explícitos contenidos en el análisis de requisitos.

Para definir el Modelo de Análisis se debe considerara los siguientes puntos:

- Especificación clara y precisa del diagrama de clases.
- Examinar todos los casos de uso y crear su ejecución.
- Cada registrador debe participar en varias actividades distintas con distintos roles
- Analiza el entorno, control y entidad.

"El Modelo de Diseño proporciona una idea completa del Sistema, enfocando los dominios de datos, funcional y comportamiento desde el punto de vista del análisis, para adaptar el entorno de implementación definido por los diagramas".[46]

2.4.2.3 Modelo de Clases

Una vez que las clases, atributos y métodos ya están definidos es mas comprensible el problema, ya que se parte del modelo básico en el que se detalla los requisitos de almacenamiento de información, requisitos funcionales permitiéndonos llegar a un diseño completo del sistema, para el desarrollo del modelo se debe considerar:

- Definir un tipo de objetos y sus operaciones
- Identificar las entidades (clases, métodos)
- Equiparar relaciones entre clases
- Atributos: contenido de información de los objetos
- Métodos: operaciones sobre los objetos
- Mensajes: invocación de las operaciones

2.4.3 Fase de Diseño Detallado

Es la estrategia de alto nivel que analiza las características específicas del ambiente de implementación incluyendo decisiones acerca de la organización del sistema en subsistemas, que determinan fundamentos conceptuales, políticas que

[46] http://www.componentsource.com/Services/WhatAreComponents.asp?bhcp=1, 2006

constituyen el marco de trabajo para el diseño detallado, por lo que el diseñador deberá:

- Agregar los detalles de implementación al modelo.
- Completar los detalles de la clase (diagrama de clases)
- Subdividir en paquetes (diagrama de paquetes)
- Asignar los subsistemas de software a los procesadores para satisfacer las necesidades de rendimiento y para minimizar la comunicación entre procesadores
- Determinar las conexiones de las unidades físicas que implementan los subsistemas.

El proceso de Diseño del Sistema para ser probado aplica las Prueba Funcional o de Validación comprobando que se cumpla con los requisitos funcionales y de rendimiento que aportan al desarrollo de interfaces bien definidas, estructuras de datos, archivos y bases de datos de acuerdo a la capacidad de almacenamiento proporcionando diversas compensaciones entre costo, tiempo de acceso, capacidad y fiabilidad..

2.4.3.1 Modelo de Interfaz

Es el entorno que esta atento a todo lo que ocurre y acontecerá en el Sistema, para probar la interfaz se construirá un juego comprensivo de transacciones que consta de:

- Definición formal de cada pantalla
- Diagrama de la pantalla, indicando cuáles objetos tiene y dónde están ubicados.
- Listado de las características tanto de la pantalla como de cada objeto (colores, tamaño de fuentes, resolución de imágenes, etc.)
- Enlaces con otros elementos de la interfaz. En caso de que algún objeto permita desplazarse a otras pantallas.

- Notas adicionales. En caso de que se requiera realizar operaciones especiales en la interfaz. Por ejemplo indicar si hay animación cuando se activa o desactiva la pantalla, si hay música de fondo.
- Diagrama de flujo de información en la Interfaz indica la relación entre las diferentes pantalla de la interfaz, estableciendo cual es la secuencia que se seguirá en la aplicación.

2.4.3.2 Ambiente de la Base de Datos

Abarca las aplicaciones, con diagramas de dependencias funcionales para explicar la normalización, definir restricciones de integridad y para representar el esquema completo que emplea el diagrama entidad/relación.

Para evitar la detección de errores sintácticos, semánticos y de organización se debe contar con el seguimiento de los siguientes puntos:

- Vigila que desde la Interfaz pase información válida al servidor
- Controla que la aplicación procese correctamente y extraiga o formatee los datos del usuario
- Se obliga que los datos del usuario pasen correctamente a la función de transformación de datos al servidor para formatear las consultas.
- Debe precisarse que las consultas pasen a la gestión de datos que se comunica con las rutinas de acceso a datos

2.4.3.3 Modelo de Control, Persistencia y Comunicación para el Sistema

Todo Sistema debe poseer una acción que asume el control antes de ser entregado al cliente, en el que `podamos inspeccionar las entradas al sistema, explorando el dominio de los requerimientos. Para establecer el modelo de control se debe considerar las siguientes recomendaciones:

- Seguimiento y control de actividades.
- Verificación objetiva de la conformidad.
- Revisión por la alta gerencia y resolución de asuntos
- Acceder al sistema con una contraseña apropiada.

- ❖ Probar las transacciones del despliegue.

Se debe complementar la fase de Diseño definiendo formalmente el modelo de Comunicación que incluye los procesos necesarios para asegurar la adecuada generación y almacenamiento de información

Es necesario que sea altamente interactiva, lo cual conlleva tener mecanismos de comunicación entre el usuario y la aplicación, interactuando con una interfaz que sea: amigable, flexible y agradable de usar; también debe ser consistente, en cuanto a los mensajes y la distribución en pantalla, el juego de colores que se presente en el sistema a implementarse.

2.2.4 Fase de Implementación

Esta fase tiene por objetivo establecer un modelo para el funcionamiento de el Sistema de acuerdo a sus necesidades, este modelo involucra la gestión de elegir el lenguaje en el que se va a programar la aplicación, estimación, y una definición clara de la interacción entre el Cliente y el sistema, estableciendo el control del manejo de actividades en caso de que se presenten cambios y la aceptación del producto.

Para su perfeccionamiento se debe considerar la infraestructura, construcción iterativa de las características establecidas en el diseño software.

2.4.4.1 Definir Estándares de Programación

Es un aspecto muy importante para un programador definir el estilo de programación que este utiliza. “Siendo su propósito el de definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también tiene que ver con el orden y legibilidad del código escrito”. [47]

2.4.4.2 Lenguaje

Describe el código que la maquina va a interpretar definiendo las instrucciones y capacidad de programación que posee el lenguaje que vamos a emplear.

[47] http://www.willydev.net/descargas/oguzman-diseno_pruebas.pdf, 2003

2.4.4.3 Codificación y Pruebas de Unidad

Permiten probar el sistema para realizar modificaciones del programa hechas durante el período de prueba del sistema, ayudando a descubrir posibles errores como resultado de su evaluación. “Son las pruebas formales que permiten declarar que un módulo está listo y terminado, estas pruebas suelen realizarlas el propio personal de desarrollo, pero evitando que sea el propio programador del módulo” [48]

2.4.4.4 Pruebas de Integración

Implican una progresión ordenada de pruebas que van desde los componentes o módulos y que culminan en el sistema completo

El orden de integración afecta a diversos factores, como son:

- El orden de codificar y probar los módulos y
- El coste de la depuración

2.4.4.5 Pruebas del Sistema

Permiten comprobar el cumplimiento de todos los requisitos funcionales, considerando al producto software completo, para el proceso de evaluación se debe tomar en cuenta:

- Funcionamiento y rendimiento en las interfaces hardware, software, de usuario y de operador
- Adecuación de la documentación de usuario
- Ejecución y rendimiento en condiciones límite y de sobrecarga

Por último, en el producto final se pasa a la Prueba de Aceptación para que el usuario compruebe en su propio entorno de explotación si lo acepta como está desarrollado.

[48] The Institute of Electrical and Electronics Engineers, Inc.345 East 47th Street, New York, NY 10017-2394, USA,1998

2.5 IEEE STD 1008

El Proceso de aplicación del estándar tiene en cuenta especificar las Pruebas Unitarias del Software creando el Plan que es la aplicación de las pruebas generadas en el proceso de desarrollo de software, permitiendo evaluar y mejorar la efectividad del Sistema a ser evaluado y aceptado.

Las Pruebas Unitarias se generan durante la determinación de las actividades a ejecutarse en la unidad que prueba su desarrollo, ejecución y verificación completa"

"El objetivo de las Pruebas Unitarias es determinar la exactitud e integridad en la aplicación con respecto a los requisitos de la unidad y documentación del plan intentando descubrir los errores".[49] Esta norma representa el acuerdo general en el desarrollo de software, involucrando nuevas políticas, normas, procedimientos y herramientas lo que ayuda a realizar cambios, requiriendo la planificación, requisitos, los objetivos de la prueba, plan, aplicación, y evaluación.

El éxito en el desarrollo de Software de Calidad se basa en cuatro pilares fundamentales: Fase Preliminar, Fase de Diseño, Fase de Diseño Detallado y la Fase de Implementación

2.6 ETAPAS DEL PROCESO DE DESARROLLO DE SOFTWARE BASADO EN EL ESTANDAR DE PRUEBAS IEEE STD 1008

2.6.1 Fase Preliminar

Para la construcción de un sistema el proceso debe describir sintéticamente los requisitos funcionales, obteniendo un conocimiento detallado de cuales son sus necesidades a través de un juego eficaz basado en los procedimientos, estructura y características de los datos

Para el perfeccionamiento de esta etapa del proceso de desarrollo de software se debe documentar los requisitos basándose en las siguientes características:

- Describir los Requisitos Funcionales

[49] http://window.to/concepcion.com.do,2005

- ❖ Identificar los Requisitos y procedimientos adicionales que pueden probarse eficazmente al nivel de las Pruebas de unidad.
- ❖ Identifica las características de los datos de entrada, salida y el rendimiento.
- ❖ Determinar los datos validos y los datos inválidos

Garantizan su correcto funcionamiento teniendo como objetivo verificar que el sistema satisface sus especificaciones y probando que el comportamiento del sistema depende de sus requisitos.

2.6.1.1 Identificar los Casos de Uso

Permiten capturar los requisitos funcionales proporcionando un medio de interacción entre el sistema y su entorno, estructurando los requisitos de acuerdo con los objetivos de los usuarios permitiendo identificar de manera que se consiga comprenderse fácilmente para lo que esta norma establece que para determinar los casos de uso se debe:

- ❖ Identificar los caminos de ejecución posibles.
- ❖ Seleccionar los valores de prueba para cada caso.
- ❖ Detalles de los resultados esperados por cada caso.

"Los casos de uso directamente reflejan la descomposición jerárquica de los objetivos del Sistema".[50]

2.6.1.2 Detalle de los Casos de Uso

Basándonos en los detalles recogidos los casos de uso, determinan los posibles caminos de ejecución que cumplen con los requisitos funcionales del sistema "Las ventajas principales son garantizar la calidad del sistema y permite validar los requisitos, a medida que vayamos detallando la funcionalidad del sistema".[51]

Es necesario disponer de todos los requisitos funcionales para poder comenzar a derivar los casos de uso en cuanto se comiencen a tener requisitos validados, para detallarlos se debe considerar:

- ❖ Precondiciones y poscondiciones
- ❖ Determinar los actores implicados y sus roles

[50] http://www.lsi.us.es/~javierj/investigacion_ficheros/Mejorando%20casos%20de%20uso%2003.pdf.
[51] http://www.turismo.uma.es/turitec/turitec2004/docs/actas_turitec_pdf/15.pdf.2004

- Los pasos de ejecución, las excepciones y las dependencias

2.6.1.3 Definir la Interfaz inicial del Sistema

Considerando que la interacción entre el usuario y el sistema es un factor clave de éxito, para dar inicio al diseño de la interfaz se basa en los datos generados por los procesos de dependencia o entidad, de manera que logremos una interfaz consistente y amigable para el usuario.

Para el diseño de la interfaz se debe considerar dos aspectos primordiales que son:

- Pantallas .- Crear formatos que sean consistentes con el diseño de la interfaz de usuario, definiendo el comportamiento de cada una de ellas y el diálogo entre las mismas, asegurándose que cumplan con requerimientos estéticos y prácticos a los usuarios.

- Reportes y documentos.- Definir los formatos de reportes que deberán generar en el sistema y los documentos utilizados por aquellos usuarios que no se comunican con la aplicación a través de una estación de trabajo.

"Cuando se habla de probar la interfaz, se relaciona en cuanto a medir la conducta real asociada con la interfaz, estado y requisitos esta norma asume que una vez que se establecen los requisitos parte el diseño de la Interfaz".[52]

2.6.1.4 Desarrollo de Diagrama de Clases

Es considerado como el modelo estructural o conceptual que consiste en identificar las clases del sistema, las asociaciones entre las clases, los atributos, representando el entorno de los procesos y las relaciones entre procesos.

Es importante destacar que para el diseño de los Diagramas de clases se debe analizar los siguientes aspectos:

- Representar claramente los roles y procesos de las clases.
- Determinar las relaciones de asociación entre dos clases.
- Debe de indicar si una clase le contiene a otra.

[52] PRESSMAN Roger,Ingenieria de Software, 4ta edición, New York,1997

- ❖ Establece la relación de herencia entre clases, apunta hacia la clase que hereda.
- ❖ Se compromete a considerar las relaciones de dependencia en ambas direcciones entre dos clases o paquetes.

2.6.2 Fase de Diseño

Es un proceso que consta de un conjunto de pasos que permiten describir todos los aspectos a construir, proporcionando una completa idea de lo que es el Sistema, enfocando los dominios de datos, funcional y comportamiento desde el punto de vista de la Implementación.

Aplica ciertas técnicas con el propósito de definir la Arquitectura del Sistema, Diseño de datos, Modelo de Análisis, Diseño y el Modelo de Clases considerando detalles que permitir su interpretación garantizando su proceso de desarrollo.

2.6.2.1 Identificar la Arquitectura

"Es la estructura fundamental de un sistema encargada de determinar los principios que orientan al diseño y evolución del Sistema, describiendo su estructura y las funciones con las que debe cumplir".[53]
Para determinar la arquitectura se debe considerar

- ❖ Definir los módulos principales
- ❖ Especificar las responsabilidades que tendrá cada uno de estos módulos
- ❖ Detallar la interacción que existirá entre dichos módulos:
- ❖ Control y flujo de datos
    - o Secuenciación de la información
    - o Protocolos de interacción y comunicación
    - o Ubicación en el hardware

[53] http://tecnologiaedu.us.es/bibliovir/pdf/30.pdf., 2004

La Arquitectura del Software aporta una visión abstracta, posponiendo el detalle de cada uno de los módulos y el contexto en el que se implantarán, orientados a su diseño y evolución.

2.6.2.2 Modelo de Análisis y Diseño

Obtienen una muestra limitada del proyecto es el que estimula para el proceso de desarrollo determinando si es adecuado para el software a desarrollar de acuerdo a su entorno de desarrollo. El Modelo de Análisis se encarga de establecer y mantener el control de:

- Una interfaz que comprende entidades asociadas.
- Mantener un control de entidades Asociadas y de comunicación
- Estructura de los diagramas de interacción.
- Estructura de los Diagramas de Clases.

El objetivo del Modelo de Diseño es describir el comportamiento del sistema que incluye todos aquellos requerimientos que crean el ambiente de procesamiento esperado, incluyendo:

- Ambiente donde funcionará.
- Eventos ante los cuales el sistema debe responder.
- Funciones que debe cumplir el sistema.
- Esquema global de datos.
- Interfaces con las que interactúa el sistema.

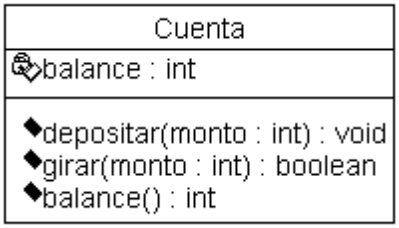
2.6.2.3 Modelo de Clases

"Representa el modelo estructural: teniendo presentes las características específicas del sistema determinando las clases, atributos, métodos, herencia y diagramas de secuencias de eventos que involucran al sistema".[54]

[54] http://martinfowler.com/articles/newMethodology.html ,2005

Para establecer el Modelo de Clases se considera los siguientes aspectos para su desarrollo de acuerdo con la norma IEEE/STD1008:

- Definir los atributos o características de una Clase que representa la comunicación y visibilidad de ellos con el entorno.
- Analiza los métodos u operaciones de una clase determinando como interactúa con su entorno.
- Debe explicar como se interrelacionan dos o más clases (cada uno con características y objetivos diferentes).



2.6.3 Fase de Diseño Detallado

Consiste en decidir y lograr un funcionamiento adecuado del producto software una vez definida la estrategia de diseño a seguir y elaborado el correspondiente diseño arquitectónico, se establece el diseño de la estructura interna, especificando los detalles relativos al formato de almacenamiento.

Consiste en expresar aspectos que contribuyen al proceso de desarrollo de software como es el Modelo de Interfaz, Base de Datos, Modelo de Control, Persistencia y Comunicación para el sistema a desarrollar.

2.6.3.1 Modelo de Interfaz

“Describe como se comunica el Sistema consigo mismo y con los operadores y usuarios que lo emplean dando solución a cada caso de uso dependiendo de la

complejidad".[55] Para establecer el Modelo de Interfaz empleando dicho estándar de pruebas se sugiere considerar los siguientes aspectos:

- Conocer el ambiente de base
- Enlazar las clases de interfaz con el modelo del mundo
- Crear diagrama de interacción (procesos)
- Crear diagrama de estados de procesos.
- Descripción textual de su funcionamiento
- Mostrar la secuencia de operaciones entre los actores involucrados
- Exponer claramente los estados de la interfaz
- Determinar el prototipo funcional de la interfaz interactuando con el usuario

2.6.3.2 Base de Datos

Se considera como la plataforma que organiza los datos y los combina para dar servicio a la aplicación, su estructura se ajusta a las necesidades del sistema para satisfacer los requerimientos funcionales Permite compartir información y mantener la integridad ya que sólo se almacena la información correcta, en definitiva para el diseño de la Base de Datos se debe considerar algunos aspectos que presentan ventajas como: mayor amplitud y capacidad de almacenamiento, por lo que para garantizar la calidad en su desarrollo es aconsejable seguir los siguientes procesos:

- Incorporar el análisis de requerimientos para una base de datos
- Identificación de las funciones e interfaces
- Parte del diagrama de entidad – relación para conocer la estructura de las tablas y sus campos.
- Formalizar los pedidos de acceso a la información.
- Debe establecer la recuperación de datos en un tiempo significativo.

[55] http://www.itba.edu.ar/capis/webcapis/trabajosfinalesdeespecialidad/merlinotrabajofinaldeespecialidad.pdf, 2004

2.6.3.3 Modelo de Control, Persistencia y Comunicación para el Sistema

Permiten identificar, representar y modificar la ejecución de tareas que se relaciona con la comunicación interna y el control de los procesos, facilitando mejorar la comunicación que debe tener el sistema con el usuario en el entorno que se desarrolla. Para establecer un Modelo de Control se debe tener en cuenta.

- Identificar el perfil de cada usuario.
- Considerar que los usuarios pueden elegir funciones del sistema por error y necesitan a menudo una salida de emergencia
- Debe establecerse acciones con disponibilidad de deshacer y rehacer.

El modelo de Comunicación del sistema es el proceso en el que se trata de evaluar la forma de transmitir un mensaje eficiente entre las partes que lo integran para generar, transformar y transferir la información determinando quien habla, con quien y con qué frecuencia.

2.6.4 Fase de Implementación

"Implementar un sistema no solo se concreta en el equipo que se va utilizar sino también en las características físicas del sistema que se va a poner en marcha".[56]

El objetivo de esta fase es comprobar la integración de los componentes establecidos en la fase de diseño detallado hay que tener en cuenta durante las Pruebas Unitarias revisar la arquitectura, diseño, base de datos y generación del producto. Para mantener el control del desarrollo de esta fase se debe comprobar:

- Interfaces desarrolladas sean compatibles entre si.
- La información que se intercambia sea entendible.
- Debe comprobarse que las funciones sean adecuadas según la especificación de requisitos.
- Analiza el comportamiento de interacción entre los módulos.

Se puede decir por tanto que el objetivo de las Pruebas a más de verificar el correcto funcionamiento del producto, es descubrir posibles errores o anomalías reportándolos para dar soporte y solucionarlos.

[56] http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf., 2004

2.6.4.1 Estándares de Programación

Visualizan un marco referencial que abarca la verificación de los dominios de entradas y salidas del programa para descubrir errores de funcionalidad, comportamiento y rendimiento

Fue emprendido en el intento de ofrecer alternativas a los sistemas de programación, específicamente para fines didácticos presentando herramientas que apoyan al programador de forma efectiva para la obtención de resultados.

2.6.4.2 Lenguaje

Tienen el propósito de enseñanza de la programación determinando los recursos adecuados impuestos por el lenguaje o el entorno de desarrollo en general, determinan los conocimientos y destrezas principales para emplear el lenguaje más favorable para el desarrollo de la aplicación.

2.6.4.3 Directorio

Es una iniciativa para diferenciar actividades, de información relacionadas a un proyecto de desarrollo de software, comprendiendo una serie de subdirectorios y archivos con extensión, de acuerdo al lenguaje en el que se a elegido para programar.

2.6.4.4 Codificación y Pruebas de Unidad

Conviene la codificación complementarla una vez que el módulo ya está construido, ya que se pueden presentar obstáculos a la hora de programar. Las pruebas de Unidad permiten declarar que un módulo está listo y terminado cumpliendo con las siguientes condiciones:

- Todos los módulos deben ser del mismo programa
- Al menos uno de ellos no es probado
- El conjunto de módulos es el objeto de un proceso de prueba
- Abarca desde un módulo hasta un grupo de módulos.

2.6.4.5 Pruebas de Integración

Son planificadas durante la fase de requisitos porque es una forma de verificar que los requisitos sean funcionales, claros y precisos, específicamente dentro de este estándar no se aplican ya que su proceso de prueba aplica el sistema de las Pruebas Unitarias.

2.6.4.6 Pruebas del Sistema

"Su propósito es el de pretender evaluar si las interfaces entre los distintos componentes del sistema funcionan de forma adecuada y consistente con el Diseño y los Requerimientos establecidos".[57]

Dentro del estudio realizado en el Estándar IEEE/STD 1008 podemos determinar que no son aplicadas este tipo de pruebas en el proceso de desarrollo de software.

2.7 ISO/IEC 9126

La calidad es uno de los principales problemas que enfrenta actualmente la Ingeniería de Software enfocando sus investigaciones al logro de dos objetivos fundamentales: Cómo obtener un Software de calidad, y cómo evaluar la calidad del Producto obtenido.

"Dicha norma establece que la medición de la calidad de un producto software debe hacerse en base a un Modelo de Calidad que permite la evaluación y estructura del software, enfocándose a la calidad interna y externa, definiendo un conjunto de seis características".[58]

- **Funcionalidad.-** Es la capacidad del software de proveer las funciones que cumplen con las necesidades implícitas y explícitas bajo ciertas condiciones.
- **Fiabilidad:** Capacidad del software de mantener un nivel específico de rendimiento bajo determinadas condiciones de uso.

[57] http://junit.sf.net/doc/testinfected/testing.htm,2001
[58] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

- **Usabilidad:** Capacidad del producto software de ser entendido, aprendido, usado bajo ciertas condiciones**.**
- **Eficiencia:** Ofrece el rendimiento apropiado con respecto a la cantidad de recursos utilizados, bajo condiciones prefijadas.
- **Mantenibilidad:** Capacidad del producto de ser modificado. Incluyendo correcciones, mejoras o adaptaciones a cambios en el entorno, requisitos y especificaciones funcionales**.**
- **Portabilidad:** la capacidad del software de ser trasladado de un entorno informático a otro.

La importancia de cada característica de calidad varía dependiendo de los puntos de vista considerados como son el punto de vista del Usuario, Desarrolladores y Clientes. Este Estándar solo cubre las partes del ciclo de vida correspondiente a las fases de Evaluación, Elaboración, Construcción y Transición del software, pero no cubre la parte del ciclo de vida correspondiente al mantenimiento.

2.8 ETAPAS DEL PROCESO DE DESARROLLO DE SOFTWARE BASADO EN EL ESTANDAR DE PRUEBAS ISO/IEC 9126

El Proceso de Evaluación de la Norma ISO/IEC 9126 presenta el siguiente esquema que se puede aplicar en Proceso de desarrollo del ciclo de vida de un producto software evaluando tres fases fundamentales del modelo y son:

- Definición de Requerimientos de Calidad
- Preparación de la Evaluación
- Procedimiento de Evaluación

Pasaremos a describir su especificación original de acuerdo a las necesidades del proceso de evaluación, describiendo procedimientos específicos para realizar las actividades del proceso de desarrollo del Producto Software.

2.8.1 Fase de Especificación

"En esta etapa inicial del desarrollo es esencial establecer las metas que se desean alcanzar por lo que es necesario identificar previamente el Análisis de usuarios, Identificación de los casos de Uso, especificaciones de la Interfaz y Diagramas de Clases".[59]

La característica de funcionalidad pretende esclarecer en esta Etapa una protección correcta y completa de los requisitos de usuario, mediante pruebas de cobertura cubriendo, la verificación de los requisitos de usuario, asegurando la calidad.

De acuerdo al Modelo de Calidad la Fase de Especificación se la llama como la fase de **Definición de Requerimientos de Calidad** donde se especifican los requerimientos de acuerdo a las características y posible subcaracterísticas prescritas para el Software. Se describirá cual será la entrada y salida de este proceso:

- **Entrada:** Se considera a todas las necesidades explícitas e implícitas del usuario.
- **Salida:** Obtendremos una especificación de requerimientos de calidad para el producto software bajo la evaluación, definiéndolos antes del desarrollo**.**

2.8.1.1 Identificación de los Casos de Uso

"El término Identificación de Casos de Uso se usa para describir un conjunto de técnicas que se preocupan de determinar la tarea o lo que debe hacer el Sistema".[60]

Es el que determina los servicios que el sistema debe proporcionar al cliente, y las limitaciones bajo las cuales éste se comprometería a operar presentando tres importantes propósitos que deben ser analizados, detallados antes de determinarlos.

---

[59] http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion-Evaluacion-6.pdf.,2005

[60] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

- **Motivación :** permiten a los desarrolladores entender cómo el cliente quiere que trabaje el sistema
- **Procedimiento:** especifican a los diseñadores la funcionalidad y las características que el sistema debe tener**.**
- **Técnicas relacionadas:** especifican al grupo de prueba lo que deben demostrar para convencer al cliente que el sistema satisface sus necesidades

2.8.1.2 Detalle de los Casos de Uso

Los casos de uso se pueden detallar dependiendo de la necesidad del problema. Tomando en cuenta los siguiente datos para describirlos.

- **Nombre del Caso de Uso** : Reflejan las tareas que el usuario necesita lograr usando el sistema, incluyendo un verbo de acción y un nombre.
- **Creados Por**:: Detalla el nombre de la persona que inicialmente documentó este caso de uso.
- **Detalle del Caso de Uso**: Determina la actividad que va a realizar.
- **Fecha que se Creó** : Es la fecha en que el caso del uso fue documentado inicialmente.
- **Actor**: Es la persona que participara para la ejecución del caso de uso.

2.8.1.3 Definir la Interfaz Inicial del Sistema

La interfaz de un sistema consiste en los aspectos con los que el usuario entra en contacto, física, perceptiva o conceptualmente, reflejando las propiedades físicas y las funciones a realizar, obteniendo un balance de control tomando en cuenta dos puntos importantes a la hora de diseñar una interfaz :

- **La visibilidad**: Para poder realizar una acción sobre un objeto este debe ser visible.
- **La comprensión intuitiva**: permite que sea evidente el objeto sobre el que se realizara la acción y cómo hacerlo.

Si se desea construir una interfaz usable, se debe primero conocer a fondo a qué usuarios específicos está destinado y cuáles son sus características principales.

- **Entrada:** Es el esquema en el que se conoce el criterio del usuario, para las tareas que se llevan a cabo para el diseño de la Interfaz.
- **Procedimiento:** Los procedimientos a llevarse a cabo son los siguientes:
  - **Visitas de campo:** es muy útil observar al usuario en su entorno de trabajo.
  - **Cuestionarios:** Permite obtener la información acerca de los usuarios mediante cuestionarios de acuerdo al entorno en el que se desarrolla.

Una vez que se haya aplicado este tipo de análisis concreto para el diseño de la Interfaz se concluirá determinando que se obtendrá un producto software con una interfaz fácil y manejable de acuerdo a las perspectivas del Usuario.

2.8.1.4 Desarrollo de los Diagramas de Clases

"Muestran la definición, especificación de las entidades para las clases software de una aplicación y esta compuesto por los siguientes elementos:

- Clase: atributos, métodos y visibilidad.
- Relaciones: Herencia, Composición, Agregación, Asociación y Uso". [61]

Se establecen antes de comenzar con el diseño del sistema dando origen a relaciones de dependencia. Para representar el diagrama de clases dentro del estándar ISO/IEC 9126 se debe considerar los siguientes puntos:

- ***Motivación*:** Es preciso tener un conjunto de especificaciones de los atributos para que puedan ser verificados.
- **Procedimiento:** Del conjunto de atributos especificados se debe seleccionar los que son mas importantes para el sistema a construir considerando que pueden crear dependencia de sus atributos por lo que existen dos tipos:

[61] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

- **Local**: Cuando en un método de una clase se define una variable local que es un objeto de otra clase, se dice que la primera tiene visibilidad local sobre la segunda. La relación de dependencia entre ambas clases se etiqueta con el estereotipo <>.
- **Global:** Cuando hay una variable global en el sistema, instancia de una clase A, y un método de una clase B llama a un método de A, se dice que la clase B tiene visibilidad global sobre la clase A. La relación de dependencia entre ambas clases se etiqueta con el estereotipo <>.

2.8.2 Fase de Diseño Preliminar

"Esta etapa comprende la Fase de Análisis y Diseño Detallado del Producto Software, Identificando la Arquitectura del Sistema que este comprende el Modelo de Análisis, Modelo de Diseño, Modelo de Clases, Modelo de Interfaz, Bases de Datos y el Modelo de comunicación para el Software a desarrollar".[62] Una vez identificadas las tareas a las que el sistema va a dar soporte, se puede empezar a diseñar el software, considerando la **Fiabilidad** como uno de los atributos internos que definen la calidad del Diseño, basándonos en el hecho de que esta característica se mide durante la etapa de pruebas para asegurar un nivel de cobertura y por ende que en su desarrollo no contengan errores, permitiendo establecer factibilidad para poderse implementar y tras habilidad para poder navegar desde un requisito hasta donde éste se encuentra representado

La Fase de Diseño en el Modelo de calidad del Proceso de Desarrollo de Software se la identifica como la Fase de Preparación de la Evaluación que especifica tres pasos generales a aplicar:

- Selección de las Métricas de Calidad
- Definición de los Niveles de Puntaje
- Definición de los Criterios de Valoración

[62] http://lsi.ugr.es/~fguti/taller/05/griman.pdf, 2004

Selección de las Métricas de Calidad

Se debe decidir y seleccionar las métricas debido a que las características definidas en el estándar son de alto nivel, relacionándose con las características respectivas, de la fase del proceso de desarrollo**.**

Definición de los Niveles de Puntaje

Define los niveles de satisfacción necesarios debido a que el valor de una métrica, expresa el nivel de satisfacción de los requerimientos, que son indicados refiriéndose a las necesidades específicas con respecto a cada evaluación en particular.

Definición de los Criterios de Valoración

Permiten definir procedimientos para resumir los resultados de las diferentes características. Por ejemplo El procedimiento puede incluir otros aspectos como tiempo y costos.

2.8.2.1 Identificar la Arquitectura

"La arquitectura del Software es estable en la medida que garantice los requerimientos funcionales del Sistema, considerándose como el puente entre los requerimientos del sistema y la implementación".[63]

Permiten aplicar técnicas de Evaluación para la estimación en el Proceso Arquitectónico del Sistema a desarrollarse y son las siguientes:

- **"Técnicas inquisitivas**: Es la técnica basada en dos instrumentos de evaluación relevantes como es la utilidad y los Perfiles del Sistema.
  - Definir los módulos principales
- **Técnicas de medición**: Son utilizadas para especificar atributos de calidad, utilizando instrumentos como los lenguajes de descripción arquitectónico y métricas, que son interpretaciones realizadas sobre la arquitectura".[64]
  - Definir las responsabilidades que tendrá cada uno de estos módulos
  - Definir la interacción que existirá entre dichos módulos

[63] http://griho.udl.es/castella/equip/invest/granollers.html,2005

[64]HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

2.8.2.2 Realizar un análisis costo – beneficio

Este análisis establece una representación en términos de análisis sin incluir aspectos de implementación, hacia una de diseño incluyendo una orientación hacia el entorno de implementación, de acuerdo al avance del proyecto

"El Análisis del Costo beneficio es un conjunto de procedimientos, reglas que definen las funciones y cual va a ser su comportamiento dando soporte a las actividades a ejecutarse para desarrollar un producto que genere beneficios".[65]

"El Costo beneficio se define como el proceso para aplicar ciertas técnicas y principios con el propósito de definir un Sistema, con detalles que puedan ser interpretados durante su desarrollo".[66]

El Análisis del Costo Beneficio se divide en dos etapas que permiten crear un ambiente de evaluación y control más amplio, garantizando la calidad en el proceso de Análisis y Diseño del Sistema y son:

- Diseño del concepto del Sistema
- Diseño de la parte Visual de la Interacción.

2.8.2.2.1 Diseño del Concepto del Sistema

Es una actividad creativa y no puede mecanizarse ya que hay principios generales que nos pueden guiar en cuanto a el desarrollo, definiendo de qué modo va a funcionar el sistema, para que sea fácilmente asimilado por el usuario, intentando lograr un desarrollo estable minimizando la posibilidad de errores. Para su evaluación se debe seguir los siguientes pasos:

2.8.2.2.2 Prototipado

Consiste en la técnica exclusiva de la Ingeniería de Usabilidad, es el primer paso del análisis del sistema, en este proceso se analizan las perspectivas del cliente,

[65] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

[66] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

sus necesidades y requerimientos, que ayudan a la identificación, planificación y desarrollo del proyecto.

2.8.2.2.3 Motivación

Determina un documento de especificaciones técnicas a un nivel abstracto y recomendable durante la identificación de las necesidades en la comunicación con el Cliente.

2.8.2.2.4 Procedimiento

Realiza un estudio de funciones, rendimiento y restricciones que afectan al desarrollo del Análisis del sistema considerando los siguientes puntos:

- **Identificación de Necesidades**

  Analiza las perspectivas del cliente, reconocimiento del problema, necesidades y requerimientos para el proceso de desarrollo de Software.

- **Estudio de Viabilidad**

  Es el Análisis de riesgos donde está relacionada la evaluación de los costos de desarrollo, funciones, rendimiento, restricciones a demás las posibles violaciones que afectan al desarrollo del Software. "En si se establece un Análisis Técnico que evalúa los principios técnicos del Sistema y al mismo tiempo recoge información adicional sobre el rendimiento, fiabilidad, características de mantenimiento y productividad obteniendo resultados técnicos que son la base para determinar perfectamente el desarrollo del Producto Software"[67].

Estas técnicas de pruebas posibilitan que el Modelo de Análisis y Diseño piense acerca de lo apropiado a las necesidades del usuario, favoreciendo un Proceso de diseño más centrado en el usuario.

[67] http://www.fi.uba.ar/materias/7547/Documento%20tipo%20para%20proyectos%20de%20Desarrollo.rtf,2003.

2.8.2.2.4 Diseño de la parte Visual de la Interacción

"Esta filosofía de diseño se conoce como el Diseño centrado en el Usuario asiéndolo responsable de las decisiones que se deben tomar para el proceso de desarrollo para el Modelo de diseño, confirmando que se desarrollará un sistema que en realidad satisface las necesidades".[68]

Su objetivo es obtener sistemas que sean fáciles de aprender, utilizar y efectivos para las tareas a las que dan soporte, considerando los siguientes ítems para mantener un mejor control y evaluación en el proceso de desarrollo del Modelo de Diseño:

- **Entrada**

  Conocimiento del dominio de la aplicación, actividades de los usuarios.

- **Actividades**

  Identificar las necesidades del usuario

  Determinar los requisitos de la aplicación

- **Salida**

  Documento de requisitos del Software

2.8.2.2.5 Determinar la tecnología necesaria

La implementación de un determinado sistema o servicio provoca un impacto importante en cualquier solución de tecnología; particularmente porque cada sistema requiere utilizar una cierta cantidad de recursos físicos (CPU, memoria, disco) lo cual hace que el rendimiento del servidor decaiga. Hacer un análisis de crecimiento, en cuanto al equipos, permite determinar adecuadamente el hardware necesario que debe adquirirse para la cantidad de usuarios actuales y un porcentaje o cantidad de crecimiento esperado en un período de tiempo determinado. Esto evita sobremanera que se produzca un impacto negativo en el trabajo cotidiano del grupo de usuarios iniciales, e incurrir en gastos o inversiones de dinero fuera de presupuesto.

[68] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

Es un análisis previo a la implementación de una solución de tecnología cuyo objetivo principal es el de poder dimensionar adecuadamente los requerimientos en hardware, software, topología y arquitectura de red necesaria para el correcto funcionamiento y trabajo de un equipo de usuarios. Los resultados del análisis de Capacity Planning mitigan completamente problemas de sobre y sub dimensionamiento de servidores. Capacity Planning incluye también esquemas de topología de red y tolerancia a fallos (clustering, NLB) en ambientes de red de gran tamaño.

2.8.3 Fase de Diseño Detallado

En si esta fase es comprendida en general en la Fase de Diseño a través de la interpretación y características del Estándar ISO 9126 consientísando que para su Modelo de Calidad la usabilidad está directamente relacionada con la forma en que el usuario percibe el producto, lo cual ha hecho de esta característica un claro objetivo de medición en las fases de pruebas, garantizando la característica de **eficiencia** en términos de mejoramiento y rendimiento en el proceso de desarrollo de el Modelo de Interfaz, Ambiente de Base de Datos y del Modelo de Control y comunicación del Sistema.

2.8.3.1 Modelo de Interfaz

El Modelo de Interfaz comprende el Ambiente de la Base de Datos y el Modelo de Control, Persistencia y Comunicación para el Sistema determinando los componentes, tareas que se van a desarrollar independiente de la aplicación y sus restricciones operacionales. Esta norma establece que para obtener el margen de control garantizando la calidad en la Interfase del Sistema existen cuatro tipos de tareas que se deben tomar muy en cuenta y son:

- **Tareas de usuario:** son las tareas realizadas por el usuario; normalmente por ejemplo decidir cual es la mejor estrategia para resolver un problema.
- **Tareas de aplicación**: son las tareas ejecutadas completamente por la aplicación, suministran información al usuario, como por ejemplo presentar los resultados de una consulta a una base de datos.

- **Tareas de interacción:** son las tareas que realiza el usuario interactuando con el sistema, por ejemplo pulsar un botón.
- **Tareas abstractas**: son las tareas que necesitan actividades complejas para su ejecución como por ejemplo una sesión del usuario con el sistema.

2.8.3.2 Ambiente de la Base de Datos

"Es un método, herramienta o una tecnología que colecciona datos que se encuentran agrupados, estructurados y organizados los mismos que van a ser usados por el sistema".[69]

Para el diseño de la Base de Datos uno de sus principales requerimientos es el de mantener un contacto estrecho con el cliente ya que es esencial la identificación de las funciones, interfaces; y la especificación de la información teniendo por objetivo que sólo se almacene la información correcta como plataforma para el desarrollo del Sistema.

Consideramos que es primordial el fortalecimiento de la calidad en el desarrollo de la base de datos, según la Norma ISO 9126 al establecer su Modelo de Calidad la Base es afectada por factores de calidad, uno de ellos es el factor de **mantenibilidad** ya que representa el mayor problema del desarrollo software.

Entrada

El diseño de la base de datos da soporte al modelo de Análisis, Diseño y Arquitectura del Sistema para su diseño se debe tomar en cuenta el siguiente proceso:

- Diseño Lógico
- Diseño Físico

El **diseño lógico** de la base de datos es un modelo abstracto que describe como los elementos en la base de datos han de quedar agrupados

El **diseño físico** muestra como la base de datos se ordena en los dispositivos de almacenamiento de acceso directo.

[69] (http://www.dsic.upv.es/asignaturas/facultad/lsi/index.html),2002

Actividades

Para el desarrollo de la Base de Datos se debe llevar a cabo la evaluación del ambiente en el que se desarrolla la Base de Datos según las actividades descritas en el diseño por lo que se recomienda:

- Determinar las relaciones entre los datos. y la manera más eficiente de agruparlos para cumplir con los requerimientos de información.
- Identificar los elementos redundantes y los agrupamientos de los elementos de datos que se requieren para programas de aplicaciones específicos.

Salida

- Oportunidad, asociado a la eficiencia y eficacia.
- Disponibilidad, permitiendo la accesibilidad de datos
- Consistencias (oportunidad + disponibilidad), como calidad de datos
- Evolución, para adaptarse al entorno
- Integridad, en el nivel de los datos así como el sistema.

2.8.3.3 Modelo de Control, Persistencia y Comunicación para el Sistema

Muestran la forma en que las operaciones del sistema son implementadas por objetos interactivos, referentes entre clases, relaciones de herencia, atributos de clases y operaciones en clases.

Entrada

Depura la estructura con detalles acerca de: los tipos de atributos y argumentos, los algoritmos de los métodos y la descomposición, tratándose de un proceso iterativo y combinado con el análisis.

Actividades

- Combina los distintos diagramas para obtener las operaciones de cada clase.
- Diseña algoritmos para cada operación.
- Evita la redundancia en los Modelos de clases para minimizar el acceso de llamadas a funciones.

- Establece el control de procedimientos para la interfaz como guía para el usuario.

Salida

Presentan gran similitud de calificación en cuanto a la fiabilidad y funcionalidad de Control, Persistencia y Comunicación para verificar que mantiene apropiadamente el control de la línea base, así como el registro completo de cambios para clases, atributos y herencias.

2.8.4 Fase de Implementación

Después de tener una visión clara y concisa del diseño, se debe escoger una plataforma de desarrollo y el lenguaje de programación para producir el código fuente, código de la base de datos, Pruebas de Unidad y documentación.

"El propósito de esta fase es el de pasar de un entorno previo a un funcionamiento productivo real, dando soporte a los usuarios de las operaciones productivas y proporcionando soporte a largo plazo".[70]

La **usabilidad** está directamente relacionada con la forma en que el usuario percibe el producto terminado, .estableciendo esta característica como un atributo interno de calidad.

Para la implementación el Software debe poseer la característica de **Mantenibilidad** para poder realizar cambios antes y después de ser Implementado, además debe comprender la habilidad para correr en diferentes entornos dando origen a la **Portabilidad.**

Para garantizar la calidad en la Fase de Implementación se aplica el Test de Usabilidad para la evaluación permitiendo conocer el nivel de usabilidad que alcanza el modelo actual del sistema, para identificar los errores que existen este estándar permite cumplir las siguientes tareas:

- Desarrollar el Prototipo del software
- Codificación
    - Planear la estructura y el diseño residual para el código
    - Auto inspeccioné su diseño o estructura

[70] PRESSMAN Roger,Ingenieria de Software, 4ta edición, New York,1997

- Teclee su código

- Pruebas del Software
  - Pruebas de Unidad
  - Pruebas del Sistema
- Corrección
- Empaquetamiento y Distribución
- Instalar el Software
- Proveer Asistencia y Ayuda a los Usuarios

En la Fase de Implementación se la conoce como la Fase de Procedimiento de Evaluación que determina el nivel alcanzado para satisfacer el desarrollo de las actividades previas detalladas en los requisitos del Sistema.

2.8.4.1 Test de Usabilidad

Consiste en una serie de tareas a realizarse, en el prototipo del sistema para conseguir resultados fiables.

Entrada

Basada en conocer el nivel de usabilidad de un sistema a través de usuarios reales.

Procedimiento

- Es preciso decidir con qué grupos de usuarios se va a probar el sistema, y el número de participantes de cada grupo.
- Diseñar las tareas de test previamente realizando el análisis enmarcadas en un contexto de uso real.
- Permite distinguir claramente los objetivos desde la etapa de análisis de requisitos, siendo un aspecto relacionado con la seguridad.
- Ayuda a depurar la aplicación durante su desarrollo, para que pueda funcionar perfectamente.
- Analiza la información obtenida por el usuario final para establecer las sugerencias y modificaciones.
- Examina el funcionamiento del sistema, acerca del dominio de aplicación.

Salidas

- Finalización del Proyecto
- Entrega de documentación final

2.8.4.2 Definir Estándares de Programación

Establece un estándar de programación y nomenclatura que puede tomar mucho tiempo. De allí la necesidad de encontrar o elaborar aquel que se ajuste más a nosotros considerando los siguientes factores:

Factor nemotécnico: Para que el programador pueda recordar el nombre de una variable fácilmente.

Factor sugestivo: Para que los programadores puedan leer y entender rápidamente nuestro código

Consistencia: Tiene que ver con usar los mismos acuerdos de nomenclatura en todo el programa y hacer que el texto del código sea legible.

Permite el uso de herramientas automáticas de verificación de nomenclaturas para identificar el tipo de dato de cada variable de tal forma que se podrá saber el uso y finalidad de dicha variable o función.

2.8.4.3 Lenguaje

Es una técnica que consiste en un conjunto de reglas sintácticas y semánticas de comunicación que permite expresar las instrucciones que han de ser ejecutadas y especificadas de manera precisa: con que datos debe operar, cómo deben ser almacenados y transmitidos.

2.8.4.4 Directorio

Permite conforme vamos construyendo nuestro software de forma iterativa, corregir errores garantizando una mejor observación para saber que pasa con nuestro código a medida que vamos avanzando en un proyecto. Por lo tanto,

podemos determinar que el software se va construyendo en base a modificaciones que nosotros vamos haciendo en forma ordenada.

2.8.4.5 Codificación y Pruebas Unitarias

Tanto la Codificación y las Pruebas Unitarias son un elemento crítico para la calidad del software, en donde un sistema se ejecuta en circunstancias especificas y los resultados se registran promoviendo la definición y aplicación de un proceso de pruebas minuciosas y bien planificadas, determinando si el software satisface los requerimientos.

2.8.4.6 Pruebas de Integración

"Se realizan para estar seguros que el sistema está funcionando como se espera o como fue diseñado".[71] Se llevan a cabo creando datos de prueba de integración de procesos y la entrada de datos, necesitan una verdadera metodología la cual exige herramientas y conocimientos que implican ponerlo en todas las situaciones posibles, este tipo de pruebas se desarrollan en forma paralela con la programación o existen circunstancias en las que primero se debe realizar la prueba para después proceder a codificar.

2.8.4.7 Pruebas del Sistema

Fueron creadas para asegurarse que el sistema es capaz de manejar el volumen de datos y el tiempo de respuesta, es la Herramienta vital para evaluar el Sistema demostrando su funcionalidad y confiabilidad, efectuando procesos extremos de tal forma que determina si el sistema responde y es confiable produciendo resultados correctos en los tiempos esperados. Es por eso que se deben realizar pruebas especiales tales como:

- Prueba de carga máxima.
- Prueba de almacenamiento.
- Prueba de tiempo de ejecución.

[71] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

- ❖ Prueba de recuperación.

Prueba de Carga Máxima

Consiste en probar si el sistema puede manejar el volumen de actividades que ocurren tanto en número de transacciones, número de terminales y magnitud de los valores.

Prueba de Almacenamiento

Determinar si el sistema puede almacenar una alta cantidad de datos según el diseño y configuración de los archivos.

Prueba de Tiempo de Ejecución

Determina el tiempo de la maquina que el Sistema necesita para procesar los datos de una transacción y si el tiempo de respuesta es adecuado para las funciones de:

- Ingreso de Datos.
- Actualización.
- Transmisión.
- Proceso.
- Reordenamiento e indexación de archivos.
- Consultas.
- Impresiones de comprobantes y listados.

Pruebas de Recuperación

Prueban la capacidad del sistema para recuperar datos y restablecerse después de una falla.

2.9 ISO/IEC 14598

"Establece la evaluación del proceso de desarrollo de un producto para incrementar la rentabilidad, siendo su compromiso lograr la satisfacción del cliente, mejorando la calidad y reduciendo el tiempo de entrega". [72]

Para lograr estos beneficios debe llevarse a cabo la ejecución de casos de prueba específicos, basados en las características de calidad de software como son Funcionalidad, Fiabilidad, Utilidad, Eficiencia relacionados con las fases del Ciclo de Vida del Software.

Para obtener éxito en la aplicación del ISO/IEC 14598 en primer lugar nos enfocamos en las necesidades del cliente: es esencial entender lo que un cliente necesita

PROCESO DE EVALUACIÓN	
Establecer requisitos de Evaluación	Establecer propósito de la evaluación Identificar los tipos de producto Especificar el modelo de calidad
Especificar Evaluación	Seleccionar métricas Establecer niveles para las métricas Establecer criterios de valoración
Diseñar Evaluación	Producir plan de evaluación
Ejecutar Evaluación	Tomar medidas Comparar con criterios Valorar resultados

El proceso de Evaluación generalmente se construye para producir software de calidad, intentando resolver la falta de calidad del producto, tiempo y costo de desarrollo. El Proceso de Evaluación del Software se relaciona directamente con:

- La especificación correcta de los requisitos

[72] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

- Ausencia de problemas en los datos y código
- Facilidad para usar la Documentación dada al Cliente
- Facilidad para mantener y poner al día el producto

2.10 ETAPAS DEL PROCESO DE DESARROLLO DE SOFTWARE BASADO EN EL ESTANDAR DE PRUEBAS ISO/IEC 14598

"El objetivo principal es Evaluar el proceso de desarrollo de software en un ambiente real que mide la integridad, efectividad, eficiencia y productividad". [73]

Este proceso de evaluación genérico representa un conjunto de subprocesos sus entradas, salidas, y se apoya en el modelo de calidad definido en el estándar ISO/IEC 9126.

"Define los subprocesos necesarios para analizar los requisitos, especificarlos, diseñarlos, planificarlos, ejecutar las acciones de evaluación, y obtener conclusiones para cualquier tipo de producto software". [74]

Sin embargo, no prescribe métodos y procedimientos específicos para realizar las tareas, sino que es responsabilidad del evaluador el seleccionar lo más apropiado para un proyecto de evaluación dado.

El proceso de evaluación según el estándar ISO/IEC 14598, comprende cinco subprocesos, con sus respectivas entradas, salidas y son los siguientes:

- ❖ Establecimiento de los Requerimientos de Evaluación
- ❖ Especificación de la Evaluación
- ❖ Diseño de la Evaluación
- ❖ Ejecución de la Evaluación, y
- ❖ Conclusión de la Evaluación

Las entradas al proceso, describe las necesidades y los componentes del producto, especificando métodos y herramientas de la evaluación. En cuanto a

[73] http://www.eqsoft.net/manuales/MEJORPR1.pdf, 2003

[74] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

las salidas del proceso encontramos los documentos de requerimientos, especificación del diseño, plan de la evaluación y el informe de evaluación.

El Proceso de Evaluación tiene el fin de proveer garantía y recursos para obtener calidad en sus actividades, estableciendo confidencialidad responsabilidad y técnicas para obtener información a tiempo y forma para lo que a continuación describiremos el objetivo de los Subprocesos y cuales son los puntos de control que se aplican en cada una de las Fases del Proceso de Desarrollo de Software.

2.10.1 Fase Especificación de Requisitos

Conlleva aplicar de manera sistemática y práctica métodos apropiados de recopilación, desarrollo y evaluación de requisitos buscando alternativas de definición y evaluación para lo cual se aplica el Proceso del Establecimiento de los Requerimientos de Evaluación su propósito es describir la meta y objetivos de la evaluación

De acuerdo al punto de vista del usuario y los riesgos asociados que no son sino los requisitos, se detalla el dominio de la aplicación del producto a evaluar considerado aspectos críticos como seguridad y calidad, para su proceso de Evaluación consideramos los siguientes procesos:

- Visión General
- Planificación y Gestión
- Proceso para Evaluadores
- Establecer los objetivos del sistema
- Identificar las necesidades del usuario
- Determinar los Requisitos de la Aplicación
- Documentar la especificación de los Requisitos Funcionales
- Definición de los Requisitos no Funcionales

Entrada

El punto de partida es proporcionar los requerimientos iniciales que permitirán determinar el alcance del Sistema a desarrollar. Por otra parte, el evaluador debe asegurar la rigidez necesaria del proceso de evaluación para determinar la calidad

del producto, por lo tanto las dos partes deben acordar sobre los requerimientos como un prerrequisito para la continuación del proceso.

Salida

Obtendremos un documento que debe contener la descripción del dominio de la aplicación y la descripción general del propósito del producto.

Esta fase permite proveer para cada requisito la especificación de la información contenida en el producto de acuerdo a las necesidades del Cliente identificando los Casos de Uso, Interfaz inicial del Sistema y el desarrollo de los Diagramas de Clases.

2.10.1.1 Identificar los Casos de Uso

Establecen el proceso de Análisis de Requisitos identificando la situación de los escenarios para establecer como será la situación actual en el futuro, detallando la información que el sistema proporcionará a los usuarios.

2.10.1.2 Detalle de los Casos de Uso

"Definen la descripción de el sistema su aplicación y las especificaciones de las actividades del sistema, pretendiendo investigar qué es lo que la gente quiere de un sistema y entender cuáles son sus necesidades en términos de diseño".[75]

Para la creación de los Casos de Uso manteniendo un control de evaluación se debe tomar en cuenta el siguiente proceso:

1. Capturar y Analizar el conjunto completo de requisitos de los usuarios y entenderlos.
2. Detallar como el Caso de Uso deberá manifestarse durante el diseño del sistema.
3. Determinar que tipo de tarea se debe ejecutar para satisfacer las necesidades del Cliente.

[75] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

2.10.1.3 Definir la Interfaz Inicial del Sistema

"Detallar la Interfaz adquiere importancia ya que consiste en la capacidad de interactuar el Sistema con el usuario, su objetivo no solo es conseguir especificaciones claras y precisas sino lograr un nivel de control, evaluación y aprobación de la Interfaz en forma clara y precisa considerando las restricciones bajo las que debe operar".[76]

En esta etapa se realiza un estudio de los usuarios que van a interactuar con el sistema indicando la visibilidad, atributos y operaciones que se van a desarrollar tomando en cuenta los siguientes pasos para su desarrollo:

- Permitir a los usuarios emplear el teclado y el ratón
- Mostrar mensajes y texto descriptivos
- Proporcionar acciones inmediatas y variables
- Adecuar a los usuarios con diferentes niveles de habilidad
- Hacer la interfaz de usuario transparente
- Permitir al usuario personalizar la interfaz
- Permitir a los usuarios manipular los objetos de la interfaz
- Diseñar la interfaz consistente lo que permitirá que los usuarios puedan preservar el argumento de trabajo de los usuarios

2.10.1.4 Desarrollo del Diagrama de Clases

Representa las principales entidades y relaciones que intervienen con el fin de establecer una base conceptual sólida contribuyendo a sintetizar el dominio de navegación e iteración.

El diseñador deberá construir el diagrama basado en las siguientes descripciones de análisis:

- Para la creación de clases, atributos, relaciones se debe de realizar un análisis y disponibilidad del entorno
- Restringir la transparencia de atributos de acuerdo a los permisos de acceso al usuario.

[76] http://www.info-ab.uclm.es/personal/caballer/download/papers/CursoVerano,2003

- ❖ Poseer cobertura de requisitos en el modelo que va a trabajar.
- ❖ Definen los caminos de navegación que un usuario puede seguir a través del sistema.
- ❖ Proporcionar al usuario formas de acceso a la información.

Por lo tanto, al establecerse el análisis entre las actividades mencionadas anteriormente se obtendrá un proceso de desarrollo de diagrama de clases bien definido.

2.10.2 Fase de Diseño

"Contribuye a mejorar la calidad en el diseño de la aplicación, motivada en la metodología de evaluación de calidad ya que su objetivo es dotar la estructura, presentación y facilidad de navegación al usuario realizando un análisis previo donde se definen los niveles de facilidad de uso que se desea alcanzar".[77]

El objetivo de esta actividad es combinar las diferentes verificaciones con los componentes del producto con el fin de documentar detalladamente los procedimientos para implementar las verificaciones de la Arquitectura, Modelo de Análisis, Modelo de Clases para lo cual se debe analizar restricciones técnicas como:

- ❖ Componentes del producto
- ❖ Presentar un formato para cada una de las actividades que comprende el Diseño.
- ❖ Predefinir los métodos de evaluación
- ❖ Herramientas que sirven de soporte a los procedimientos específicos.

Presenta disponibilidad de recursos para programar las actividades tomando en cuenta los requerimientos ya que de ellos depende el Diseño del Sistema a desarrollar, descripción de la arquitectura que describe los principales componentes del software, diseñando las interfaces internas y externas que describen las unidades de software y se establece la habilidad entre los requerimientos y el Diseño del software.

[77] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

2.10.2.1 Identificar la Arquitectura

Describe los procesos de Software especificando los detalles de cómo implementar o llevar a cabo las actividades y tareas incluidas en los procesos determinando rendimiento, mantenibilidad, fiabilidad, seguridad física y seguridad de acceso al software.

Esta actividad consta de varias tareas que el diseñador deberá llevar a cabo proporcionando apoyo según lo requiera:

- Deberá identificar los elementos de hardware, software y operaciones manuales.
- Asegúrese que todos los requisitos del sistema se distribuyen entre los elementos de configuración de hardware, software y operaciones manuales.
- Deberá evaluarse la Arquitectura del sistema y los requisitos teniendo en cuenta los siguientes criterios de evaluación:
  - Trazabilidad hacia los requisitos del Sistema
  - Consistencia con los requisitos del Sistema.
  - Adecuación de las normas y métodos de diseño usados
- Viabilidad de los elementos de software para cumplir con los requisitos asignados.
- Posibilidad de las operaciones y mantenimiento.

2.10.2.2 Modelo de Análisis y Diseño

"Nos ayuda a definir las guías de calidad integrando las herramientas para automatizar las medidas y elaborar una correspondencia entre los procesos, actividades, tareas adecuadas para el proyecto Software, basado en los criterios de decisión para obtener resultados que satisfagan la necesidad de información para ser entendido, aprendido y usado por los usuarios bajo condiciones concretas".[78]

[78] http://www.pcm.gob.pe/portal_ongei/Banco_Normas/IT-PROC_CICLO_VIDA_SW.pdf,2004

Para un análisis sistemático de la aplicación se necesita un Modelo de Diseño con el cual se pueda describir un diseño de alto nivel en cuanto a la estructura del Sistema mostrando las funciones de la aplicación.

Para detallar el modelo de análisis y diseño se debe aplicar los siguientes criterios de evaluación como son:

- Facilidad para tener acceso a la información y para comprender las operaciones.
- Permitir a los usuarios tener la idea del propósito de lo que el sistema debe realizar.
- Legibilidad, expresa un sentimiento de la validez de toda la aplicación considerando los objetos, operaciones y propósitos del Software a desarrollarse.

Dependiendo del modelo de Análisis y Diseño de software, basado en este estándar ISO/IEC 14598 se establece un ambiente adecuado garantizando un producto integro y fiable.

2.10.2.3 Modelo de Clases

Proporcionan un conjunto de medidas para valorar la complejidad y la calidad del Diseño, automáticamente verificando si existen entidades de fracasos que proporcionan las herramientas de apoyo que puede controlar fácilmente.

Se declara en términos de clases determinando la estructura, comportamiento y funcionalidad, construyendo modelos dinámicos que describen las entidades siguiendo un proceso de evaluación que permite la verificación y validación del modelo.

- Definir la estructura y las relaciones entre clases identificadas en el dominio del problema.
- Describir las posibles secuencias de servicios y los aspectos relacionados con la interacción entre objetos.
- Asociar los cambios de estado entre los objetos originados por los eventos o servicios.
- Definir la semántica asociada a las clases para la navegación permitida en la aplicación del sistema.

- ❖ Definir, de una manera abstracta la estructura lógica de presentación de los objetos en la interfaz de usuario.

2.10.3 Fase de Diseño Detallado

Para el desarrollo de esta Fase se aplica el proceso de Evaluación que describe las unidades de software, que pueden ser codificadas, compiladas y probadas, desarrollando un diseño de alto nivel para el Modelo de Interfaz en esta fase se realiza un estudio de los tipos de usuarios que pueden interactuar con el sistema, indicando qué operaciones podrán ejecutar.

El propósito de este proceso es obtener los resultados a medida en que se vayan realizando las actividades para verificar si el producto conforme a sus requerimientos especificados es desarrollado. Durante la ejecución de la evaluación, se producen resultados intermedios y finales que deben ser revisados lo que podrá ocasionar cambios tanto en su Interfaz, Base de Datos y en el Modelo de Control y Comunicación.

2.10.3.1 Modelo de Interfaz

"Se Evalúa el Modelo para asegurar que la información fluye de forma adecuada conservando su integridad durante todos los procesos de ejecución para asegurarse que funcione correctamente en los límites establecidos".[79]

La aplicación de pruebas pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que su resultado sea correcto, así como que la integridad de la información se mantenga

Para lograr un Modelo de Interfaz consistente esta norma se basa en los siguientes principios:

- ❖ Proporcionar al usuario una ayuda activa.
- ❖ Hacer que los objetos estén disponibles al usuario de forma que pueda utilizarlos en cualquier secuencia y momento.
- ❖ Crear acciones predecibles y reversibles.
- ❖ Construir la Interfaz a partir del conocimiento previo del usuario.

[79] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001
232323

- Forjar los objetos y controles visibles e intuitivos.
- Permitir al usuario personalizar la Interfaz
- Evitar los errores que se pueden originar al momento de interactuar el usuario con el sistema.
- Dar al usuario el control de las técnicas de interacción

2.10.3.2 Ambiente de la Base de Datos

Abarca los requisitos globales a nivel del sistema realizando el análisis y diseño para determinar la funcionalidad de la aplicación. "Su objetivo es analizar las respuestas del sistema al usuario, transacciones y volumen de datos para mejorar el rendimiento"[80]

Para determinar el ambiente, análisis y diseño de la Base de Datos se aplica pruebas que permiten probar la precisión e integridad de los datos almacenados en el servidor examinando las transacciones enviadas por las aplicaciones del cliente para asegurarse que los datos se almacenen, actualicen y se recuperen adecuadamente:

- **Pruebas De carga** .- prueba diferentes combinaciones y niveles de carga que reflejan situaciones de uso real
- **Pruebas De tensión.-** permiten incrementar el ingreso de datos hasta el punto de determinar la capacidad máxima que puede manejar el entorno de la aplicación.

El desarrollador deberá probar la base de datos asegurándose satisfacer los requisitos, documentando los resultados obtenidos, facilitando a la actualización en el caso de que se deba modificar los requisitos para la Integración del sistema y lograr ventajas competitivas.

2.10.3.3 Modelo de Control, Persistencia y comunicación para el sistema

Según la Norma ISO 14598, el control es la gestión orientada al cumplimiento de los requisitos del Software a desarrollar. "El Modelo de Control y Persistencia son

[80] http://quercusseg.unex.es/jhernandez/dsoa03/papers/proceedings.pdf,2001

las técnicas y actividades de carácter operativo, utilizadas para satisfacer los requisitos relativos a la calidad, centradas en dos objetivos fundamentales:

- Mantener bajo control un proceso y
- Eliminar las causas de los defectos en las diferentes fases del ciclo de vida".[81]

Comprende actividades que permiten evaluar y controlar el producto software desarrollado, ejercitando caminos específicos de la estructura de control asegurando una detección máxima de errores.

Modelo de Persistencia permite mantener la información asociada a los objetos almacenada en la bases de datos para funcionar correctamente cumpliendo requisitos, tales como facilidad de uso y transparencia**.**

Las pruebas se centran en el proceso de verificación y validación en el diseño del software: y son:

- Prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada
- Examinan las estructuras de los datos para asegurar que los datos que se mantienen temporalmente conservan su integridad durante todos los pasos de ejecución.
- Debe asegurarse que funcione correctamente en los límites establecidos.

2.10.4 Fase de Implementación

Su proceso de evaluación tiene el propósito de revisar el desarrollo del producto entre el Usuario y el Desarrollador poniendo a disponibilidad los documentos finales antes de Implementarlo, para dar inicio al proceso de evaluación conjunta de todo el proceso de desarrollo de Software aquí es donde el usuario debe tener la oportunidad de realizar comentarios sobre el informe, para que se proceda a realizar los cambios respectivos para satisfacer sus necesidades.

[81] HAUG Michael, OLSEN Eric, Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001

“Esta Etapa se basa en elementos que son genéricos tratando de definir los Estándares de Programación, Codificación y Pruebas de Unidad que deben ser analizados dependiendo del ambiente en el que se vaya a desarrollar”.[82]

2.10.4.1 Definir Estándares de Programación

Es la mejor alternativa de programación que permite definir únicamente el nivel de programación abarcando todas las Fases del Ciclo de desarrollo desde los requisitos hasta su implementación de acuerdo a sus necesidades, Se fundamenta tanto en el Lenguaje como en el Directorio a la Hora de gestionar el desarrollo del Software.

2.10.4.2 Lenguaje

Se debe seleccionar, adaptar y usar lenguajes de programación estipulados, para que sean apropiados estableciéndolos para llevar a cabo las actividades incluyendo la seguridad física y de acceso.

2.10.4.3 Directorio

Su propósito es técnicamente el de almacenar información que contiene: los atributos del proceso de desarrollo Software dónde se encuentra físicamente las características de almacenamiento concreto y permite la navegación rápida y fácil para las modificaciones que pueden establecerse en el transcurso de su desarrollo.

2.10.4.4 Codificación y Pruebas de Unidad

Evidentemente para conseguir un buen desarrollo tanto de codificación y pruebas de Unidad existen aspectos que miden sus líneas de código o sus posibles variantes de acuerdo a los requisitos del producto software incluyendo análisis de los requisitos, diseño hasta la aceptación durante el proceso del ciclo de vida del software.

[82] PRESSMAN Roger,Ingenieria de Software, 4ta edición, New York,1997

2.10.4.5 Pruebas de Integración

Llevadas a cabo para demostrar que el producto software cumple sus especificaciones, desarrollando una estrategia de integración que verifica la integridad del software usando los criterios de aceptación y permitiendo determinar si se cumple un requisito.

2.10.4.6 Pruebas del Sistema

Se producen para integrar componentes, como el manual de operación y el hardware, para que el sistema satisfaga las expectativas del usuario expresadas en los requerimientos del sistema

2.11 ANSI/IEEE STD 829

"El objetivo es generar un documento de Pruebas de Software determinando un nivel de calidad superior para obtener lógica en el funcionamiento y desarrollar el producto con una estructura que sea fácil de usar, manteniendo la integridad de los procesos como parte primordial en las Etapas de desarrollo del producto Software".[83] A la hora de puntualizar la calidad del software se pueden adoptar diferentes aproximaciones como primera aproximación es importante diferenciar entre la calidad del Producto software y la calidad del Proceso de desarrollo, ya que la calidad del producto va a estar en función de la calidad del proceso de desarrollo, sin un buen proceso de desarrollo es casi imposible obtener un buen producto.

Opera en un sistema bajo ciertas condiciones, abarcando actividades que van desde la validación de los requerimientos hasta la verificación de las actividades realizadas por los clientes para aceptar el producto.

Esta técnica se basa en el Plan de Gestión de Software es el documento de control para gestionar un proyecto software que define los procesos técnicos y tareas necesarios para el Proceso de Desarrollo del Software y son:

- Visión General del Proyecto.
- Organización del Proyecto

[83] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

- Procesos de Gestión
- Procesos Técnicos

2.12 ETAPAS DEL PROCESO DE DESARROLLO DE SOFTWARE BASADO EN EL ESTANDAR DE PRUEBAS ANSI/IEEE 829

2.12.1 Fase Preliminar

Describe el propósito y alcance de las actividades, de acuerdo con los objetivos del proyecto obteniendo la visión de determinar los requemitos o requisitos que en lista todos los términos que deben ser desarrollados para satisfacer las necesidades del cliente o usuario.

2.12.1.1 Identificación de los Casos de Uso

Es una técnica para capturar información de cómo un sistema trabaja, o de cómo se desea que trabaje para capturar requisitos. Los casos de uso intervienen durante todo el ciclo de vida del Software permitiendo:

- Definir los límites del sistema, las relaciones entre el sistema y el entorno.
- Describen la funcionalidad del sistema independiente de la implementación.
- Cubren la carencia existente en métodos previos en cuanto a la determinación de requisitos.

"Los Casos de Uso se identifican observando y precisando, actor por actor, las secuencias de interacción, los escenarios, desde el punto de vista del usuario".[84]

2.12.1.2 Detalle de los Casos de Uso

Es una tarea específica que debe ser simple, inteligible, claro y conciso,.a realizarse tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso

[84] http://dis.um.es/jmolina/as.htm,2005

Para detallar los Casos de Uso se debe considerar los siguientes parámetros que no son más que preguntas claves:

- Cuáles son las tareas del actor?
- Qué información crea, guarda, modifica, destruye o lee el actor?
- Se compromete el actor a notificar al sistema los cambios externos?
- Debe al sistema informar al actor de los cambios internos?

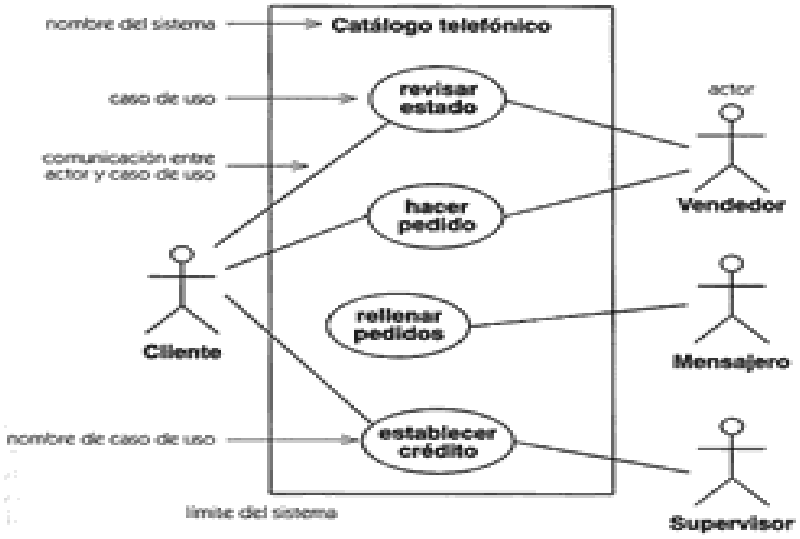


Figura 2.1 Diagrama de los Casos de Uso

2.12.1.3 Definir la Interfaz Inicial del Sistema

"Detalla los límites funcionales de gestión del software y cada una de las entidades de la organización que se encarga del proyecto dependiendo de las actividades que vayan a realizar los usuarios o cualquier otra entidad organizativa que interaccione con el sistema". [85]

[85] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

Establece las funciones de soporte del proyecto tanto para la gestión de configuración, aseguramiento de calidad, verificación, validación del producto construyendo una interfaz. Inteligente fácil de aprender y usar permitiendo a los usuarios hacer su trabajo o desempeñar las tareas especificas para las que fue diseñado.

Para validar la interfaz del sistema existen pasos a seguirse en el proceso de diseño centrados en las siguientes tareas:

- No se deben colocar demasiados objetos en la pantalla, y los que existen deben estar bien distribuidos.
- Análisis de Color no es sólo decorativo, se debe reforzar los mensajes de error
- Determinar el sonido adecuado, permite personalización, volumen y desactivación comprende como ejemplo un mensaje de aviso de correo o de bienvenida, respectivamente, al iniciar una sesión de trabajo.
- Análisis Animación define un cambio en el tiempo de la apariencia visual de un elemento gráfico o explicar su comportamiento.

2.12.1.4 Desarrollo del Diagrama de Clases

Son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones incluyendo herencia siendo utilizados tanto para mostrar lo que el sistema puede hacer, como para mostrar cómo puede ser construido.

Todo lo que un objeto conoce está representado en sus atributos, variables y estado actual, y todo lo que puede realizar está definido en sus métodos comportamiento, y en sus interacciones con otros objetos a través del intercambio de mensajes dinámica del ciclo de vida.

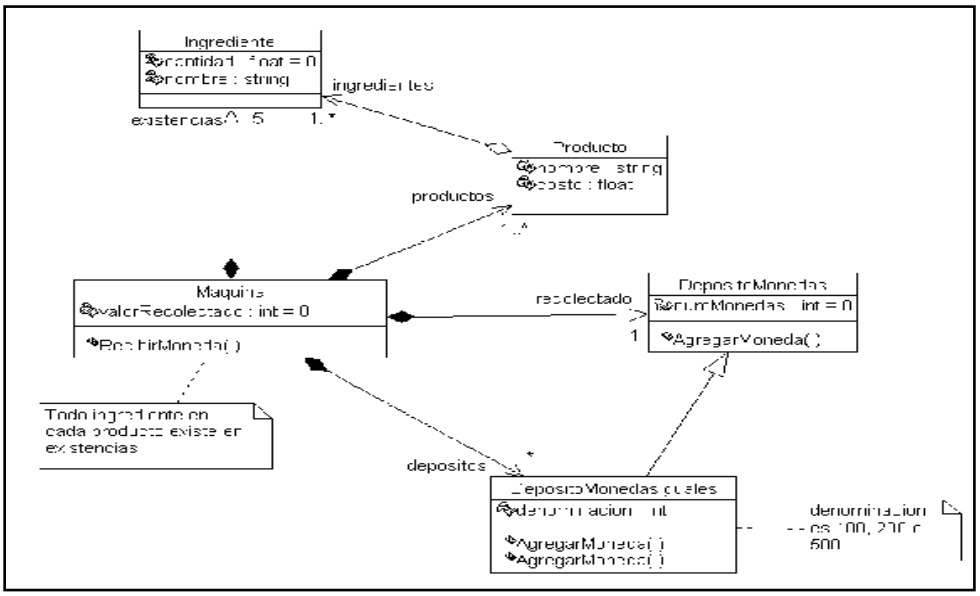


Figura 2.2 Esquema del Diagrama de Clases

2.13.1 Fase de Diseño

Es el proceso que aplica técnicas y principios con el propósito de definir un producto que satisfaga especificaciones, limitaciones traduciendo los requisitos controlando el rendimiento, utilización de recursos, coste impuestos por el medio de destino.

Se considera como la etapa técnica que produce un modelo o representación técnica del producto a desarrollarse, el proceso de validación y verificación dentro del Diseño es determinar y evaluar el estado en el que se encuentra el proceso de diseño, así como descubrir errores en la especificación de requisitos y en el modelo de Análisis, Diseño, Clases.

2.13.1.1 Identificar la Arquitectura

Establece los fundamentos para que analistas, diseñadores, programadores trabajen en una línea común que permita alcanzar los objetivos y necesidades del software no solo de tipo funcional si no también objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas.

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, para establecer el proceso de prueba basado en la norma ANSI/IEEE 829 se considera los siguientes requerimientos:

- ❖ Disponibilidad del sistema
- ❖ Rendimiento
- ❖ Escalabilidad

"Una vez que se cumpla con el respectivo análisis de estos requerimientos se permitirá al equipo de desarrollo comprender mejor la topología de un software".[86]

2.13.1.2 Modelo de Análisis y Diseño

Proporciona métodos, técnicas de supervisión y control para el Análisis y Diseño del software desde el punto de vista funcional.

Su objeto es caracterizar su estructura y funcionamiento, marcando las reglas que permitan alcanzar los objetivos propuestos y evaluar sus resultados agrupando formalmente las tareas que constituyen el análisis en una serie de etapas que suceden de forma iterativa hasta producir y validar el proceso completo.

El Modelo de Diseño describe inicialmente el software, generando un catálogo de requisitos generales que se describiendo de forma detallada los requisitos funcionales que el software debe cubrir, lo que permite mantener un control que identifica los requisitos no funcionales del sistema, es decir las facilidades que proporciona el sistema y las restricciones a que estará sometido, en cuanto a rendimiento y seguridad.

A través de esta norma no existe una especificación clara y precisa como se debe llevar a cabo el Modelo de Análisis y Diseño del Software.

2.13.1.3 Modelo de Clases

Busca lograr una mayor eficiencia en el proceso de los datos considerando las características específicas del sistema, los requisitos establecidos y los detalles del entorno tecnológico a utilizar.

Una vez definidas cada una de las clases, se incorporan al modelo la actividad de análisis, donde se identifican sus atributos, responsabilidades y relaciones para

[86] http://www.vico.org/aRecursos/TRAD_Cards/TRAD_arquitectura/TRAD_Arquitectura_5_esp.pdf,2004

aplicar el proceso de validación, verificación se debe identificar los siguientes tipos de clases:

- Clases de Entidad representan la información manejada en el caso de uso.
- Clases de Interfaz de Usuario se utilizan para describir la interacción entre el sistema y sus actores, representando ventanas de interfaces para la comunicación y formularios.
- Clases de Control son responsables de la coordinación de transacciones y control de los objetos relacionados con un caso de uso.

2.13.2 Fase de Diseño Detallado

Especifica en detalle el entorno tecnológico del software, con la planificación de sus requisitos de operación, administración, seguridad y control de acceso.
"El diseño detallado del software, sigue un enfoque estructurado, que comprende un conjunto de actividades que se llevan a cabo en paralelo a la Definición de la Arquitectura del Sistema".[87] Para alcanzar un efectivo Diseño Detallado del Software se considera tas siguientes actividades en base al estándar ANSI 829:

2.13.2.1 Realizar el diseño lógico de la Base de Datos

Diseño de la Arquitectura de Soporte, establece las normas y requisitos propios del diseño y construcción, así como la identificación y definición de los mecanismos genéricos de diseño y construcción.

2.13.2.2 Realizar el diseño físico de la Base de Datos.

Diseño de la Arquitectura de Módulos del Sistema, realiza el diseño de detalle específico del sistema y la revisión de la interfaz de usuario.

Diseño Físico de Datos, incluye el diseño y optimización de las estructuras de datos del sistema, así como su localización en los nodos de la arquitectura propuesta y la comunicación.

[87] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

2.13.2.3 Modelo de Interfaz

Determina los formatos de pantalla, diálogos entre el usuario y el sistema, su objetivo es realizar un análisis de los procesos del sistema en los que se requiere una interacción del usuario, con el fin de crear una interfaz que satisfaga todos los requisitos establecidos.

- Diseñar el Modelo de Interfaz

  - ✓ Conocer al usuario
  - ✓ Comprensión de la función de negocios
  - ✓ Entender los principios del buen diseño de pantallas
  - ✓ Seleccionar el tipo adecuado de ventana
  - ✓ Desarrollar menús del sistema
  - ✓ Seleccionar los controles basados en dispositivos adecuados
  - ✓ Seleccionar los controles basados en pantalla adecuados
  - ✓ Organizar y distribuir ventanas
  - ✓ Elegir los colores adecuados

- Diseñar las pantallas de entrada – salida
- Diseñar los reportes.
- Diseñar la documentación

La interfaz será gráfica e interactiva considerando los siguientes lineamientos que son:

- Activar operaciones que se produzcan mediante opciones de una barra de menú y botones opcionales.
- Emplear ventanas con formularios que contienen campos de texto, listas de selección simple y botones de selección simple o múltiple.
- Deben las ventanas tener en general, un botón para aceptar los datos provistos, un botón para cancelarlos dependiendo de la funcionalidad corresponde establecerse botones auxiliares para realizar otro tipo de operaciones..

- Conviene los mensajes de error mostrarse mediante ventanas emergentes.
- En todos los formularios, menús y ventanas de la aplicación, sólo estarán activadas las opciones que pueden ser utilizadas. Aquellas opciones que no pueden ser utilizadas en un momento determinado, se encontrarán deshabilitadas.
- Cualquier operación de cancelación o cierre de una ventana exigirá la confirmación del usuario

2.13.2.4 Ambiente de la Base de Datos

"Constituye la herramienta que facilita la aplicación estable de un proceso de administración de datos, pretendiendo que la herramienta cuente con una interfaz gráfica que facilite su utilización y con la posibilidad de generar un reporte de los resultados obtenidos luego de una sesión de identificación de riesgos para un proyecto".[88] Provee mecanismos específicos de seguridad y recuperación ante una eventual necesidad y así continuar operando sin interrupciones.

2.13.2.5 Modelo de Control, Persistencia y Comunicación para el Sistema

El objetivo de esta tarea es definir los procedimientos de seguridad y operación necesarios para el correcto funcionamiento del sistema y garantizar el cumplimiento de los servicios que exigirá el sistema en cuanto a la gestión de operaciones, seguridad y comunicación.

Tomando como referencia los requisitos establecidos permiten llevar a cabo la definición de los requisitos de seguridad y control de acceso necesarios para garantizar la protección del sistema para ello dentro de la norma ANSI 829 se a considerado el análisis de los siguientes procedimientos:

- Acceso al sistema y a sus recursos datos, transacciones y librerías.
- Mantenimiento de la integridad y confidencialidad de los datos.
- Control y registro de accesos al sistema.
- Recuperación de datos.

---

[88] http://www.mvp-access.com/rubenvigon/pdf/testing_software.pdf,2001

- Recuperación ante catástrofes.

2.13.2 Fase de Implementación

"Este proceso tiene como objetivo principal la entrega y aceptación del sistema en su totalidad, estudia su alcance en función de sus características y la realización de todas las actividades necesarias para la producción del mismo".[89]

Incluye la preparación de la infraestructura necesaria para configurar el entorno considerando las siguientes actividades en el proceso de documentación de pruebas:

- Producción y entrega del informe final
- Plan de monitoreo y mantenimiento de los resultados del proceso de Explotación del Software
- Revisión del Proyecto
- Distribución de los resultados.

La Documentación de las pruebas de implementación cubren un rango muy amplio, que va desde la comprobación de cualquier detalle de diseño hasta la explotación de Datos comprobando que el sistema puede encargarse de obtener la información requerida, a justándose a los procedimientos de respaldo, seguridad e interfaces con otros sistemas y que funcione correctamente.

2.13.2.1 Definir Estándares de Programación

Especifica los métodos, herramientas y técnicas del sistema, metodologías de desarrollo, estructura del equipo, lenguaje de programación que van a ser usados para especificar, diseñar, construir, probar integrar, documentar, entregar y modificar el software, así llevar a cabo el desarrollo y la modificación del producto.

2.13.2.2 Lenguaje

Toma en cuenta las restricciones de la plataforma tecnológica y arquitectura de desarrollo sobre la cual se construye el sistema, considerando la mejor

[89] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

alternativa de desarrollo, para que posea facilidades en el proceso de codificación con un alto nivel de madurez y aceptación dentro de la industria del software.

2.13.2.3 Directorio

Está relacionado con el entorno operativo, los principios de programación, la arquitectura utilizada, los algoritmos, y otras posibles soluciones. Si el programador ha trabajado antes con el código, el conocimiento que tenga incluye además cualquier modelo mental del software que posea.

2.13.2.4 Codificación y Pruebas de Unidad

El objetivo de esta actividad es la codificación a partir de las especificaciones obtenidas en el proceso de Diseño del Software, construyendo los procedimientos de operación y seguridad establecidos, además desarrolla las actividades relacionadas con las pruebas unitarias y de integración del sistema basadas en la cobertura de la funcionalidad requerida.

2.13.2.4.1 Pruebas de Integración

Estas pruebas son realizadas por el Diseñador del Proyecto, quien tomará como criterio de evaluación el cumplimiento de los requisitos funcionales, por parte del sistema.

2.13.2.4.2 Pruebas del Sistema

Son orientadas a los métodos de partición equivalencias y análisis de valores límites examinando aspectos externos del modelo del sistema sin tener en cuenta la estructura lógica interna del software. Una vez que todos los casos de prueba han sido superados exitosamente, la aplicación estará lista para ser entregada.

2.14 ANSI/IEEE 1012

“Es la clave para el Plan de Verificación y Validación basado en Pruebas que permiten el cumplimiento de las actividades definidas en el proceso de desarrollo

de Software, con el fin de asegurar la calidad en el producto final llevando a cabo revisiones durante todo el ciclo de vida del software".[90]

Especifica las pautas a seguir durante el proceso de desarrollo para poder asegurar la calidad del producto a construir, detallando todos los atributos y métodos de evaluación de calidad bajo, situaciones ideales de comprobación y evaluación, asegurando características de fiabilidad, mantenibilidad, reusabilidad, para lograr estas metas es necesario entender las características del software

Un software basado en el Plan de Pruebas de Validación y Verificación prueban cada proceso a desarrollarse en forma individual, no solo para medir el grado de calidad de un producto Software sino también construir la calidad durante el proceso de desarrollo del producto, generalmente el proceso de comprobación sigue las siguientes Fases de observación:

- Durante la fase de Análisis de Requisito, deben asegurarse que los requisitos se especifiquen claramente para lograr una norma de especificación que establezca la calidad de requisitos.
- Mientras tanto la fase del Plan, debe proporcionar un modelo de arquitectura que se encarga del diseño de los modelos, interfaz consistente y fácil de manejar.
- En tanto que la fase de Aplicación, debe dirigirse al nivel de código fuente para verificar si los requisitos están propiamente incorporados dentro del código de fuente.
- Finalmente la fase de Comprobación, las pruebas deben verificar que las funciones que realiza el Software satisfacen las necesidades y requerimientos del Usuario.

El objetivo de convicción de calidad de software es establecer y mejorar el proceso claramente definido ejerciendo la disciplina de evaluación para cada paso del proceso, llevando a cabo las siguientes pruebas que son:

[90] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

- Prueba modular, prueba unitaria o prueba de componentes
- Prueba de integración
- Prueba del sistema
- Prueba de aceptación
- Prueba de regresión.

2.15 ETAPAS DEL PROCESO DE DESARROLLO DE SOFTWARE BASADO EN EL ESTANDAR DE PRUEBAS ANSI/IEEE 1012

2.15.1 Fase Preliminar

"Es la fase de definición de requisitos que Identifican y definen un juego completo de características solicitadas a través del usuario o grupo de enfoque".[91]

Determina un detalle completo de requisitos y especificaciones claras, esta revisión se realiza para asegurar que se cumple con los requerimientos especificados por el Cliente, para asegurar la consistencia y disposición técnica de los Casos de Uso, Diseño de la Interfaz y el Diseño de los Diagramas de Clases.

A través del Estándar ANSI/IEEE 1012 establece que para el desarrollo de esta etapa se genera la "Prueba Modular que consiste en mantener un mecanismo integrando de herramientas independientemente de cada modulo seleccionando un acercamiento de tres-fases, la integración de los datos, integración del Diagrama de Clases, e Integración de la interfaz". [92]

Esto lleva a un juego más detallado de requisitos del Software considerando las siguientes características:

- Definición y especificación de requisitos del proceso.
- Los requisitos deben ser impuestos por el usuario y por el desarrollador dar opiniones.
- Deben ser detallados los requisitos en todas las fases;

[91] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

[92] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

- ❖ Integridad de los requisitos aportan a la determinación de las tareas que realizara el Sistema.
- ❖ Declarar propiedades que pueden evaluarse directamente examinando la estructura y funcionalidad de los componentes como atributos, métodos y clases para ser evaluados de forma objetiva.

Al cumplir con cada uno de estos procesos determinados para el control de calidad en la Etapa Preliminar, construyen la parte fundamental para el desarrollo de la arquitectura del Sistema.

2.15.1.1 Identificar los Casos de Uso

Con el fin de asegurar la calidad en el producto, se utilizan estándares y metodologías, para mejorar, entender las actividades y procesos en que se trabaja. para analizar, modelar y diseñar, los Casos de Uso que son manejados dentro del problema.

Es un proceso utilizado para capturar los requerimientos del cliente, en un sistema nuevo a desarrollarse, para estandarizar y elaborar los diagramas requeridos en la etapa de diseño se eligió la herramienta de Rational Rose que permite seguir un estándar en la realización de los diagramas de Casos de Uso, secuencias y clases.

2.15.1.2 Detalle de los Casos de Uso

Es un método común para los requisitos del usuario identificando claramente los diferentes grupos de usuarios.

"Los Casos de Uso son exactamente desprendidos del Documento de Requisitos que cumplen un solo objetivo claramente definido",[93] para detallarlos se debe considerar:

- ❖ Asignar nombres a los casos de uso para que comiencen con un verbo en infinitivo trasmitiendo claramente cuál es su objetivo

[93] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

- ❖ Descripciones de los casos de uso debe ser seguro en base a los requisitos.
- ❖ Las entradas, salidas y condiciones deben estar correctamente definidas para cada caso de uso.
- ❖ Los diagramas de los casos de uso deben ser acordes con las descripciones de los mismos.

2.15.1.3 Definir la Interfaz inicial del Sistema

Para definir la Interfaz se considera la medida de Facilidad de Uso, basándose en la probabilidad de que el Usuario no tenga problemas en la interfaz del Sistema durante el periodo de operación.

Se debe recoger los datos necesarios para calcular la función de probabilidad de los problemas que pueden existir con la Interfaz, asignando mensajes de error informativos y funciones de ayuda, contribuyendo a la facilidad de uso que contribuye a la comprensibilidad o legibilidad.

2.15.1.4 Desarrollo del Diagrama de Clases

"Es el diagrama principal para el análisis y diseño, presenta las clases del sistema con sus relaciones estructurales, herencia incluyendo definiciones para atributos y operaciones".[94] El modelo de casos de uso aporta información para establecer las clases, objetos, atributos y operaciones, definiendo cada objeto a que clase pertenece.

Cada clase se representa en un rectángulo con tres compartimientos:

- ❖ nombre de la clase
- ❖ atributos de la clase
- ❖ operaciones de la clase

Diagrama de Clases muestran la abstracción de una parte del dominio que posee el sistema.

[94] http://www.yahoo.com/Computers_and_Internet/Software/Testing/diagramas,2005

2.15.2 Fase de Diseño

Especifica como el software será construido para satisfacer los requerimientos proponiéndose describir los componentes y sub.-componentes del diseño incluyendo interfaces.

- Todo elemento del diseño debe contribuir a algún requerimiento agregado atributos de calidad al software.
- La implementación de todo requerimiento debe estar contemplada en por lo menos un elemento del diseño.
- Debe ser consistente con la calidad del producto

Para la aplicación del Plan de Validación y Verificación en esta Etapa del desarrollo de Software se aplica la "Prueba de Integración proponiendo el análisis de sistema basado en las perspectivas del-usuario". [95]

El objetivo fundamental de esta prueba es comprobar que las interfaces son correctas, estableciendo comprobaciones que son necesarias realizarlas y son:

- Corrección en la invocación de procedimientos y funciones.
- Compatibilidad de los tipos de funciones y parámetros de llamada.
- Corrección de las especificaciones de los módulos.

Se pueden utilizar tres posibles estrategias de Integración:

- De arriba a abajo: Consiste en empezar la Prueba de integración por los módulos que están en los niveles superiores de abstracción, e integrar incrementalmente los niveles inferiores.
- De abajo a arriba: Consiste en empezar la prueba por los módulos que están en los niveles inferiores de abstracción, e integrar incrementalmente los niveles superiores.
- De big-bang: Consiste en integrar y probar todo al mismo tiempo

[95] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

Las Pruebas de Integración se basan en la comprobación de las funciones del Sistema a desarrollar, por consiguiente cuando una función declara una interfaz crea una relación de dependencia de datos o de otra función que se ejecutara, derivando relaciones de dependencia, y comprobación del Análisis, Diseño y Modelos.

2.15.2.1 Identificar la Arquitectura

Es un juego a nivel de la aplicación que pone énfasis en la identificación y definición de las funciones, analizando la interacción entre el usuario y el sistema, Durante esta fase, nosotros definiremos el Modelo de Análisis, Diseño y Modelo de Clases normalmente incluyen los funcionamientos que se realizarán en las interfases del Software manteniendo la estandarización de entidades, atributos y relaciones. Describe la arquitectura del Sistema de los procesos que se realizara especificando los detalles de cómo realizar las actividades y tareas incluidas en el proceso de desarrollo del Software.

2.15.2.2 Modelo de Análisis y Diseño

La Implementación de Modelos tiene sus ventajas, desde el punto de vista del desarrollo del Software facilitando analizar, todos los requisitos sobre las funciones, actuación, datos y objetos que se emplearan en el ambiente que opera.
El Modelo de análisis normalmente lleva a cabo la especificación de lo que el sistema hará y como nosotros vamos a hacerlo., incluyendo la integración, comprobación, verificación del sistema.
El Modelo de Diseño, es un factor muy importante que afecta al Proceso de Desarrollo de Software ya que sobre el se crean las bases del producto a desarrollar, refiriéndose a la magnitud, capacidad y funcionamiento que atribuye, al desarrollo de la Interfaz y conducta tomando en consideración las siguientes funciones que garantizan la calidad en su desarrollo, como son:

- Comprobación de la sintaxis y la semántica de la especificación de Requisitos.
- Comprobación de consistencia y completitud de la especificación de requisitos.

- Seguimiento de los requisitos.
- Descripción del Modelo de Dominio que nos ayudara a identificar los Usuarios que interactúan con el Sistema.

2.15.2.3 Modelo de Clases

Consiste en dividir las posibles unidades del sistema y sus relaciones de tal forma que todos los miembros de una misma clase prueben las propiedades en el sistema, por lo que sólo va a ser necesario seleccionar un elemento de cada clase, en este estándar no se especifica con claridad como se debe diseñar el Modelo de Clases.

2.15.3 Fase de Diseño Detallado

El objetivo de estas revisiones es determinar y evaluar el estado en el que se encuentra el proceso de Diseño, así como descubrir errores o contradicciones entre la especificación de requisitos y el diseño de las interfaces entre módulos.

Se aplica la Prueba del Sistema para integrar todos los módulos, su objetivo es comprobar que el sistema satisface los requisitos del usuario, tanto los funcionales como los no funcionales".[96]

Para su proceso de evaluación se considera un conjunto de preguntas que permiten la comprobación y son las siguientes

- ¿Que exista uniformidad en el Diseño?
- ¿Se han definido correctamente las interfaces entre módulos?
- ¿Se han definido correctamente las interfaces externas?
- ¿Cubre el diseño todas las funciones incluidas en la especificación de requisitos?
- ¿Cumple el diseño todos los requisitos no funcionales?
- ¿Resulta confusa la documentación del diseño?
- ¿Se ha aplicado la notación de diseño correctamente?

[96] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

- ¿Es el diseño lo suficientemente detallado como para que sea posible implementarlo en el lenguaje de programación elegido?

Una vez realizado el análisis de estas preguntas correspondientemente se garantizara el Diseño de un producto final integro y fiable.

2.15.3.1 Desarrollar el Modelo de Interfaz

Para el desarrollo del Modelo de Interfaz, se debe considerar un grado de precisión adecuado a nivel de la facilidad de uso de un sistema que se desea desarrollar y así conseguir aplicar técnicas para desarrollar software con un enfoque más centrado en el usuario.

Se evaluará la interfaz del sistema .ya que es importante que sea interactiva y que proporcione una buena navegación al usuario, para que no se pierda tiempo en entender el funcionamiento. Lo que se desea probar es que:

- Cumpla con los requerimientos funcionales del sistema
- Unificar el diseño de la Interfaz del sistema.
- Correcto funcionamiento y desempeño.
- Poseer facilidad de uso se puede basar en la probabilidad de que el operador del sistema no se encuentre con problemas en la interfaz durante el periodo de operación.
- Los mensajes de error informativos y las funciones de ayuda deben de estar bien definidos y estructurados.
- Debe de existir comprensibilidad o legibilidad de las pantallas.

Para cada uno de estos aspectos de prueba, en el Modelo de Interfaz se debe revisar y aplicarlos para profundizar correctamente su funcionamiento.

2.15.3.2 Conocer el Ambiente de la Base de Datos

Se asume que los requisitos descritos en este documento son estables una vez que sea aprobado, de tal manera que parten para el diseño de la Base de Datos que permitirá garantizar la integridad de la información que en ella se almacenara.

Establecerá para su manejo la configuración y administración de esta forma se obtendrá un control de cambios que dependerá del servidor de bases de datos con el que cuenta el Sistema a desarrollarse.

En este estándar no especifica con claridad como se debe de realizar el diseño de la Base de Datos por lo que lo realiza en forma general.

2.15.3.3 Desarrollar el Modelo de Control, Persistencia y Comunicación para el Sistema.

Dentro de las disciplinas de control del Sistema, cuyo objetivo es asegurar un cierto nivel de calidad en el producto software desarrollado, definiendo los mecanismos para generar informes. Lo que está claro a partir de estas definiciones es que el Modelo de Control es algo relativo que siempre va a depender de los requisitos o necesidades que se desee satisfacer. Por eso, la evaluación del Modelo de Persistencia de un producto siempre va a implicar una comparación entre unos requisitos preestablecidos y el producto realmente desarrollado.

El control del Sistema es uno de los factores importantes de las pruebas del producto a desarrollar, es la facilidad de producir un rendimiento específico, eficaz declarando el dominio de la entrada.

Considera tres puntos principales para establecer el modelo de control y persistencia para el Sistema y son:

- El mando y conducta del Sistema
- Las características especificas en el campo funcional del Sistema
- La instalación del Sistema para ser implementado.

El mando y conducta del Sistema*:* se encarga de controlar los datos que se obtienen como resultado respondiendo a sus funciones y datos de la entrada.

Las características específicas en el campo funcional del Sistema: se refiere a la capacidad de apoyo y configuración de sus funciones interiores.

La instalación del Sistema para ser implementado: se refiere a la capacidad de instalación del Sistema e Implementación.

Son esenciales para el componen de apoyo en el proceso de desarrollo de Software probando el funcionamiento y contribuyendo al desarrollo bien definido.

El Modelo de Comunicación será el canal de comunicación entre las partes implicadas, dependiendo las plataformas que participarán en su elaboración

previamente establecidas. Esta especificación será revisada tanto por el equipo de desarrollo, quienes realizarán revisiones de las versiones, con las que interactúa hasta alcanzar su aprobación. Una vez aprobado, servirá de base al equipo de desarrollo para dirigir la lógica funcional y toma de decisiones en el ambiente del funcionamiento.

2.15.4 Fase de Implementación

"Incluye la tarea mayor para determinar la aceptación del producto en el tiempo real de ejecución, analiza los posibles fracasos y el mantenimiento para contar con un mejoramiento continuo". [97]

Esta etapa Consta de factores que permiten asegurar la ejecución de todas las actividades como es:

- Simplicidad: Posibilita la implementación de las funciones de la forma más comprensible posible.
- Completitud: Implementa todas las funciones requeridas.
- Trazabilidad: Proporcionan un diseño desde los requisitos a la implementación con respecto a un entorno operativo concreto.
- Auto descripción: Proveen explicaciones sobre la implementación de las funciones .proporcionando amplitud.

Se aplica en esta fase la Prueba de Aceptación, realizada una vez que el sistema se ha implantado en su entorno real de funcionamiento, y su objetivo es demostrar al usuario que el sistema satisface sus necesidades.

2.15.4.1 Definir Estándares de Programación

Especifica y describe los datos, entradas de control requeridos, así como las limitaciones necesarias para la ejecución exitosa del software. De acuerdo a los requerimientos específicos del proyecto uno de sus objetivos es determinar que el Lenguaje de Programación baya acorde con el diseño realizado, examinando los siguientes aspectos para la codificación como son:

[97] http://www.inf.uach.cl/rvega/asignaturas/info265/G_Calidad.pdf,2004.

- Interfaces
- Estructura del programa
- Utilización de variables
- Fórmulas
- Entradas y salidas
- Comentarios

Los estudios realizados demuestran que mediante las inspecciones de los Estándares de Programación, se pueden detectar los errores de programación.

2.15.4.2 Lenguaje

Es el grado de independencia del proceso de desarrollo de Software de acuerdo a las características del sistema operativo y restricciones del entorno, considera dos características principales para determinar el lenguaje en el que se desarrollara la aplicación:

- Trazabilidad - Consiste en seguir una representación del diseño real del programa hasta los requisitos.
- Formación - Es la ayuda que presta el Software para que sea manejado por el nuevos usuarios.

"Las revisiones y validaciones del lenguaje se basan en la lectura del mismo, lo que exige a los desarrolladores un esfuerzo para emplear un lenguaje legible".[98]

2.15.4.3 Directorio

No existe ninguna correlación entre las medidas de calidad del estándar ANSI/IEEE 1012 de acuerdo al plan de verificación y validación para determinar que es un directorio y como se lo debe desarrollar.

2.15.4.4 Codificación y Pruebas de Unidad

Facilita las actividades informando y verificando métodos que enfocan la aprobación de los rasgos funcionales requeridos basados en las especificaciones del módulo, enfocando la aprobación de la estructura del programa, conductas y

[98] ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.

lógica de módulos del software estos métodos de validación y verificación se diseñan enfocándose en:

- Verificar la estructura del módulo y la lógica funcional;
- Confirmar los datos y objetos de clase.
- Identificar las funciones, interfaz, conductas, así como la actuación en un módulo del software.

Una vez que se haya codificado el programa en el ambiente específico que opera, abarca el proceso de mantenimiento para resolver los inconvenientes que se pueden descubrir o presentar.

En esta fase se aplica la Prueba de Regresión que tiene como objetivo comprobar que toda nueva versión de un producto Software que introduce cambios no reduce la valoración de ninguna de las características de calidad que tiene el producto.

CAPITULO III

DISEÑO DE LA PROPUESTA METODOLOGICA MGCM

El desarrollo de software no es una tarea fácil. Prueba de ello es que existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Por una parte tenemos aquellas propuestas más tradicionales que se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los recursos que se deben producir, las herramientas y notaciones que se usarán. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas debido a que no existe metodología que se encargue de analizar y validar cada una de las tareas con las que se debe cumplir desde la fase de Requisitos hasta la fase de Implementación.

De ahí la necesidad de someter a estos procedimientos a un esquema referencial que de alguna forma estandarice su desarrollo, debido que no existe metodología que se encargue previamente del análisis y validación de cada una de sus tareas, únicamente se encargan de validar al momento en que el sistema a sido codificado. No es lo mismo hacer uso de una metodología para el desarrollo de una aplicación con propósitos que garantizar la calidad en todo el ciclo de vida de una determinada aplicación, orientada a satisfacer varias necesidades.

La estrategia de desarrollar la Metodología basada en los Estándares de Pruebas se fundamentan en una debida sistematización de tareas para su desarrollo previas a su diseño e implementación. Es decir, se están creando sistemas en el

menor tiempo posible de lo esperado, puede deberse a factores externos - como la necesidad de satisfacer las necesidades de una organización, sometiéndose a procesos, reglas y procedimientos que garanticen la integridad y mantenimiento del sistema a medio y largo plazo.

La integración de desarrollar una Metodología centrada en los Estándares de Pruebas para el proceso de desarrollo del software es necesaria para comenzar a mejorar de forma sustantiva la usabilidad de las aplicaciones. Va a mejorar el escenario de desarrollo introduciendo el control y validación de cada una de sus Etapas de desarrollo, como también de cada un de las tareas que debe cumplir sus fases de forma sistemática.

En primer lugar se tratan dos aspectos genéricos: una propuesta que creemos más estructurada de ciclo de vida que permita la incorporación sistemática de las tareas de Ingeniería de Usabilidad en el desarrollo de las aplicaciones; y una traducción metodológica de principios de diseño conceptual propuestos por los Estándares de Pruebas.

La primera va en el camino de hacer más sistemática la ingeniería de usabilidad dentro de la ingeniería del software; la segunda intenta incorporar la reflexión y evaluación en los primeros momentos del análisis, diseño e implementación siguiendo el principio de que los errores detectados al inicio son más fáciles y menos costosos de corregir que en momentos más tardíos.

La Metodología MGCM (**Metodología que Garantiza la Calidad y Mantenimiento del Software**) aportara con un marco de trabajo al proceso de desarrollo de Software basándose en los Estándares de Pruebas, cuyo propósito es atribuir un proceso disciplinario en el ciclo de vida con el fin de hacerlo más predecible y eficiente de acuerdo a sus necesidades.

A la Metodología que Garantiza la Calidad y Mantenimiento del Software, se la puede definir como el conjunto de métodos y técnicas que poseen lógica para solucionar o ayudar a resolver los problemas en la gestión de desarrollo de

Software, el éxito depende en gran medida de cuán rápido pueden obtenerse resultados positivos optimizando los procesos

Su objetivo central está relacionado con el desarrollo de sistemas incluyendo el estudio de cómo los Estándares de Pruebas trabajan, estableciendo la estructura apropiada para el proceso de desarrollo, aplicando nuevos métodos, técnicas y procedimientos.

El modelo que se emplea permite identificar las Fases del Proceso de software que son aceptadas por la mayoría de Ingenieros en Sistemas,. especificando la gestión que determina y aplica las políticas de calidad, los objetivos y las responsabilidades, a través de medios tales como la planificación de la calidad, el control de la calidad y la garantía de la calidad. La presente figura muestra el ciclo de vida del producto software, que se aplicara.

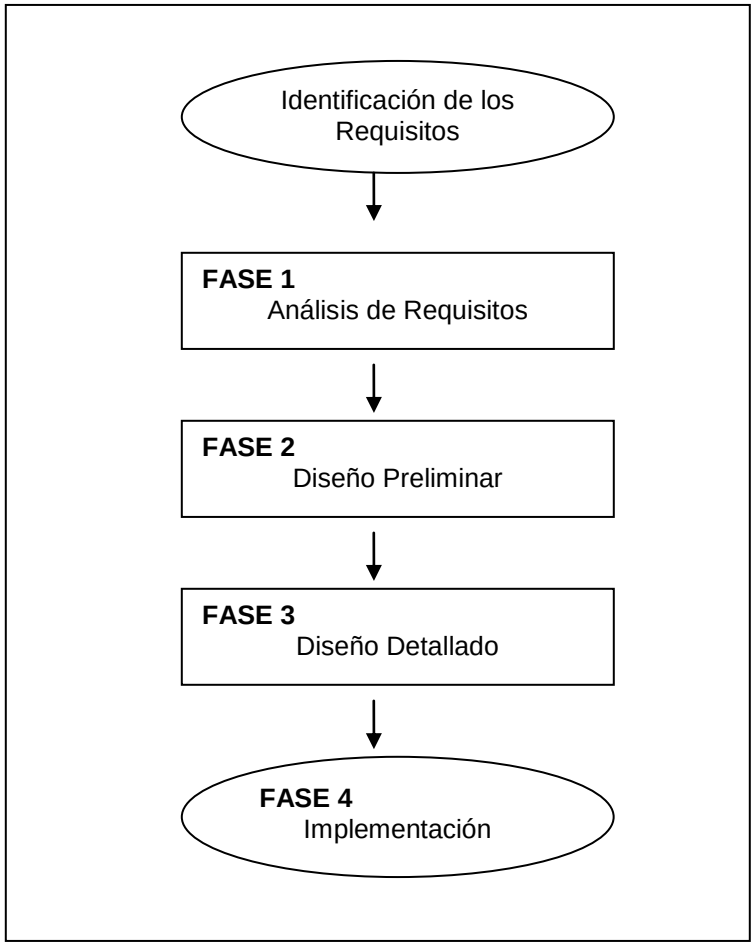


Figura 3.1 Etapas del Ciclo de vida del Software

3.1 APLICACIÓN Y VALIDACIÓN DE LOS ESTÁNDARES DE PRUEBAS EN EL CICLO DE VIDA DEL SOFTWARE

El uso de estándares de pruebas ha permitido un entendimiento más claro, o por lo menos, más consistente para asegurar la calidad en el proceso de desarrollo de software, tratando con suficiencia de prioridad y recursos, de tal forma que los proyectos de software se desarrollen satisfactoriamente.

Todos los Estándares de Pruebas y herramientas tienen un único fin producir software de calidad de acuerdo con los requisitos funcionales establecidos y centrados en dos objetivos fundamentales:

3.1.1 Mantener el control de un proceso

El propósito es controlar cuantitativamente la ejecución del proceso de desarrollo software obteniendo resultados más eficientes.

3.1.2 Eliminar las posibles causas de error

Esto permitirá edificar una base sólida que pueda apoyar no solo las necesidades actuales sino también las futuras, implementando mejores prácticas que nos ayudan a construir software, perfeccionando significativamente la flexibilidad y escalabilidad de los sistemas.

La Metodología MGVM comprende cuatro fases para el Proceso de desarrollo de Software:

- **Análisis de Requisitos.-** Se enfoca a las necesidades del Usuario.
- **Diseño Preliminar.-** Determina el grado óptimo de la Arquitectura del software a desarrollar.
- **Diseño Detallado:-** Obtiene un buen funcionamiento operacional de acuerdo a sus actividades.
- **Implementación.-** Es el cumplimiento de todas las expectativas del proyecto.

Cada una de estas etapas es desarrollada mediante la iteración de Estándares de Pruebas establecidos en función del ciclo de vida bajo estrictas disciplinas.

3.2 FASE DE ANÁLISIS DE REQUISITOS

El propósito es desarrollar una metodología que defina la calidad en términos de satisfacer necesidades explícitas, declaradas, especificadas e implícitas dentro del proceso de desarrollo.

El Estándar más indicado para aplicarlo en la Fase de Requisitos es el Estándar **ISO/IEC 14598** trata esencialmente de normalizar el proceso de evaluación del producto software con el propósito de establecer los requisitos del sistema de una forma mas explicita y detallada, prescribiendo, recomendando métodos, procedimientos específicos para las actividades, representando un marco conceptual genérico personalizable a las distintas necesidades del proceso de evaluación de requisitos del software.

Prioriza el análisis de las características correspondientes a la situación actual, el dominio que poseerá el sistema, analizando tanto el impacto positivo como el negativo para mejorar su eficiencia, contribuyendo a mejorar la calidad del producto, siendo su meta alcanzar la calidad necesaria y suficiente para cumplir con las necesidades reales de los usuarios.

Describe cómo obtener los requisitos, cómo organizarlos, cómo perseguir las decisiones, cómo captar, documentarlos y comunicar las tareas con las que deberá cumplir el sistema de software a desarrollar.
Para obtener los requisitos se debe mantener reuniones con los usuarios, conocer el dominio del problema, el contexto organizacional y operacional.

3.2.1 Ventajas de la aplicación del Estándar ISO/IEC 14598 en la Fase de Requisitos.

El estándar ISO 14598 permite realizar una secuencia de tareas para captar las necesidades o requisitos del usuario, aportando a obtener una Fase de Requisitos de calidad, las ventajas de su aplicación son:

- Permite emplear de manera sistemática y practica métodos apropiados de recopilación, desarrollo y evaluación de requisitos.

- Admite realizar un análisis de los usuarios que van a interactuar con el sistema para determinar los dominios antes de empezar con el desarrollo.

- Establece un marco de procedimientos organizativos que llevan a conseguir una alta calidad para determinar los casos de uso y el diagrama de clases.

- Identifica las condiciones y restricciones del formato o contenido de los datos de entrada.

Objetivos

- Conocer la situación actual y el dominio del problema
- Analizar las necesidades del cliente.
- Identificar los requisitos funcionales que deberá cumplir el sistema software a desarrollar.
- Determinar los actores del sistema a desarrollar.

3.2.2 Tareas Recomendadas

Las tareas recomendadas para obtener los requisitos descritos en esta metodología son las siguientes:

Tarea 1: Visión General

Es hacer explícito el conocimiento oculto sobre el origen de la empresa dando a conocer su estructura organizacional y funcional con la finalidad de definir sus niveles jerárquicos, determinando cuales son los objetivos que persiste.

Tarea 2: Planificación y Gestión

Asume implícitamente que durante la realización de las actividades, surge el efecto de planificar metódicamente la visión y misión de la empresa, conociendo cual será el dominio, definiendo los acrónimos y abreviaturas gestionando el interés del mando de su Gerente.

Tarea 3: Proceso para Evaluadores

Esta actividad, es considerada como la más importante, tiene como objetivos principales detectar los requisitos orientados por el cliente, estableciendo las bases del diseño del sistema, conociendo la situación actual de la empresa, manteniendo la interacción entre clientes con los ingenieros de requisitos, conociendo el dominio del problema, descubriendo las necesidades reales de clientes y usuarios.

El principal objetivo de esta actividad es el método de recolección de datos que consiste en mirar detenidamente las particularidades de una investigación de campo dividiéndose en cuatro tipos:

- **Observación Directa**

La observación directa se caracteriza por la interrelación que se da entre el investigador y los sujetos de los cuales se habrán de obtener ciertos datos. En ocasiones este mismo investigador adopta un papel en el contexto social para obtener información más “fidedigna” que si lo hiciera desde fuera.

- **Observación Indirecta**

La observación indirecta consiste en tomar datos del sujeto(s) a medida que los hechos se suscitan ante los ojos del observador, quien desde luego podría tener algún entrenamiento a propósito de esa actividad.

- **Observación por Entrevista**

La entrevista es la práctica que permite al investigador obtener información de primera mano. La entrevista se puede llevar a cabo en forma directa, por vía telefónica, enviando cuestionarios por correo o en sesiones grupales.

- **Entrevista Personal**

Esta puede definirse como una entrevista cara a cara, donde el entrevistador (Ingeniero en Sistemas) pregunta al entrevistado y recibe de éste las respuestas pertinentes a las hipótesis de la investigación. Las preguntas y su secuenciación marcarán el grado de estructuración del cuestionario, objeto de la entrevista.

- **Entrevista por teléfono**

Para lograr éxito en este tipo de encuesta, se recomienda iniciar con una breve conversación, la cual cumple con la misma función de la carta o la introducción en la entrevista personal. Las preguntas deben derivarse de un cuestionario estructurado, preferentemente cuando éste haya sido revisado por más de un integrante del equipo de investigación, pero sobre todo que lleve el visto bueno del responsable del proyecto.

- **Observación por Encuesta**

La encuesta es un proceso interrogativo que finca su valor científico en las reglas de su procedimiento, se le utiliza para conocer lo que opina la gente sobre una situación o problema que lo involucra, y puesto que la única manera de saberlo, es preguntándoselo, luego entonces se procede a encuestar a quienes involucra, pero cuando se trata de una población muy numerosa, sólo se le aplica este a un subconjunto, y aquí lo importante está en saber elegir a las personas que serán encuestadas para que toda la población esté representada en la muestra; otro punto a considerar y tratar cuidadosamente, son las preguntas que se les hará.

El tipo de información que se recoge por este medio por general, corresponde a: opiniones, actitudes y creencias, etc.; por lo tanto, se trata de un sondeo de opinión.

Tarea 4: Establecer los objetivos del sistema

Su meta es Identificar los objetivos que se esperan alcanzar mediante el software a desarrollar, entendiendo las verdaderas necesidades del cliente antes de describir qué pasos deben cumplirse.

En términos generales podemos definir los objetivos basándonos en la definición clara del problema, diferenciando entre las cosas como se perciben y las cosas como se desean Aquí vemos la importancia que tiene una buena comunicación entre desarrolladores y clientes; de esta comunicación con el cliente depende que entendamos sus necesidades determinando el desempeño y funcionalidad de la aplicación.

Tarea 5: Identificar las necesidades del usuario

Su objetivo es lograr un producto que satisfaga las necesidades del usuario, con un enfoque interactivo en el desarrollo de software y una continua retroalimentación desde la perspectiva del usuario

Es la tarea que describe y especifica las necesidades de los usuarios, estos requerimientos especificados, deberán ser usados como razonamientos para el proceso de desarrollo de software.

Tarea 6: Determinar los Requisitos de la Aplicación

Los requisitos de la aplicación son los que deberá cumplir el sistema a desarrollar a partir de la información obtenida en las tareas que desarrolla, inicialmente se partirán de conceptos generales para posteriormente ir detallándolos hasta obtener todos los datos relevantes.

Es la clave para lograr un proceso de especificación de requisitos exitoso y por consiguiente asegura en gran medida la calidad del software, ofreciendo un método satisfactorio para las especificaciones textuales de los requisitos identificados.

Tarea 7: Documentar la especificación de los Requisitos Funcionales

Los requisitos funcionales especifican los servicios que debe proporcionar la aplicación, revisando previamente los conflictos que pueden ser identificados entre ellos.

Inicialmente se identificarán los actores que interactuarán con el sistema, es decir aquellas personas que serán el origen de la información que producirá el sistema a desarrollar y que forman su entorno.
A continuación se identificarán los casos de uso asociados a los actores, los pasos de cada caso de uso con las posibles excepciones hasta definir todas las situaciones posibles.

Tarea 8: Definición de los Requisitos no Funcionales

Los requerimientos no funcionales describen aspectos del sistema visibles para el usuario que no están relacionados en forma directa con el comportamiento funcional del sistema. Los requerimientos no funcionales abarcan varios asuntos, desde la apariencia de la interfaz de usuario hasta los requerimientos de tiempo de respuesta y las cuestiones de seguridades.

Para identificar los Requisitos no Funcionales se debe considerar:

- ✓ Requerimientos de Desempeño

Especifican las restricciones de tiempo que debe observar la aplicación en tiempo real, donde cada una de las acciones deben terminar dentro de limites de tiempo especificados.

Para su determinación considera las siguientes características de desempeño:
¿Qué tan sensible debe ser el sistema? ¿Qué usuarios concurrentes debe soportar?.

- ✓ Requerimientos de Confiabilidad y Disponibilidad

Especifican la confiabilidad en términos cuantificados. Este tipo de requerimiento reconoce que es poco probable que las aplicaciones sean perfectas, por lo que circunscribe su grado de imperfección.
Para su determinación se debe de considerar: ¿Qué tan confiable, disponible y robusto debe de ser el sistema? ¿Cuál es la participación del cliente en la valoración de la calidad del sistema o en el proceso de desarrollo?.

- ✓ Requerimientos de Manejo de Errores

Explica como debe responder la aplicación a los errores de su entorno. Para el manejo de errores y condiciones externas se debe considerar:
¿Cómo debe manejar el sistema las excepciones? ¿Qué excepciones debe manejar el sistema? ¿Cuál es el peor ambiente en el que se desempeña el sistema? ¿Hay requerimientos de seguridad en el sistema?.

- ✓ Requerimientos de Interfaz

Describen el formato con el que la aplicación se comunica con su entorno. Para este tipo de requisito se debe de considerar: ¿Qué tipo de interfaz debe proporcionar el sistema? ¿Cuál es el nivel de experiencia de los usuarios?.

- ✓ Restricciones

Describen los límites o condiciones para diseñar o implementar la aplicación, especificando las condiciones que el cliente impone al proyecto.
Para determinar las restricciones se debe de considerar: ¿Cuáles son las restricciones en los recursos consumidos por el sistema? ¿Cuál es el alcance previsto de cambios futuros? ¿Quién realizara los cambios?.
Una vez que se haya descrito todos los requerimientos no funcionales se les asigna prioridad de acuerdo con su importancia.

3.2.2.1 Identificación de los Actores

Los actores representan entidades externas que interactúan con el sistema. Un actor puede ser un sistema humano o uno externo que intercambia información, teniendo interacciones especificas para definir las clases de funcionalidad.
Cuando se trata de identificar a los actores, los desarrolladores pueden hacer las siguientes preguntas:

- ✓ ¿Cuales grupos de usuarios son apoyados por el sistema para realizar su trabajo?
- ✓ ¿Cuáles grupos de usuarios ejecutan las funciones principales del sistema?
- ✓ ¿Cuáles grupos de usuarios realizan funciones secundarias, como el mantenimiento y la administración?
- ✓ ¿Interactuará el sistema con algún sistema de hardware o software externo?

Una vez identificados los actores, el siguiente paso es determinar la funcionalidad a la que tiene acceso cada actor. Esta información puede extraerse usando escenarios y formalizando el uso de los casos de uso.

3.2.2.2 Identificación de Escenarios

Un escenario es la descripción narrativa de lo que la gente hace y experimenta cuando trata de utilizar sistemas. Es una descripción concreta, enfocada e informal de una sola característica del sistema desde el punto de vista de un solo actor, mejorando la obtención de requerimientos para proporcionar una herramienta comprensible con facilidad para usuarios y clientes.
Para describir un escenario se debe de emplear el siguiente formato:

Nombre del Escenario	Descripción de la situación actual.
Instancia de Actores participantes	Nombre de la persona: Cargo
Flujo de Eventos	Narra la acción que el actor esta realizando hasta cuando se reporta la situación actual.

Para identificar los escenarios se pueden usar las siguientes preguntas:

- ✓ ¿Cuáles son las tareas que el actor realiza al interactuar con el sistema?
- ✓ ¿Qué información consulta el actor? ¿Quién crea esos datos? ¿Se los puede modificar o eliminar? ¿Quién lo hace?
- ✓ ¿Qué cambios externos necesita informar el actor al sistema? ¿Con cuánta frecuencia? ¿ Cuando?.
- ✓ ¿Cuáles eventos necesita el actor que le informe el sistema?.

El énfasis en la identificación de actores y escenarios es para que los desarrolladores comprendan el dominio de aplicación y definan el sistema correcto. Esto da como resultado una comprensión compartida de los procesos de trabajo del usuario que necesitan ser apoyados y el alcance del sistema. Una vez que los desarrolladores identifican, describen a los actores y escenarios, formalizan los escenarios hacia casos de uso.

3.2.2.3 Identificar los Casos de Uso

Un caso de uso especifica todos los escenarios posibles para una parte de funcionalidad dada y es iniciado por un actor.
Un caso de uso es detallado por el siguiente formato:

Nombre del Caso de Uso	Nombre del reporte de la función
Actor Participante	Iniciado por
	Se comunica con
Condición Inicial	La persona quien activa la función

Flujo de Eventos	Cada una de las actividades que realiza para reportar la función.
Condición de salida	La persona quien recibe la respuesta
Requerimientos especiales	Detalla el recibo de reporte de la función Ejecutada.

Una vez que se enfocan los casos de uso, el sistema obtiene estabilidad al tratarse de asuntos de rastre habilidad y redundancia, consolidando y reorganizando los actores y casos de uso.

3.2.2.4 Validación de los Casos de Uso

Para implantar las pruebas en los casos de uso estos deben ser modelados mediante la automatización y captura de los requisitos funcionales basados en la identificación de los actores y escenarios:

- ✓ Analizar que para definir un actor este debe representar un conjunto coherente de roles que juegan los usuarios de los casos de uso, con el sistema.
- ✓ Determinar que los escenarios descubiertos sean precisos, expresados de forma correcta y sin ambigüedad, verificable, trazable y modificable.
- ✓ Revisar que cada caso de uso debe tener un nombre que lo distinga de otros casos de uso.
- ✓ Examinar que el modelado de los Casos de uso este centrado en la secuencia de estados del sistema, proporcionando soporte para el comportamiento lógico de la aplicación.

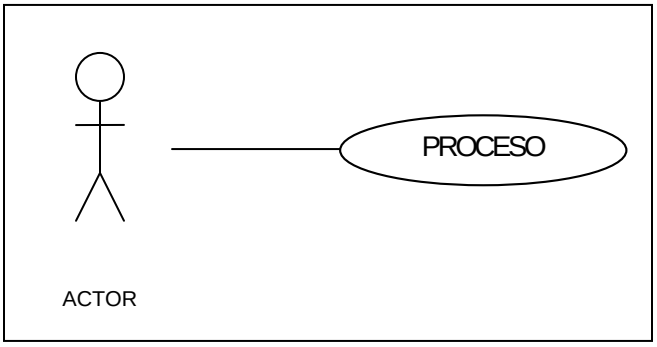


Figura 3.2 Esquema de los Caso de Uso

3.2.2.5 Modelo de Diagramas de Clases

Los diagramas de clases son diagramas de estructura estática que muestran las clases del sistema y sus interrelaciones incluyendo herencia, agregación y asociación.

Son el pilar básico del modelado del sistema, utilizados tanto para mostrar lo que el sistema puede hacer (análisis), como para mostrar cómo puede ser construido (diseño), cumpliendo con las siguientes actividades.

- ✓ Representar y determinar claramente las clases y sus relaciones, la clase se determina por un rectángulo con tres divisiones internas que alojan:

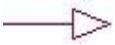
  - **Nombre Clase** Es la unidad básica que encapsula el nombre principal con el que se definirá una clase de otra.
  - **Atributos** Son las características de una Clase definiendo el grado de comunicación y visibilidad de ellos con el entorno.
  - **Operación** Contiene los métodos u operaciones, los cuales son la forma como interactúa el objeto con su entorno.

Nombre Clase
Atributo Atributo: Tipo Atributo: Tipo: Valor Inicial Operación ()

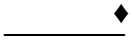
3.2.2.6 Identificar relaciones de Herencia

Interrelacionar dos o más clases, cada una con características y objetivos diferentes, predominando la cardinalidad que no es más que el grado de dependencia:

- **uno o muchos**: 1..* (1..n)
- **0 o muchos**: 0..* (0..n)

✓ Herencia  Indica que a más de poseer sus propios métodos y atributos, poseerá las características y atributos visibles de la Súper Clase.

✓ Asociación:  Es la relación entre clases, permite asociar objetos que colaboran entre si

✓ Agregación:  Es la relación que existe entre dos instancias, simbolizando la asociación que hay entre los componentes y la clase que se compone de ellos.

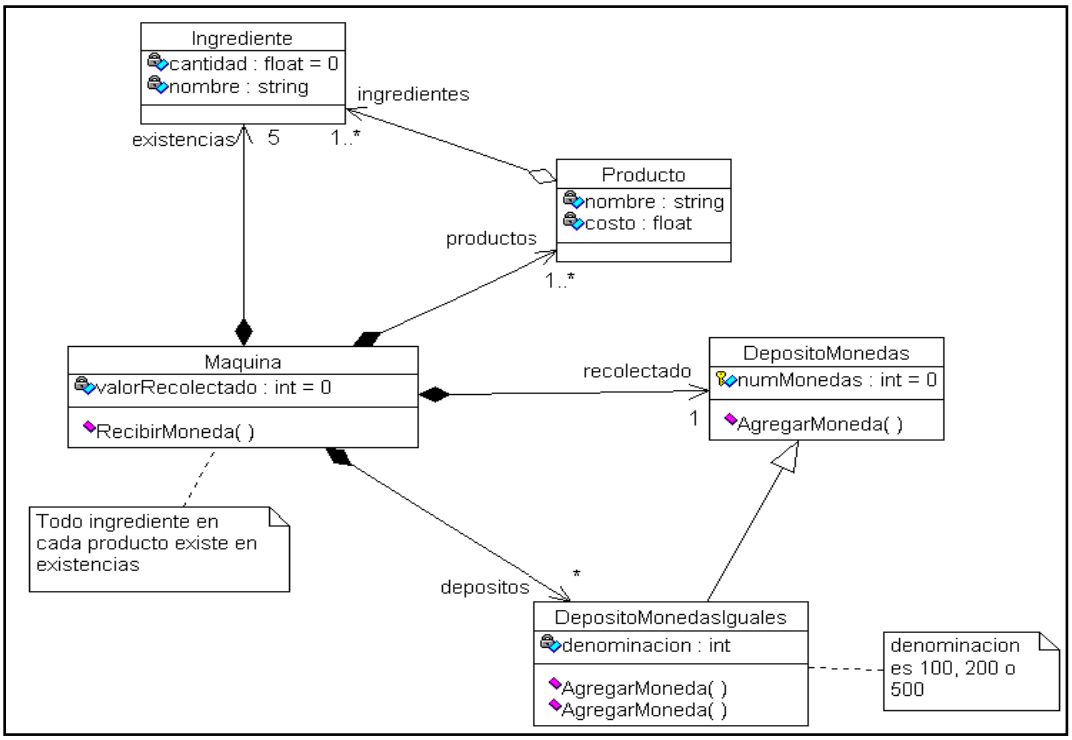


Figura 3.3 Diseño del Diagrama de Clases

3.2.2.7 Validación del Diagrama de clases

Los Diagramas de Clases son notaciones que identifican las clases del dominio, claves y sus relaciones considerando su respectiva validación que permite definir las vistas de herencia, analizando los siguientes puntos:

- ✓ Examina la notación dentro del diagrama de clases, considerando las tres separaciones nombre de la clase, los atributos y la operación.
- ✓ Debe analizar que los atributos que distinguen a una clase puedan almacenar valores para el mismo en cada instancia que genere la clase.
- ✓ Considera que cada clase debe tener un nombre y el tipo de dato que va a recibir. Los atributos se colocan en la segunda parte del rectángulo de la clase, primero se coloca el nombre del atributo, después precedido de dos puntos (:) el tipo de dato que recibirá y en algunos casos se podrá especificar el valor inicial que recibe precedido por un signo de igual (=).

- ✓ Inspeccionar que las operaciones deben ser funciones que puedan realizar las instancias de una clase, mediante ellas se pueden visualizar cuales son las responsabilidades de cada clase dentro del sistema.

Se visualiza si existe un tipo de clase especial llamada clase abstracta, este tipo de clase no genera instancias directas, lo único que hace es heredar a otras clases que pueden generar instancias directas.

3.2.2.8 Diagramas de Secuencia

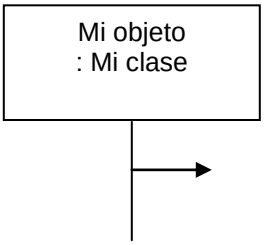
El Diagrama de Secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia se modela para cada caso de uso, el diagrama de secuencia contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario.

Un diagrama de secuencia muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como vectores horizontales. Los mensajes se dibujan cronológicamente desde la parte superior del diagrama a la parte inferior; la distribución horizontal de los objetos es arbitraria.

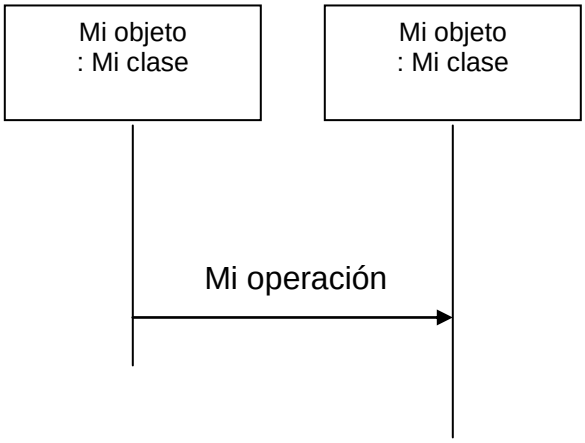
Para describir un Diagrama de Secuencia se debe de considerar:

- ✓ Identificar el caso de uso cuyo diagrama de secuencia se construirá
- ✓ Identificar que entidad inicia el caso de uso
    - El usuario
    - Un objeto de una clase
    - Nombre de la clase
    - Nombre del objeto
- ✓ Dibujar un rectángulo para representar este objeto arriba a la izquierda
- ✓ Dibujar un rectángulo alargado debajo de este para representar la ejecución de una operación.

- ✓ Dibujar una flecha que sale del rectángulo para indicar la funcionalidad convocada.



- ✓ Identificar que entidad maneja la operación iniciada.
    - Un objeto de una clase
    - Nombre de la clase
    - Nombre del objeto
- ✓ Etiquetar la flecha con el nombre de la operación
- ✓ Mostrar el inicio del proceso con un rectángulo alargado
- ✓ Continuar con cada nueva sentencia del caso de uso



3.2.2.9 Validación de los Diagramas de Secuencia

Los Diagramas de Secuencia resaltan la ordenación temporal de los mensajes que se intercambian, muestran la secuencia de mensajes entre objetos durante un

escenario concreto, considerando los siguientes principios para validar los Diagramas de Secuencia:

- ✓ En la parte superior se detallar los objetos que intervienen.
- ✓ La dimensión temporal se indica verticalmente (el tiempo transcurrido).
- ✓ Las líneas verticales indican el periodo de vida de cada objeto.
- ✓ El paso de mensajes se indica con flechas horizontales u oblicuas.
- ✓ La realización de una acción se indica con rectángulos sobre las líneas de actividad del objeto que realiza la acción.

Formalmente el determinar los diagramas de secuencia, permiten trazar eventos de interacción de objetos, utilizados con frecuencia para validar los casos de uso.

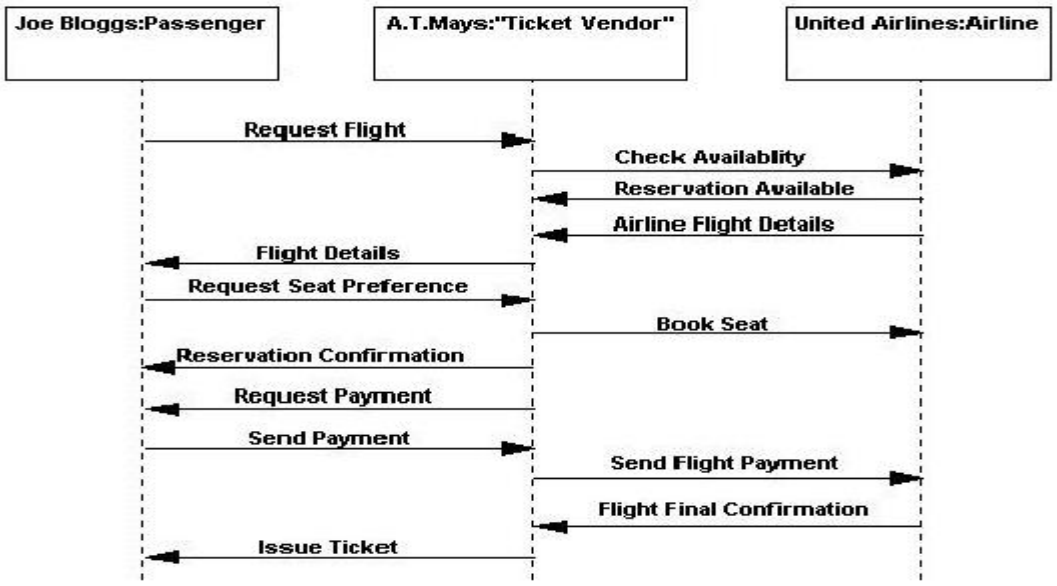


Figura 3.4 Diagrama de Secuencia para un escenario

3.2.2.10 Validación de la Fase de Análisis de Requisitos

Este proceso se centra en la estructura orientada al proceso de desarrollo que determina los requisitos, siendo un modelo que contiene descripciones sobre el

dominio del problema, los objetivos que el sistema debe alcanzar, identificación de los requisitos y la documentación.

El método de verificación y validación se apoya en el uso de técnicas que ayudan en la evaluación facilitando la detección de problemas, el propósito es la revisión de los requisitos definidos en relación a las necesidades establecidas, considerando los siguientes elementos:

- ✓ Explorar la situación actual y las necesidades que deben ser atendidas en función de las características que debe alcanzar el sistema.
- ✓ Evaluar si todos los interesados están siendo considerados en los requisitos, o si las ausencias son justificadas.
- ✓ Confirmar eventualmente las situaciones de conflictos e inconsistencias entre requisitos de la aplicación.
- ✓ Comprobar la existencia de requisitos incompletos, ambiguos o no verificables.
- ✓ Verificar si los requisitos del software contemplan aspectos funcionales y no funcionales.
- ✓ Evidenciar que el sistema tenga mecanismos de claves de entrada, estado de salida y estados esperados.
- ✓ Fortalecer la Documentación de los requisitos, permitido un entendimiento más claro, consistente entre los actores y la persona quien desarrollara la aplicación.

El proceso de evaluación logra resultados más objetivos creando un ambiente de comprobación interactivo que permite el conocimiento absoluto en la importancia de calidad del software.

3.3 FASE DE DISEÑO PRELIMINAR

Es el proceso que determina el diseño efectivo, eficiente de las funciones y contiene la información del análisis de dominio, cumpliendo con las necesidades o funciones específicas, que son requeridas por el usuario.

Un control sobre la calidad en la Fase de Diseño Preliminar, es cumplir con los objetivos establecidos, en cuanto al diseño de la Arquitectura, Costo beneficio y las herramientas técnicas que se aplicarán durante el desarrollo y en el producto final, garantizando el cumplimiento de la funcionalidad global del sistema.

La fase de Diseño Preliminar se basa en el Estándar ISO/IEC 9126, garantizando su desarrollo, empleando pocos recursos utilizados eficientemente, haciendo hincapié en las funciones y en la inexistencia de errores en el tiempo de ejecución, mediante las cuales se evalúa el diseño y desarrollo de la arquitectura, proporcionando un marco de trabajo que especifica los requisitos y evalúa la calidad de la aplicación; entendiéndose por característica de calidad de un producto a un conjunto de propiedades mediante las cuales se evalúa y describe su calidad.

Objetivos

- Identificar soluciones tecnológicas para cada una de las funciones del sistema.
- Asignar recursos materiales para cada una de las funciones.
- Establecer métodos de validación del diseño para la arquitectura e interfaz.
- Ajustar las especificaciones del producto para que el sistema cumpla con los requerimientos identificados durante la fase de análisis.

3.3.1 Ventajas de la aplicación del Estándar ISO/IEC 9126 en la Fase de Diseño Preliminar

El proceso de desarrollo de la Fase de Diseño Preliminar se fundamenta en las características del estándar ISO/IEC 9126 y son:

- ✓ **Funcionalidad**.- Capacidad del producto software para proporcionar funciones que satisfagan las necesidades específicas e implícitas.
- ✓ ***Confiabilidad***.- Capacidad del producto software para mantener un nivel especifico de rendimiento.
- ✓ ***Eficiencia***.- La capacidad del producto software para proporcionar el rendimiento apropiado, relativo a la cantidad de recursos utilizados.
- ✓ ***Usabilidad***.- La capacidad del producto software de ser entendido, aprendido, utilizado y atractivo al usuario.
- ✓ ***Mantenimiento***.- La capacidad del producto software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptación del software a cambios en el entorno, en los requisitos o en las especificaciones funcionales.
- ✓ ***Portabilidad***.- La capacidad del producto software de ser transferido de un entorno a otro.
- ✓ **Reusabilidad.-** Conjunto de atributos que respaldan la posibilidad de que el producto de software se use en otro dominio.

Basado en las características, las ventajas de la aplicación del Estándar ISO/IEC 9126 son:

- Permite la reutilización de los componentes reduciendo el tiempo de desarrollo, mejorando la fiabilidad del producto final y siendo más competitivos.
- Proporciona una retroalimentación para mejorar el diseño de la arquitectura e interfaz evaluando la respuesta.
- Contribuye a mejorar la calidad del producto y la calidad en uso.
- Posee la capacidad de organizar el proceso evaluando periódicamente los casos de uso para determinar hasta qué punto el sistema es eficaz en lograr sus metas.

- Determina especificaciones de seguridad relacionadas con los métodos de operación y mantenimiento.
- Admite la especificación precisa y sin ambigüedad de la arquitectura, partiendo del análisis de los requisitos de usuario.

3.3.2 Tareas Recomendadas

Tarea 1: Identificar la Arquitectura del Software

La especificación clara de la arquitectura para el proceso de desarrollo de software es indispensable para una aplicación, eligiendo la arquitectura más adecuada para satisfacer todas las metas

En el sentido más amplio podríamos estar de acuerdo en que la Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de:

- ✓ Definir los módulos principales

Un modulo es una asignación de trabajo para un programador, incorporando abstracciones funcionales o de datos que controlan y manejan aspectos locales del problema por resolver.

Definir la arquitectura de una aplicación por módulos tiene como meta producir sistemas modulares de programación bien estructurados; la modularizacion permite al diseñador descomponer un sistema en sus unidades funcionales con el fin de imponer un ordenamiento jerárquico en el uso de las funciones, igualmente permite la instrumentación de abstracciones de datos y el desarrollo independiente de subsistemas útiles:

Un módulo debe cumplir con los siguientes criterios:

- Cada módulo de un proceso debe ser claramente definido.
- Cada función del módulo tiene un propósito específico claramente definido.

- Cada función debe manejar no más de una estructura de datos principal del sistema.
- Las funciones deben compartir datos globales en forma selectiva.
- Las funciones deben manejar un tipo abstracto de datos para ser encapsulados.

✓ Definir las responsabilidades que tendrá cada uno de estos módulos

Para definir las responsabilidades que tendrá un modulo, se considera como una entidad definida que debe cumplir las siguientes características:

- Deben contener instrucciones, lógicas de procesos y estructura de datos.
- Pueden ser compilados aparte y almacenados.
- Tienen que ser incluidos dentro de un programa
- Deben los segmentos de un modulo ser utilizados al invocar un nombre con algunos parámetros.
- Pueden los módulos usar a otros módulos.
- Mejoran el desempeño del producto de programación, facilitando la depuración, pruebas, integración, ajustes y la modificación del sistema.

✓ Definir la interacción que existirá entre dichos módulos

    - Control y flujo de datos
    - Secuenciación de la información
    - Protocolos de interacción y comunicación
    - Ubicación en el hardware.

Tarea 2: Realizar un análisis costo – beneficio

La estimación de costos de un producto de programación es una de las más difíciles y erráticas tareas de la Ingeniería de Software; las estimaciones durante la fase de diseño preliminar representan el refinamiento obtenido como resultado de las actividades de trabajo, costos y tiempo de entrega.

Los principales factores que influyen en los costos de una aplicación son:

- ✓ Capacidad del programador
- ✓ Complejidad del producto
- ✓ Tamaño del programa.
- ✓ Tiempo disponible.
- ✓ Confiabilidad requerida.
- ✓ Nivel tecnológico.

La técnica que se emplea para la estimación de costos es el Modelo constructivo de costos o COCOMO (Constructive Cost Model) es un modelo de costos por algoritmos que proporciona los valores nominales de la estimación de meses del programador y del calendario de desarrollo para cada unidad de trabajo, basándose en el número de instrucciones de código fuente entregadas de cada unidad; después se utilizan factores multiplicadores para ajustar la estimación de acuerdo con los atributos del producto, computadora, del personal dedicado y del proyecto.

El Modelo COCOMO se basa en la existencia de tres niveles que ha de aplicarse según el estado en que se encuentre el desarrollo del proyecto. Estos tres niveles son:

- ✓ Modelo básico.

Se utiliza al principio del proyecto, para estimaciones de proyecto rápidas y poco precisas. La información que facilita es una estimación en cuanto al orden de magnitud del esfuerzo. Las ecuaciones que rigen este modelo son:

$$E = a\,(KLOC)b$$

Donde E es el esfuerzo expresado en personas mes, el número de líneas de código estimadas excluyendo comentarios (en miles) viene indicado por KLOC. Los valores a y b son valores constantes que dependen de la clase o “modo” de proyecto que estemos evaluando.

Las ecuaciones que rigen este modelo son:

$$T = c\,E\quad d$$

Donde T es el número de meses estimados para el desarrollo, E es el valor del esfuerzo calculado en la anterior ecuación. Los valores c y d son valores constantes que dependen de la clase o modo de proyecto que estemos evaluando.

Los posibles modos que nos encontramos son:

- ✓ Orgánico (organic): Equipos de trabajo pequeños. Existe un buen conocimiento de la aplicación y del sistema utilizado. Poca influencia de las comunicaciones. Poca innovación técnica y tamaño relativamente pequeño.

  E= 2.4*(KDSI) 1.05 TDEV=2.5*(E) 0.38

- ✓ Semiacoplado (semidetached): Se sitúan en una posición intermedia en cuanto a complejidad y tamaño, entre el modo orgánico y el Integrado. Equipo formado por expertos y principiantes. Se han de satisfacer requisitos no excesivamente estrictos. Por ejemplo aplicaciones bancarias o que impliquen transacciones con bases de datos.

  E=3.0*(KDSI) 1.12 TDEV=2.5*(E) 0.35

- ✓ Integrado (embedded): Sistema hw/sw complejo con influencia clara de la seguridad o tiempo real. Costes de validación muy elevados. Requisitos estrictos e inamovibles. Entorno de gran innovación técnica. El problema a resolver es único y es difícil basarse en experiencias anteriores, puesto que quizá no existan.

  E = 3.6*(KDSI) 1.20 TDEV=2.5*(E) 0.32

La tabla para los valores de las constantes de las ecuaciones de esfuerzo (E) y tiempo (T) en función del “modo” del proyecto es:

Modo	**a**	**b**	**c**	**d**
Orgánico	2.4	1.05	2.5	0.38

Semiacoplado	3.0	1.12	2.5	0.35
Integrado	3.6	1.20	2.5	0.32

- ✓ Modelo Intermedio

Se aplica cuando el proyecto ha sido dividido en subsistemas. En este modelo se han de considerar factores relativos a atributos del producto software. y de recursos (materiales, métodos y personal). Los valores de las constantes también se ven afectados. En este caso la ecuación de esfuerzo que rige este modelo es:

$$E = a\,(KLOC)^{b}\,m(x)$$

Donde E, KLOC, a y b tienen el mismo significado que en caso del modelo básico, aunque el valor de a y b es diferente.

La tabla para los valores de las constantes de las ecuaciones de esfuerzo (E) para este “modo” de proyecto es:

Modo	***a***	***b***
Orgánico	*3.2*	*1.05*
Semiacoplado	*3.0*	*1.12*
Integrado	*2.8*	*1.20*

El valor de m(x) es el peso del factor de coste xj, y cuya expresión matemática es: $m(x) = \Pi\, m(xj)$ El valor de m(x) en el caso del modelo básico tiene como valor fijo la unidad.

Boehm consideró quince factores de coste diferentes, agrupados según el siguiente esquema:

- ✓ Atributos del producto

- Fiabilidad requerida.
- Tamaño de la base de datos.
- Complejidad del producto.

✓ Atributos de los recursos

- El material
- Restricciones del rendimiento en tiempo de ejecución
- Restricciones de memoria
- Inestabilidad de la máquina virtual.

- Tiempo de espera requerido.

✓ Atributos de los recursos

➢ El personal

- Capacidad de análisis.
- Capacidad del ingeniero de software.
- Experiencia en aplicaciones.
- Experiencia con máquina virtual.
- Experiencia con lenguaje de programación.
- Métodos y herramientas
- Práctica de los métodos modernos de programación.
- Utilización de herramientas software.

➢ Tiempo

- Planificación temporal del desarrollo requerida.

Cada factor es valorado por separado en una escala ordinal de seis puntos (muy bajo / bajo / nominal / alta / muy alta / extra alta). A partir de las tablas hechas públicas por Boehm se asigna un valor numérico a cada factor y se aplica la ecuación; el resultado es el factor de ajuste del esfuerzo.

✓ Modelo detallado

Ideado para determinar las estimaciones en cada fase del proyecto. Se basa en dividir el esfuerzo en fases, de forma que para cada una de ellas se obtenga el

factor de coste correspondiente. Finalmente se ha de sumar cada uno de ellos para obtener el global.

Como guía para el uso del modelo de COCOMO, en cualquiera de sus variedades, podemos indicar los siguientes pasos a seguir:

- Identificar el “modo” de desarrollo para el nuevo proyecto.
- Estimar el tamaño del proyecto en KLOC y derivar una predicción del esfuerzo.
- Determinar el valor de los 15 factores de ajuste.
- Calcular y hacer uso de las ecuaciones de estimación del esfuerzo y tiempo de desarrollo.

Señalaremos que este modelo introduce dos diferencias respecto a los anteriores:

- Los factores de coste se aplican a cada fase del proyecto, ya que algunas fases se ven más afectadas que otras por ellos. El modelo detallado proporciona un conjunto de valores para cada factor de coste.

- Se puede aplicar a tres niveles de jerarquía de un producto de software (módulo, sistema y subsistema), agregando resultados según el esquema de descomposición.

Para finalizar indicaremos los aspectos a favor y en contra de la utilización de este modelo:

Pros:

- Transparencia del modelo, así como el acierto de los factores definidos para obtener el factor de ajuste.
- Uno de los modelos más documentados; correcto con referencia a los 63 Proyectos utilizados para su creación.

Contras:

- La selección del modo de desarrollo es extremadamente importante.
- Fuertemente dependiente de los datos históricos de la organización, que no siempre están disponibles.
- No se puede asumir que sea totalmente válido en todos los entornos y proyectos.

FACTORES MULTIPLICADORES DE ESFUERZO EN COCOMO

FACTOR MULTIPLICADOR	INTERVALO DE LOS VALORES
Atributos del producto	
▪ Confiabilidad requerida	0.75 a 1.40
▪ Tamaño de la base de datos	0.94 a 1.16
▪ Complejidad del producto	0.70 a 1.65
Características de la maquina	
▪ Limitantes en el tiempo de ejecución	1.00 a 1.66
▪ Limitaciones en memoria principal	1.00 a 1.56
▪ Volatilidad de la virtualidad en la maquina	0.87 a 1.30
▪ Tiempo de entrega de programas	0.87 a 1.15
Características del personal	
▪ Capacidad de los analistas	1.46 a 0.71
▪ Capacidad de los programadores	1.42 a 0.70
▪ Experiencia en programas de aplicación	1.29 a 0.82
▪ Experiencia en maquinas virtuales	1.21 a 0.90
▪ Experiencia en lenguajes de programación	1.14 a 0.95

Características del proyecto	
▪ Uso de técnicas modernas de programación	1.24 a 0.82
▪ Uso de herramientas de programación	1.24 a 0.83
▪ Tiempo requerido para el desarrollo	1.23 a 1.10

Se utilizan los multiplicadores de esfuerzos para cubrir las siguientes actividades:

- ✓ Abarca desde el diseño hasta las pruebas de aceptación.
- ✓ Incluye los costos de documentación y revisiones.
- ✓ Incluye los costos del gerente del proyecto y del bibliotecario de programas.

La mayor ventaja del modelo es que sirve para tener una visión intuitiva de los factores de costos dentro de una organización; para desarrollar una estimación de costos por medio del COCOMO debe de considerar los siguientes pasos:

1. Identificar todos los subsistemas y los módulos del producto.
2. Estimar el tamaño de cada modulo y calcular el tamaño de cada subsistema y del sistema en total.
3. Especificar los factores multiplicadores de módulos para cada uno; estos son: la complejidad del producto, la capacidad de programación, la experiencia en maquinas virtuales y la experiencia en lenguajes modernos de programación.
4. Calcular el esfuerzo para cada modulo, así como el tiempo de desarrollo; usando las ecuaciones de estimación nominal junto con los factores relevantes a cada modulo.
5. Especificar los once multiplicadores restantes de cada subsistema.
6. Calcula el esfuerzo y el tiempo de desarrollo, estimados para cada subsistema.

7. Del paso 6, calcular el esfuerzo y tiempo de desarrollo totales para el sistema.
8. Efectuar un análisis de sensibilidad sobre la estimación, estableciendo comparaciones para diversos factores.
9. Sumar los otros ingredientes en el costo del desarrollo, como la plantación y el análisis, que no se hayan incluido antes.
10. Comparar la estimación con otra obtenida a partir de la técnica DELFI, identificando y corrigiendo las diferencias en la estimación.

La versión del COCOMO descrita representa un modelo detallado que considera diferencias en los multiplicadores de esfuerzo para la fase de desarrollo, su propósito es realizar un análisis exhaustivo, señalando los factores que influyen en el costo del desarrollo de un producto de programación.

Ejemplo

Estimamos que un proyecto va a tener **32,000** DSI, estime el esfuerzo y la duración para todos los casos.

Solución

Esfuerzo = 3.0*(32) 1.12 = 146 p-m

Tiempo = 2.5*(146) 0.35 = 14 meses

Media de técnicos = 146 /14 = 10

Solución (caso de estudio)

Esfuerzo = 2.4*(3.069) 1.05 = 8 p-m

Tiempo = 2.5*(8) 0.38 = 6 meses

Media de técnicos = 8 / 6 = 1.3

Precio = 8 * 152 * 70 = 85000 €

Tarea 3: Determinar la tecnología necesaria

El determinar la tecnología necesaria para el desarrollo de la aplicación refleja el lenguaje empleado, el equipo como los programas de apoyo, las prácticas y herramientas de programación empleadas.

- ✓ *Hardware* Comprende la abstracción de la maquina se refiere al conjunto de facilidades del equipo y de los paquetes del sistema utilizados durante el proceso de desarrollo, la familiaridad, estabilidad y facilidad de acceso que brinda, influyendo en la productividad del programador.
- ✓ *Software (Lenguaje de Programación)* Se debe de elegir un lenguaje que brinde características adicionales que permitan mejorar la productividad y confiabilidad del producto de programación entre estas características están la verificación de los tipos de datos, abstracción de datos, compilación, manejo de excepciones y de interrupciones así como el mecanismo de concurrencia.
- ✓ *Herramientas de Programación*. Las herramientas de programación van desde las herramientas más elementales como ensambladores y depuradores hasta compiladores, incluyendo editores interactivos de texto y el sistema de la Base de Datos, herramientas de configuración y verificación automática del desarrollo de la aplicación.

3.3.3 Validación de la Fase de Diseño Preliminar

La Fase de Diseño Preliminar posee seis características principales funcionalidad, fiabilidad, facilidad de uso, eficiencia, mantenibilidad y portabilidad, con el propósito de definir un proceso o sistema con suficientes detalles para permitir su desarrollo físico.

Los problemas que podemos encontrar durante la fase de diseño pueden ser solucionados a través de pruebas que se llevaran a cabo a medida que se realiza el proceso de desarrollo, de modo que los errores aparezcan lo antes posible, utilizando notaciones sistemáticas, los pasos a validar son:

- ✓ ***I***dentificar los requerimientos, recopilados durante la investigación, ya que son el apoyo para transformar los datos y arquitectura del software.
- ✓ . Crear un diseño modular con funciones independientes en base a los requisitos de software, las funciones deben tener un propósito especifico.

- ✓ Tener un comportamiento fiable, lo que significa que no se bloquee ni corrompa los datos; debe además realizar todas sus funciones correctamente y de la forma prevista por el usuario.
- ✓ Tener disponibilidad es una característica importante que debe poseer la arquitectura del sistema, no tanto que el sistema sea rápido, sino que complete las tareas en el tiempo previsto.
- ✓ Evitar inconvenientes es importante evaluar el diseño con los usuarios probando el conocimiento y expectativas sobre el rendimiento de las tareas.

- ✓ Proporcionar el acceso personalizado al sistema para los distintos roles de usuario (administrador, usuario, registrador).

- ✓ Implementar la funcionalidad que el sistema debe proporcionar a los distintos usuarios facilitando y simplificando su uso.

- ✓ Poseer la capa lógica de acceso a datos que es el conjunto de componentes que facilitan el acceso a datos.

La gestión efectiva de la validación no solo reduce el esfuerzo necesario para la Fase de Diseño Preliminar, si no que reduce la posibilidad de introducir errores durante el diseño

3.4 FASE DE DISEÑO DETALLADO

El Diseño Detallado es la actividad técnica que sigue a la definición de la arquitectura, su meta es preparar por completo el proyecto para su implementación, comienza con los resultados de la etapa de arquitectura y termina con un bosquejo completo para la etapa de programación

Dentro del proceso de desarrollo de la fase de diseño detallado el Estándar ANSI/STD 829 plantea una tecnología específica, basada en el Plan de Gestión de Software y el ambiente en el que opera, implementado un número importante

de mejores prácticas que nos ayudan a construir la Fase de Diseño Detallado robusta.

La operación principal de esta fase es diseñar el Modelo de la Base de Datos y la Interfaz del software a partir de su especificación asegurando entregar un sistema de calidad. Este estándar establece un conocimiento cuantificable de la calidad planteada, alcanzando los objetivos específicos que implican definir, satisfacer siempre desde la perspectiva del usuario, permitiendo realizar el seguimiento de los requisitos.

Su finalidad se centra en suministrar de modo oportuno toda la información necesaria para adoptar el apoyo relativo a los procesos y la tecnología de la información. Esta fase permitirá perfeccionar la estructura general de las actividades, preparar un plan detallado de ejecución para adoptar una decisión bien fundamentada respecto al Modelo de Base de datos, Interfaz, menús e hipervínculos en general, convirtiéndolo en un sistema integro y confiable.

3.4.1 Ventajas de la aplicación del Estándar ANSI/STD 829 en la Fase de Diseño Detallado

Una de las principales ventajas al aplicar dicho estándar en la Fase de Diseño Detallado es mantener el Control de Calidad en el ámbito de garantizar la calidad en el proceso de desarrollo del sistema, es un desafío el cual propone modelos de prueba adecuados que dirigen la comprobación del proceso de diseño, alcanzando ventajas que evalúan el control y calidad del proceso:

- Permite orientar cada una de las actividades que se ejecutan en la Fase de Diseño Detallado con simplicidad y confiabilidad.

- Proporciona facilidades para realizar el diseño de la Base de Datos en base a los requerimientos del sistema, determinando la facilidad de mantenimiento.
- Permite desarrollar el diseño de la Interfaz analizando el comportamiento, para cumplir con las expectativas del usuario y la aplicación.

- Verifica que el diseño sea completo incluyendo y asegurando el modelo de los casos de uso.
- Asegura que las interfaces del modulo sean claras.
- Evalúa el proceso de Diseño Detallado valorando su efectividad para alcanzar un Diseño con bases sólidas donde prevalezca la calidad e integridad.

Objetivos

- Determinar el diseño e implementación de la Base de Datos, utilizando los requerimientos obtenidos

- Analizar y diseñar la interfaz desde una perspectiva conceptual a una física.
- Minimizar los costes de desarrollo mediante la optimización de recursos consiguiendo calidad y práctica.

3.4.2 Tareas Recomendadas

Esta fase busca elaborar un diseño detallado que muestre como se construirán los subsistemas de datos y el subsistema programado, produciendo el paquete de diseño, el cual contiene todas las especificaciones para la construcción del sistema e implantación considerando las siguientes tareas:

Tarea 1: Realizar el diseño lógico de la Base de Datos

Este proceso de diseño elabora un modelo de datos que representa las entidades, sus atributos y las relaciones existentes entre esas entidades, verificando si el modelo de datos obtenido satisface todos y cada uno de los requisitos detallados.

El diseño de la estructura del modelo de datos, debe considerar las siguientes tareas:

- Analizar los flujos de datos que entran y salen de cada tabla del prototipo del sistema.
- Derivar la estructura de datos contenida en cada tabla, identificando las entidades que representa y los atributos que poseen.
- Establecer las relaciones que existan entre las diferentes entidades y construir el modelo de entidad-relación correspondiente.

Tarea 2: Realizar el diseño físico de la Base de Datos.

Dependiendo del tipo y característica del sistema la base de datos debe estar dispuesta a traducir el modelo de datos a un esquema que describe la estructura lógica de los datos estableciendo métodos de acceso que se utilizaran, en términos de lenguaje de descripción de datos.

Para determinar el diseño físico de la base de datos para una aplicación se debe considerar

- ✓ Analizar el tiempo de respuesta, lógicamente se trata de minimizar el tiempo que tarde en darnos la información solicitada y en almacenar los cambios realizados.
- ✓ Facilidad de manejo de la información.
- ✓ Seguridad de la información (acceso a usuarios autorizados), protección de información, de modificaciones, inclusiones, consulta.

Tarea 3: Diseñar el Modelo de Interfaz

Esta actividad tiene como meta diseñar la Interfaz cuyo énfasis esta en la usuabilidad que va a tener el sistema a desarrollar, el grado de complejidad que

dicha interfaz varia dependiendo del proyecto, pudiendo ir desde solo unas cuantas imágenes de pantalla hasta un esquema de interfaz funcional.

La herramienta que guía el proceso de desarrollo del diseño de la Interfaz es GUI que significa (Grafic User Interface) proporcionando los siguientes pasos para su respectivo diseño:

- ✓ Conocer al usuario

Este paso recomienda entender la naturaleza de los usuarios que pueden existir para interactuar con la aplicación, describiendo las características básicas de los usuarios, ayudándonos a identificar las aptitudes y nivel educativo de los usuarios y factores involucrados.

- ✓ Comprensión de la función de negocios

Este paso permite al diseñador que entienda el propósito de la interfaz de usuario en términos del propósito global de la aplicación, reflejando la secuencia de pantallas de manera en que los usuarios llevan a cabo sus tareas.

Para entender los principios importantes para un buen diseño de pantalla se deben de considerar los siguientes pasos:

- Prever donde es común que el usuario empiece

  Con frecuencia, esquina superior izquierda; colocar ahí el "primer" elemento

- Hacer la navegación lo mas sencilla posible
  - ➢ Alinear elementos parecidos
  - ➢ Agrupar elementos parecidos
  - ➢ Considerar bordes alrededor de elementos parecidos
- Aplicar la jerarquía para hacer énfasis en el orden de importancia
- Aplicar los principios de un visual atractivo

  Balance, simetría, regularidad, que sea predecible

  Sencillez, unidad, proporción, economía
- Proporcionar títulos

- ✓ Entender los principios del buen diseño de pantallas

Enumera algunos elementos importantes del buen diseño de pantalla incluyendo factores que con frecuencia se aplican en el proceso de hacer una interfaz

atractiva dentro de estos principios tenemos que en la esquina superior izquierda se encuentran los elementos mas importantes y se agrupan los elementos similares siendo uno de los principios del buen diseño de pantallas.

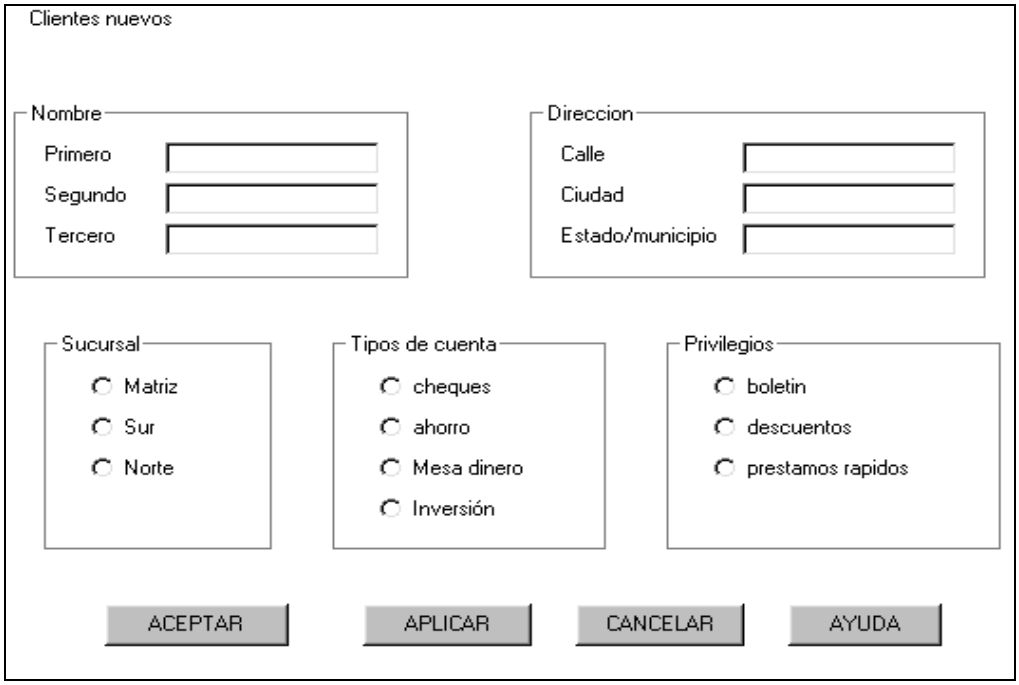


Figura 3.6 Principios del buen Diseño de Pantalla

- ✓ Seleccionar el tipo adecuado de ventana

El propósito de esta tarea es que cada interfaz del sistema pueda cumplirse con más efectividad con uno o dos tipos específicos de ventanas.

Para seleccionar el tipo adecuado de ventana la GUI (Grafic user Interface) enumera cinco propósitos que son:

- Propósito desplegar propiedades de una entidad
- Propósito de obtener información adicional para realizar una tarea o comando especifico.
- Propósito proporcionar información
- Propósito presentar un conjunto de controles
- Propósito ampliar información.

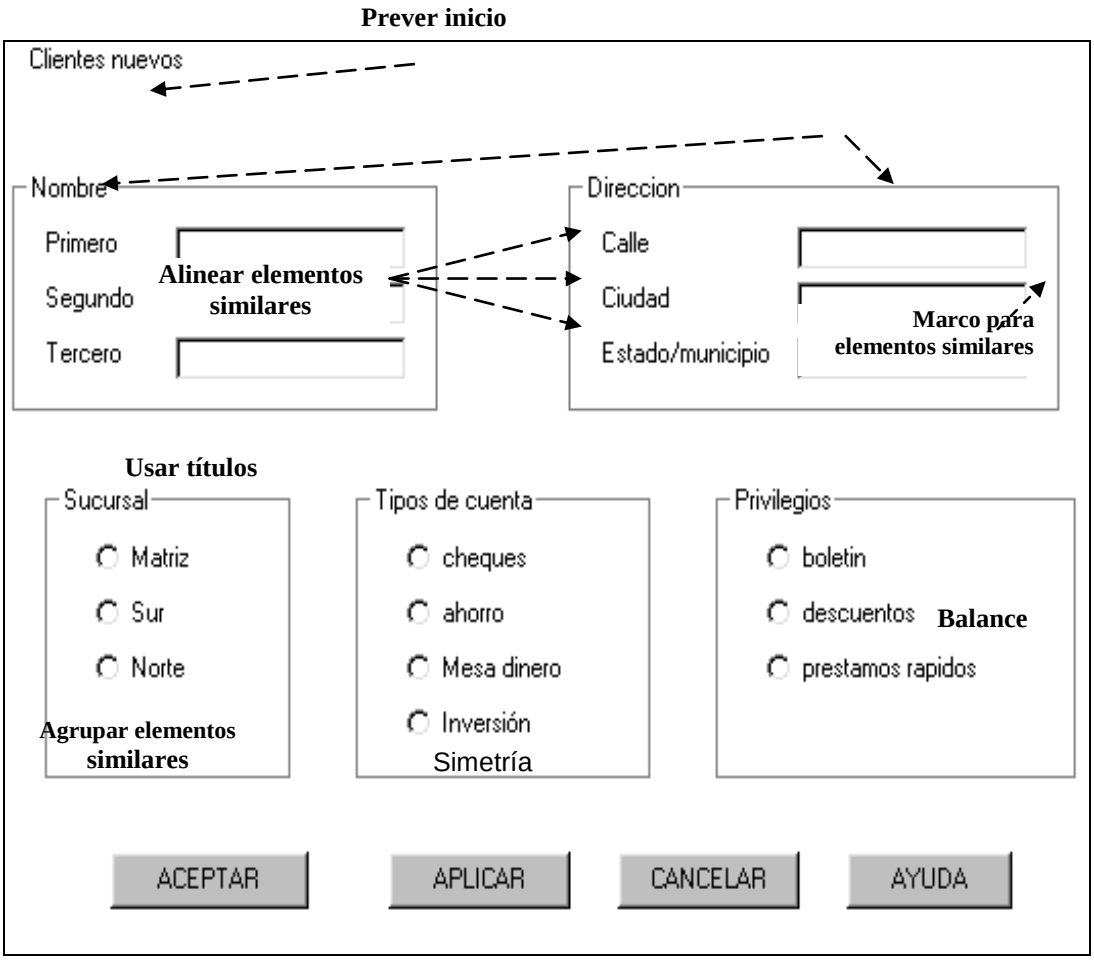


Figura 3.7 Diseño del tipo adecuado de pantalla

- ✓ Desarrollar menús del sistema

Las reglas que se siguen para la creación de los menús principales de la aplicación se basan en los siguientes pasos:

- Proporcionar un menú
- Desplegar todas las alternativas relevantes
- Amoldar la estructura del menú con la estructura de la tarea de la aplicación.
- Minimizar el número de niveles de menú.

- ✓ Seleccionar los controles basados en dispositivos adecuados

Los controles basados en dispositivos son los medios físicos por los que los usuarios comunican sus deseos respecto a la aplicación. Incluyendo tablas de graficas, pantallas de contacto, ratones, micrófonos y teclados.

- ✓ Seleccionar los controles basados en pantalla adecuados

Los controles basados en pantalla son símbolos que aparecen en el monitor mediante los cuales el usuario notifica la aplicación de su entrada o intención, incluyendo iconos, botones, cajas de texto, botones de selección y cuadros de activación.

- ✓ Organizar y distribuir ventanas

Las reglas para colocar ventanas múltiples son las mismas que para el diseño de una ventana individual que incluyen simetría y proporción.

- ✓ Elegir los colores adecuados

Para definir los colores adecuados se debe emplear la aptitud y buen gusto, el color puede resaltar las pantallas convirtiéndose en la complejidad personificada de la aplicación, para el uso de colores el Ingeniero en sistemas debe ser muy conservador y reservado con el uso del color ya que de èl dependerá el grado de funcionalidad, se considera el siguiente proceso:

- Intentar el diseño primero a blanco y negro.
- Cuando exista un propósito claro introducir un color
- Asegurar que la imagen que presenta ayude al usuario.
- Pensar con detenimiento antes de agregar mas colores
- Realizar el análisis conjuntamente con los procesadores de palabras ya que pueden sugerirle como usar los colores.
- Determinar con frecuencia que exista simetría en los colores.

Estos principios intervienen en el diseño de la interfaz del sistema a desarrollar, aportando información a partir del enfoque propuesto conjuntamente con la información relativa a los requisitos.

Tarea 5: Diseñar las pantallas de entrada – salida

Consiste en diseñar la estructura o formato de cada pantalla de entrada de datos al sistema y de salida de información a los usuarios.

La elaboración de una pantalla, bien diseñada, exige una gran dedicación utilizando ventanas, íconos y menús que dependerán de las tareas que se tiene enfrente, enfocándose en los siguientes aspectos:

- ✓ Diseñar pantallas fáciles de aprender a usar, permitiendo a los usuarios hacer su trabajo.
- ✓ Definir pantallas específicamente para la gente que la usará y que va a interactuar con el.
- ✓ Crear un prototipo que consiste en diseñar algo más concreto y actuar con una descripción más detallada para un trabajo más amplio.

Las pantallas dentro de la aplicación deben ser implementadas de una manera sencilla y se tiene un control muy amplio en cuanto al desempeño y presentación.

Tarea 6: Diseñar los reportes.

En esta actividad se diseña aquellos reportes que son básicamente, los listados gráficos y diagramas especificando su estructura, formato y contenido.

Tarea 7: Diseñar la documentación

Esta actividad se ocupa de determinar el formato y contenido de cada uno de los manuales que forman la documentación que no es más que los instructivos del sistema.

3.4.2.1 Validación de la Fase de Diseño Detallado

Este proceso busca verificar que el Diseño Detallado satisfaga los requerimientos del sistema y cumpla con los objetivos del mismo, empleando técnicas, herramientas que siguen un enfoque estructurado basado en un conjunto de actividades que se llevan a cabo en paralelo con el proceso de diseño.

3.4.2.1.1 Validación y mantenimiento de la Base de datos.

Esta actividad permite iniciar o cargar la base de datos con los datos provenientes de fuentes con volumen considerable, los mismos que serán operados y mantenidos por el administrador de la base de datos.

Se realiza una verificación extensiva de las condiciones de error y genera los mensajes correspondientes a los datos en manipulación de la Base de Datos..

- ✓ El diseño conceptual parte de las especificaciones de requisitos de usuario, su esquema es una descripción de alto nivel de la estructura de la base de datos, su objetivo es describir el contenido de información que se necesitará para manejar la información.
- ✓ El diseño lógico parte del esquema conceptual, un esquema lógico es una descripción de la estructura de la base de datos representando el esquema concreto del sistema.
- ✓ El diseño físico parte del esquema lógico y da como resultado un esquema físico. Un esquema físico es una descripción de la implementación de una base de datos en memoria, la estructura de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos.

3.4.2.1.2 Validación del Modelo de Interfaz del Software

Un buen diseño depende de una continúa evaluación de la interfaz a partir de la especificación obtenida en el proceso de análisis, de acuerdo con el entorno

tecnológico definido, teniendo en cuenta los siguientes criterios que permiten validar la Interfaz:

- ✓ Los perfiles definen la funcionalidad en la interfaz, su objetivo es obtener los requerimientos de los usuarios y plasmarlos en documentación textual que puede incluir bocetos de pantallas.
- ✓ Se debe comprobar que en los formularios, menús y ventanas de la aplicación solo deberán estar activadas las opciones que pueden ser utilizadas y las demás en un momento determinado deben de encontrarse desactivadas.
- ✓ El Formato de la interfaz debe ser individual con adaptaciones oportunas, teniendo en cuenta los requisitos de rendimiento, seguridad improvisando un botón para ejecutar la acción y otro para cancelarla dependiendo de la funcionalidad.
- ✓ Analizar en detalle la navegación entre ventanas y la información precisa para la ejecución de cada diálogo, identificando las relaciones de dependencia entre los datos para establecer la secuencia de presentación más apropiada.
- ✓ Realizar el diseño de los mensajes de error, mensajes de aviso o advertencia que genera el sistema en función del tipo de acción realizada por el usuario en el contexto del diálogo, así como las facilidades de ayuda que proporciona la interfaz durante la interacción con el sistema.
- ✓ Proporcionar sólo la información solicitada para entender la salida, de forma que enmascaren la información que realmente ha solicitado el usuario.
- ✓ Seguir un orden lógico, la información se debe pedir y presentar con una secuencia lógica de los datos.
- ✓ Considerar que el usuario se puede equivocar en múltiples ocasiones y debe permitirse la recuperación de dichos errores de la manera más sencilla posible.

- ✓ Crear asistentes o tutóriales, es conveniente introducir un mecanismo de ayuda que informe al usuario sobre la forma en la que tiene que realizar su trabajo.

- ✓ Si el proceso que se está realizando es largo, desplegar un mensaje para que el usuario no piense que el sistema se ha detenido emplear barras de progreso, iconos con relojes.

Si la Fase de Diseño Detallado cumple de forma satisfactoria con estos niveles mencionados anteriormente, se procede a ejecutar la Fase de Implementación.

3.5 FASE DE IMPLEMENTACIÓN

Es la fase de desarrollo en la que se implementa el sistema, es decir se crea el código correspondiente al resultado de la fase de diseño detallado, siguiendo los esquemas y la arquitectura escogida.

La calidad hoy en día, es uno de los principales objetivos perseguidos para asegurar el éxito de los procesos, entendida como el grado de satisfacción de los requisitos y expectativas planteadas, en la fase inicial de análisis de requisitos. En el caso de abordar el desarrollo de la Fase de Implementación se basa en el Estándar ISO/IEC 9126 su tecnología juega un papel clave en lograr mejores utilidades y reduce costos.

Este estándar permite tareas de verificación proporcionando reglas y métodos de medición prácticos de las características de calidad, usabilidad directamente se relaciona con el esfuerzo necesario para desarrollar la implementación del sistema de forma que el usuario percibe el producto terminado.

Proporciona no solo soporte al Ingeniero de sistemas en el proceso de implementación, sino también la manera como se ofrece la información garantizando al usuario la calidad de lo producido. Es el instrumento fundamental para que pueda precisar la calidad del software que desarrolla incluyendo la justificación de las características y sub-características consideradas como indispensables en esta etapa.

Es conocida también como fase de codificación, que hace posible que el sistema finalmente implementado cumpla con las especificaciones establecidas en la fase de análisis de requisitos y responda al diseño del sistema descrito en la fase anterior. Habitualmente esta fase se centra en las tareas y actividades que se requieren para implementar el sistema en el entorno de producción, el punto de entrada para esta fase es la determinación de que el sistema está listo para implementarse en producción y que se ha obtenido la aprobación.

3.5.1 Ventajas de la Aplicación del Estándar ISO/IEC 9126 en la Fase de Implementación

Es importante destacar que este estándar es una guía o requerimiento condicional que definen la usabilidad, efectividad, eficiencia y satisfacción de la aplicación alcanzando objetivos específicos en un contexto de uso específico**.**

La implantación de un sistema de calidad viene fundamentada por los principios del estándar de pruebas ISO/IEC 9126 manteniendo la consistencia de la Fase de Implementación estableciendo ventajas claras y descriptivas como son:

- Proveer al Ingeniero de sistemas de una guía sobre cómo controlar el desarrollo, mantenimiento del software, cómo debe evolucionar hacia una cultura y administración excelente.
- Usar herramientas de verificación conforme se programe detectando automáticamente errores de nomenclatura.
- Orientar al proceso de Implementación del sistema a satisfacer las necesidades y expectativas de los destinatarios.
- Permitir mejorar permanente todo lo que el sistema pretende alcanzar sobre la base de empaquetamiento y distribución.
- Garantizar la calidad de los procesos internos como medida, para alcanzar la eficacia del producto.
- Guiar las estrategias del proceso de implementación, determinando la razón del proceso actual e identificando los problemas más críticos para la calidad y el mejoramiento del proceso.

Objetivos

- Transferir la actualización de software aprobada y cualquier contramedida en su entorno de producción.
- Satisfacer los requerimientos especificados en el diseño detallado.
- Admitir herramientas que ofrecen seleccionar el estilo de código, evitando errores de nomenclatura.
- Construir el código fuente efectuando pruebas en el transcurso de su desarrollo.
- Planificar las tareas individuales de implementación en cuanto al empaquetamiento y distribución de la aplicación.
- Examinar el código fuente para descubrir y eliminar posibles vulnerabilidades de seguridad, constituyendo un paso fundamental a la seguridad del software durante el proceso de desarrollo.

3.5.2 Tareas Recomendadas

Esta fase determina el lenguaje de programación a utilizar para la implementación del proyecto, precisa los sistemas de intercomunicación de procesos garantizando la usabilidad y accesibilidad del sistema final.

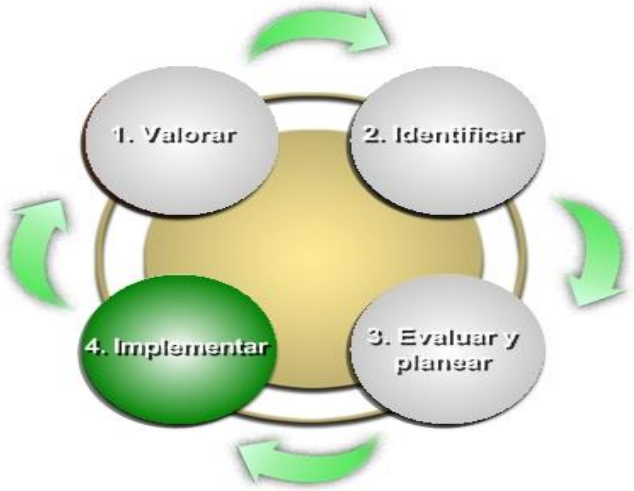


Figura 3.8 Diagrama de tareas que sigue la Fase de Implementación

Para mejorar el proceso de desarrollo de la Fase de Implementación, parte de un enfoque de tareas cuyo propósito es mejorar de forma continua la eficiencia y efectividad del sistema a implementar.

Tarea 1: Desarrollar el Prototipo del software

Es el diseño que traduce los requisitos de software a un conjunto de representaciones basadas en lenguajes que describen la estructura de la base de datos, la arquitectura, el procedimiento algorítmico y las características de la interfaz.

Se centra en las tareas y actividades que se requieren para implementar el software en el entorno de producción, requiriendo de las siguientes tareas adicionales para implementarlo:

- Confirmar el desarrollo del diseño detallado que será implementado.
- Preparar la medición del tiempo dedicado a:
  - Revisión del Diseño detallado
  - Codificación
  - Revisión del código

- Compilación y reparación de defectos de sintaxis
- Pruebas de Unidades
- Reparación de los defectos encontrados en las pruebas

Tarea 2: Codificación

Es la representación del diseño traducido a un lenguaje de programación convencional, dando como resultado instrucciones ejecutables en la computadora.
Su meta real dentro de la Ingeniería de Software es crear el código correcto, es decir justo el apropiado para los requerimientos, verificando la sintaxis y generando un código objetivo.
Por lo tanto el programador esta en el deber de cumplir con los siguientes procesos esenciales:

- Planear la estructura y el diseño residual para el código
  - Note las precondiciones y poscondiciones
  - Registre el tiempo dedicado
- Auto inspeccioné su diseño o estructura
  - Observe tiempo dedicado, tipos de defectos, fuente (etapa) y severidad
- Teclee su código
  - No compile todavía
  - Intente los métodos enumerados a continuación
  - Aplique los estándares requeridos
  - Codifique de manera que la verificación sea sencilla
- Auto inspeccioné el código; no compile todavía
  - Asegure que su código realiza el trabajo requerido
  - Registre tiempo dedicado, defectos encontrados, tipo, fuente y severidad
  - Inspección de código
- Compile su código
  - Repare los defectos de sintaxis

- Registre tiempo dedicado, tiempo de defectos, severidad y líneas de código.

- Pruebe su código
  - Aplique los métodos de prueba de unidad.

La ejecución de estas tareas durante el desarrollo permite incluir un código basado en pruebas y verificación, asegurándose que cumple con sus funciones.

3.5.2.1 Definir Estándares de Programación

El uso de Estándares de Programación mejora la disciplina, lo legible y lo portátil de un programa.
Un aspecto muy importante para un Ingeniero en Sistemas es definir el estilo de programación que este utilizara, eligiendo aquel que se ajuste más al estilo de programación. Un buen estándar de programación generalmente considerará los siguientes factores:

- ✓ **Factor nemotécnico**

Para que el programador pueda recordar el nombre de una variable fácilmente debe usar convenciones que determinen como se bebe declarar el nombre de una variable o constante.

Normalmente las variables se declaran al principio de los bloques de algoritmos, o subprogramas, también se pueden declarar variables fuera de los subprogramas, estando así visibles y accesibles por todos los algoritmos que se desarrollen después sin necesidad de declararlas de nuevo.

Una variable es un espacio que reservamos en la memoria para almacenar un dato. Cada vez que queramos reservar un trocito de memoria para poder almacenar un dato tendremos que indicárselo al intérprete que biene hacer el lenguaje en el cual programas.
Para declarar una variable se debe:

- Saber de que tipo queremos declararla.

- Elegir la zona de declaración correcta
- Declarar finalmente el nombre que debemos dar a la variable es muy simple, en pseudocódigo, basta con poner primero el nombre el tipo utilizando letras minúsculas y después una lista de las variables que se van a declarar con ese tipo.

Para nuestro caso las variables se las definirá en relación del intérprete de PHP comenzando el nombre por un símbolo del dólar.

Ejemplo:

$variable = “Hemos reservado un trozo de memoria\n”;

- Declara dos variables naturales 0..max

  Z m, n;
- Declara dos variables enteras -..0..+

  R x, y;
- Declara una variable para valor lógico T,F

  N k = 33;
- Declara e inicializa

  R z = 2.0, v = 1.0E-9;

Aparentemente las constantes y las variables son lo mismo. Sin embargo, utilizaremos constantes cuando queramos almacenar un dato en la memoria que NO vaya a cambiar durante la ejecución del programa. En teoría se utilizan las constantes para definir datos en la memoria porque ocupan menos memoria que las variables. Al contrario que las variables, las constantes no llevan el símbolo $ delante.

Las constantes las declaramos con la función define();y por orden escribiremos todas las constantes en mayúsculas, así te será mas fácil distinguirlas, y lo haremos de la siguiente manera: define(“nombre_variable”,valor);o

define(“nombre_variable”,”valor”); , dependiendo si el valor es string o si es un numero, pero para aclarar tus posibles dudas te lo explicare con

Ejemplo

```
< ?php cafecit
define ("CONSTANTE", "Hola Mundo");
printf (CONSTANTE);
?> haslo
```

CONSTANTES

- Declarar una constante de cuantos metros son una milla (467m)

  define(“MILLA”,1467);//aquí definimos una constante numérica

- Declarar una constante del nombre de nuestro sitio

  define(“SITIO”,”www.tjnix.com);//aquí definimos una constante de tipo string.

- ✓ **Factor sugestivo**

El factor sugestivo permite que otros programadores puedan leer y entender rápidamente nuestro código, por lo que debe de existir legibilidad del código

Debemos poner énfasis especial en la legibilidad del código, en cualquiera que sea el proyecto, los miembros encargados de programar deberán pasar mucho mas tiempo escribiendo, leyendo y revisando el código fuente, de esta forma permitirán que el programador pueda en el menor tiempo tratar de entender "que es lo que tal o cual bloque de código hace". Podemos hacer este trabajo mucho más fácil siguiendo las convenciones de nomenclatura de los estándares y a la

vez haciendo nuestro código legible y bien documentado. A continuación les daré algunas pautas de cómo puede dar mayor legibilidad a su código fuente:

- Esto tiene que ver con los comentarios explicatorios y aclaratorios que establecemos en nuestro código para futura referencia. Muchas veces podemos olvidar la finalidad de un proceso con complicada algoritmia.

- Los comentarios nos ayudan a recordar los puntos claves de cada parte de nuestro código (sobre todo cuando ha pasado un tiempo largo de haberlo codificado). Además sirve como que los demás miembros del equipo de desarrollo puedan entender también dichos bloques de código.

- Podemos establecer una convención de documentación que puede darse de muchas formas. A continuación una forma sencilla y practica de comentar nuestro código.

Ejemplo

```
'Created by: Helkyn Coello
'Date of creation: 18/08/2004
'Function description: Descripción de lo que hace la función
'Parameters description: descripción de la finalidad de cada parámetro
Function Process1 (par1 as integer, par2 as integer) as Integer
   code goes here
   ...
   ...
End Function
```

Tarea 3: Prueba del software

Una vez que el software ha sido implementado en una forma ejecutable por la maquina, debe ser probado para descubrir los defectos que puedan existir, en la función, en la lógica y en la implementación.
En el entorno que se desarrolla la codificación de la aplicación se debe aplicar dos tipos de pruebas como son:

3.5.2.2 Pruebas de Unidad

Se requiere un enfoque sistemático para las pruebas el primer aspecto es identificar que unidades deben probarse y quien lo hará. Es sencillo establecer que cada parte del trabajo debe probarse; sin embargo esto tiene muy poco significado por que se asigna solo una cantidad finita de recursos, duración y personas – hora a la etapa de pruebas. Así la meta es detectar cuantos errores sea posible con los recursos disponibles
El plan ideal y la ejecución de las pruebas de unidad debe realizarlo alguien diferente al desarrollador y de hecho, en ocasiones se hace en la parte de aseguramiento de la calidad de la organización.
Una forma de aplicar las pruebas de unidad es considerando el siguiente proceso:

- Decida la filosofía de las pruebas de unidades

Es el primer aspecto que se debe de identificar que unidades deben probarse y quien lo hará considerando las siguientes preguntas:

  - ¿es responsable un ingeniero individual (común)?
  - ¿diseñadas, realizadas y revisadas por otros?

- Decida que / donde / como documentar.

La documentación de las pruebas de unidad es consistente en los procedimientos de prueba, los datos de entrada, el código que ejecuta las pruebas y los datos de salida considerando las siguientes preguntas:

  - ¿conjunto de documentos personales individuales común?
  - ¿Cómo / cuando incorporar otros tipos de pruebas?
  - ¿Se incorporan documentos formales?

- ¿Se usan utilidades de herramientas / pruebas?

- Determine el alcance de las pruebas de unidad

Dado que es imposible probar todo, debe definirse el alcance de las pruebas con cuidado determinando cuando debe de terminar el proceso de pruebas y el enfoque de sus prioridades considerando:

- No solo realice pruebas hasta que el tiempo expire
- Asigne prioridades para que las pruebas importantes se hagan

- Decida como y donde obtener los datos para las pruebas

Se han mencionado entradas legales para estos datos requiriendo cierta generación aleatoria de datos de entrada para las pruebas analizando el código fuente y detectando las fronteras y ramificaciones de los datos.

- Estime los recursos requeridos

Se debe de identificar las personas, mes y la duración requerida para realizar las pruebas de unidad, la fuente mas importante de esta estimación consiste en los datos considerando:

- Usar datos históricos si están disponibles

- Registre tiempo, cuenta de defectos, tipo y fuente

Los ingenieros que participan determinan la forma exacta en la que registraran el tiempo dedicado a las pruebas de unidades, los datos obtenidos se usan para evaluar el estado de la aplicación y pronosticar la calidad que tendrá el producto y su fecha de terminación.

Las Pruebas de unidad comprueban que módulos concretos cumplan las funcionalidades esperadas. Todo el código debe tener sus correspondientes pruebas de unidad antes de ser liberado o publicado.

Estas pruebas se hacen de acuerdo con el usuario facilitando la aportación de opiniones sobre como se están cubriendo sus necesidades permitiendo conocer el grado de funcionamiento del proyecto completo.

La aplicación de las Pruebas de Unidad determinan ventajas que ayudan a construir un sistema integro entre las que podemos describir:

- ✓ Permite mantener con facilidad el código funcional y utilizable.
- ✓ Evita los problemas de compilación que causan grandes retrasos en la integración del código.
- ✓ Integra el proceso de compilación detectando inmediatamente cuando se introduce un error.

3.5.2.3 Pruebas del Sistema

El objetivo de las Pruebas del sistema es comprobar la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica.

Verifica que cada elemento encaje de forma adecuada y que alcance la funcionalidad y el rendimiento del sistema total. La prueba del sistema está constituida por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema. Algunas de estas pruebas son:

- ✓ Prueba de validación: Proporciona una seguridad final de que el software satisface todos los requerimientos funcionales y de rendimiento. Además, valida los requerimientos establecidos comparándolos con el sistema que ha sido construido. Durante la validación se usan exclusivamente técnicas de prueba de caja negra.

- ✓ Prueba de recuperación: Al encontrar un fallo del software verifica que la recuperación se lleva a cabo apropiadamente.
- ✓ Prueba de seguridad: Verificar los mecanismos de protección.
- ✓ Prueba de resistencia: Enfrenta a los programas a situaciones anormales.
- ✓ Prueba de rendimiento: Prueba el rendimiento del software en tiempo de ejecución.
- ✓ Prueba de instalación: Se centra en asegurar que el sistema software desarrollado se puede instalar en diferentes configuraciones hardware y software y bajo condiciones excepciones, por ejemplo con espacio de disco insuficiente o continuas interrupciones.

En la realización de estas pruebas es importante comprobar la cobertura de los requisitos, dado que su incumplimiento puede comprometer la aceptación del sistema por el equipo de operación responsable de realizar las pruebas de implantación del sistema, que se llevarán a cabo en el proceso implantación y aceptación del sistema.

La aplicación de las pruebas de sistemas se encarga de analizar los resultados de las pruebas del sistema y efectuar su evaluación. Dicha evaluación basada en las pruebas de unidad determinan ventajas que consisten en:

- Comparar los resultados obtenidos con los resultados esperados
- Identificar el origen de cada problema detectado para poder remitirlo a quien proceda, determinar la amplitud de las modificaciones y qué acciones deben llevarse a cabo para resolverlo de forma satisfactoria.
- Indicar si el plan de pruebas debe volver a realizarse total o parcialmente, y si será necesario contemplar nuevos casos de prueba no considerados anteriormente.

Tarea 4: Corrección

Es muy probable que el cliente descubra defectos en el software por lo que debe poseer un mantenimiento adoptivo que consiste en modificarlo antes de implementarlo.

Los desarrolladores se enfrentan de forma constante con que hacer con los datos potencialmente ilegales. Nuestra meta primordial es la prevención de errores y su corrección empleando un proceso bien definido ayudando a corregir errores.

Una forma de implementar la corrección de errores es considerando el siguiente proceso:

- Siga el proceso de desarrollo acordado; inspeccione.
- Considerar introducir clases para encapsular los valores legales de los parámetros.
    - Constructor privado, funciones privadas para crear instancias
    - Detecta muchos errores durante la compilación
- Cuando los requerimientos especifican el manejo de errores, debe Implementarse según se requiere.
    - Use excepciones si se pasa la responsabilidad del manejo de errores
- Para las aplicaciones que nunca deben detenerse, prevea todos los defectos de implementación posibles (como uso de predeterminados o default).
    - Solo si la operación desconocida es mejor que nada (poco Usual)
- Siga una política consistente para verificar los parámetros.
    - Debe apoyarse mas que nada en el buen diseño y proceso de desarrollo

La ejecución durante el desarrollo permitirá incluir un código de pruebas y verificación, verificando las precondiciones de la aplicación.

Tarea 5: Empaquetamiento y Distribución

El empaquetamiento es un programa especial que comprime los archivos de un sistema, es el proceso que contribuye a disminuir el tamaño de la aplicación se inició, cuando no existía suficiente tamaño en los discos duros y disquetes eran pequeños, lo que hace que el tamaño del programa tenga menor significado.

Empaquetar el programa resuelve el problema de protegerlo de virus, brinda funciones adicionales de autodefensa destinadas al respectivo análisis por medio de depuradores, al momento de comprimirlo se notara el cambio de un sólo byte en el fichero ejecutable lo que permitirá al nuevo conjunto de bytes en el fichero empaquetado sea completamente diferente.

La aplicación de la distribución dentro del software desarrollado ayudara en el perfeccionamiento consistente de acuerdo a dos tipos de módulos claramente diferenciados. Por una parte existe el módulo de navegación convencional, el cual permite realizar conexiones a un servidor web y mostrar sus contenidos. Por otra, incluye un módulo de gestión accesible desde un servidor de control remoto, que se inicia automáticamente una vez cargado la aplicación.

Tarea 6: Instalar el Software

Es Importante ya que determina las condiciones con las que se puede instalar el software, requiriendo derechos de administrador para estar disponible automáticamente para todos los perfiles de usuarios.

En nuestro caso detallaremos claramente la secuencia de pasos que se siguen para implementar una aplicación desarrollada en el ambiente web.

Un servidor web no es más que un programa que se ejecuta de forma continua en un ordenador, utilizando términos para referirse al ordenador que lo ejecuta, manteniéndose a la espera de peticiones por parte de un cliente de forma adecuada, utilizando una página web que será mostrada en el navegador o mostrando el mensaje correspondiente si se detectó algún error.

Para poder instalar una aplicación que esta orientada a la web, necesitamos un servidor web con PHP y MySQL, a demás se requiere de las siguientes herramientas:

- DNS dinámicas con No-IP
- Instalar un servidor web Apache
- Instalación de PHP y MySQL
- Acelerando PHP: Zend Optimizer
- Facilitando el uso de MySQL: phpMyAdmin

Antes de proceder debemos crear una base de datos, para lo cual podemos utilizar phpMyAdmin, en la que Wordpress almacenará información como las entradas y comentarios escritos. Crear una nueva base de datos en phpMyAdmin es tan simple como introducir un nombre para la base de datos y pulsar Create (Crear). Vamos a crear una base de datos llamada, por ejemplo, wordpress.

Descargamos la última versión de Wordpress desde su página web y descomprimimos su contenido en el DocumentRoot de Apache. El directorio en el que se encuentran los archivos que sirve Apache.. Renombramos el archivo wp-config-sample.php a wp-config.php y lo editamos para configurar la aplicación.

Modificaremos los valores para el nombre de la base de datos (db_name), nuestro nombre de usuario en MySQL (db_user), nuestra contraseña en MySQL (db_password) y la dirección del PC en el que está ejecutando la base de datos (db_host) en el caso de que no esté instalada en el mismo PC en el que instalamos Wordpress. Los nuevos valores serán wordpress (o el nombre que le hallamos dado a la nueva base de datos), root y la contraseña que introdujimos al instalar MySQL.

Abre el navegador y ejecuta el script install.php, dentro del directorio wp-admin, que a su vez se encuentra en la carpeta en la que instalamos Wordpress. En mi caso, al haber copiado los archivos de wordpress en un directorio de nombre wordpress en DocumentRoot, http://localhost/wordpress/wp-admin/install.php.

Esto lanzará un pequeño wizard en el que se nos preguntará por el nombre de la bitácora y la dirección de correo del escritor. El script pasará entonces a crear las tablas necesarias en la base de datos que creamos anteriormente utilizando el nombre de usuario y la contraseña generada automáticamente. Lo primero que deberás hacer es cambiar la contraseña y el nick desde la pestaña Users de la interfaz de administración de Wordpress y esta concluida la instalación para ser utilizada.

Tarea 7: Proveer asistencia y ayuda a los usuarios

Esta actividad es una de las fortalezas claves para el manejo del sistema, orientando la manipulación interactiva del sistema desarrollado.

Es necesario que una de las ayudas principales para el usuario es la documentación. Es necesario que cada ingeniero conserve documentación de su trabajo actual, esta documentación tendrá un nombre, como archivo de documentación de software o documentación personal de software.

Este documento permite al ingeniero reportar el estado en todo momento y parte de el se convierte en parte del archivo del proyecto incluyendo los siguientes elementos:

- ✓ Código Fuente
- ✓ Notas personales de defectos
    - Tipo de defecto
    - Etapa personal en la que se incluyo
    - Etapa personal en la que se elimino.

Las etapas personales son:

- ✓ Diseño detallado adicional (si se aplica)
- ✓ Código (registro de defectos incluidos o detectados y reparados en el código fuente antes de compilarlo)

Prueba Unitaria

- ✓ Notas de tiempo
  - Tiempo dedicado a diseño detallado adicional, codificación, Compilación y pruebas.
- ✓ Notas de ingeniería
  - Incluye el estado de diseño detallado adicional, si se aplica y Código.
  - Incidentes, aspectos de desarrollo notorios.

La documentación constituye una medida útil perfecta para el desarrollo de la aplicación para llevar un seguimiento del proceso de desarrollo.

3.5.2.4 Validación de la Fase de Implementación

Las aportaciones de evaluar en esta fase son altamente valiosas y recomendables para comprobar la consistencia global del producto justo antes de su puesta en escena. Su objetivo primordial es asegurarse de que se escriba correctamente el código para mitigar las amenazas de manera efectiva.

Los procesos de validación que se aplican en la fase de implementación permiten comprobar si se cumple con las expectativas del usuario previas a las actividades y comprensión de los requisitos apoyándose en las siguientes tareas:

- ✓ Aplicar reglas de codificación evitando que los programadores incluyan errores que puedan producir vulnerabilidades de seguridad, centrándose en el funcionamiento correcto de las características y funciones del software.
- ✓ Emplear herramientas de comprobación de seguridad ofreciendo entradas estructuradas para validar a las interfaces de programación de software,

maximizando la probabilidad de detectar errores que puedan ocasionar vulnerabilidades de seguridad del software.

- ✓ Realizar revisiones del código complementando a examinar el código fuente constituyendo un paso fundamental para eliminar las vulnerabilidades de seguridad del software durante la fase de implementación.
- ✓ Verificar que se haya instalado el hardware y que funcione correctamente.

El diseño de esta propuesta metodología sea planteado con el fin de establecer una metodología con objetivos que permitan evaluar la calidad tanto del proceso de desarrollo como en si de la aplicación, definiendo una serie de procedimientos que deben ser controlados a lo largo del proceso

CAPITULO IV

APLICACIÓN DE LA METODOLOGIA MGCM EN EL PROCESO DE DESARROLLO DE SOFTWARE, IMPLEMENTADA EN EL SISTEMA DE ADMINISTRACION DE LA GRANJA AVICOLA "REGALO DE DIOS "

INTRODUCCION

La aplicación de la Metodología MGCM (Metodología que Garantiza la Calidad y Mantenimiento del Software) provee una visión integral para el proceso de desarrollo de Software, que esta basada en Estándares de Pruebas.

Está metodología esta dirigida a los Encargados de Desarrollar Sistemas, permitiendo controlar el Ciclo de Vida de Desarrollo del Software
Hoy en día los profesionales de Sistemas requieren de la parte procedimental para el desarrollo de sistemas, lo más importante es aplicar la metodología MGCM que permitirá usar correctamente tareas secuenciales y herramientas confiables para la garantía de la calidad y mantenimiento del Software, detallando la fase de Análisis de Requisitos, Diseño Preliminar, Diseño Detallado y la Fase de Implementación que determina la visión integra de todo el proceso de desarrollo.

Objetivo General

Garantizar que la metodología MGCM es aplicable para el desarrollo de sistemas de gestión y sistemas Web, logrando comprobar en el sistema On Line de Administración para la Granja Avícola Regalo de Dios.

Objetivos Específicos:

- Conocer e identificar los requisitos básicos necesarios (opciones) para cada proceso

- Diseñar la arquitectura del sistema en base a las necesidades requeridas por los usuarios.

- Diseñar la Base de Datos e interfaz del Sistema de Administración de la Granja Avícola Regalo de Dios.

- Implementar el Sistema de Administración, que permita el control de las operaciones de compra y venta que realiza la empresa.

Propósito

El Propósito de implementar una aplicación basada en la Metodología MGCM (Metodología que Garantiza la Calidad y Mantenimiento del Software) es demostrar la objetividad que posee, en cuanto al control interno y externo del proceso de desarrollo, brindando la información necesaria de acuerdo a las necesidades y actividades de desempeño de los usuarios.

Alcance

La metodología MGCM “Metodología que Garantiza la Calidad y Mantenimiento del Software” describe en si un proceso de desarrollo de software que garantiza la eficacia del Sistema para la gestión de la Granja Avícola Regalo de Dios, que se ocupa directamente del control de las compras, ventas, control de stop de productos (huevos de consumo, aves de postura) con datos reales y en línea.

4.1 FASE DE ANÁLISIS DE REQUISITOS

Determina las necesidades del cliente de forma clara y precisa, definiendo todas sus funcionalidades y restricciones de lo que se quiere lograr.

4.1.1 Visión General

La Explotación Avícola en la Provincia de Cotopaxi crece constantemente, por lo tanto se han ido incrementando las granjas avícolas de aves de postura. Cabe mencionar que en el campo de la avicultura existen empresas que se dedican a la reproducción y distribución de pollitas bebé de diferente razas entre las cuales tenemos: LOHMAN, SHAWER, WRARREN, ISABROWN.

Con el crecimiento de la actividad avícola, se forja el ideal de la creación de un plantel avícola que ayudará a abastecer la demanda del mercado, dicho plantel avícola lleva por nombre Granja Avícola " Regalo de Dios", sus propietarios el Ing. Gustavo Villacís S. y la Sra. Lourdes Mora de Villacís, fue creada en el año de 1989 siendo esta una organización de carácter privado.

La Empresa Avícola " Regalo de Dios " inició sus actividades con 3.500 aves en producción, en la actualidad cuenta con 28.000 aves en producción y 7.000 en crecimiento, así como también cuenta con una planta de producción de alimento balanceado para las mismas.

Esta granja viene explotando de entre las razas mencionadas anteriormente la denominada ISABROWN, por ser una gallina eficiente en la cantidad de huevos que produce, así como el peso promedio de los mismos, resultados que se consiguen con el cumplimiento eficaz de técnicas de manejo desde el período de crianza hasta el período de producción o madurez.

La Granja Avícola "Regalo de Dios" esta formada por dos zonas:

ZONA 1.-Compuesta por 4 galpones de producción, un galpón es la infraestructura en si, provista de las instalaciones necesarias y complementarias para un adecuado alojamiento de las aves, jaulas de hierro galvanizado de tres pisos y tres módulos, bebederos automáticos de copa, y comederos e instalaciones de

energía eléctrica, cada galpón da cabida a 7.000 aves, cada una de las jaulas tiene capacidad para seis aves.

En estos galpones se encuentran aves de diferentes edades, de lo que dependerá su alimentación, existen cuatro tipos de balanceados: Balanceado Inicial, Desarrollo, Prepostura y de Postura con el propósito de obtener una producción máxima; permitiendo producir huevos de diferente tamaño, proceso que es controlado a través del ajuste de consumo de nutrientes, a partir de las 20 a 21 semanas las aves ponen sus primeros huevos que son pequeños, desde las 27 a 28 semanas comienzan a poner huevos medianos, a partir de las 35 semanas comienzan a poner Huevos grandes hasta la culminación de su ciclo de producción que es en un promedio de sus 75 semanas.

ZONA 2.- Un galpón de crianza para pollitas bebé (un día de nacidas) con una capacidad de 8.000 aves, las mismas que reemplazarán a las gallinas de producción una vez que han culminado su ciclo de postura (ciclo que comprende de 10 a 12 meses).
Además cuenta con una bodega en la cual se almacenan y clasifican los huevos, la misma que cumple con las condiciones de higiene, ventilación y temperatura adecuada, dispone también de una planta procesadora de balanceados para el abastecimiento de alimento de las aves.

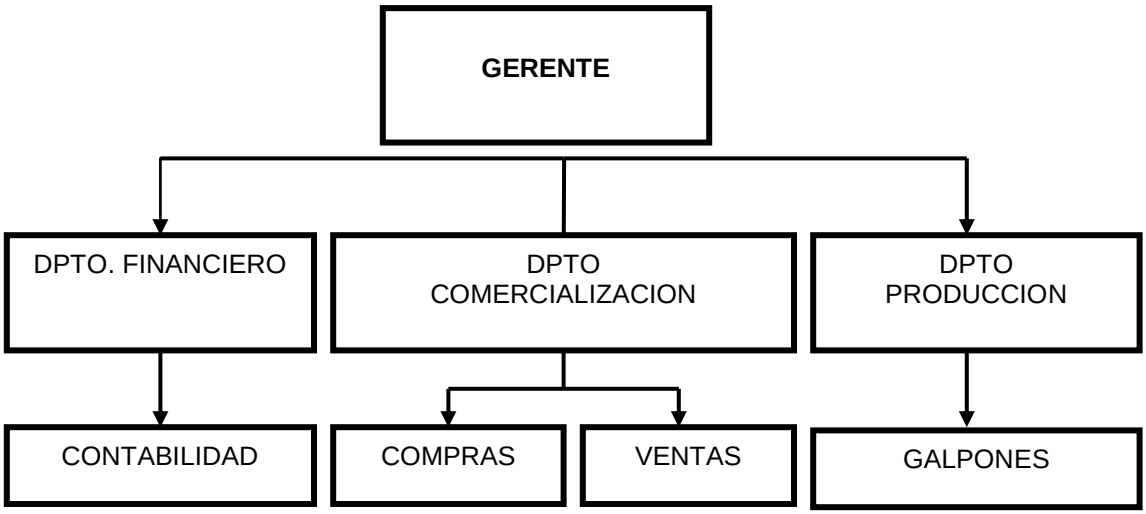
Con respecto al personal que labora en la empresa se tiene tres personas que se desempeñan como galponeros, los mismos que son los encargados de la elaboración del alimento, abastecimiento, manejo y cuidado de las aves ; adicionalmente se cuenta con cuatro personas las que se encargan de recoger y seleccionar la producción diaria de huevos para posteriormente proceder a la comercialización y distribución de los mismos.

ORGANIZACIÓN DE LA EMPRESA

En toda empresa es necesario identificar a través de un gráfico su estructura organizacional, con la finalidad de definir sus niveles jerárquicos, así como también determinar las funciones a cada uno de los niveles con que cuenta la empresa, procurando de esta manera ahorrar los recursos humanos, materiales y económicos.

ORGANIGRAMA ESTRUCTURAL

EMPRESA AVICOLA " REGALO DE DIOS"

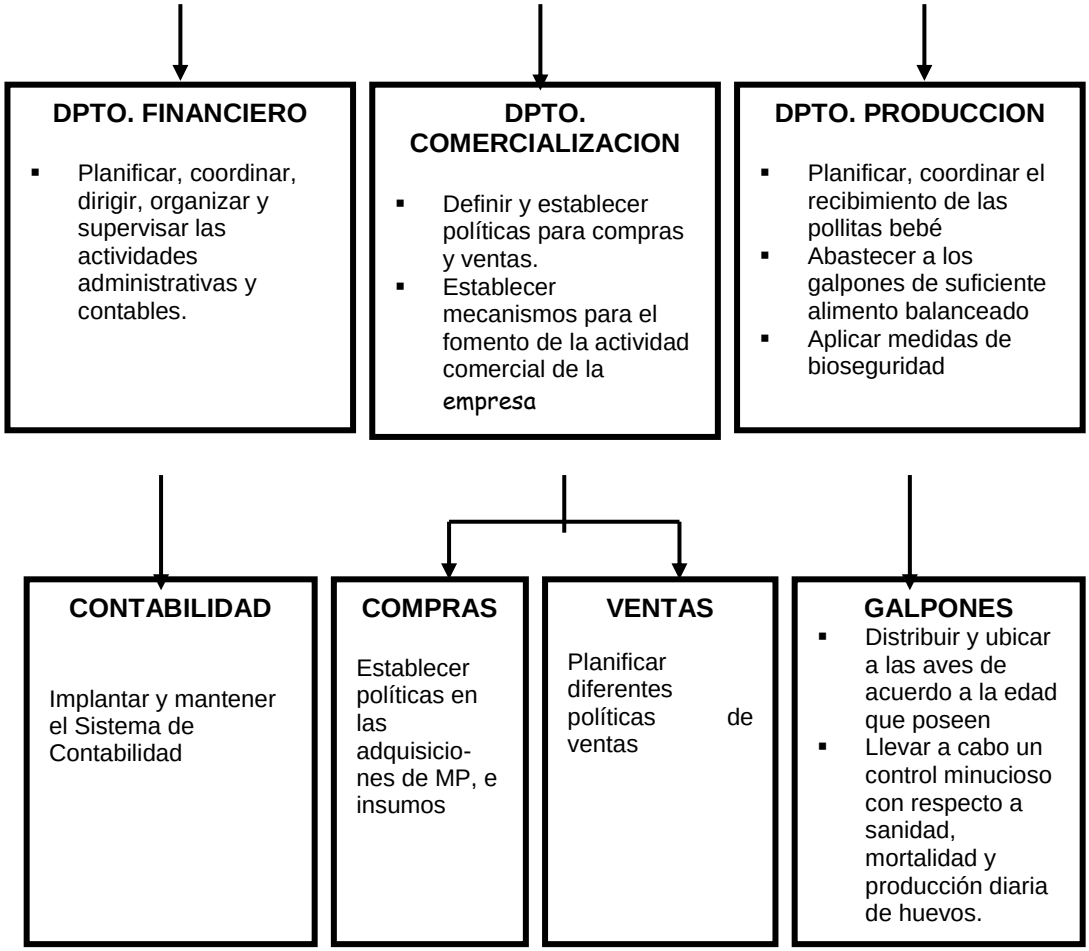


ORGANIGRAMA FUNCIONAL

EMPRESA AVICOLA " REGALO DE DIOS"

GERENTE

- Planificar, organizar, la Administración del Plantel Avícola
- Controlar los bienes económicos del Plantel
- Aprobar y desaprobar a los empleados
- Proveer al Plantel Avícola con los suficientes recursos económicos
- Establecer un ambiente de trabajo.



OBJETIVOS DE LA EMPRESA

- Producir y comercializar huevos de buena calidad, con miras a satisfacer las necesidades del consumidor, abasteciendo los mercados de la región.
- Planificar, controlar, y dirigir las actividades financieras a través de mecanismos que permitan la captación de recursos financieros con la finalidad de propender al mejoramiento de la empresa.
- Establecer canales de comercialización con la finalidad de abastecer el mercado provincial y nacional con miras a satisfacer las necesidades del consumidor.

- ❖ Establecer lineamientos de producción, con la finalidad de alcanzar una mayor productividad y por ende ofrecer un producto de buena calidad para satisfacer la demanda del mercado.

4.1.2 Planificación y Gestión

Localización de la Empresa

La Empresa Avícola " Regalo de Dios "se encuentra ubicada en:

PROVINCIA: Cotopaxi
CANTON: Salcedo
LUGAR: Salache Barbapamba (Al norte del centro poblado de Salcedo 6 Km.)

Los factores ambientales de esta zona (altura, clima, ubicación), han favorecido para el establecimiento de la Granja Avícola de Aves de Postura, dedicada a la producción de huevos para el consumo humano.

Misión

Producir y comercializar huevos de calidad para el consumo humano, satisfaciendo de esta manera las necesidades y requerimientos de nuestros clientes, así como también generar fuentes de trabajo, propender al desarrollo integral de nuestros colaboradores y lograr rentabilidad para nuestra empresa.

Visión

La Empresa Avícola "Regalo de Dios" será siempre una empresa de alta rentabilidad, de reconocido prestigio, confianza y credibilidad, por la calidad de su producto y servicios que ofrece a los sectores que atiende, y por la seguridad y bienestar que brinda a todos sus miembros.

La Empresa Avícola "Regalo de Dios", se caracterizara : por ser una empresa en desarrollo y mejoramiento continuo en el campo de la avicultura; por su flexibilidad para adaptarse a los cambios del entorno y demandas de sus clientes; por el incremento permanente de su productividad; por mantener en su cultura de ventas la filosofía de que "EL CLIENTE ES LO PRIMERO", por el trabajo en equipo y por la preocupación permanente y constante de satisfacer las necesidades alimenticias de nuestro consumidor final.

Definiciones

Gerente	Constituye el representante legal de la empresa
Granja	Finca dedicada a la cría de animales en la que se enseña el cuidado de los animales.
Avícola	Perteneciente o relativo a la avicultura. Personas que se encargan a la cría de aves.
Avicultura	Arte de criar y fomentar la reproducción de las aves y de aprovechar sus productos.
Galpón	Casa grande de una planta que se destina a las aves.
Jaula	Armazón, cerrado o no según los casos, hecho con barras, generalmente de hierro destinada a encerrar animales.
Molino	Maquina de moler, compuesta de una muela, una solera y los mecanismos necesarios para transmitir y regularizar el movimiento productivo por una fuerza motriz (fabricación balanceado).

Lote de Aves	.Es una cierta cantidad de aves que ingresan a un galpón, en el que permanecerán por un cierto periodo de tiempo.
Transferencia	.Es el proceso que se realiza al cumplir las pollitas BB 16 semanas de vida, inmediatamente se les traslada al galpón de producción donde permanecerán hasta cumplir con su ciclo de producción.
Plantel Avícola	Es un lugar destinado a la explotación de aves, con propósito definido ya sea para producción de pollos de carne o de gallinas ponedoras (huevos) para el consumo humano.
Galponeros	Son las personas encargadas de abastecer el balanceado a las aves, aplicando medidas y técnicas de bioseguridad las mismas que permitan alcanzar una sanidad eficiente durante el periodo de crianza y madures de las aves.
Activo Corriente	Representan los productos que la empresa posee a una fecha o periodo determinado.
Activo Corriente Consumible	Representa los productos que serán adquiridos para procesar el alimento (balanceado) de las aves.
Activo Corriente Fabricado	Representa el producto una ves procesado que servirá de alimento indispensable para las aves
Activo Corriente para la Venta	Representa el producto terminado disponible para la venta que oferta la empresa en el mercado (gallinas y huevos).
Balanceado Inicial	Es un alimento rico en proteínas y energía

	que se provee a las aves desde la 1 hasta la 8 semana.
Balanceado de Desarrollo	Es un alimento con menor porcentaje de proteína que se suministra a las aves desde la 8 hasta las 16 semanas.
Balanceado de Prepostura	Es un alimento que contiene un porcentaje de proteína y energía necesario para que las aves inicien con su ciclo de producción, a partir de la semana 16 hasta que las aves alcanzan el 5% de producción.
Balanceado de Postura	Es un alimento con un porcentaje adecuado de proteínas ,energía y calcio que se provee a las aves durante el ciclo de producción

Acrónimos y Abreviaturas

MGCM	Metodología que Garantiza la Calidad y Mantenimiento del Software.
COCOMO	.Es el entorno de la herramienta que permite la Estimación del Esfuerzo del desarrollo del Software.

4.1.3 Proceso para Evaluadores

Para la obtención de requisitos de Software se aplicó dos métodos de recolección de información como es la Observación Directa, conociendo en sí las instalaciones de la empresa determinando los procesos internos que lleva la empresa y la Entrevista Personal en forma directa con el Gerente, Bodeguero, Jefe de

Producción y el Contador de la empresa, obteniendo una información fidedigna, analizando la Situación actual de la Empresa.

El proceso de comercialización de la empresa Avícola “Regalo de Dios” se encuentra estrechamente ligada con dos funciones fundamentales: compras y ventas de productos, los mismos que se detallan a continuación:

Las compras, es el aprovisionamiento de materias primas para la producción de balanceados, de pollitas bebe para crecimiento y de vacunas para ayudar al ciclo de crecimiento de las aves, entendiendo por tal, el conjunto de operaciones que tiene por objeto dotar a la empresa de los stocks adecuados, en las cantidades requeridas y en el tiempo preciso, para que la empresa pueda realizar los procesos de fabricación.

Se entiende por **ventas** al conjunto de operaciones realizadas por la empresa que tiene por objeto poner a disposición de los consumidores y usuarios los productos por ella elaborados como son huevos y gallinas de postura, en las mejores condiciones de calidad, precio y situación.

Las ventas constituyen por consiguiente, el fin de todo el proceso productivo y está basada naturalmente, en la demanda que presente el mercado del producto. La empresa realiza sus ventas al contado cuando se trata de cantidades pequeñas, y a crédito no mayor a 8 días, cuando realiza sus ventas en cantidades al por mayor.

La competencia en el campo avícola esta sujeta a la oferta y demanda del producto (huevos para el consumo), en algunos casos el mercado se satura por cuanto existe una sobreproducción de huevos, originándose una competencia desleal por parte de muchos avicultores.

La empresa no se ha visto mayormente afectada en cuanto a la competencia, puesto que su producto tiene aceptación en el mercado por su calidad, tamaño y peso del huevo; características que le hacen merecedora de la confianza de sus clientes al momento de ser adquirido.

Uno de los problemas que aqueja a la Empresa Avícola " Regalo de Dios " y en especial al Departamento de Contabilidad es la falta de un Sistema de

Administración, puesto que no existe un control adecuado de los productos que se adquieren, procesan y produce para la venta, razón por la cual es prioritario el disponer de un Sistema de Administración de Inventarios y Facturación, el mismo que coadyuve al desarrollo interno y externo de la empresa, con el objeto de obtener la información necesaria para facilitar y mejorar la toma de decisiones, con miras a obtener resultados eficientes.

El Sistema de Administración deberá ser diseñado de acuerdo a las necesidades de la Empresa Avícola “Regalo de Dios”, el cual nos permita conocer el stock de materia prima disponible, así como la cantidad de producto disponible para la venta, con la finalidad de estimar y prever los hechos futuros de la empresa para una mejor toma de decisiones

4.1.4 Objetivos del Sistema

El avance tecnológico, las múltiples transacciones de compra y venta de productos en la empresa Avícola Regalo de Dios, tiene la necesidad de transformar los procesos manuales a un Sistema On Line de Administración de Facturas y Control de Inventarios, que brinde mayor eficiencia, eficacia en los procesos para el servicio al cliente, y además ayude a la toma rápida de decisiones del empresario.

5 Establecer el proceso formal para la administración de la información del sistema en línea de facturación e inventarios aplicando el proceso descrito en la metodología MGCM.

6 Actualizar con eficiencia y en línea los datos de materias primas y productos que maneja la empresa.

7 Informar con anterioridad el abastecimiento de insumos requeridos para todo el proceso de producción interna.

8 Generar informes automáticos de ingresos, egresos, consumos y transferencias de productos y materias primas.

4.1.5 Identificar las Necesidades del Usuario

El propósito fundamental es otorgar apoyo a los usuarios que se desenvuelven en el ámbito de trabajo de dicha organización, brindando solución a los inconvenientes prácticos de contar con la información segura y oportuna en tiempos rápidos de respuesta, logrando eficiencia y eficacia en el proceso y por ende una disminución de costos totales de operación. Las principales necesidades son:

- Manejo de perfiles de usuarios para restringir el manejo de información.
- Administración de facturas de compras materias primas, medicinas y pollitas bebes manteniendo los registros actualizados en tiempo real.
- Administración de facturas de ventas de huevos de diferentes tamaños y gallinas de postura, manteniendo los registros actualizados en tiempo real.
- Control de inventarios de compras y ventas, conociendo la disponibilidad de sus productos e insumos en cada momento.
- Visualización de reportes de inventarios de materia prima y medicinas para una mejor toma de decisiones.
- Generar reportes on line de las transacciones de compra y venta de insumos y productos para un período determinado.
- Garantizar el respaldo de la información de la Base de Datos.
- Administración del consumo de materias primas, utilización de medicinas, nivel de mortalidad y producto vendido por periodo o galpón.

- Soporte para la toma de decisiones en la adquisición de materia prima cuando llegue a sus mínimos, y en la producción cuando llegue a sus máximos.

4.1.6 Determinar los Requisitos de la Aplicación

El sistema debe proporcionar soporte a cada uno de los procesos, del modulo de Facturación, Inventarios y de Perfiles de Usuarios.

- ✓ En cuanto al manejo de perfiles de usuarios, una de las obligaciones que tendrá la aplicación, es la posibilidad de crear diferentes perfiles de usuario. Estos perfiles permiten que varios usuarios utilicen el mismo sistema, cada uno con su propia identidad que comprende un login y password, otorgándoseles permisos de las actividades que pueden realizar en cuanto al manejo de la información.
- ✓ Para garantizar el respaldo de la información almacenada, el sistema guarda la información mensualmente por versiones y las reproduce además en un CD's.
- ✓ La aplicación administrara documentos no negociables como son Facturas, que tienen por objeto detallar la mercadería vendida y comprada, manteniendo los registros de inventarios actualizados en tiempo real.
- ✓ Control de inventarios, este procedimiento proveerá ciertas utilidades para el manejo de inventarios físico de materias primas, de medicinas, de producción, de mortalidad, de balanceado, actualizando la información automáticamente en el Kardex.
- ✓ . La aplicación debe entregar un informe de productos comprados, vendidos, elaborados, y totales de ventas por cliente y vendedor.
- ✓ Administración del consumo de materias primas, nivel de mortalidad, por ciclo de crianza, por lote de galpones o por periodo.
- ✓ La aplicación ayuda a la toma de decisiones a través de la administración de máximos y mínimos, permitiendo obtener un reporte para conocer la cantidad disponible que se tiene por Tipo de Producto.

4.1.6.1 Gestión Administrador del Sistema

Permite realizar las funciones de ingreso, eliminación, modificación, consulta individual o general de los usuarios que manipulan el sistema, recalcando que en la eliminación y modificación se podrán hacer solo en campos que el sistema permita, y no tengan datos relacionados con ellos.

El primer usuario que interactúa con el sistema se le llamara Administrador, es la persona encargada de verificar la configuración inicial del sistema, creación de usuarios, este proceso valida los datos que se ingresen de forma correcta.

El segundo usuario es el Gerente que interactúa con el sistema, es el usuario que visualizara los reportes de ventas, compras, producción, mortalidad, stock en la avícola, además de visualizar los usuarios que se encuentran activos en la red trabajando en el sistema; con la información obtenida ayudara para la toma de decisiones.

El tercer usuario es la Secretaria que interactuara con el sistema para el ingreso, eliminación y actualización de la información de compras, ventas, producción y mortalidad de los productos que dispone en la Granja Avícola Regalo de Dios.

Cada vez que un usuario acceda o salga del sistema se activará el estado en que se encuentra.

4.1.6.2 Gestión Proveedor

Permite realizar las funciones de ingreso, eliminación, modificación, consulta individual o general de proveedor, son empresas o individuos que proporcionan los recursos e insumos requeridos por la empresa; la modificación se podrá hacer solo en aquellos campos que el sistema permita.

En proveedores se registraran todos los datos personales que permitan la identificación de los distribuidores de materias primas y medicinas, cabe recalcar

que se mantendrá datos actualizados, permitiendo automáticamente generar el reporte detallado de los proveedores por productos.

4.1.6.3 Gestión Cliente

Permite realizar las funciones de ingreso, eliminación, modificación, consulta individual o general del cliente, es la persona que adquiere los productos de la avícola; la modificación se podrá hacer solo en aquellos campos que el sistema permita.

En clientes se registrara todos los consumidores que establecen un mismo procedimiento para comprar productos (huevos o gallinas), se mantendrá datos actualizados, permitiendo automáticamente generar el reporte detallado de los clientes por compras de productos, por fechas o periodos de compra, ordenación de clientes de acuerdo al monto de compra por periodo e información detallada del cliente.

4.1.6.4 Gestión Vendedor

Permite realizar las funciones de ingreso, eliminación, modificación, consulta individual o general de vendedor, es la persona que se encarga de vender los productos que la empresa oferta en el mercado, la modificación se podrá hacer solo en aquellos campos que el sistema permita.

En vendedor se registrara todo el personal que esta apto para realizar una venta de los productos que la empresa oferta (huevos y gallinas ponedoras), permitiendo al momento de extender una factura de venta automáticamente registrar la venta realizada, actualizando el registro de mercadería disponible.

4.1.6.5 Gestión Tipos de Productos

Permite realizar las funciones de ingreso, eliminación, modificación, y consulta de los tipos de productos (Activo que puede ser activos creados para la venta, activos consumibles y activos fabricados) o cualquier tipo producto que ingresa y sale de la empresa, estos se utilizarán para la gestión de tipos de Activos Corrientes.

4.1.6.6 Gestión Tipos de Activos Corrientes

Permite realizar las funciones de ingreso, eliminación, modificación, y consulta de los tipos de activos corrientes como son materia prima, vacunas, balanceado, huevos y aves de postura.

Los activos corrientes manipulan la siguiente información: Tipo de Producto, código, nombre, descripción, cantidad máxima, cantidad mínima que mantendrá el control a través de reportes para el manejo de stock, que ayuda a tomar decisiones sobre productos en abundancia o en escasez.

- ✓ M**aterias Primas.-** Para la elaboración de los balanceados la empresa adquiere materia prima que puede ser maíz, soya, pescado, afrecho y otros.

- ✓ **Vacunas.-** Es un medicamento convenientemente preparado que se inocula a las aves para preservarlos de una enfermedad, destinada a prolongar la inmunidad.

- ✓ **Balanceado.-** Es el alimento básico de las aves, la empresa por lo general realiza diferentes clases de alimentos balanceados, esto se debe a que las aves necesitan un balanceado que vaya acorde a la edad que poseen.

- ✓ **Producción (huevos).-** Para mantener un control del proceso de producción de huevos de consumo humano en los respectivos galpones, se debe alcanzar un nivel de producción estable, depende del consumo de alimento balanceado, determinando características procedentes rico en calcio lo que ayuda a determinar el tamaño y peso del huevo dependiendo de ello su costo.

- ✓ **Aves.-** Para mantener un control de las aves en sus respectivos galpones, la empresa Avícola Regalo de Dios considera dos periodos que

son: el periodo de ***crecimiento*** (desde el primer día de nacimiento hasta las 17 semanas de vida) y el de ***producción*** (desde las 18 semanas hasta la semana numero 75).

4.1.6.7 Gestión Transacciones

Permite realizar las funciones de ingreso, eliminación, modificación, y consulta de los tipos de transacciones que se realizo de acuerdo al Activo Corriente como son ingresos, egresos, transferencia y mortalidad.

Para la manipulación de las transacciones tiene la siguiente información: Tipo de Transacción, bodega de origen, bodega de destino, descripción, activo corriente, cantidad, valor, se mantendrá el control a través de kardex permitiendo conocer el detalle respectivo de acuerdo a la transacción que se realizo en determinada fecha y hora.

. ✓ **Ingreso.-** Es la acción de ingresar a un espacio de la infraestructura de la empresa un activo de acuerdo a los tipos de Activos Corrientes (materias primas, vacunas, balanceados, huevos y pollitas bebes).

. ✓ **Egreso.-** Es el proceso permite registrar la salida o consumo interno que tiene la empresa, de acuerdo a los tipos de Activos Corrientes como (materias primas, vacunas y balanceados).

. ✓ **Transferencia.-** Es el proceso que permite registrar el traslado de las aves del galpón de crecimiento al galpón de postura esto ocurre de las 15 a 17 semanas de vida de las aves donde permanecerán durante todo su ciclo de producción. .

. ✓ . **Mortalidad.-** Es el proceso que permite registrar diariamente las aves que se mueren en cada galpón.

4.1.6.8 Gestión de Facturación

Permite realizar las funciones de generar, anular, modificar y consultar la información de las facturas tanto de compras como de ventas, recalcando que en la modificación se podrán hacer solo en aquellos campos que el sistema permita, la anulación de la factura se puede realizar solamente en el proceso de compra mas no en el proceso de venta, considerando que cuando el producto es entregado no se acepta devolución.

Cuando se genera y anula o modifica una factura, los datos de los productos que se detallan serán actualizados en el inventario, el movimiento de los productos se registran en el kardex y quedan actualizadas las cantidades disponibles en bodega.

En facturación se registraran todas las facturas de compras, de ventas y retenciones creadas, cabe recalcar que las retenciones para ser emitidas dependerán del producto adquirido y proveedor, existen productos (pollitas) que no tienen retención, permite también llevar el control de totales de ventas por clientes, por fechas, por producto; las ventas pueden ser a contado o a crédito máximo por 8 días por considerarse un producto de primera necesidad.

4.1.6.9 Gestión Cuentas por Cobrar

Permite realizar reportes de las facturas vendidas a los clientes, en un periodo de fecha determinado, según el tipo de pago, el número de factura, el total de la factura, o facturas que incluyen o no IVA y por cliente.

4.1.6.10 Gestión Cuentas por Pagar

Permite realizar reportes de las facturas compradas a los proveedores, en un periodo de fecha determinado, según el tipo de pago, el número de factura, el total de la factura, o facturas que incluyen o no IVA y por proveedor.

4.1.6.11 Gestión de kardex

Genera reportes individuales de acuerdo al Tipo de Producto(Activos Corrientes Consumibles, Activo Corrientes Fabricados y Activos Corrientes para la Venta) de

acuerdo a ellos se puede elegir el activo corriente y el periodo en el que desea consultar.

Kardex imprimirá todo lo relativo a los Activos Corrientes de acuerdo a su Tipo de Producto de forma individual, detallando los Ingresos, Egresos, Ventas y Saldo que dispone.

4.1.7 Documentar la Especificación de los Requisitos Funcionales

Significativamente la documentación de los requisitos funcionales establece preceptos que admiten mayor flexibilidad describiendo de forma mas explicita, los requisitos que posee la aplicación a desarrollar, con el fin de demostrar la planificación, operación y control eficaz de sus procesos.

El sistema permitirá cumplir con los siguientes requisitos:

4.1.7.1 Gestión Administrador de Usuarios

Req(1) Ingreso de datos generales del usuario: Login, Password, nombre, dirección, teléfono, e-mail, tipo, estado.

Req(2) Verificación del ingreso de datos en los capos obligatorios con presentación de mensajes.

Req(3) El usuario puede tener máximo 3 intentos para ingresar correctamente su login y password, caso contrario saldrá del sistema y regresara a la pantalla del escritorio de su PC.

Req(4) Actualización de campos de Usuario del Sistema notificando con mensajes la actividad realizada.

Req(5) Control de acceso de usuarios, notificando el Estado en que se encuentra.

4.1.7.2 Gestión Proveedor

Req(6) Ingreso de datos generales de proveedor: serán los especificados en el req 1 excepto Login, Password y estado que no deberán ser registrados, al registro deberá incluir los campos de código, RUC, contacto, fax, IVA, IRF y Descripción Empresa.

Req(7) Verificación del ingreso de datos en los capos obligatorios con presentación de mensajes.

Req(8) Eliminación de Proveedor, notificando con mensajes la actividad realizada, cuando no se ha realizado una compra al proveedor.

Req(9) Actualización de campos de proveedor, notificando con mensajes la actividad realizada.

Req(10) Imprimir proveedor en forma particular.

Req(11) Consultar proveedor seleccionando el Activo Corriente registrado en el req 27, identificando que proveedor nos abastece con esa materia prima o vacuna.

4.1.7.3 Gestión Cliente

Req(12) Ingreso de datos generales de cliente: serán los especificados en el req 1 excepto Login, Password, tipo y estado no deberán ser registrados, al registro deberá incluir los campos de código, RUC, dirección de oficina, móvil y descripción.

Req(13) Verificación del ingreso de datos en los capos obligatorios con presentación de mensajes..

Req(14) Eliminación de Cliente, notificando con mensajes la actividad realizada, solo si el cliente no ha realizado compras.

Req(15) Actualización de campos de Cliente, notificando con mensajes la actividad realizada.

Req(16) Imprimir Cliente en forma particular.

4.1.7.4 Gestión Vendedor

Req(17) Ingreso de datos generales de vendedor: serán los especificados en el req 1 excepto Login, Password, tipo y estado no deberán ser registrados, al registro deberá incluir los campos de código, RUC, móvil y descripción

Req(18) Verificación del ingreso de datos en los capos obligatorios con presentación de mensajes.

Req(19) Eliminación de Vendedor, notificando con mensajes la actividad realizada, solo si no tiene registrado ventas realizadas.

Req(20) Actualización de campos de Vendedor, notificando con mensajes la actividad realizada.

Req(21) Imprimir Vendedor en forma particular y en general

4.1.7.5 Gestión Tipo Producto

Req(22) Ingreso de datos de Tipo de Producto con la siguiente información: Código. Nombre, Activo.

Req(23) Verificación del ingreso de datos en los capos obligatorios con presentación de mensajes.

Req(24) Eliminación de Tipo de Producto, notificando con mensajes la actividad realizada.

Req(25) Actualización de campos de Tipo Producto notificando con mensajes la actividad realizada.

Req(26) Imprimir Tipo de Producto en forma particular.

4.1.7.6 Gestión Activo Corriente

Req(27) Ingreso de datos del Activo Corriente con la siguiente información:
Tipo de Producto, Código. Nombre, Descripción, Cantidad Máxima, Cantidad Mínima.

Req(28) Verificación del ingreso de datos en los capos obligatorios con presentación de mensajes.

Req(29) Eliminación de Activo Corriente, notificando con mensajes la actividad realizada.

Req(30) Actualización de campos de Activo Corriente notificando con mensajes la actividad realizada.

Req(31) Consultar el registro de forma general de todos los Activos Corrientes de forma general.

4.1.7.7 Gestión Transacciones

Req(32) Ingreso de datos de una Transacción con la siguiente información: Tipo de Transferencia, bodega de origen, bodega de destino, descripción, activo corriente, cantidad y valor.

Req(33) Verificación del ingreso de datos en los capos obligatorios con presentación de mensajes.

Req(34) Eliminación del detalle de la transacción, notificando con mensajes la actividad realizada antes de ser aprobada.

Req(35) Actualización del detalle de una transacción notificando con mensajes la actividad realizada antes de ser aprobada.

Req(36) Consultar el registro de forma particular de la transferencia que previamente se selecciona.

4.1.7.8 Gestión Factura de Venta

Req(37) Ingreso de datos de factura con la siguiente información: Fecha, #_Factura, nombre cliente, nombre vendedor, forma de pago, Bodega, activo corriente, cantidad, precio unitario, total.

Req(38) El sistema debe calcular el subtotal que es la multiplicación de la cantidad por el precio unitario.

Req(39) Comprobar si al cliente que se le emitirá la factura esta registrado en la base de datos, caso contrario se debe realizar el req 12, notificando con mensajes la actividad realizada.

Req(40) Verificación del ingreso de datos en los campos obligatorios con presentación de mensajes.

Req(41) Actualización de campos de encabezado de factura, notificando con mensajes la actividad realizada.

Req(42) Consultar facturas por fechas, # de Factura, IVA, subtotal y forma de pago.

4.1.7.9 Gestión Factura de Compra

Req(43) Ingreso de datos de factura con la siguiente información: #_Factura, Proveedor, Fecha de Emisión, Fecha de Pago, Descripción, Forma de pago, IVA, IRF, Activo Corriente, cantidad, valor unitario, total.

Req(44) Comprobar si el proveedor esta registrado en la base de datos, caso contrario se debe realizar el requisito Req 6, notificando con mensajes la actividad realizada.

Req(45) Verificación del ingreso de datos en los campos obligatorios con presentación de mensajes.

Req(46) Anulación de Factura de Compra seleccionando la respectiva factura, dando clik en el icono anular.

Req(47) Actualización de campos de factura antes de ser aprobada.

Req(48) Consultar facturas por Activo corriente consumible en un periodo de tiempo.

4.1.7.10 Gestión Cuentas por Cobrar

Req(49) Ingresar para obtener el reporte de Facturas de Venta la siguiente información: Fecha de inicio, fecha final, forma de pago.

Req(50) Consultar Facturas de Ventas por forma de pago (contado o a crédito) en un periodo de tiempo en concordancia con el req 37.

Req(51) Consultar Facturas de Ventas por cliente en un periodo de tiempo determinado.

4.1.7.11 Gestión Cuentas por Pagar

Req(52) Ingresar para obtener el reporte de Facturas de Compra la siguiente información: Fecha de inicio, fecha final, forma de pago.

Req(53) Consultar Facturas de Compra por forma de pago (contado o a crédito) en un periodo de tiempo en concordancia con el req 43.

Req(54) Consultar Facturas de Compra por proveedor en un periodo de tiempo determinado.

4.1.7.12 Gestión Kardex

Req(55) Ingreso de datos de Kardex del sistema con la siguiente información: Tipo de producto, activo corriente, fecha inicial, fecha final.

Req(56) Verificación del ingreso de datos en los capos obligatorios con presentación de mensajes.

Req(57) Imprimir Kardex en forma general por un periodo de tiempo.

Req(58) El sistema generara automáticamente un número de identificación único para cada gestión al momento de su registro siendo este el serial.

4.1.8 Requisitos No funcionales

Para determinar los requisitos no funcionales dentro de nuestra aplicación se analizado el desempeño, disponibilidad, confiabilidad, manejo de errores y las restricciones que tendrá el sistema:

Req(59) Establece una interacción fácil de entenderla, permitiendo que interactúen dos o más usuarios a la vez, su desarrollo esta orientado a la Web por lo que no existirá problemas.

Req(60) El sistema manejara detección y resolución de problemas admitiendo mensajes que indiquen continuamente el error cometido.

Req(61) Presenta recuperabilidad e integridad de los datos proporcionando soporte a la gestión de la información manipulada.

Req(62) Establece una Arquitectura, Diseño integro para conseguir la disponibilidad requerida proporcionando soporte a las funciones que desarrolla el sistema.

Req(63) Facilidad en la gestión de operaciones, determinando seguridad y recuperación de datos, minimizando o eliminando la pérdida y alteración de la información.

Req(64) El sistema para el manejo de errores de funcionamiento y detección de problemas, administra mensajes de error que le permiten conocer el problema que se presenta.

Req(65) Emplea funciones para registros que ayudan fácilmente a implementar un nuevo modulo, continuando con su desarrollo ya sea un

Ingeniero, Analista o Programador de sistemas, gracias a que su programación es estructurada fácil de comprenderla y esta totalmente documentado.

4.1.9 Identificación de los Actores

Para definir los actores de la Granja Avícola Regalo de Dios que interactúan, con los elementos externos se identifican:

- ✓ *Administrador* del Sistema.- Persona encargada del manejo de perfiles de usuario, registrando o actualizando la información necesaria para que el usuario interactué correctamente con el sistema.

- ✓ *Gerente.*- Es la persona que se encarga de inspeccionar todos los reportes de Inventarios de la empresa y de la toma de decisiones según la información obtenida.

- ✓ *Cliente*.- Es el consumidor que establece un mismo procedimiento para comprar una serie de ítems diferentes, relacionado con el nivel de satisfacción que el va a experimentar.

- ✓ *Proveedor*.- Es considerada la organización a quien compramos medicinas, materias primas y pollitas bebe usadas para el proceso de fabricación del producto

- ✓ *Secretaria*.- Persona encargada de la gestión de manejo de la información organizando las compras y ventas de acuerdo a los requerimientos de la empresa.

4.1.10 Identificación de los Escenarios

El significado de la identificación de escenarios se basa en determinar el dominio que tiene la aplicación analizando el proceso de trabajo del usuario con el sistema.

Primer Escenario

Nombre del Escenario Ingresar Usuario	**Descripción de la situación actual.** Se desarrolla un sistema que me permita mantener un control de inventarios y facturación. El usuario (Administrador) (maneja perfiles de usuarios que quieran acceder al sistema para interactuar con el.
Instancia de Actor participante Ingreso Usuario al sistema (login y password).	**Nombre de la persona:** Edwin Reyes **Cargo** Ingeniero
Flujo de Eventos • Usuario (administrador) ingresa al sistema con su Login y Password. • Sistema valida (login y password). • Sistema presenta Menú Principal. • Usuario selecciona opción de Parámetros. • Sistema presenta contenido del (Menú de Parámetros).	**Narra la acción del actor** El Usuario (Administrador) es la persona a la que se le permite manejar perfiles de Usuarios. Admitiendo ingresar los datos en el formulario de insertar nuevo usuario guardando la información para posteriormente proceder a ingresar al sistema con el respectivo login y password del nuevo Usuario registrado.

• Usuario (administrador) selecciona Usuarios del Sistema. • Sistema le presenta el formulario con el respectivo botón para ingresar, editar, eliminar y consultar. • Usuario da click sobre el botón de Insertar. • Sistema presenta el formulario de Ingreso de un nueva usuario a interactuar con el sistema. • Usuario ingresa los datos de acuerdo a los campos solicitados. • Sistema verifica ingreso de datos. • Usuario envía a guardar. • Sistema confirma la operación y registra nuevo usuario.	

Segundo Escenario

Nombre del Escenario	**Descripción de la situación actual.**
Emitir Factura de Venta	El sistema permitirá gestionar Facturas de Venta de acuerdo al producto que se oferta en el mercado (huevos o gallinas) sin mantener un registro de

	control del producto vendido y disponible.
Instancia de Actor participante Secretaria Ingresa (Login y Password), emite Facturas de Venta.	**Nombre de la persona:** Ximena Villacís **Cargo** Ingeniera Comercial
Flujo de Eventos • Secretaria selecciona en menú (Ventas). • Sistema presenta opciones de (menú ventas). • Secretaria selecciona la opción de Factura de Ventas. • Sistema presenta el formulario de Factura de Ventas con los respectivos botones para insertar, editar, eliminar y consultar (Factura de Venta). • Secretaria da click sobre el botón de Insertar. • Sistema presenta el formulario de Encabezado de Factura. • Secretaria Ingresa los datos del respectivo formulario y envía a guardar. • Inmediatamente Sistema presenta el formulario de Detalle de Factura.	**Narra la acción del actor** Usuario (Secretaria) es la persona que se encarga de emitir Facturas de Venta ingresa los datos en el formulario de encabezado de la factura de acuerdo a la información solicitada, ingresa los datos del detalle de factura detallando el Activo Corriente que sale a la venta con su respectiva cantidad y valor unitario.

• Secretaria ingresa los datos de detalle de Factura especificando el activo corriente, cantidad y valor unitario y envía a guardar detalle. • Sistema verifica el ingreso de datos. • Secretaria da click sobre el botón de aprobar Factura de Venta. • Sistema comprueba operación realizada por el usuario y guarda el registro de Factura de Venta.	

4.1.11 Casos de Uso

Una vez que sean definido los actores que interactuaran con el sistema de administración de la empresa Avícola, detallaremos los casos de uso que representan un flujo de eventos que realiza el sistema, en el sentido que describe una serie de interacciones que son necesarios para el desempeño de las actividades de dicha organización y son los siguientes:

4.1.11.1 Gestión Perfiles de Usuario

Caso de Uso: Ingresar Usuario
Actor: Administrador del Sistema
Condición Inicial: Ingresar un usuario en el sistema
Visión general: Si lo que desea el administrador es ingresar un nuevo usuario, deberá seleccionar el icono de Insertar que se encuentra en la parte superior izquierda, inmediatamente se le desplazara el formulario de datos que deberán ser

registrados por el nuevo usuario que pretende interactuar con el sistema, una vez que son ingresados correctamente debe seleccionar la opción de guardar, donde el sistema se encarga de almacenar los datos y confirma la operación.

Condición de salida: Primario real

Requerimientos especiales:

Flujo de Eventos

Actor	Sistema
1.- Este caso de uso comienza cuando el administrador solicita la operación de ingresar usuario.	2.- Presenta botón de Insertar Usuario en la parte superior izquierda.
3.- Da click sobre el botón de Insertar	4.- Presenta formulario de ingreso de usuario.
5.- Ingresa de forma individual cada uno de los datos del formulario de Usuario.	6.-Almacena los datos de usuario y confirma la operación.

***Caso de Uso*:** Eliminar Usuario

***Actor*:** Administrador del Sistema

***Condición Inicial*:** Registro de Usuarios que interactúan con el Sistema.

***Visión general*:** Si lo que desea el administrador es eliminar un Usuario, el sistema presentara el registro de Usuarios, administrador selecciona el serial del Usuario que se pretende eliminar, el sistema presenta en la parte superior el icono de eliminar, Administrador da click sobre el icono, inmediatamente el sistema confirma con mensaje la operación a eliminar registro de Usuario.

Condición de salida: Primario real

***Requerimientos especiales*:**

Flujo de Eventos

Actor	Sistema
1.- Este caso de uso comienza cuando el administrador solicita la operación de eliminar Usuario.	2.- Presenta el registro de Usuarios.
3.- Selecciona el serial de un Usuario de la lista.	4.- Pinta todos los datos del Usuario seleccionado.
5.- Da click en el botón de eliminar	6.- Confirma la operación.

Caso de Uso: Editar Usuario

Actor**:** Administrador del Sistema

Condición Inicial: Registro de Usuarios que se pretenda actualizar su información.

Visión general**:** Si lo que desea el administrador es actualizar los datos de un Usuario, el sistema presentara el registro de Usuarios, el administrador selecciona el serial de uno del registro, sistema sombrea todos los datos del Usuario seleccionado, presenta el icono de Editar en la parte superior izquierda, administrador da click sobre el botón de editar, sistema presenta el formulario del usuario seleccionado permitiendo editar sus campos, administrador edita el campo a cambiar, sistema guarda los cambios y confirma la operación de actualización.

Condición de salida**:** Primario real

Requerimientos especiales**:**

Flujo de Eventos

Actor	Sistema
1.- Este caso de uso comienza cuando el administrador solicita la operación de editar Usuario.	2.- Presenta el registro de Usuario.
3.- Selecciona el serial de un Usuario	4.-.Marca todos los datos del Usuario

de la lista.	seleccionado.
5.- Da click sobre el botón de Editar	6.-Presenta los datos del Usuario seleccionado
5.- Actualiza los datos.	6.- Confirma la operación.

Esquema Casos de Uso

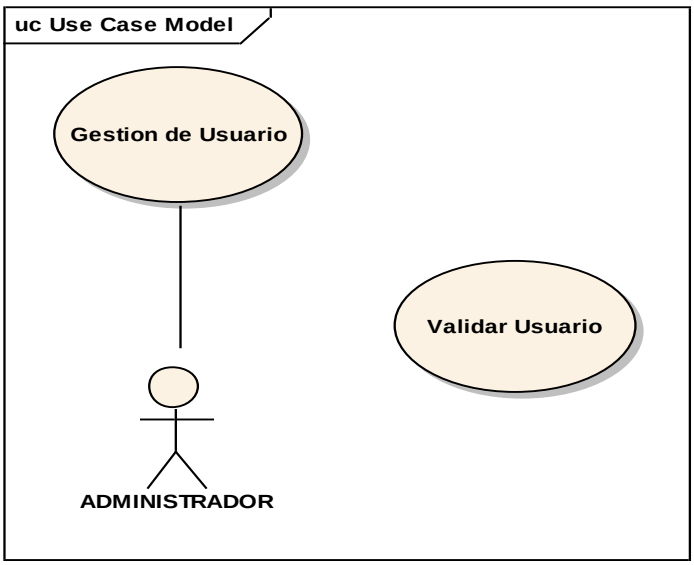


Figura 4.1 Caso de Uso (Gestión Usuario (Administrador)

4.1.11.2 Gestión Factura de Venta

Caso de Uso: Ingresar Factura de Venta

Actor**:** Secretaria

Condición Inicial**:** Registro de Activos Corrientes para la Venta, Registro de Cliente, Visualización reporte de Mercadería Disponible y Registro de Vendedor**.**

Visión general: La Secretaria para ingresar los datos de una factura de venta, selecciona la opción de Ventas, el sistema desplaza el menú de ventas donde

existe la opción de Factura de Venta, sistema presenta el encabezado de del formulario de factura, secretaria llena el formulario de encabezado, una vez que sistema confirma el ingreso de datos en todos los campos, debe dar click en el botón de guardar, inmediatamente el sistema presenta el formulario de detalle de factura, seguidamente se selecciona el Activo Corriente para la venta detallando la cantidad y su valor unitario, para guardar el detalle debe dar click en el botón de guardar que se encuentra en la parte superior del detalle, para registrar la factura se debe de dar click en el icono de aprobar, el sistema se encarga de registrar Factura de Venta y confirma la operación.

Condición de salida: Primario real

Requerimientos especiales:

Flujo de Eventos

Actor	Sistema
1.- La secretaria solicita la operación de ingresar datos en una Factura de venta.	2.- Presenta formulario de encabezado de Factura de venta.
3.- Ingresa de forma individual cada uno de los datos del formulario de encabezado de Factura de venta.	4.-Guarda los datos del encabezado de Factura de venta y presenta el formulario de detalle de Factura de Venta.
5.- Ingresa de forma individual cada uno de los datos del detalle de la Factura de venta	6.- Guarda los datos del detalle de Factura de venta
7.- Da click en el botón de Aprobar Factura de Venta.	8.- Confirma la operación y registra la factura.

***Caso de Uso*:** Editar Encabezado de Factura de Venta

Actor**:** Secretaria

Condición Inicial**:** Visualizar Cliente, Visualizar Vendedor, Visualizar Fecha de Emisión y Forma de pago.

Visión general: Si lo que desea la Secretaria es editar los datos del encabezado de una Factura de Venta, debe de seleccionar el serial de la Factura de Venta a ser editada, el sistema marca los datos de esta factura en el registro, la secretaria da click sobre el botón de Editar que se encuentra en la parte superior, sistema desplaza el formulario de encabezado de la factura previamente seleccionada, realiza la modificación respectiva en los campos del formulario de encabezado de Factura de Venta, sistema guarda los cambios de encabezado, confirmando la operación.

Condición de salida**:** Primario real

Requerimientos especiales**:**

Flujo de Eventos

Actor	Sistema
1.- Este caso de uso comienza cuando la secretaria solicita la operación de Editar encabezado de Factura de venta.	2.- Presenta el registro de Facturas de venta
3.- Selecciona el serial de una Factura de venta de la lista.	4.- Presenta los datos del encabezado de la Factura de venta seleccionada.
5.- Actualiza los respectivos campos datos.	6.- Confirma y guarda la operación.

Caso de Uso: Consultar Facturas de Venta

Actor**:** Gerente

Condición Inicial: Registro de Facturas de Venta

Visión general: El gerente solicita la operación de consultar Facturas de venta emitidas en un periodo de tiempo, el sistema presenta las opciones de consulta de Facturas de venta, gerente ingresa la información de los campos del formulario de la consulta respectiva, el sistema presenta un botón de imprimir que se encuentra localizado en la parte superior, gerente da click en el botón de imprimir, sistema presenta reporte de consulta de Facturas de Venta de acuerdo a la opción que se selecciono (por fechas, subtotal, Numero de Factura, IVA(S/N), Forma de pago y por Cliente), sistema confirma la operación de consulta y desplaza la información, gerente imprime el reporte o guarda como archivo,

***Condición de salida*:** Primario real

***Requerimientos especiales*:**

Flujo de Eventos

Actor	Sistema
1.- Este caso de uso comienza cuando el gerente solicita la operación de consultar Facturas de venta.	2.- Presenta las diferentes opciones de consulta como son: (por fechas, subtotal, Numero de Factura, IVA(S/N), Forma de pago y por Cliente)
3.- Selecciona una de las opciones de consulta.	4.- Presenta el formulario de consulta de Facturas de Venta de acuerdo a la opción elegida.
5.- Ingresa la información de cada uno de los campos del respectivo formulario de consulta.	6.- Confirma la operación y desplaza la información.

Esquema Casos de Uso

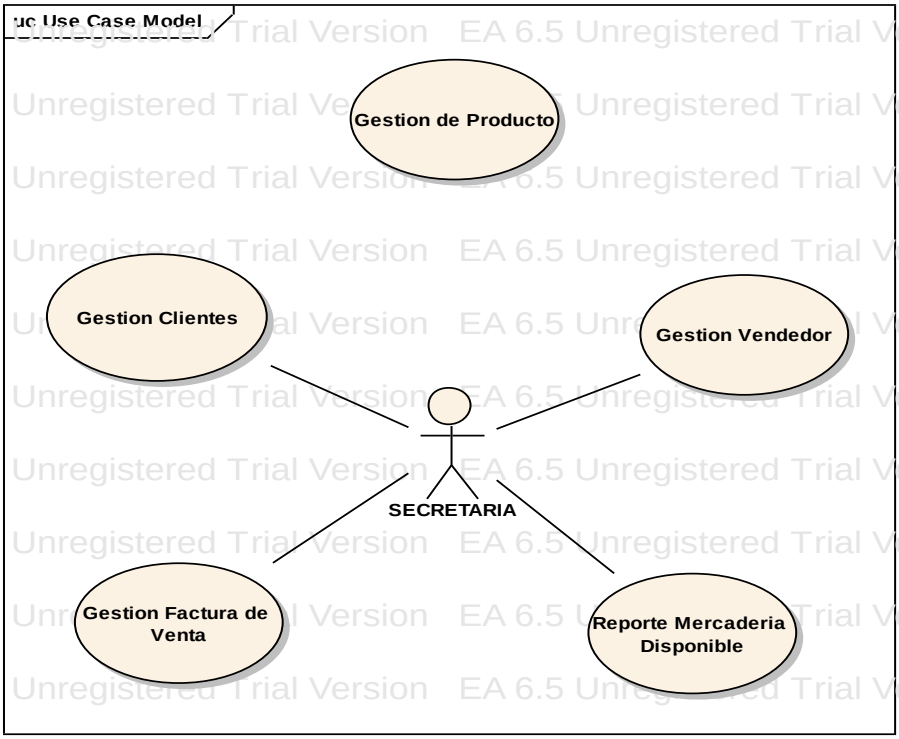
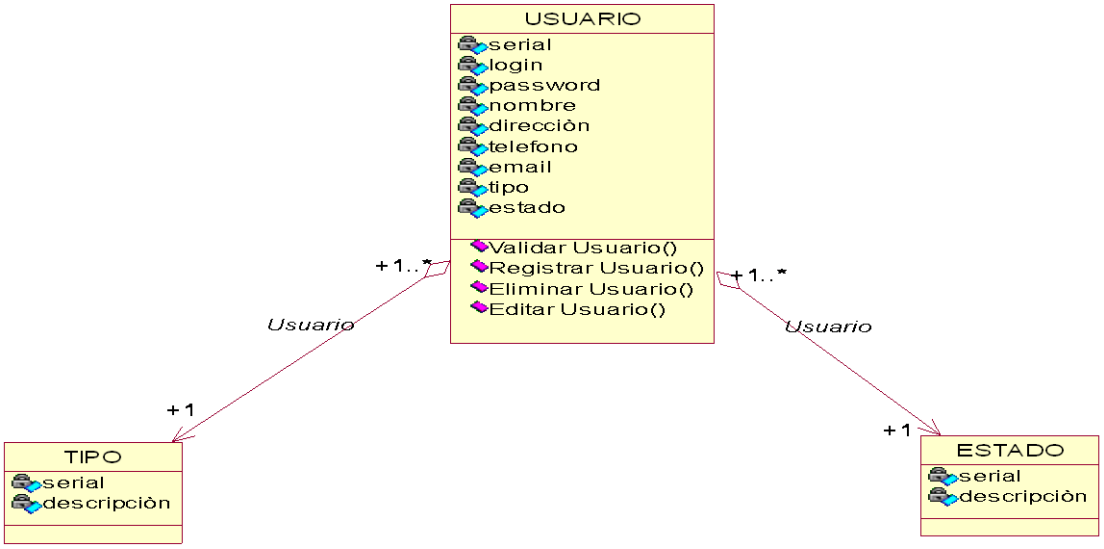


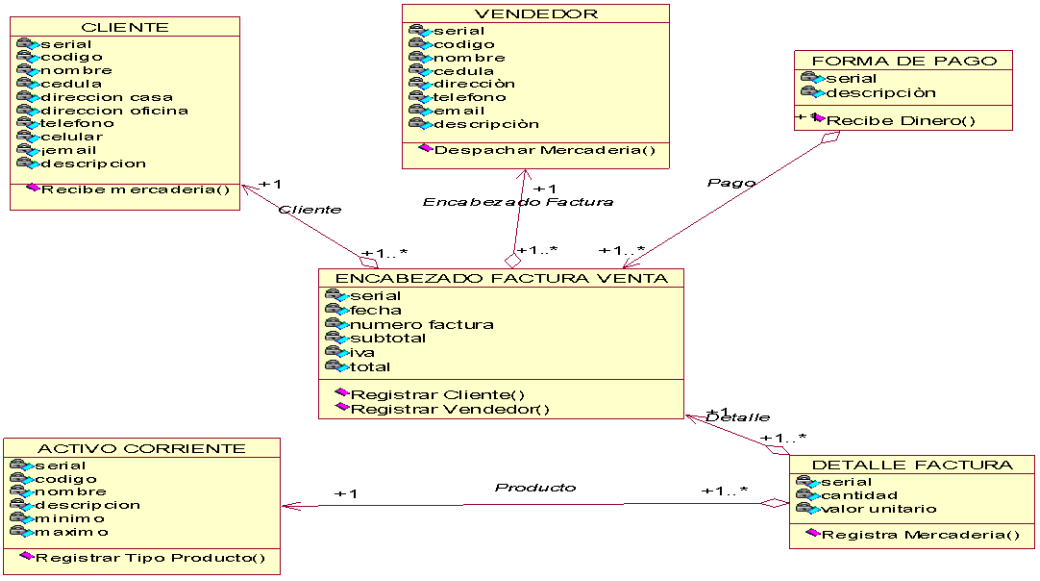
Figura 4.2 Caso de Uso (Gestión Factura de Venta)

4.1.12 Modelo de Diagramas de Clases

Esquema del Diagrama de Clases Perfiles de Usuarios



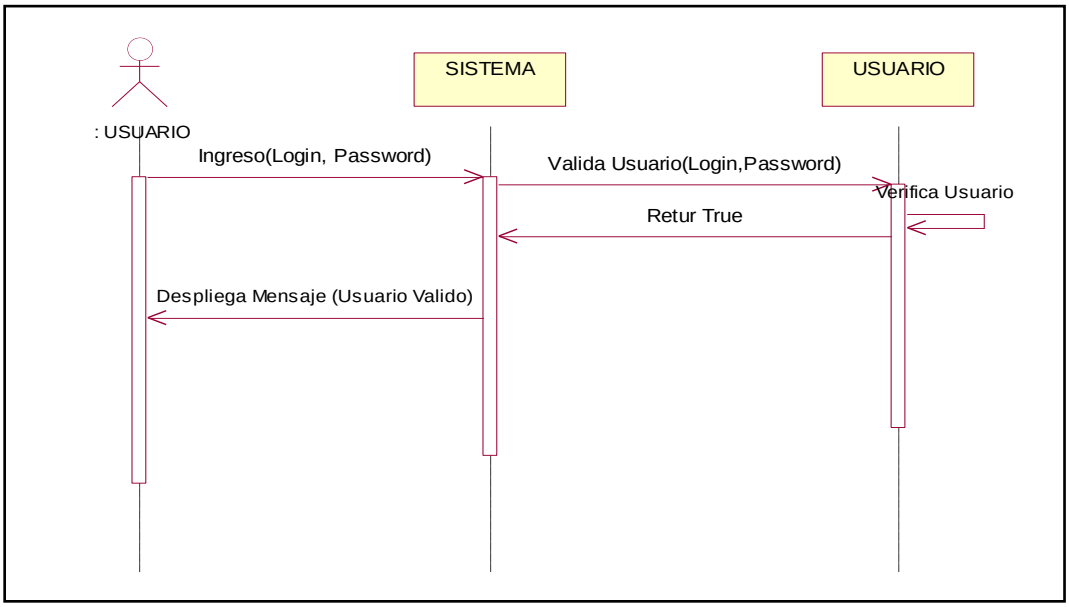
Esquema del Diagrama de Clases Factura de Venta



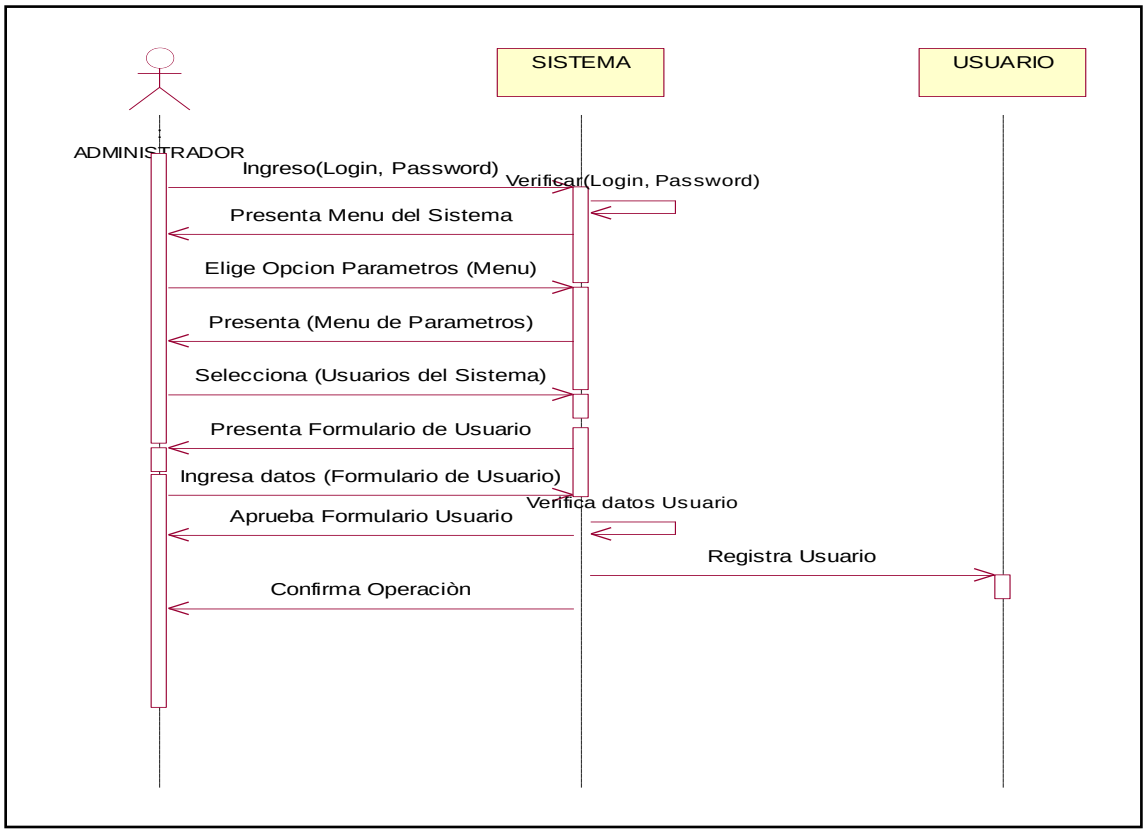
4.1.13 Modelo de Diagramas de Secuencia

4.1.13.1 Gestión Perfiles de Usuario

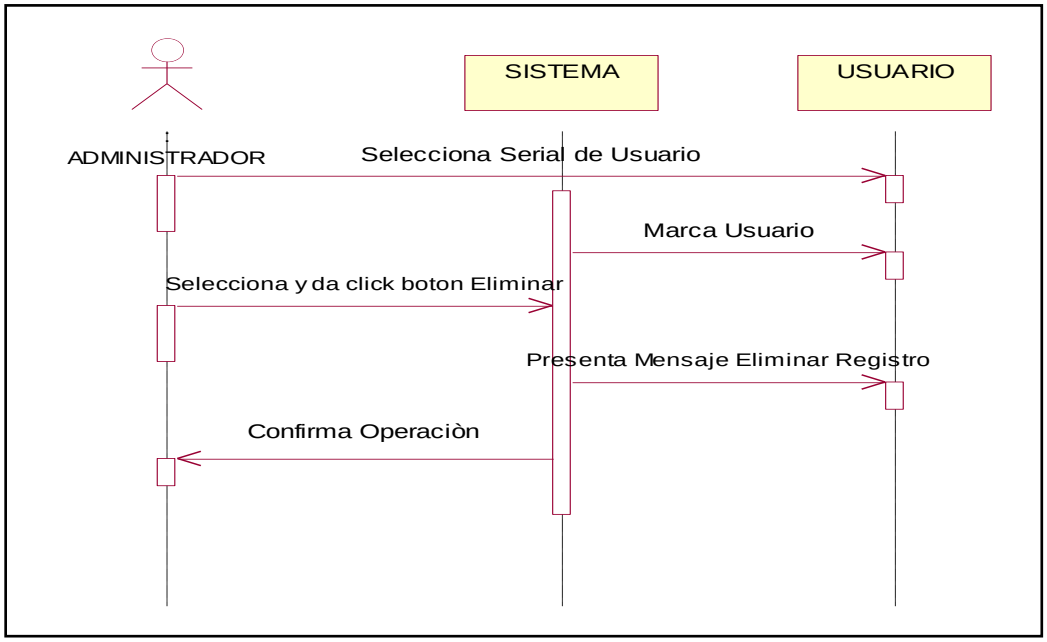
Validar Usuario



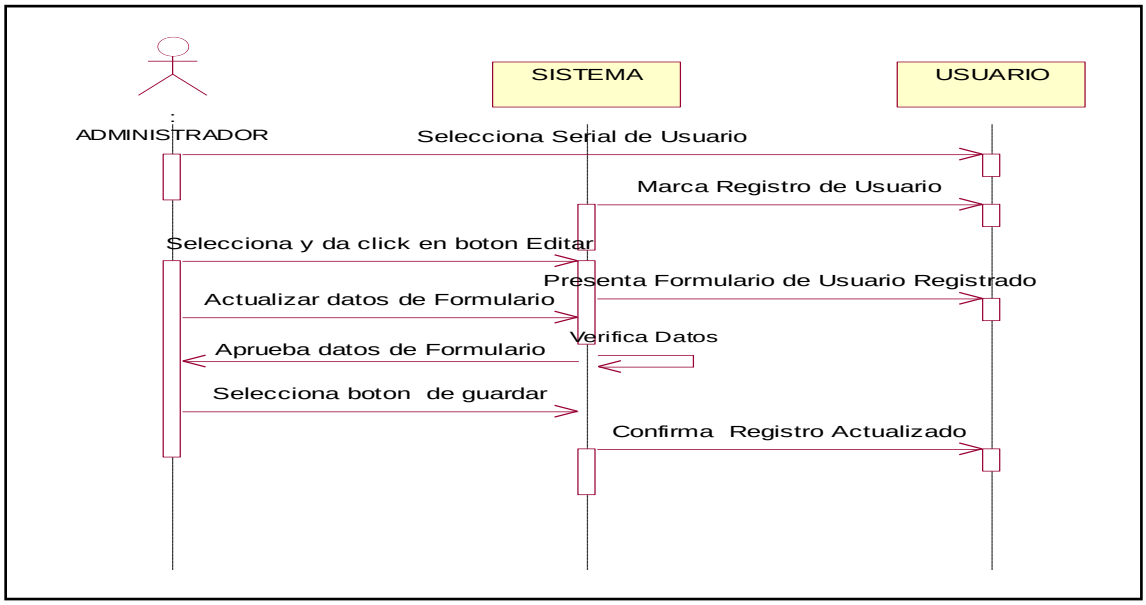
Ingreso Usuario



Elimina Usuario

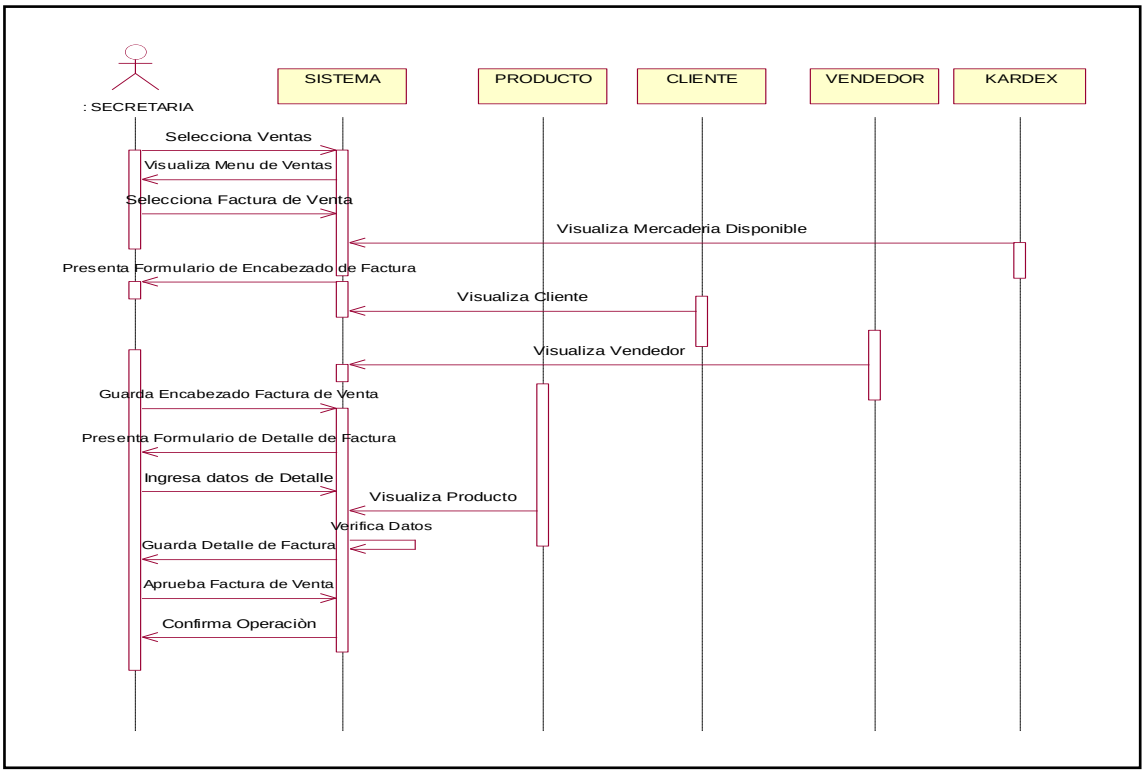


Editar Usuario

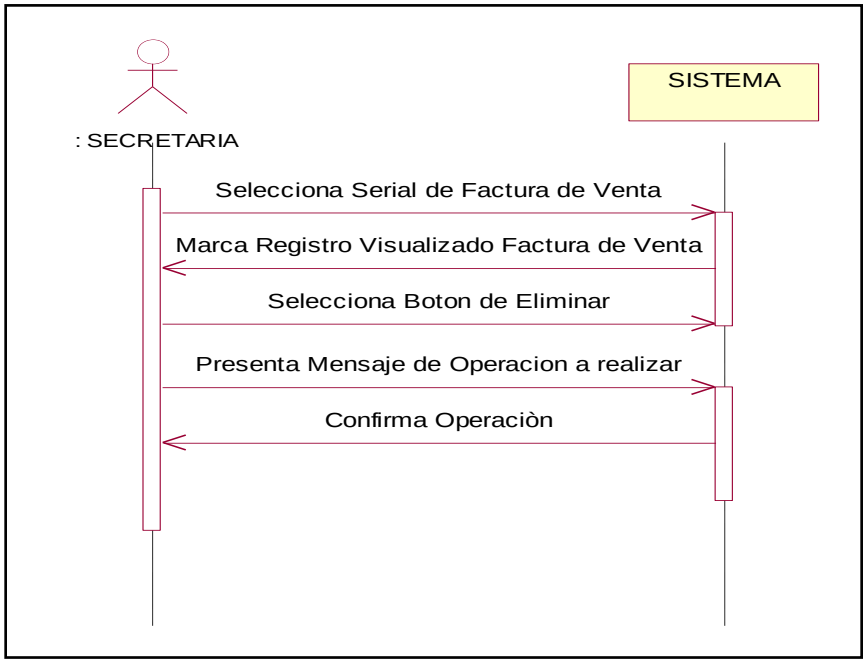


4.1.13.2 Gestión Factura de Venta

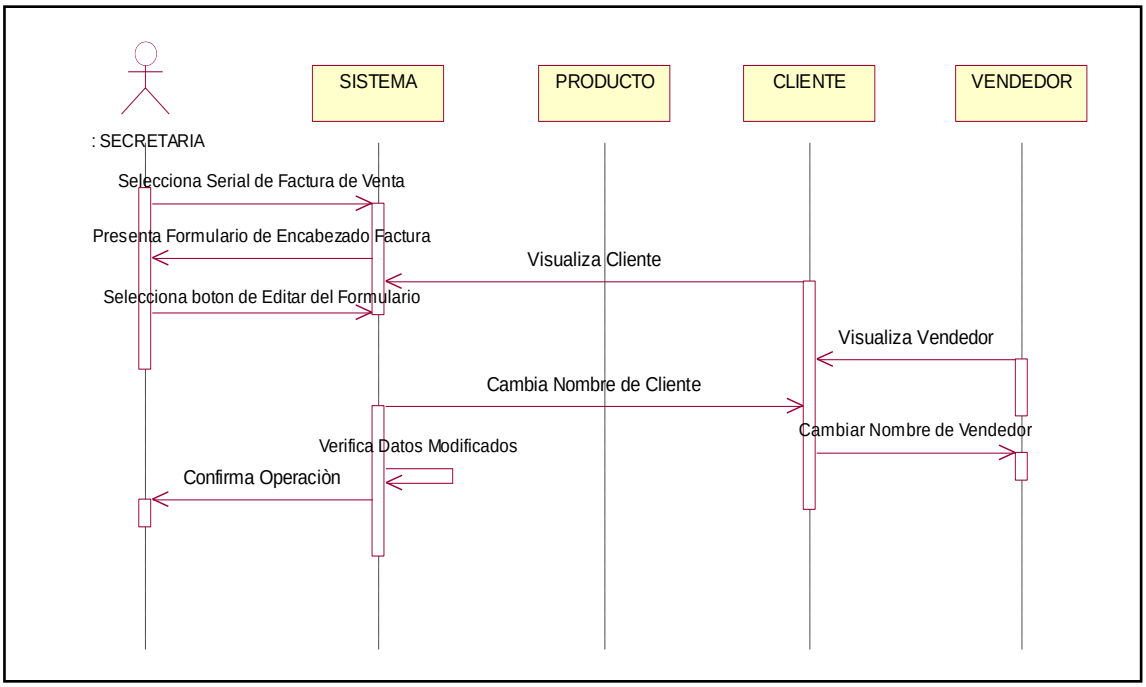
Ingresar Factura de Venta



Eliminar Factura de Venta



Edita Encabezado de Factura de Venta



4.2 FASE DE DISEÑO PRELIMINAR

Esta fase admite el diseño de la Arquitectura del Sistema determinando el Costo Beneficio y las herramientas técnicas que se requieren para el correcto desarrollo y desempeño del Sistema de Administración para la Granja Avícola Regalo de Dios.

4.2.1 Definir los Módulos Principales

Una vez obtenida la información necesaria del entorno, es necesario sintetizarla, darle prioridades y distribuir las necesidades de cada parte, delimitando los límites del sistema, para definir su interacción con el entorno.

La Arquitectura para el Sistema de Administración de la Granja Avícola Regalo de Dios esta estructurada básicamente en el manejo de cinco módulos. Este modelo de arquitectura presenta una visión abstracta de los sub-sistemas que identifican y configuran la funcionalidad del sistema.

4.2.1.1 Modulo de Inicialización

Este módulo permite ingresar los Tipos de Producto, Porcentaje de Retención, los mismos que serán manejados dentro de la Empresa, este modulo realiza las siguientes funciones:

- ✓ Admite obtener un registro de Proveedores con información detallada de cada uno de ellos, actualizada automáticamente desde su respectiva gestión
- ✓ Permite detallar el Activo Corriente de acuerdo al proveedor que nos provee con determinadas materias primas, insumos y pollitas bebes.

4.2.1.2 Modulo de Activos Corrientes

Es el encargado de registrar las operaciones de los Activos Corrientes, esta opción le permite manejar todo lo relativo a los productos de los Tipos de Producto gracias a esta herramienta el usuario podrá conocer el detalle de su movimiento de Activos Corrientes (Kardex), control de sus máximos y mínimos, emitiendo reportes de mucha utilidad para el departamento de Bodega.

En este módulo se pueden hacer consultas directamente a la base de datos que contiene la información de los Activos Corrientes, desplegándolo en pantallas para una mejor lectura por fecha o descripción, este modulo realiza las siguientes funciones:

- ✓ Conocer las cantidades y costo de los Activos Corrientes en bodega.
- ✓ Ejercer un control total de las existencias de los Activos Corrientes.

4.2.1.3 Modulo de Ventas

Consiste en la rápida y eficaz emisión de facturas que afecta directamente los registros de Activos Corrientes y ventas, para almacenar su información con fines contables. Este módulo cuenta con la capacidad de dar soporte a nuestro sistema, controlando las ventas de la empresa y realizando reportes de las ventas realizadas en un periodo de tiempo.

Cuando realiza una factura, ésta es automáticamente actualizada en el modulo de Activos Corrientes y en el módulo de Reportes de Ventas. Así mismo el movimiento de los Activos Corrientes se registran en el kardex y quedan actualizadas las cantidades disponibles en bodega cumpliendo las siguientes funciones:

- ✓ Elabora la factura de venta.
- ✓ Permite obtener reporte de la Mercadería disponible de acuerdo al Tipo de Producto (Activo Corriente Consumible, Activo Corriente Fabricado y el Activo Corriente para la Venta).
- ✓ Maneja un registro de Clientes con información detallada de cada uno de ellos y de su relación con la Empresa, actualizada automáticamente desde su respectiva gestión,
- ✓ Accede a obtener un registro de los vendedores de la empresa que son los responsables de las ventas que se realizan.

4.2.1.4 Modulo de Reportes de Ventas

Este módulo le permite tener un reporte de las facturas de Venta, cumpliendo las siguientes funciones:

- ✓ Podrá consultar las facturas de ventas en un periodo de tiempo por Fechas.
- ✓ Podrá consultar las facturas de ventas en un periodo de tiempo por Subtotal.
- ✓ Podrá consultar las facturas de ventas en un periodo de tiempo por Número de Factura.
- ✓ Podrá consultar las facturas de ventas en un periodo de tiempo por IVA(s/n).
- ✓ Podrá consultar las facturas de ventas en un periodo de tiempo por Forma de Pago.
- ✓ Podrá consultar las facturas de ventas en un periodo de tiempo por Cliente.

4.2.1.5 Modulo de Reportes de Compras

Este módulo le permite tener un reporte de las facturas de Compra, cumpliendo las siguientes funciones:

- ✓ Podrá consultar las facturas de compras en un periodo de tiempo por Fechas.
- ✓ Podrá consultar las facturas de compras en un periodo de tiempo por Subtotal.
- ✓ Podrá consultar las facturas de compras en un periodo de tiempo por Número de Factura.
- ✓ Podrá consultar las facturas de compras en un periodo de tiempo por IVA(s/n).
- ✓ Podrá consultar las facturas de compras en un periodo de tiempo por Forma de Pago.
- ✓ Podrá consultar las facturas de compras en un periodo de tiempo por Proveedor.

4.2.1.6 Modulo de Parámetros del Sistema

Es usado para controlar el ingreso y garantizar la seguridad del sistema, permitiendo denegar el acceso a los usuarios dentro de cada sitio, admitiendo a cada uno el acceso a distintas secciones o la realización de algunas acciones.

Este modulo garantiza el ocultamiento de la clave (password) de los usuarios que pretenden interactuar con el sistema, este modulo realiza las siguientes funciones:

- ✓ Especificación de claves de acceso al sistema.
- ✓ Especificaciones claves de acceso de los datos o funciones especificas del sistema.
- ✓ Limitación del número de intentos de acceso.
- ✓ Registra las Bodegas que posee la Infraestructura de la Empresa.

4.2.2 Definir la interacción que existirá entre dichos módulos

Los Módulos gestionan la interacción, el control de flujo de datos y la secuencia de información del servicio ofrecido por el sistema. La interacción implica la conectividad total entre los seis módulos y la comunicación de los mismos, permitiendo la centralización de la información.

Esto significa determinar la interacción o tareas del usuario, para que el sistema las pueda soportar, tienen que estar definidas en la Arquitectura del Software, enumerándose a continuación los cinco primeros escenarios:

- ✓ Agregación de datos: Poder aplicar simultáneamente una acción a un conjunto de datos.
- ✓ Agregación de comandos: Agrupar acciones que se puedan ejecutar de una sola vez.
- ✓ Comandos de cancelación
- ✓ Uso de aplicaciones de forma concurrente
- ✓ Validación automática de la entrada de datos

Su desarrollo esta orientado a la dependencia del motor de la bases de datos a utilizar con dicha aplicación lo que permite una mejor escalabilidad del proyecto, en forma codificada y clasificada para la recuperación de información.

Permite la asociación de operaciones realizadas en algún modulo directamente con las aplicaciones que manejan y procesan toda la información de la empresa a través de la administración. Cada operación realizada en algún módulo se considera como única en el sistema pudiendo ser las mismas modificada, actualizada o ampliada pero nunca ingresada nuevamente ni a través de mecanismos de importación y exportación lo que evita el error de recarga de información y agiliza el ingreso de datos.

4.2.2 Análisis del Costo Beneficio

Existen algunos factores que influyen en el costo de un producto de programación en efecto aplicaremos la herramienta del Cocomo que entre los factores primordiales que se observan y afectan en forma primordial es, la capacidad individual del personal asignado al proyecto y su familiaridad, complejidad del producto, el tamaño del programa, el tiempo asignado, el nivel de confiabilidad, el nivel tecnológico utilizado, la disponibilidad, familiaridad y estabilidad del sistema donde se desarrolla el producto, obteniendo una estimación del desarrollo del Sistema de Administración para la Granja Avícola Regalo de Dios.

Estimamos que el desarrollo de este proyecto va a tener **3600** Líneas de código**,** estimando el esfuerzo y la duración muestra los resultados calculados. Incluye el tamaño total del proyecto en líneas de código, y una tabla con una estimación de:

	Más Probable
Esfuerzo (Effort)	12
Duración (Sched)	6 meses
Productividad (PROD)	299.4
Costo (Cost)	2405.16
Instrucción (INST)	0.7
Personal Necesario (Staff)	2

De acuerdo a la tabla de resultados sea llegado a determinar que va a tener un costo de 2405.16 USD, un tiempo de desarrollo de 6 meses, en cuanto al recurso humano se requiere de dos personas y el costo por línea de código será de 0.70 centavos.

4.2.4 Determinar la tecnología necesaria

La tecnología necesaria para el desarrollo de la aplicación refleja el lenguaje empleado, el equipo como los programas de apoyo, las prácticas y herramientas de programación empleadas.

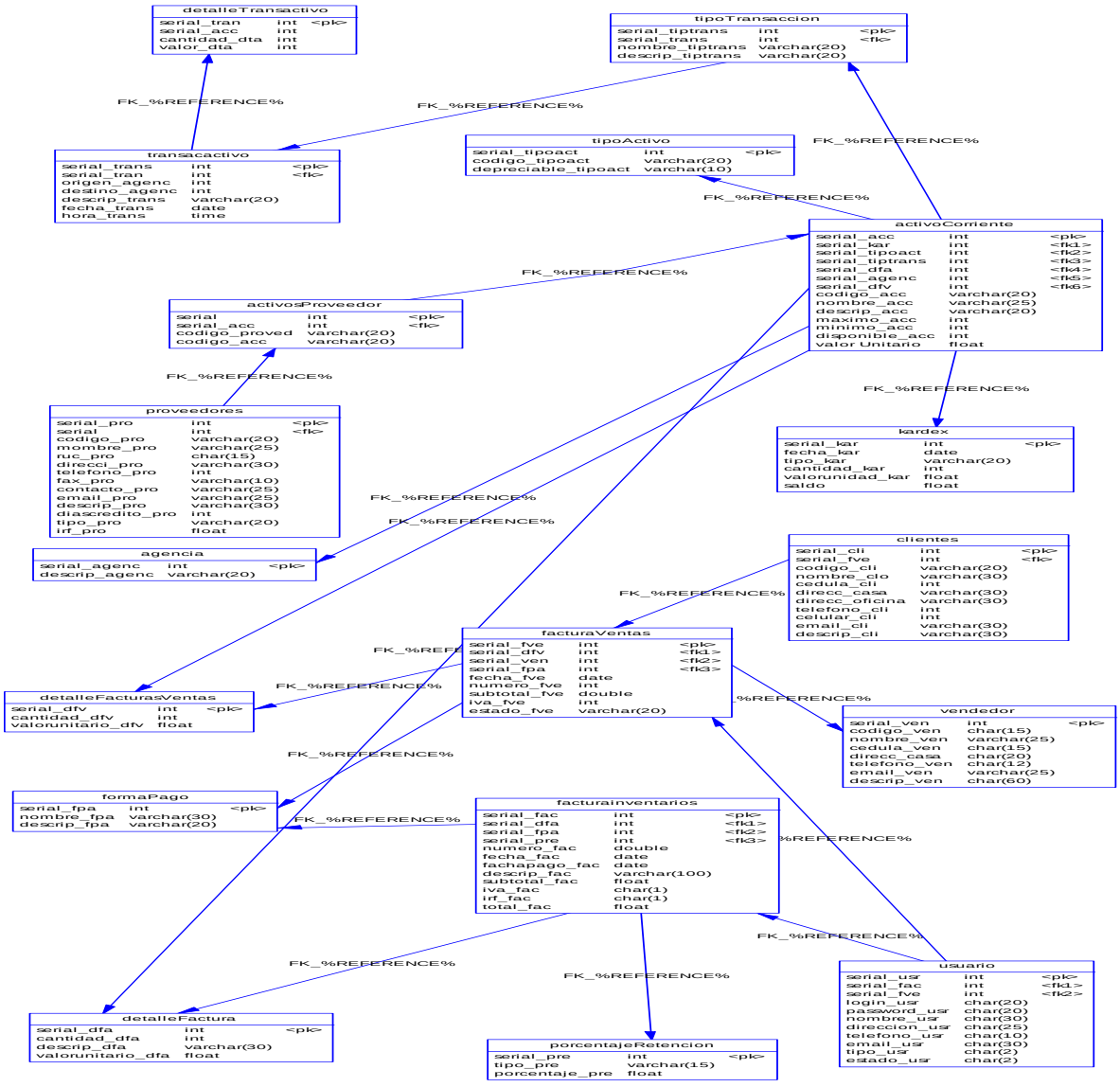
Hardware	Modelo 945 C280 Procesador Intel Pentium 4 de 3.2 GHz Cache 2.0Mb Placa Intel D945G Bus frontal 800 MHz Memoria RAM 512Mb Disco Duro 160 Gb Drive Óptico CDRW 52x32 Monitor LG/Samsung 17 Tubo Red Modems Intel 10/100 PCI 56kbps
Software (Lenguaje de Programación)	Enterprise Architect Rational Rose Cocomo PHP Microsoft Windows (2000 Professional o Windows XP SmartFTP Cliente PowerDesigner
Herramientas de Programación.	Versión compilada de PHP Un servidor web (Apache, PWS, IIS, Etc.). Base de datos se recomienda Mysql Server Internet Explorer

4.3 FASE DE DISEÑO DETALLADO

Esta actividad técnica permite determinar el diseño e implementación de la Base de Datos en dos formatos tanto físico y lógico dependiendo del tipo y características que tiene el sistema a desarrollar para la Granja Avícola Regalo de Dios.

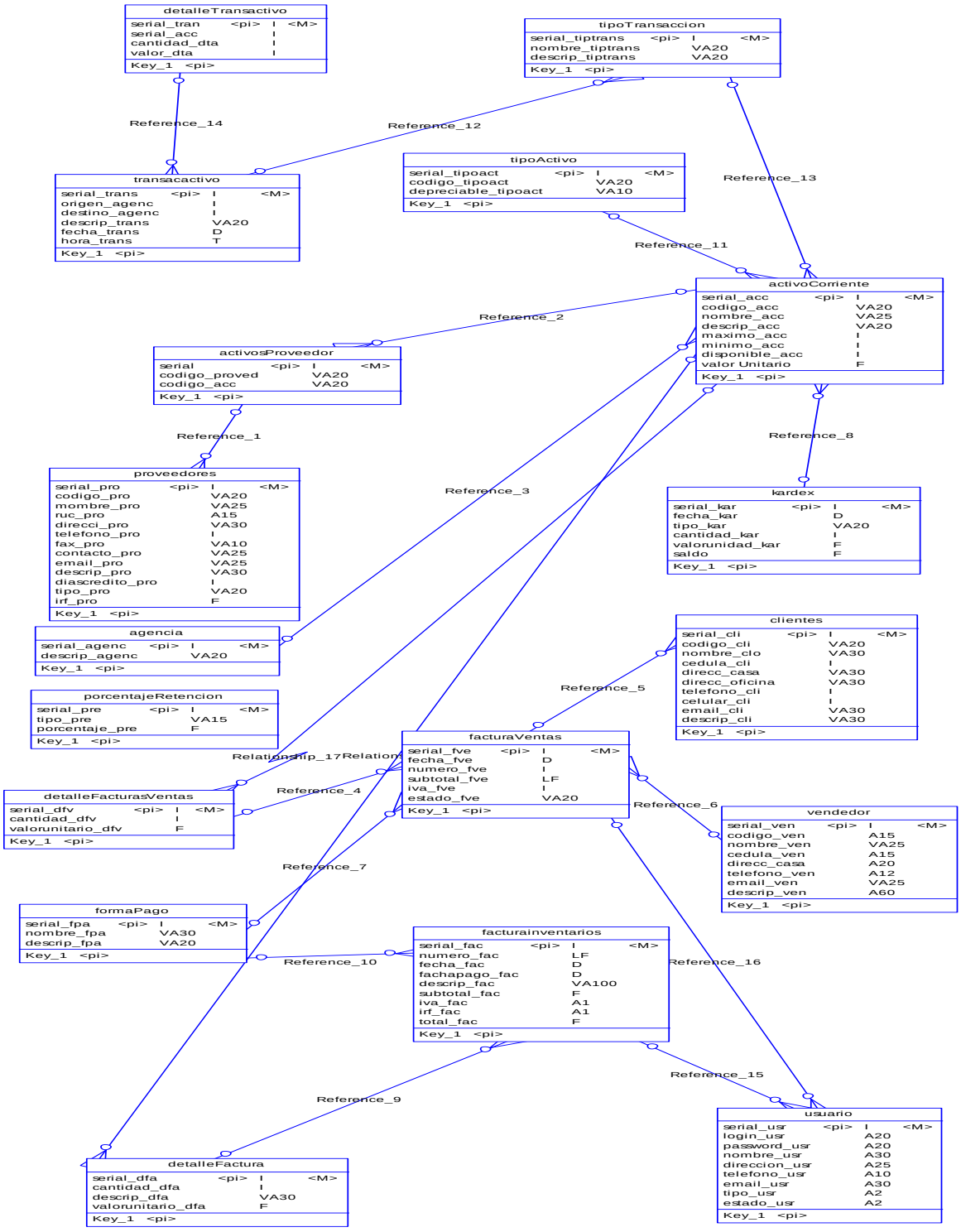
4.3.1 Diseño Lógico de la Base de Datos

El diseño lógico esta basado en satisfacer todos y cada uno de los requisitos detallados identificando las entidades, atributos y las relaciones que existen entre entidades.



4.3.2 Diseño Físico de la Base de Datos

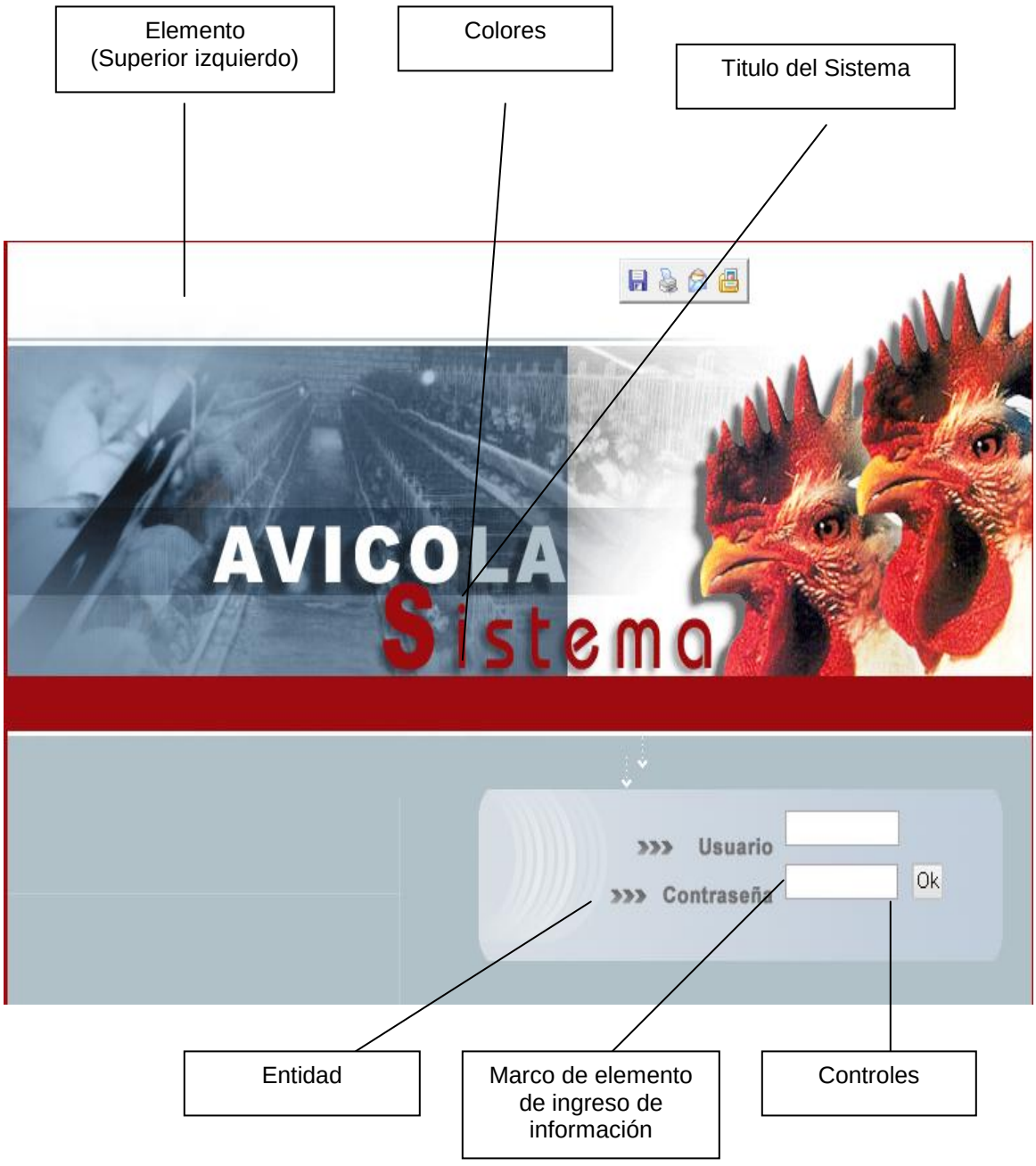
El diseño físico esta basado en el análisis de minimizar el tiempo de repuesta y manejo de la información, considerando el acceso de los usuarios y la protección de la información que deberá tener.


detalleTransactivo
serial_tran <pi> I <M>
serial_acc I
cantidad_dta I
valor_dta I
Key_1 <pi>
tipoTransaccion
serial_tiptrans <pi> I <M>
nombre_tiptrans VA20
descrip_tiptrans VA20
Key_1 <pi>
Reference_14
Reference_12
transacactivo
serial_trans <pi> I <M>
origen_agenc I
destino_agenc I
descrip_trans VA20
fecha_trans D
hora_trans T
Key_1 <pi>
tipoActivo
serial_tipoact <pi> I <M>
codigo_tipoact VA20
depreciable_tipoact VA10
Key_1 <pi>
Reference_13
Reference_11
activoCorriente
serial_acc <pi> I <M>
codigo_acc VA20
nombre_acc VA25
descrip_acc VA20
maximo_acc I
minimo_acc I
disponible_acc I
valor Unitario F
Key_1 <pi>
Reference_2
activosProveedor
serial <pi> I <M>
codigo_proved VA20
codigo_acc VA20
Key_1 <pi>
Reference_1
Reference_8
proveedores
serial_pro <pi> I <M>
codigo_pro VA20
mombre_pro VA25
ruc_pro A15
direcci_pro VA30
telefono_pro I
fax_pro VA10
contacto_pro VA25
email_pro VA25
descrip_pro VA30
diascredito_pro I
tipo_pro VA20
irf_pro F
Key_1 <pi>
Reference_3
kardex
serial_kar <pi> I <M>
fecha_kar D
tipo_kar VA20
cantidad_kar I
valorunidad_kar F
saldo F
Key_1 <pi>
agencia
serial_agenc <pi> I <M>
descrip_agenc VA20
Key_1 <pi>
clientes
serial_cli <pi> I <M>
codigo_cli VA20
nombre_clo VA30
cedula_cli I
direcc_casa VA30
direcc_oficina VA30
telefono_cli I
celular_cli I
email_cli VA30
descrip_cli VA30
Key_1 <pi>
Reference_5
porcentajeRetencion
serial_pre <pi> I <M>
tipo_pre VA15
porcentaje_pre F
Key_1 <pi>
RelationShip_17
Relation
facturaVentas
serial_fve <pi> I <M>
fecha_fve D
numero_fve I
subtotal_fve LF
iva_fve I
estado_fve VA20
Key_1 <pi>
Reference_4
Reference_6
detalleFacturasVentas
serial_dfv <pi> I <M>
cantidad_dfv I
valorunitario_dfv F
Key_1 <pi>
vendedor
serial_ven <pi> I <M>
codigo_ven A15
nombre_ven VA25
cedula_ven A15
direcc_casa A20
telefono_ven A12
email_ven VA25
descrip_ven A60
Key_1 <pi>
Reference_7
formaPago
serial_fpa <pi> I <M>
nombre_fpa VA30
descrip_fpa VA20
Key_1 <pi>
facturainventarios
serial_fac <pi> I <M>
numero_fac LF
fecha_fac D
fachapago_fac D
descrip_fac VA100
subtotal_fac F
iva_fac A1
irf_fac A1
total_fac F
Key_1 <pi>
Reference_10
Reference_16
Reference_15
Reference_9
usuario
serial_usr <pi> I <M>
login_usr A20
password_usr A20
nombre_usr A30
direccion_usr A25
telefono_usr A10
email_usr A30
tipo_usr A2
estado_usr A2
Key_1 <pi>
detalleFactura
serial_dfa <pi> I <M>
cantidad_dfa I
descrip_dfa VA30
valorunitario_dfa F
Key_1 <pi>

4.3.1 Diseñar el Modelo de Interfaz

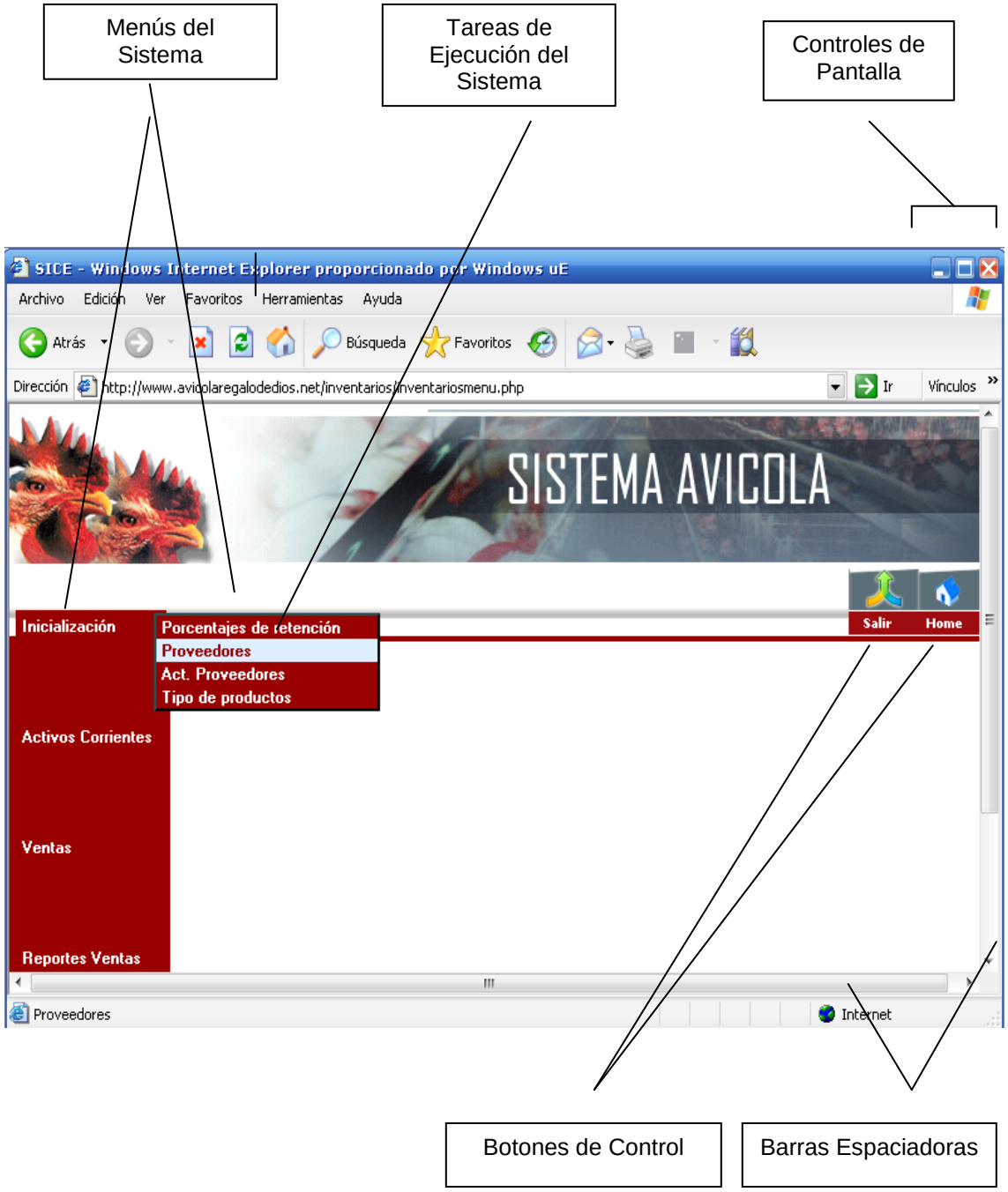
Para el diseño de la interfaz del Sistema Administrativo para la Granja Avícola Regalo de Dios, su enfoque esta orientado al diseño e implementación de una interfaz fácil de operar, evitando la complejidad.

La interfaz del Sistema de Administración se baso previamente en los colores del logotipo y actividad comercial que posee la empresa.



4.3.1.1 Diseño de los Menús del Sistema

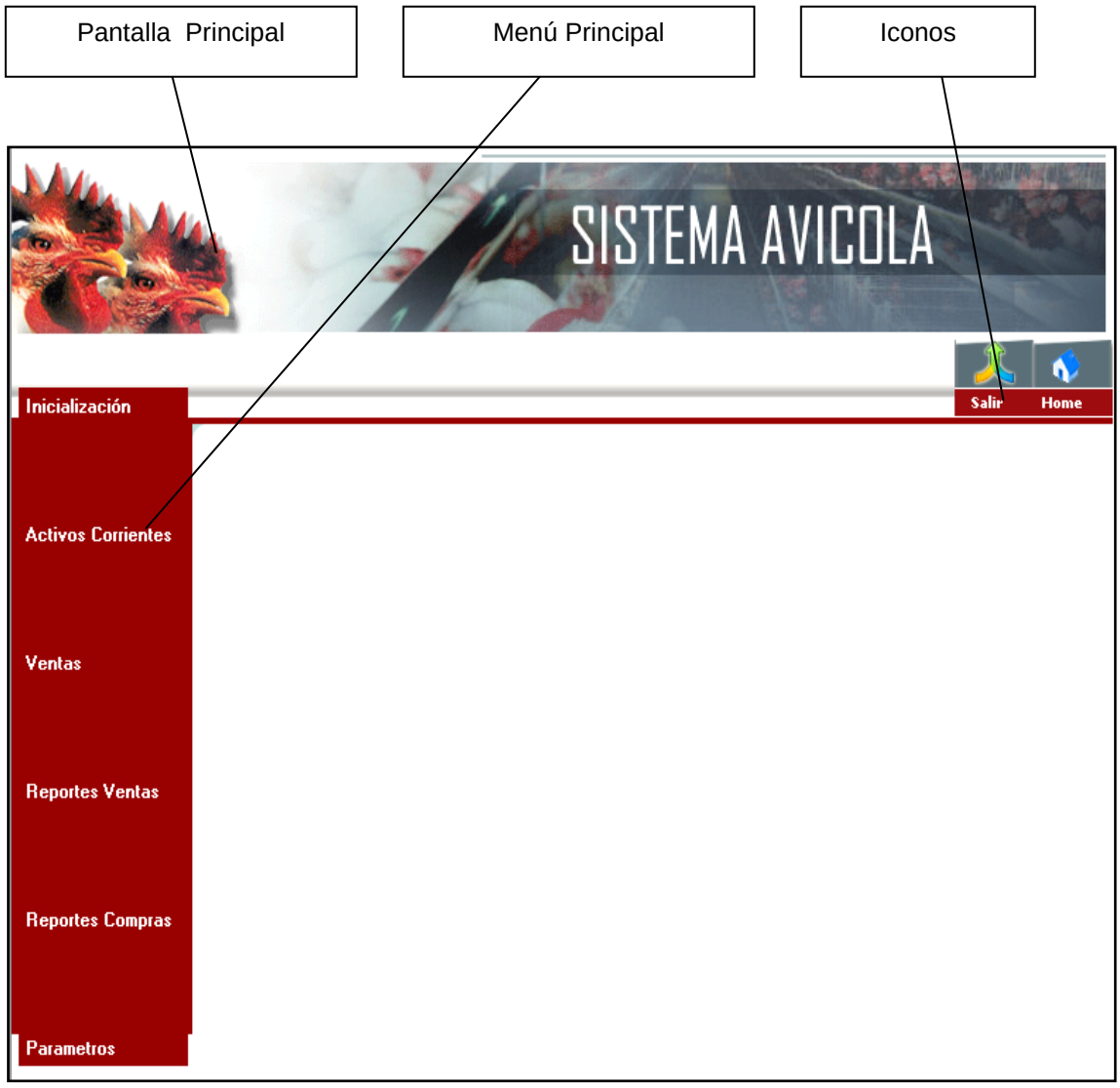
Para el diseño de los menús que manejara la aplicación se basa de acuerdo a las necesidades, requerimientos de acuerdo al desarrollo de la aplicación, analizando la forma correcta el manejo de la información.

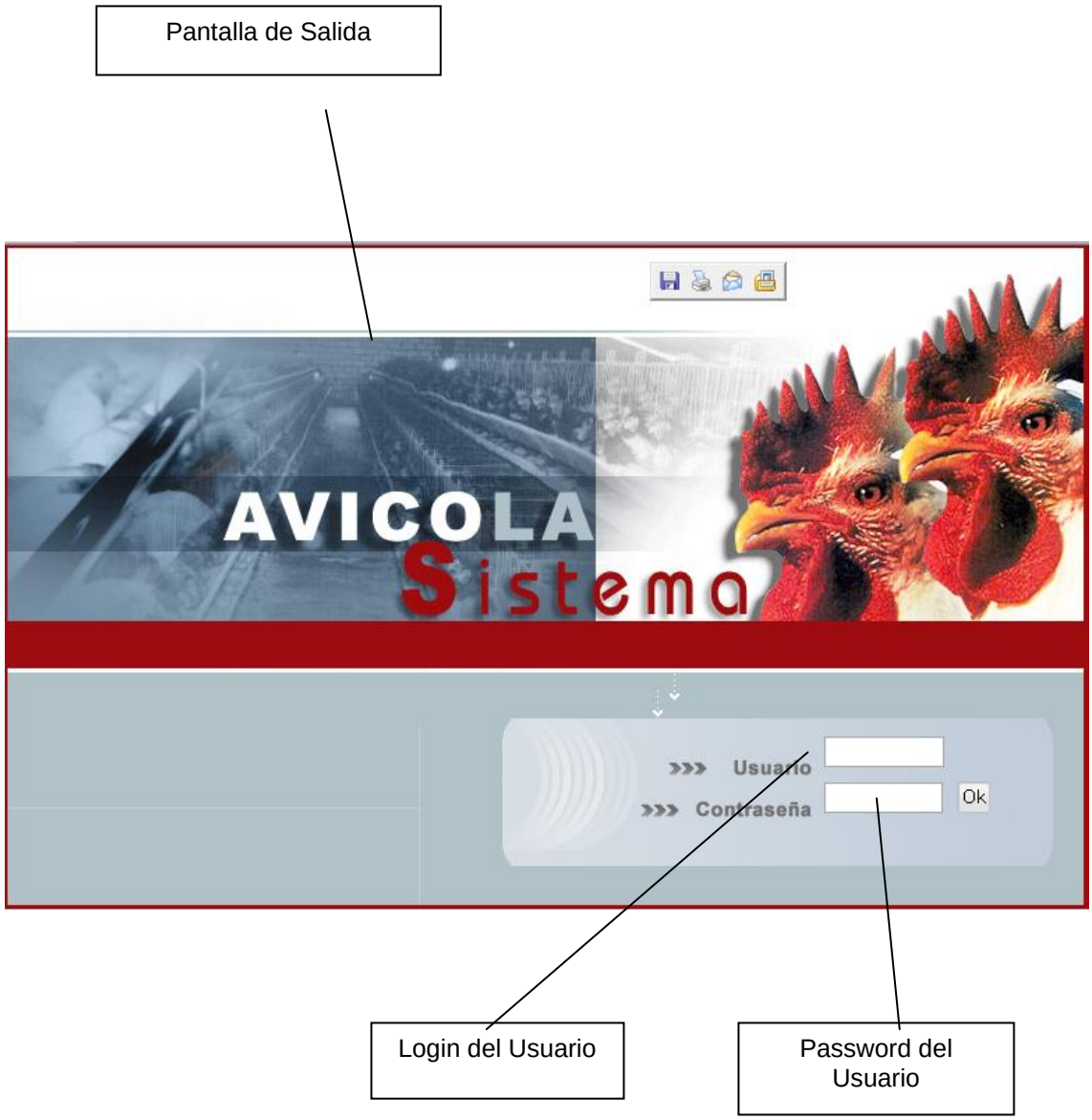


4.3.2 Diseñar las pantallas de entrada – salida

El diseño de la pantalla (entrada) para que el usuario pueda interactuar con el sistema, presenta un Menú Principal con seis opciones: Inicialización, Activos Corrientes, Ventas, Reportes de Ventas, Reportes de Compras y parámetros permitiendo seleccionar cada una de ellos de acuerdo al proceso o tarea que se va a ejecutar.

Para el diseño de la pantalla de salida se realizo bajo el entorno de que es un sistema on line por lo que regresa a la pantalla de ingreso de Login y Password del usuario que va a trabajar en el sistema.

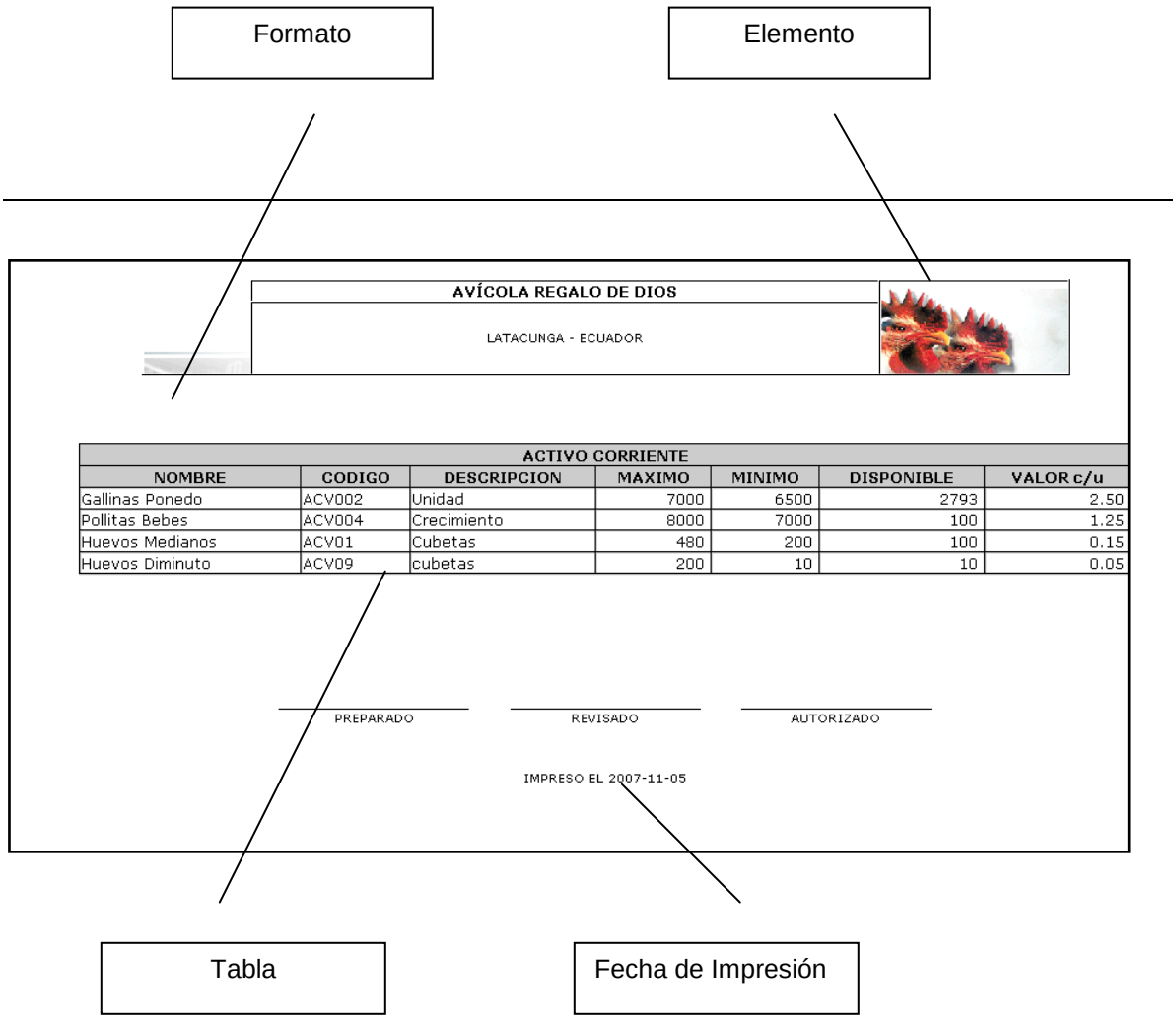




4.3.3 Diseñar los Reportes

Para el diseño de los reportes que emitirá la aplicación se basa en una estructura propiamente de forma general de acuerdo a cada Tipo de Producto sea este Activos Corrientes Consumibles, Activos corrientes de fabricación y Activos corrientes para la venta.

Detallando un formato que consta de la siguiente información: Nombre, código, descripción, cantidad máxima, cantidad mínima, cantidad disponible y el valor unitario.



4.3.4 Diseñar la Documentación

MANUAL DEL USUARIO DE REFERENCIA

Para poder ingresar a la aplicación es necesario que usted tenga permisos y este registrado, estos permisos son asignados por el Usuario Administrador del Sistema (de acuerdo al cargo).

INDICE

- Factura de Venta

- **SECCIÓN 4: Reporte de Ventas**
  - Reporte de Ventas por Fecha
  - Por Subtotal
  - Numero de Factura
  - IVA(s/n)
  - Forma de Pago
  - Por Cliente

- **SECCIÓN 4: Reporte de Compras**
  - Reporte de Compras por Fecha
  - Por Subtotal
  - Numero de Factura
  - IVA(s/n)
  - Forma de Pago
  - Por Proveedor

INTRODUCCIÓN

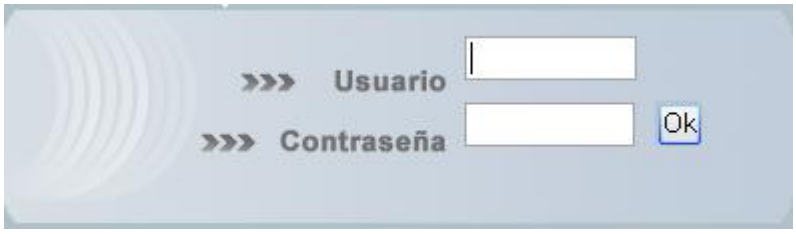
¿Qué es Sistema de Administración?

Sistema de Administración es un Sistema que permite administrar Facturas de Compra, Venta y Inventarios, muy simple y fácil de usar. Entre sus características se tienen:

- Inicialización
- Manejo de Activos Corrientes.
- Emisión de Reportes de Máximos, mínimos y Facturas.
- Registro de Transacciones
- Ventas
- Registro de Ventas.
- Registro de Compras.
- Manejo de Privilegios de Usuario Administrador y Usuarios Operadores.

SECCIÓN 1: Inicialización

Para inicializar el Sistema de Administración de la Granja Avícola Regalo de Dios., se mostrará la Pantalla de ingreso al sistema en la que se debe de digitar el Login y el Password del Usuario.:



Ventana de Ingreso al Sistema

Presione OK, espere un segundo para que ingrese automáticamente al Menú Principal del sistema. Luego el Sistema le permitirá ingresar a la opción del menú dando click en Bodega, Usuario.

Definir Porcentajes de Retención

Utilice las opciones de esta ventana para definir Porcentajes de retención aplicables a los montos de las facturas de compra calculados por el Sistema de Administración (IVA Y IRF).

Ventana de Definición de Porcentaje de Retención

En la parte superior de la ventana se encuentra la opción de Impuesto IVA o IRF. Permitiendo seleccionar una de los dos porcentajes que serán aplicados a los Montos.

Introduzca los datos referentes a un Porcentaje de Retención.

- En 'Código', previamente seleccione una de las dos opciones (IVA o IRF).
- En el campo de 'Valor', introduzca el porcentaje del Impuesto.

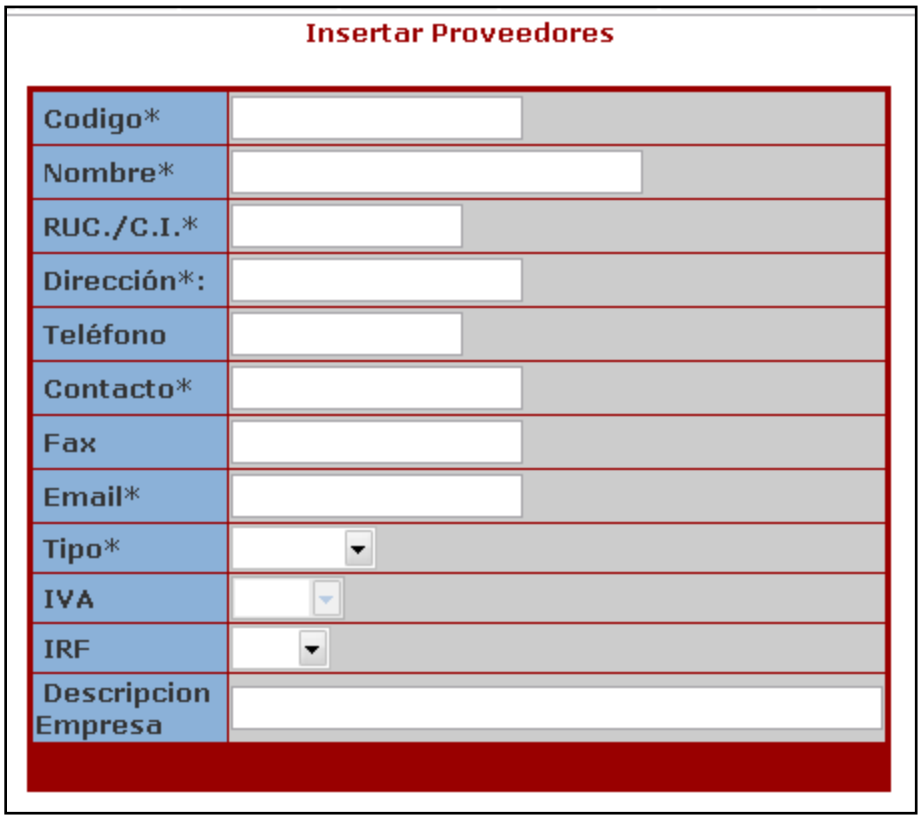
Luego de introducir los datos correspondientes, haga click sobre el botón "Guardar", para agregar el recién creado porcentaje en la Lista de Impuestos retenidos.

En esta ventana también podrá realizar operaciones como "Editar" y "Eliminar".

Sólo seleccione de la Lista un Porcentaje de Retención al que se le aplicará la operación, y a continuación, haga click sobre el botón correspondiente.

Proveedor

Para ingresar un nuevo Proveedor el sistema presenta un formulario con los siguientes campos:



Ventana de Proveedores

La Lista mostrada a continuación, describe cada uno de los campos de datos que se visualizan en el formulario de proveedores:

- Código del. Proveedor: Indica el número de Proveedor con el que se le identificara, su propósito es sólo de organización dentro de la Lista.
- Nombre Proveedor: Ingresa el nombre del Proveedor. Este nombre se establece durante su compra.
- RUC / CI Define el numero de cedula o ruc personal del cliente.
- Dirección.- Detalla exactamente el lugar donde mantiene las instalaciones la empresa.
- Teléfono.- Ingresa el numero de teléfono local

- Contacto:- Es el nombre de la persona con la que esta relacionada la empresa.
- Fax.- Es el número de teléfono al que podemos enviar un documento.
- E-mail .- Ingresa el correo electrónico que posee el proveedor.
- Tipo:- Permite seleccionar una de las dos opciones si es una empresa natural o jurídica.
- IVA.- Permite seleccionar el porcentaje de IVA que debe ser considerado dentro de una factura.
- IRF.- Es el impuesto de valor agregado a la fuente.
- Descripción.- Determina que tipo de producto oferta la empresa proveedora.

Luego de introducir los datos correspondientes, haga click sobre el botón "Guardar".

En esta ventana también podrá realizar operaciones como "Editar" y "Eliminar". Sólo seleccione de la Lista un Proveedor al cual se le aplicará la operación y a continuación haga click sobre el botón correspondiente.

Activos Proveedores

Permitirá realizar la selección de un Activo Corriente detallando que proveedores nos pueden proveer dicho producto, presentando el siguiente formulario:

Ventana Inserta producto dependiendo de Proveedor

Este formulario consta de dos campos:

Proveedor:- Muestra el código y Nombre del Proveedor.

Activo:- Muestra el código del Activo Corriente Consumible y su nombre.

Luego de seleccionar los datos correspondientes, haga click sobre el botón "Guardar".

En esta ventana también podrá realizar operaciones como "Editar" y "Eliminar". Sólo seleccione de la Lista un Proveedor al cual se le aplicará la operación y a continuación haga click sobre el botón correspondiente.

Tipo de Producto

Dentro de la ventana para ingresar un Tipo de Producto este presenta el siguiente formulario:

Ventana de Tipos de Productos

Este formulario consta de tres campos en los que se debe de ingresar la siguiente información:

Código:- Se debe de escribir con la letra C mayúscula seguida por dos ceros y el respectivo numero en el orden en que se realiza el ingreso.

Nombre.- Se describe el nombre del Tipo de Producto con las primeras letras en mayúsculas.

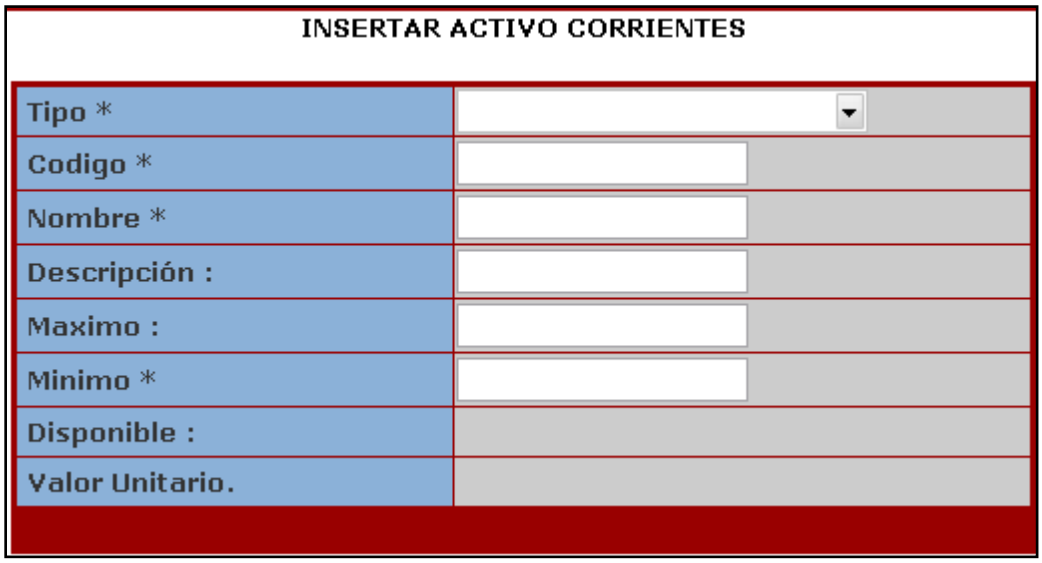
Activo Para:-Tiene tres opciones (Venta, Consumo Interno y medicinas) permitiendo seleccionar una de las opciones.

Luego de seleccionar los datos correspondientes, haga click sobre el botón "Guardar". En esta ventana también podrá realizar operaciones como "Editar" y "Eliminar". Sólo seleccione de la Lista un Proveedor al cual se le aplicará la operación y a continuación haga click sobre el botón correspondiente.

SECCIÓN 2: Activos Corrientes

Inventario Activos Corrientes

Podrá crear el Inventario en esta Ventana. Para configurar el Inventario, diríjase al menú "Principal", opción "Inventario".



Ventana de Manejo de Inventario (Activos Corrientes)

Para introducir la definición de Activos Corrientes del Inventario, primero haga click en "Insertar". Esto habilitará los campos de textos para recibir la entrada de datos.

- En 'Tipo', introduzca el Tipo de Producto.
- En 'Código', introduzca el código del Activo Corriente de acuerdo si es (Activo Corriente Consumible, Activo Corriente Fabricado o Activo Corriente para la Venta).
- En 'Nombre', introduzca el Nombre del Activo Corriente.
- En 'Descripción', introduzca que tipo de unidad es...
- En ' Cantidad Máxima', introduzca la cantidad máxima del Activo Corriente..
- En 'Cantidad Mínima', introduzca la cantidad mínima del Activo Corriente con la que se puede contar.

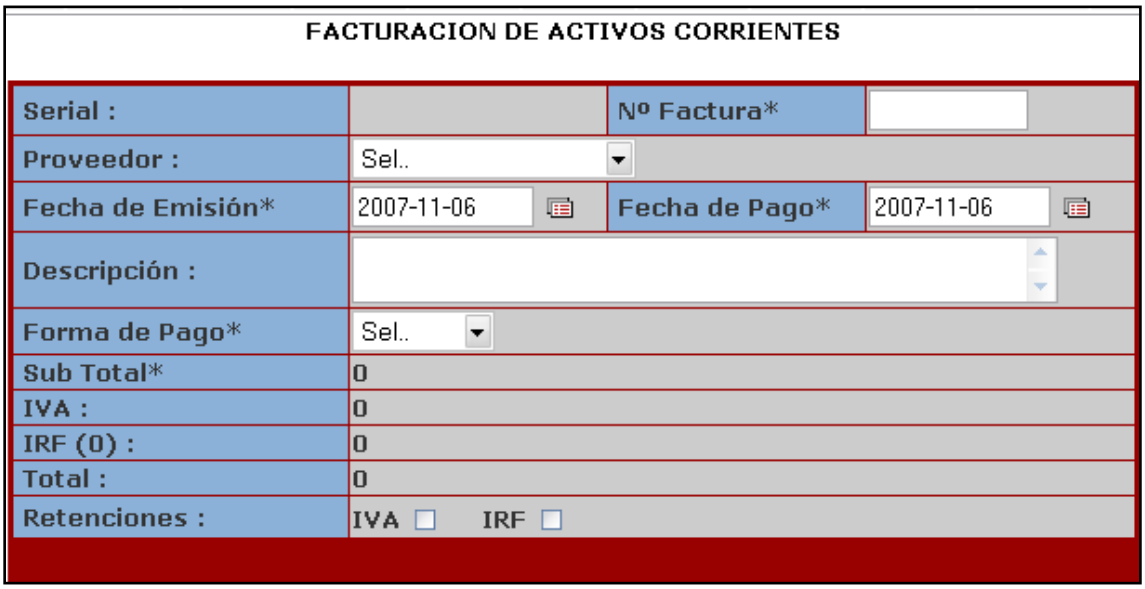
Luego de introducir los datos correspondientes, haga click sobre el botón "Guardar".

Cuando finalice con la creación de la Lista de Inventario, haga click en el botón "Imprimir" para cerrar la ventana.

En esta ventana también podrá realizar operaciones como "Editar" y "Eliminar". Sólo seleccione de la Lista el Activos Corrientes a cual se le aplicará la operación, y a continuación, haga click sobre el botón correspondiente.

Registrar Factura de Activo Corriente

En la parte inferior de la Ventana Principal de Inventario, se encuentra el Panel de Activos Corrientes la Vista Previa de la factura de activos corrientes generada para el Proveedor seleccionado en la Lista:



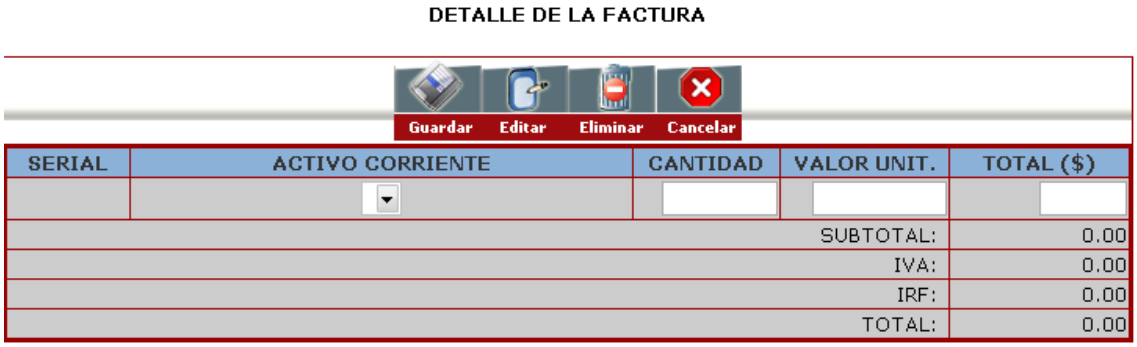
Ventana de Facturas de Activos Corrientes

Para editar un Factura de Activo Corriente haga click en el botón “Insertar” y tiene los siguientes campos:

- En 'Numero de Factura', registra el numero propiamente de la factura que fue emitida por el proveedor.
- En 'Proveedor', Permitirá elegir el proveedor.

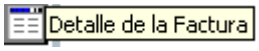
- En 'Fecha de Emisión', Registra el día y mes que se compro.
- En 'Fecha de Pago', Registra específicamente el plazo de la fecha de pago indicando el día, mes y año.
- En 'Descripción', Registra la descripción física (quintal, funda, frasco).
- En 'Forma de Pago', permite seleccionar una de las dos opciones (contado o a crédito).
- En 'Retención', permitirá dar click en las dos opciones o independientemente de acuerdo al calculo del porcentaje de retención (IVA, IRF).

Para ingresar los datos en la parte de detalle se debe de dar click en el botón de "Guardar", inmediatamente se desplazara el detalle que consta de:



Ventana de Detalle de Facturas de Activos Corrientes

- En 'Activo Corriente', seleccionamos uno de los productos que nos oferta el proveedor que fue previamente seleccionado.
- En 'Cantidad', Registra la cantidad del producto que se adquirió.
- En 'Valor Unitario', Registra el costo del producto de forma individual.
- En 'Total ', Registra el calculo automático de la cantidad por el valor unitario y los porcentajes de retención. Luego de haber llenado todos los campos correspondientes, presione nuevamente el botón "Guardar". Para continuar con la operación, finalmente presione el botón "Aprobar" para que se registre.

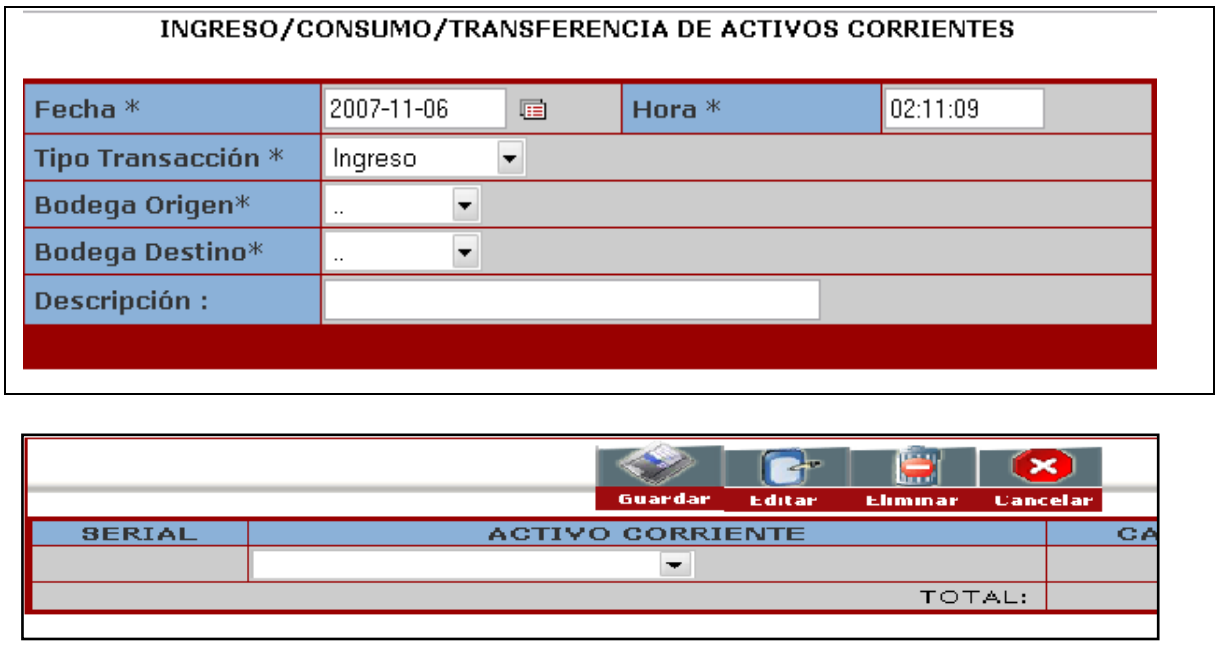
No se Utilizara los botones "Modificar" y "Editar" una vez que se aprobó el registro, en caso de que desee “Anular” una factura, respectivamente, debe dar click 

para que pueda ingresar al registro y pueda encontrar habilitado el icono de anular 

dando click en el la factura se registrara como anulada y se actualiza la información de inventario.

Transacciones

Dentro del Menú de Inventarios en la opción de Activos Corrientes permitirá una opción de elegir transacción, inmediatamente al dar click en ella se presentará la ventana de Transacciones permitiendo registrar los Ingresos, Egresos, Producción y Mortalidad,

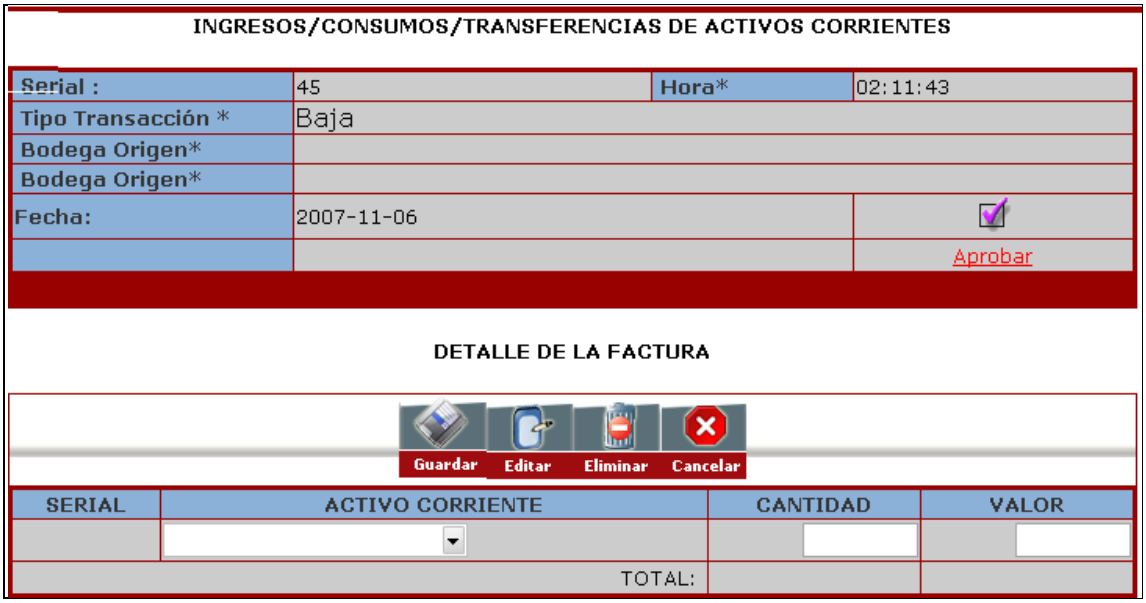


Ventana de Detalle de Transacción

Las operaciones en esta ventana son descritas en el campo Tipo Transacción de materias primas, vacunas, aves, balanceado, producción y mortalidad.

Registrar Transacciones

En la parte superior, se encuentra la fecha y la hora en que se registro determinada transacción.



Ventana de Detalles de Consumos

Para agregar un Ingreso, Egreso, Transacción y Mortalidad, haga click en el botón "insertar", y llene los campos correspondientes:

- En 'Tipo de Transacción', Permitirá elegir (Ingreso, Egreso, Transferencia y Mortalidad.
- En ‘Bodega de Origen’ indicara el lugar donde se encuentra el producto.
- En ‘Bodega de Destino’ determina el lugar exacto en el que fue consumido o suministrado un producto.
- En 'Descripción', introduzca la descripción física del Activo Corriente (quintal, funda, frascos, etc).

Para ingresar los datos de la segunda parte de la transacción presione el botón “Guardar” inmediatamente le desplazara la parte de detalle que contiene:

- En ‘Activo Corriente’ permite seleccionar el nombre del Activo Corriente del cual se va a realizar la transacción.

- En 'Cantidad', introduzca la cantidad por unidad del respectivo Activo Corriente.

- En 'Valor', introduzca el precio del Activo. Luego de haber llenado todos los campos correspondientes, presione nuevamente el botón "Guardar". Para continuar con la operación, y finalmente presione el botón "Aprobar" para que se registre.

No se Utilizara los botones "Modificar" y "Editar" una vez que haya sido aprobado el registro, en caso de que desee modificar o eliminar, respectivamente, debe realizar antes de ser aprobada..

Kardex

Sistema de Administración puede emitir Kardex de acuerdo al Tipo de Producto (Activos Corrientes Consumibles, Activos Corrientes Fabricados y Activos Corrientes para la Venta. Para emitir Kadex, haga click en el menú "Activos Corrientes", y luego seleccione la opción "Kardex". Esta acción mostrará la siguiente Ventana:

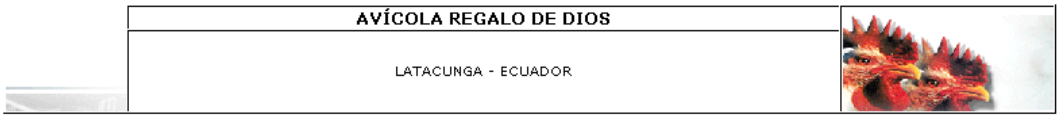
Ventana de Kardex

- En 'Tipo de Articulo', permitirá seleccionar una de las opciones que presenta sea este un producto (consumible, fabricado y para la venta.

- En 'Activo Corriente', permitirá seleccionar una de las opciones de activo corriente de acuerdo al tipo de Producto que fue previamente seleccionado.

- En 'Fecha desde', introduzca la fecha inicial a tomar en cuenta para generar el Kardex (en formato año/mes/día).
- En 'Fecha hasta', introduzca la fecha inicial a tomar en cuenta para generar el Kardex.
- En 'Enviar', seleccione una de las dos opciones con las que cuenta sea estas (pantalla o archivo).

Luego de haber establecido e ingresado los datos para que se genere el Kardex, haga click sobre el botón "Imprimir", para mandar el Kardex.



KARDEX DEL ACTIVO CORRIENTE Gallinas Ponedo

Del 2007-10-28 Al 2007-11-06

FECHA	REF.	INGRESOS			EGRESOS			VENTAS			SALDO		
		CANT.	VALOR c/u	TOTAL	CANT.	VALOR c/u	TOTAL	CANT	VALOR c/u	TOTAL	CANT	VALOR c/u	TOTAL
2007-10-31	ING.Galpon1	1000	2.50	2,500.00							2793	2.5	6,982.50
2007-10-31	EGR.Galpon1				793	2.50	1,982.50				2793	2.5	6,982.50
2007-10-31	ING.Galpon1	793	2.50	1,982.50							2793	2.5	6,982.50

PREPARADO REVISADO AUTORIZADO

Ventana de Kardex

Reportes

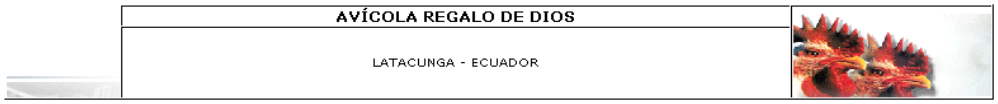
Sistema de Administración de Facturas de Compra, Venta y Inventarios puede emitir Reportes de acuerdo al Tipo de Producto (Activos Corrientes Consumibles, Activos Corrientes Fabricados y Activos Corrientes para la Venta. Para emitir Reportes, haga click en el menú "Activos Corrientes", y luego seleccione la opción "Reportes". Esta acción mostrará la Ventana de Reporte:

Ventana de Reporte

- En 'Tipo de Articulo', seleccione una de las opciones que comprende(Activos Corrientes para la Venta, Activos Corrientes Consumibles, Activos Corrientes para la Venta)

- En 'Enviar a ', permite seleccionar una de las dos opciones si desea un reporte en pantalla o para guardarlo como archivo. Una ves que se ingresaron los datos para continuar con el proceso se debe de dar click en el botón 

inmediatamente desplazara la pantalla que contiene el reporte de acuerdo al Tipo de Producto que se selecciono.

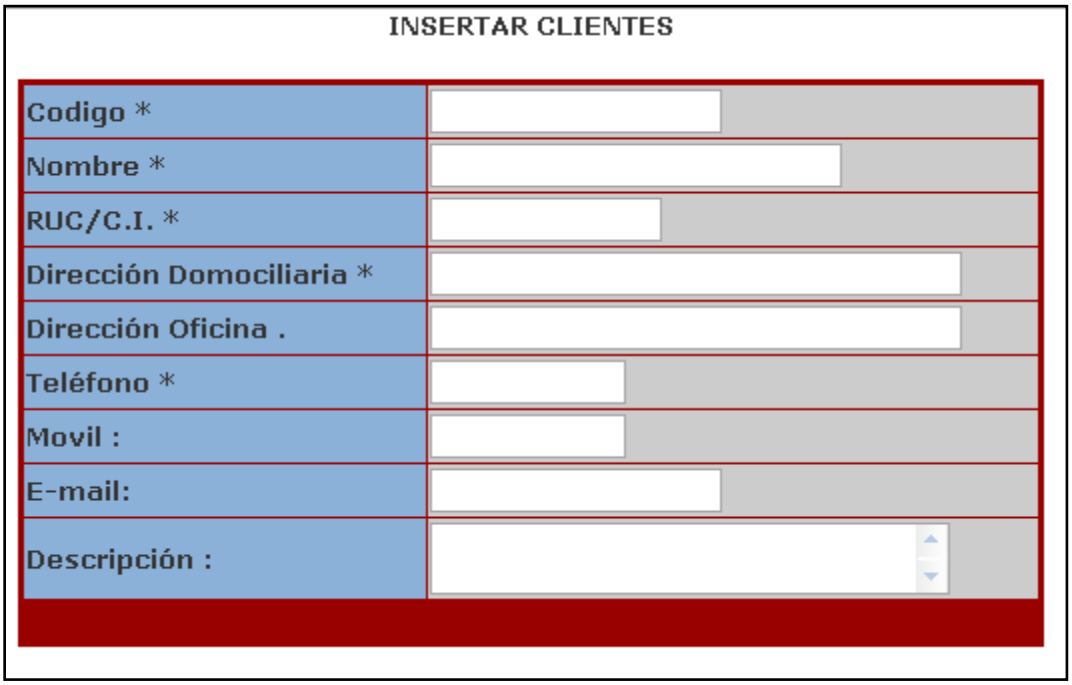


ACTIVO CORRIENTE						
NOMBRE	CODIGO	DESCRIPCION	MAXIMO	MINIMO	DISPONIBLE	VALOR c/u
bicarbonato	AC03	funda	10	1	0	0.00
Maiz	AC04	Nacional	800	100	398	10.50
Metionina	AC05	Americano	10	2	0	0.00
Balanceado de P	ACF01	Nacional	18	17	0	0.00
Balanceado de C	ACF02	Quintal	20	16	0	0.00
Balanceado Inic	ACF03	Quintal	12	1	20	10.00
Gallinas Ponedo	ACV002	Unidad	7000	6500	2793	2.50
Pollitas Bebes	ACV004	Crecimiento	8000	7000	100	1.25
Huevos Medianos	ACV01	Cubetas	480	200	100	0.15
Huevos Diminuto	ACV09	cubetas	200	10	10	0.05

SECCIÓN 3: Ventas

Cliente

Para mantener un registro actualizado de los clientes es necesario tener un formulario el mismo que permita almacenar datos personales de los clientes con los que la empresa esta relacionada, presenta el siguiente formulario.



Ventana para Ingresar Cliente

La Lista mostrada a continuación, describe cada uno de los campos de datos que se visualizan en la Lista de Clientes:

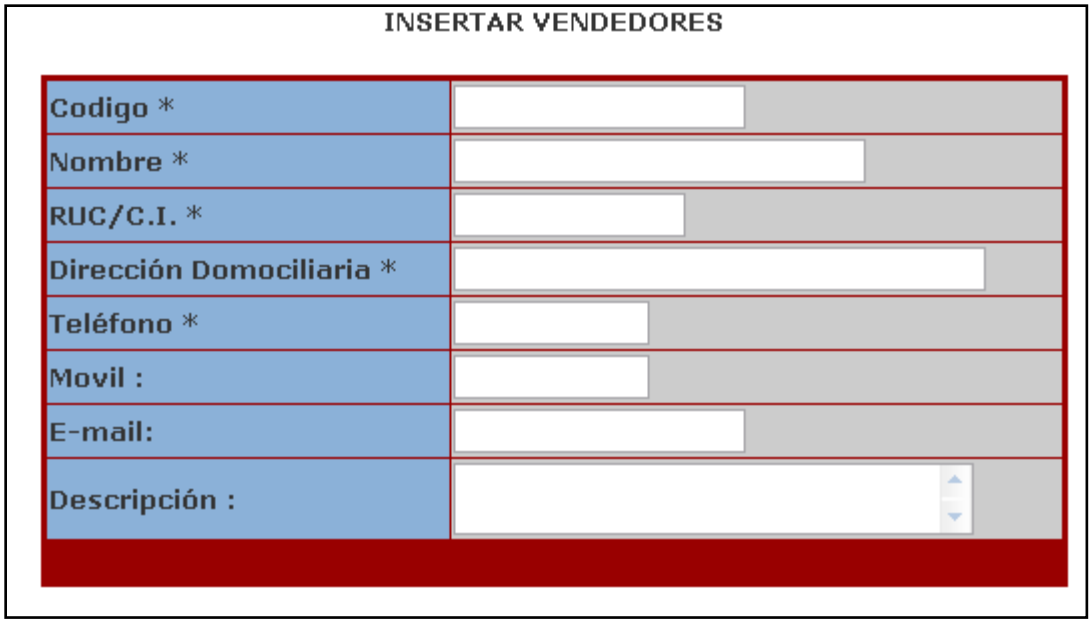
- Código del. Cliente: Indica el número de Cliente. Este número es asignado cuando se admite el nuevo Cliente en el Sistema, y su propósito es sólo de organización dentro de la Lista.
- Nombre Cliente: Visualiza el nombre del Cliente. Este nombre se establece durante su admisión, al ser despachado.
- RUC / CI Define el numero de cedula o ruc personal del cliente.
- Dirección Domiciliaria.- Detalla exactamente el lugar donde reside el cliente.
- Teléfono y Móvil.- Ingresa el numero de teléfono tanto local como móvil
- E-mail .- Ingresa el correo electrónico que posee el cliente.

- Descripción.- Determina que tipo de producto compra determinado cliente.

Luego de introducir los datos correspondientes, haga click sobre el botón "Guardar". En esta ventana también podrá realizar operaciones como "Editar" y "Eliminar". Sólo seleccione de la Lista un Proveedor al cual se le aplicará la operación y a continuación haga click sobre el botón correspondiente.

Vendedores

Dentro del modulo de ventas es necesario mantener un registro de los vendedores los mismos que asumen la responsabilidad de una venta realizada a determinado cliente, presenta el siguiente formulario con los siguientes campos.



Ventana para Insertar Vendedores

La Lista mostrada a continuación, describe cada uno de los campos de datos que se visualizan en el formulario de Vendedores:

- Código del. Vendedor: Indica el número de Cliente. Este número es asignado con la V (mayúscula) seguida un cero y el número correspondiente en su orden numérico.

- Nombre Vendedor: Ingresa el nombre del vendedor.
- RUC / CI Define el numero de cedula o ruc personal del vendedor.
- Dirección Domiciliaria.- Detalla exactamente el lugar donde reside el vendedor.
- Teléfono.- Ingresa el número de teléfono local.
- Móvil.- Ingresa el número de teléfono del celular personal.
- E-mail .- Ingresa el correo electrónico que posee el vendedor.
- Descripción.- Determina si es un vendedor autorizado para realizar las ventas.

Luego de introducir los datos correspondientes, haga click sobre el botón "Guardar". En esta ventana también podrá realizar operaciones como "Editar" y "Eliminar". Sólo seleccione de la Lista un Proveedor al cual se le aplicará la operación y a continuación haga click sobre el botón correspondiente.

Mercadería Disponible

Para obtener el reporte de mercadería disponible de los Activos Corrientes Consumibles, Activos Corrientes de Fabricación y Activos Corrientes para la venta presenta el siguiente formulario.

Ventana de Mercadería Disponible

La Lista mostrada a continuación, describe cada uno de los campos de los datos que se visualizan en el formulario de Mercadería Disponible:

- Tipo de Articulo.- Va a permitir seleccionar una de las opciones que posee de acuerdo al tipo de Activo que deseamos visualizar el reporte de Mercadería disponible.

Una ves seleccionada una de las respectivas opciones el sistema desplazara los Activos Corrientes que se encuentran dentro del Tipo de producto seleccionado con el siguiente formato:

TIPO DE MATERIAL : Activos Creados Venta

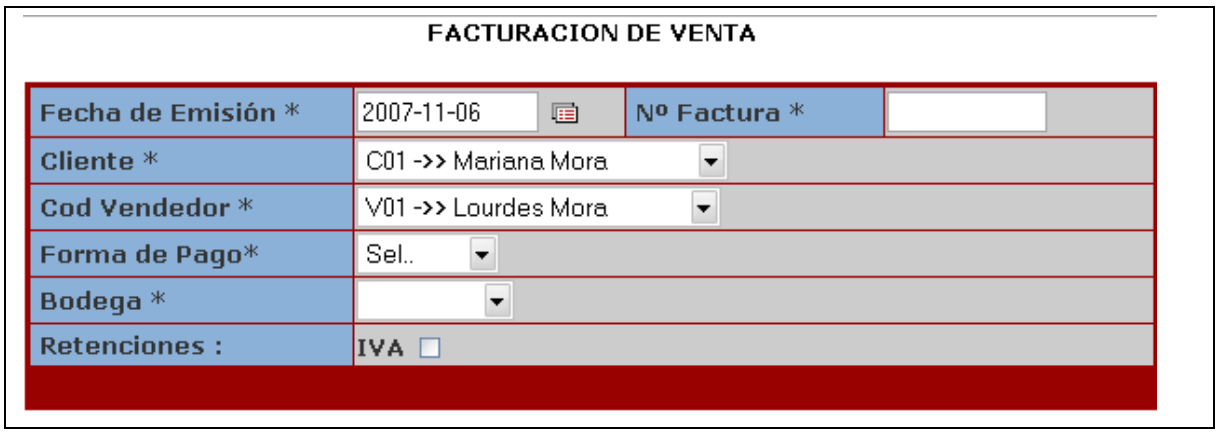
CODIGO	NOMBRE	DISPONIBLE	VALOR c/u
ACV002	Gallinas Ponedo	2793	2.50
ACV004	Pollitas Bebes	100	1.25
ACV01	Huevos Medianos	105	0.15

.

Ventana Reporte de la Mercadería Disponible

Factura de Venta

En la parte inferior de la Ventana Principal de Inventario, se encuentra el menú de Ventas permitiendo seleccionar la opción de Factura de Venta, generada para el Cliente seleccionado y el Vendedor responsable:

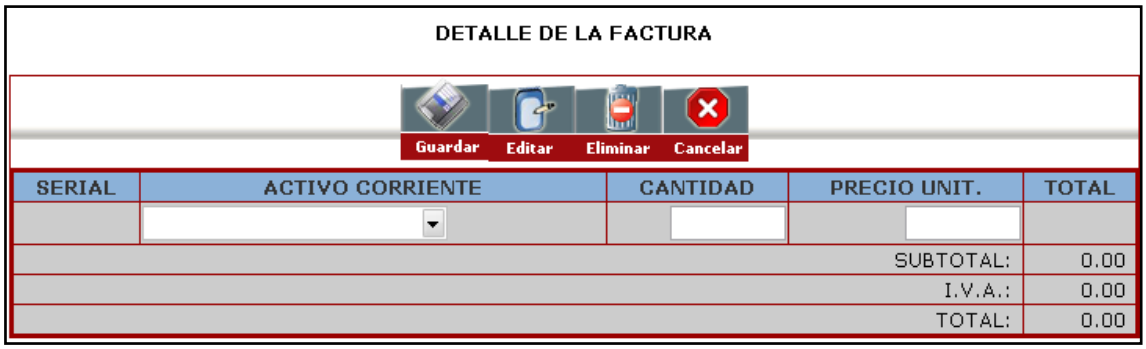


Ventana de Facturas de Venta

Para editar una Factura de Venta haga click en el botón “Insertar” y tiene los siguientes campos:

- En 'Fecha de Emisión', Registra el día y mes en el que se vendió.
- En 'Numero de Factura', registra el numero o código con el que se identifica una factura.
- En 'Cliente', Permitirá elegir el código y nombre del cliente a quien se le realizo la venta.
- En 'Vendedor', Permitirá elegir el código y nombre del vendedor quien es el responsable de la venta.
- En 'Forma de Pago', permite seleccionar una de las dos opciones de pago (contado o a crédito).
- En 'Bodega', Permitirá registrar de que galpón o bodega se realizo la venta del producto.
- En 'Retención', permitirá dar click en la opción de porcentaje (IVA).

Para ingresar los datos en la parte de detalle se debe de dar click en el botón de “Guardar”, inmediatamente se desplazara el detalle de la factura que consta de:



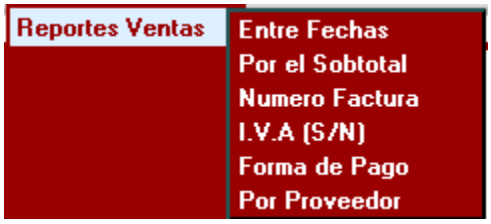
Ventana de Detalle de Facturas de Venta

- En 'Activo Corriente', seleccionamos uno de los productos que la empresa oferta al mercado.
- En 'Cantidad', Registra la cantidad del producto que se vendió.
- En 'Precio Unitario', Registra el costo del producto por unidad.
- En 'Total ', Registra el calculo automático de la cantidad del producto. Luego de haber llenado todos los campos correspondientes, presione nuevamente el botón "Guardar". Para continuar con la operación, finalmente presione el botón "Aprobar" para que se registre.

Se Utilizara los botones "Eliminar" y "Editar" antes y después que se aprobó el registro, seleccionando la factura con la que se pretende ejecutar una de las dos opciones y automáticamente se actualiza modulo de Activos Corrientes.

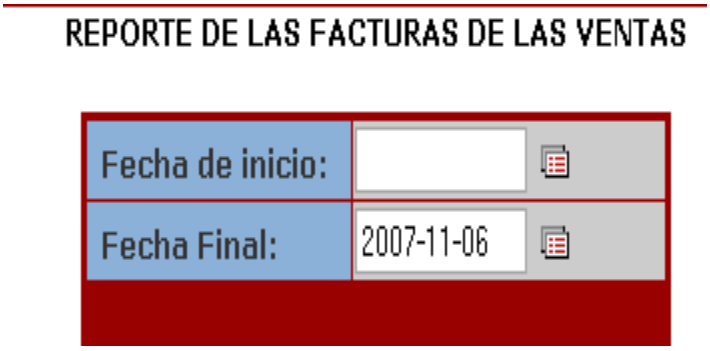
SECCIÓN 4: Reporte de Ventas

Sistema de Administración guarda un registro en el modulo de Reporte de Ventas todas las Ventas realizadas de acuerdo a la fecha, SubTotal, Numero de Factura, IVA(S/N), Forma de Pago, Por Cliente. Este registro se le conoce como Reportes de Ventas donde debe dar click y le presentara las opción de reportes que puede permitir:



Ventana de Reportes de Ventas

Para obtener un reporte de Facturas de Venta entre fechas presenta la siguiente ventana:

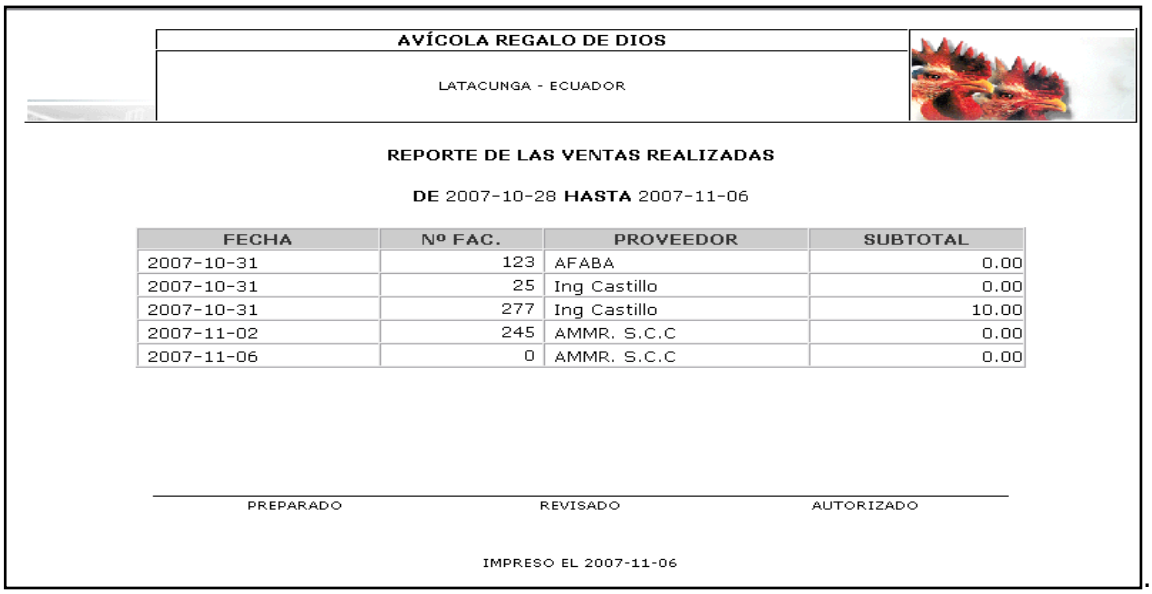


Ventana para reporte de Facturas de Venta por Fechas

Para editar un Reporte de Factura de Venta debe ingresar la información de los siguientes campos:

- En 'Fecha Inicio', introduzca la fecha inicial a tomar en cuenta para generar el reporte (en formato año/mes/día).
- En 'Fecha Final', introduzca la fecha final a tomar en cuenta para generar el reporte. Una ves ingresada la información para continuar con el proceso debe dar click en el botón 

(imprimir) para que se le genere la consulta.



AVÍCOLA REGALO DE DIOS

LATACUNGA - ECUADOR

REPORTE DE LAS VENTAS REALIZADAS

DE 2007-10-28 HASTA 2007-11-06

FECHA	Nº FAC.	PROVEEDOR	SUBTOTAL
2007-10-31	123	AFABA	0.00
2007-10-31	25	Ing Castillo	0.00
2007-10-31	277	Ing Castillo	10.00
2007-11-02	245	AMMR. S.C.C	0.00
2007-11-06	0	AMMR. S.C.C	0.00

PREPARADO REVISADO AUTORIZADO

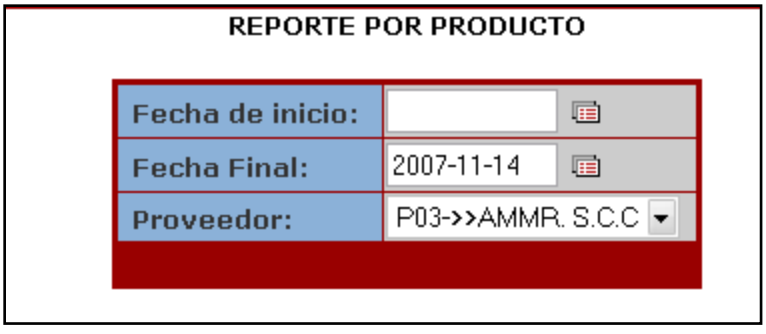
IMPRESO EL 2007-11-06

Ventana de reportes de Facturas de Venta (fechas)

De ese modo se mostrará en forma detallada todas las Facturas emitidas en un periodo de tiempo. Nota .- Para obtener los reportes de Facturas de Venta por las demás opciones que se permite se deberá ingresar los datos de acuerdo a la consulta que se realizara.

SECCIÓN 5: Reporte de Compras

Para obtener un reporte de Facturas de Compra por Proveedor presenta la siguiente ventana:

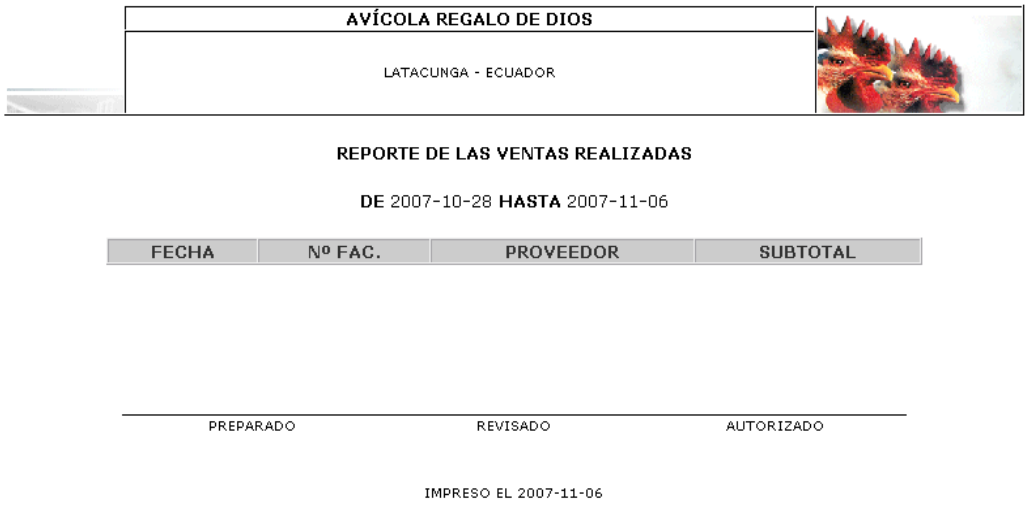


Ventana para reporte de Facturas de Compra por Proveedor

Para editar un Reporte de Factura de Compra debe ingresar la información de los siguientes campos:

- En 'Fecha Inicio', introduzca la fecha inicial a tomar en cuenta para generar el reporte (en formato año/mes/día).
- En 'Fecha Final', introduzca la fecha final a tomar en cuenta para generar el reporte.
- En 'Descripción', introduzca el código y nombre del Activo Corriente del que se desea consultar, Una ves ingresada la información para continuar con el proceso debe dar click en el botón 

(imprimir) para que se le genere la consulta.


AVÍCOLA REGALO DE DIOS

LATACUNGA - ECUADOR

REPORTE DE LAS VENTAS REALIZADAS

DE 2007-10-28 HASTA 2007-11-06

FECHA	Nº FAC.	PROVEEDOR	SUBTOTAL

PREPARADO REVISADO AUTORIZADO

IMPRESO EL 2007-11-06

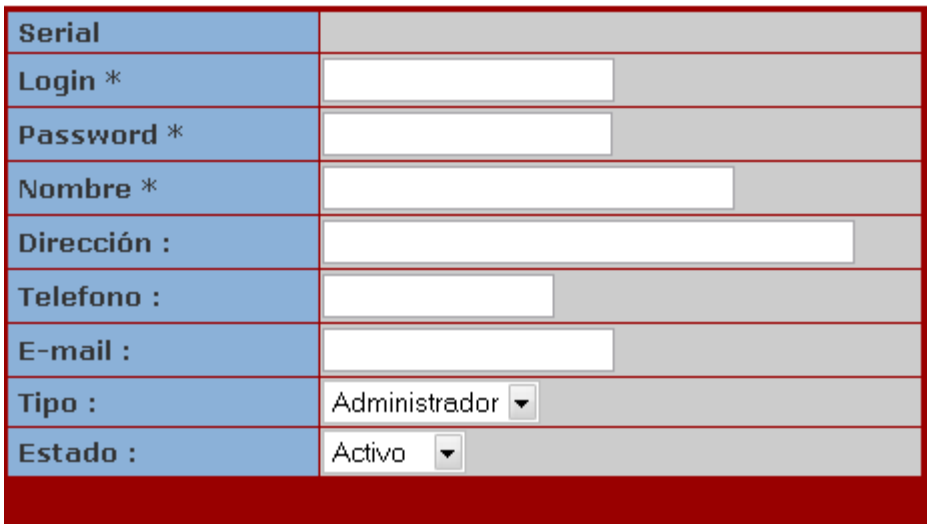
.**Ventana de reportes de Facturas de Compra (Proveedor)**

De ese modo se mostrará en forma detallada todas las Facturas de las compras que realiza la empresa por Proveedor en un periodo de tiempo.

SECCIÓN 6: Parámetros del Sistema

Dentro del modulo de parámetros se maneja el siguiente menú:

Usuarios del Sistema



Ventana para el Manejo de Usuarios

En esta ventana, usted podrá definir los Usuarios que tendrán acceso a Sistema de Administración. Existen tres Tipos de Usuarios: (a) Administrador, el cual tiene acceso a todas las funciones de Sistema, (b) Gerente, el cual tiene acceso a las opciones de consultas de reportes, (c) Secretaria, la cual tiene acceso a ingresar la información y limitado el acceso a las opciones de reportes. Al final de esta sección se detallan las operaciones permitidas y restringidas por cada Tipo.

Para agregar un Usuario, haga click sobre el botón "Insertar", y llene todos los campos requeridos:

- En 'Login', introduzca el nombre que este Usuario utilizará para identificarse.

- En 'Password', introduzca la Contraseña o Clave de Acceso que este usuario utilizará para que el Sistema de Administración compruebe la autenticidad del Usuario. El campo de Contraseña oculta en forma de asteriscos lo que usted escribe, por medidas de seguridad. Cualquier diferencia entre ambos campos impedirá el Registro del Nuevo Usuario.

- En 'Nombre', introduzca el nombre completo del Usuario.

- En ‘Dirección’, Introduzca la dirección del Usuario.

- En ‘Teléfono’, introduzca el teléfono del Usuario.

- En ‘Email’ introduzca el e-mail personal del Usuario.

- En la parte inferior de la ventana, se encuentran los botones de Tipo de "Usuario" y "Estado", indicando el Tipo que este usuario tendrá en el Sistema de Administración.

Todos los campos son obligatorios. Si cambia de opinión, y no desea registrar el Usuario, haga click sobre el botón "Cancelar" (este se encuentra al lado del campo 'Confirmar Contraseña', y sólo aparece durante el proceso de registro o modificación). Si desea proseguir con el registro, haga click sobre el botón "Insertar ".

Usted puede modificar los datos de cualquier usuario introducido en el Sistema de Administración. Para ello, sólo debe seleccionar el usuario en la Lista, y a continuación, hacer click sobre el botón "Editar". Esto le permitirá realizar las modificaciones que usted requiera en cualquier campo de los datos del Usuario. Cuando finalice, presione el botón "Guardar".

Para eliminar un usuario completamente del Sistema, selecciónelo y haga click sobre el botón "Eliminar".

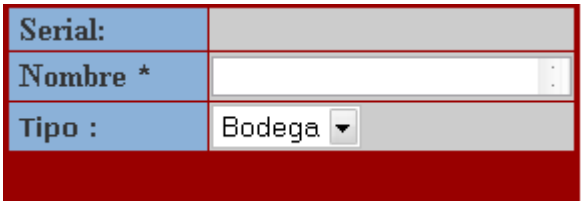
La siguiente tabla muestra los privilegios del Usuario Administrador y los del Usuario Operador.

Operación	**Usuario Administrador**	**Usuario Operador**
Opciones de Impresión	Sí	Sí
Manejar Activos Corrientes	Sí	Sí
Manejar Clientes	Sí	Sí
Emitir Facturas	Sí	Sí
Cambiar de Usuario	Sí, pero si el otro Usuario también es Administrador, se requerirá su Contraseña	Sí, pero se requerirá la Contraseña del otro Usuario
Consultar / Manejar Registro	Sí	No
Manejar Usuarios	Sí	No
Manejar Impuestos	Sí	No

Podrá configurar el resto de las opciones de Sistema de Administración desde su Ventana Principal, a través del menú "Principal".

Bodega

Manejará sitios que representan los lugares necesarios donde se almacenara los productos que la empresa adquiere. De esa manera, podrá visualizar correctamente bodegas con el nombre de acuerdo a la infraestructura que posee Para ingresar la Bodega, diríjase al menú "Bodega".



Ventana de Definición de Bodega

En esta ventana, llene los campos correspondientes con la información acerca de la Bodega:

- En 'Nombre', introduzca el nombre de la Bodega.
- En ‘Tipo’, Seleccione una de las dos opciones que se le presentan determinando si es una oficina o una Bodega.

Cuando finalice con la introducción de datos, haga click sobre el botón "Guardar".

En esta ventana también podrá realizar operaciones como "Editar" y "Eliminar". Sólo seleccione de la Lista una bodega a cual se le aplicará la operación, y a continuación, haga click sobre el botón correspondiente.

Ventana Principal de Sistema de Administración

De arriba hacia abajo, en la Ventana Principal se encuentran: la Barra de Menú, la Barra de Herramientas, la Lista de Clientes y la ventana de previsualización de la Factura.

Desde la Barra de Menú, usted puede tener acceso a todas las opciones y módulos de Sistema de Administración. La Barra de Herramientas contiene las opciones más comúnmente utilizadas.

La Barra de Herramientas contiene botones relacionados con operaciones sobre los Clientes.

Barra de Herramientas

Para aplicar alguna operación sobre un Cliente, seleccione éste previamente en la Lista de Clientes y luego haga click sobre el botón correspondiente. A continuación se describe brevemente cada uno de los botones de la Barra de Herramientas:

Nuevo Cliente: Se utiliza para admitir nuevos Clientes. Sistema de Administración creará una nueva entrada para este Cliente en la Lista de Clientes.

Guardar Cliente: Se utiliza para almacenar los datos de un Cliente.

Permitirá modificar uno de loa campos del formulario de cliente.

Eliminar.- permitirá borrar de la lista de clientes previamente seleccionándolo.

Cancelar es una de las opciones que permitirá cancelar el proceso que se realizo.

Imprimir.- Se utilizara para imprimir una factura, reporte o kardex.

Salir - Permitirá salir a la pantalla principal de ingreso al sistema.

Home.- Permite salir a la pantalla del Menú Principal del Sistema de Administración, permitiendo seleccionar una de opciones que presenta.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- ✓ En el desarrollo del presente trabajo investigativo, el Estudio y Análisis de los Estándares de Pruebas dentro del proceso de desarrollo del software, ayuda a garantizar el diseño de la Metodología MGCM garantizando la calidad del producto.

- ✓ Con la aplicación de los Estándares de Pruebas ISO, IEEE y ANSI al integrarles dentro del proceso de desarrollo de software provocan la implementación de un sistema de calidad.

- ✓ . La Metodología MGCM (Metodología que Garantiza la Calidad y Mantenimiento del Software), establece hoy en día la mejor opción para diseñar y construir sistemas íntegros, fiables y confiables, aun para las aplicaciones mas complejas.

- ✓ Al estructurar cada una de las fases basadas en la Metodología MGCM dentro del proceso de desarrollo de Software, permitió determinar que la fase de Análisis de Requisitos es fundamental para la ejecución de las fases complementarias.

- ✓ Como resultado del diseño de la Metodología MGCM, se desarrolla el Sistema de Administración (Inventarios y Facturación) para la Granja Avícola “Regalo de Dios “, acorde a los requisitos del cliente, el mismo que contribuirá con las tareas que se realizan, permitiendo de esta manera un mayor rendimiento y aprovechamiento del recurso informático.

5.2 RECOMENDACIONES

✓ Considerar la aplicación de los Estándares de Pruebas ISO, IEEE y ANSI, para alcanzar la calidad optima dentro del proceso de desarrollo del software.

✓ . Es importante recomendar el diseño de la metodología MGCM puesto que permite ser una herramienta fundamental para minimizar la complejidad y nivel de errores dentro del proceso de desarrollo de Software.

✓ Se sugiere realizar el uso práctico de la metodología MGCM, el cual permitirá evaluar y controlar la evolución del ciclo de vida del Software diseñado de acuerdo a las necesidades de la Granja Avícola “Regalo de Dios”.

BIBLIOGRAFIA

1. PRESSMAN Roger, Ingeniería de Software, 4ta edición, New York,1997
2. PIATTINI Mario. G, Calidad en el desarrollo y mantenimiento del Software,1era edición, México,2002
3. HAUG Michael,OLSEN Eric,Software Quality Approaches: Testing, Verification, and Validation, 1ra edition, Berlin Heidelberg New York,2001
4. ZEYU GAO Jerry, TSAO Jacob,Testing and Quality Assurance For Component-Based Software, 2da edition,Boston.London,2003.
5. http://congreso.hispalinux.es/congreso2001/actividades/ponencias/robles.pdf ,2006
6. http://sunsite.uakom.sk/sunworldonline/swol-08-1996
7. http://www.shu.ac.uk/tfre/web.links.html,2004
8. http://sunset.usc.edu/publications/TECHRPTS/2000/usccse2000501/usccse2000-501.pdf ,2004
9. http://ltsc.ieee.org/ 2004
10. http://www.internettime.com 2002
11. http://www.digitalearth.gov/ 2006
12. http://geogratis.ISO. cgdi.gc.ca/ 2006
13. http://clearinghouse.cnr.gob.sv/metadatos/princ.htm ,2006
14. http://www.google.com/search?q=cache:CTycFYk8c5UJ:adala.org/encuentros/jasl2/ponencias/008.pdf+estandar+ANSI&hl=es&ie=UTF-8,2006
15. http://www.lcc.uma.es/~av/Publicaciones/02/CalidadDSBC.pdf, 2005
16. http://www.fi.uba.ar/materias/7500/schenonetesisdegradoingenieriainformatica.pdf,200
17. http://window.to/concepcion.com.do, 2000
18. http://www.mvp-access.com/rubenvigon/pdf/testing_software.pdf,2003.
19. http://www.lsi.us.es/docs/doctorado/memorias/Memoria_ProyectoInvestigador.pdf,2005[1]
20. http://www.procuno.com/users/taller/Presentaciones/PresentacionISO.ppt.,2005
21. http://www.lcc.uma.es/~av/misConfs/Calidad%20de%20Componentes%20CR%20Junio%202004.ppt,2004

22. http://www.yahoo.com/Computers_and_Internet/Software/Testing,2000
23. http://quercusseg.unex.es/jhernandez/dsoa03/papers/proceedings.pdf.
24. The Institute of Electrical and Electronics Engineers, Inc.345 East 47th Street, NewYork, NY 10017-2394, USA
25. http://www.fing.edu.uy/inco/cursos/ingsoft/iis/files/examenes/ex_030226.doc,2002
26. http://www.lisi.usb.ve/publicaciones/02%20calidad%20sistemica/calidad_29.pdf, 2003
27. http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion-Evaluacion-6.pdf, 2004
28. http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion-Evaluacion-6.pdf, 2005
29. http://www.lsi.us.es/docs/doctorado/memorias/1147448819Memoria_ProyectoInvestigador.pdf,2004
30. http://www.yahoo.com/Computers_and_Internet/Software/Testing,2004
31. http://www.softwaretechnews.com/stn5-4/inspections.html,2000
32. http://www.iie.org.mx/bolAT02/aplica1.pdf, 2000
33. http://blogs.msdn.com/askburton/archive/2004/09/20/232065.aspx, 2005
34. http://msdn.microsoft.com/architecture/journ/default.aspx?pull=/library/enus/dnmaj/html/aj3softfac.asp
35. http://www.componentsource.com/Services/WhatAreComponents.asp?bhcp=1, 2006
36. http://www.willydev.net/descargas/oguzman-diseno_pruebas.pdf, 2003
37. http://window.to/concepcion.com.do,2005
38. http://www.lsi.us.es/~javierj/investigacion_ficheros/Mejorando%20casos%20de%20uso%2003.pdf.
39. http://www.turismo.uma.es/turitec/turitec2004/docs/actas_turitec_pdf/15.pdf2004
40. http://tecnologiaedu.us.es/bibliovir/pdf/30.pdf., 2004
41. http://martinfowler.com/articles/newMethodology.html ,2005
42. http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf., 2004
43. http://junit.sf.net/doc/testinfected/testing.htm,2001
44. http://is.ls.fi.upm.es/udis/docencia/erdsi/Documentacion-Evaluacion-6.pdf.,2005
45. http://lsi.ugr.es/~fguti/taller/05/griman.pdf, 2004
46. http://griho.udl.es/castella/equip/invest/granollers.html,2005
47. http://www.fi.uba.ar/materias/7547/Documento%20tipo%20para%20proyectos%20de%20Desarrollo.rtf,2003.

48. http://www.dsic.upv.es/asignaturas/facultad/lsi/index.html),2002
49. http://www.eqsoft.net/manuales/MEJORPR1.pdf, 2003
50. http://www.info-ab.uclm.es/personal/caballer/download/papers/CursoVerano,2003
51. http://www.pcm.gob.pe/portal_ongei/Banco_Normas/ITPROC_CICLO_VIDA_SW.pdf,2004
52. http://quercusseg.unex.es/jhernandez/dsoa03/papers/proceedings.pdf,2001
53. http://dis.um.es/jmolina/as.htm,2005
54. http://www.mvp-access.com/rubenvigon/pdf/testing_software.pdf,2001
55. http//www.yahoo.com/Computers_and_Internet/Software/Testing/diagramas
56. ,2005
57. http://www.inf.uach.cl/rvega/asignaturas/info265/G_Calidad.pdf,2004.

ANEXOS

ANEXO I

GALPON DE PRODUCCION DE HUEVOS PARA EL CONSUMO HUMANO

ANEXO II

GALPON DE PRODUCCION DE HUEVOS PARA EL CONSUMO HUMANO

ANEXO III

PLANTA PROCESADORA DE ALIMENTO BALANCEADO

ANEXO IV

BODEGA DE ALMACENAMIENTO DE MATERIAS PRIMAS Y BALANCEADOS