UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TEMA

"SISTEMA INFORMÁTICO DE SUBASTA GANADERA (SUBGANA)."

APLICATIVO

"MÓDULO FINANCIERO"

Autor: Claudio Rafael Guerrón Andrade

Director: Ing. Xavier Mauricio Rea Peñafiel

Ibarra – Ecuador

2012

CERTIFICACIÓN

Certifico que la Tesis "**SISTEMA INFORMÁTICO DE SUBASTA GANADERA (SUBGANA)"** con el aplicativo **"MÓDULO FINANCIERO**" ha sido realizada en su totalidad por el señor: Claudio Rafael Guerrón Andrade portador de la cédula de identidad número: 100305576-9.

……

Ing. Mauricio Rea

Director de la Tesis

CERTIFICACIÓN

Ibarra, 23 de julio de 2012

Señores
UNIVERSIDAD TÉCNICA DEL NORTE
Presente

De mis consideraciones.-

Siendo auspiciantes del proyecto de tesis del Egresado CLAUDIO RAFAEL GUERRÓN ANDRADE con CI: 100305576-9 quien desarrolló su trabajo con el tema “SISTEMA INFORMÁTICO DE SUBASTA GANADERA (SUBGANA)” con el aplicativo "MÓDULO FINANCIERO", me es grato informar que se han superado con satisfacción las pruebas técnicas y la revisión de cumplimiento de los requerimientos funcionales, por lo que se recibe el proyecto como culminado y realizado por parte del egresado: CLAUDIO RAFAEL GUERRÓN ANDRADE. Una vez que hemos recibido la capacitación y documentación respectiva, nos comprometemos a continuar utilizando el mencionado aplicativo en beneficio de nuestra empresa/institución.

El egresado CLAUDIO RAFAEL GUERRÓN ANDRADE puede hacer uso de este documento para los fines pertinentes en la Universidad Técnica del Norte.

Atentamente,

_____
Eco. Hernán Valencia
GERENTE GENERAL
EP-FYPROCAI

UNIVERSIDAD TÉCNICA DEL NORTE

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE INVESTIGACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, CLAUDIO RAFAEL GUERRÓN ANDRADE, con cedula de identidad Nro. 1003055769, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de propiedad intelectual del Ecuador, articulo 4, 5 y 6, en calidad de autor del trabajo de grado denominado: "**SISTEMA INFORMÁTICO DE SUBASTA GANADERA (SUBGANA)"** con el aplicativo **"MÓDULO FINANCIERO**", que ha sido desarrollada para optar por el título de Ingeniería en Sistemas Computacionales, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes mencionada, aclarando que el trabajo aquí descrito es de mi autoría y que no ha sido previamente presentado para ningún grado o calificación profesional.

En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la biblioteca de la Universidad Técnica del Norte

……..

Nombre: CLAUDIO RAFAEL GUERRÓN ANDADE

Cédula: 100305576-9

Ibarra a los 23 días del mes de julio del 2012

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA
AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA.

La UNIVERSIDAD TÉCNICA DEL NORTE dentro del proyecto Repositorio Digital institucional determina la necesidad de disponer los textos completos de forma digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la universidad.
Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual ponemos a disposición la siguiente investigación:

DATOS DE CONTACTO	
CEDULA DE IDENTIDAD	1003055769
APELLIDOS Y NOMBRES	GUERRÓN ANDRADE CLAUDIO RAFAEL
DIRECCIÓN	La Victoria, Carlos Barahona 10-26
EMAIL	claudio_guerron@yahoo.com
TELÉFONO FIJO	062959592
TELÉFONO MOVIL	082627625

DATOS DE LA OBRA	
TITULO	"SISTEMA INFORMÁTICO DE SUBASTA GANADERA (SUBGANA) – MODULO FINANCIERO "
AUTOR	CLAUDIO RAFAEL GUERRÓN ANDRADE
FECHA	23 DE JULIO DEL 2012
PROGRAMA	PREGRADO
TITULO POR EL QUE	INGENIERÍA EN SISTEMAS COMPUTACIONALES
DIRECTOR	ING. MAURICIO REA

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD.

Yo, CLAUDIO RAFAEL GUERRÓN ANDRADE, con cedula de identidad Nro. 100305576-9, en calidad de autor y titular de los derechos patrimoniales de la obra o trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en forma digital y autorizo a la Universidad Técnica del Norte, la publicación de la obra en el Repositorio Digital Institucional y el uso del archivo digital en la biblioteca de la universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión, en concordancia con la Ley de Educación Superior Artículo 143.

……..
Nombre: CLAUDIO RAFAEL GUERRÓN ANDRADE
Cédula: 1003055769
Ibarra a los 23 días del mes de julio del 2012

DEDICATORIA

A Dios, por iluminar mi camino y guiarme hasta llegar a este punto tan importante de mi vida brindándome salud, fortaleza y tranquilidad.

Para ti Madrecita, por cuidarme y brindarme tu apoyo incondicional durante toda mi vida. Por enseñarme el principio de la Solidaridad con nuestros compañeros de aulas; por tu cariño y amor que me has dado a mí, y a mi familia.

A ti Papá que con tu ejemplo de lucha hemos logrado alcanzar nuestras metas, sin olvidar nuestros principios morales y éticos. Por ayudarme en momentos difíciles de mi vida y también por guiarme por el camino del bien.

A ti Vero, por ser mi amiga, compañera y la mejor Esposa del mundo, por demostrarme que cuando las cosas se hacen con amor y tranquilidad salen bien; por brindarme tu amor día a día, por ayudarme y comprenderme en momentos difíciles de mi vida, por llenar de alegría y felicidad nuestro hogar. También por compartir conmigo la carrera universitaria y momentos hermosos dentro de las aulas junto a nuestros grandes amigos y compañeros.

A ti mi Hijit@, aunque eres demasiado chiquito y todavía estas en el vientre de tu mami, te dedico con todo mi amor y cariño este trabajo de grado y que gracias a ti hemos comprendido la importancia de estar juntos y luchar por nuestras metas. Mi amorcito esperamos con alegría tu llegada para quererte y amarte todos los días de nuestras vidas.

A mis hermanos Christian y Marilyn, que siempre fueron mis compañeros y amigos en todas las etapas de la vida. A ti Christian que con tu ejemplo de hermano mayor has fortalecido nuestros deseos y anhelos de ser cada día mejor en todas las actividades que desarrollamos. A ti Marilyn que con tu cariño y ternura lograste comprenderme en etapas críticas de mi vida; también por tu dedicación y ñeque que le pones a las actividades que realizas todos los días.

A mis angelitos que están en el Cielo, Abuelito Claudio, Manuel y mi Abuelita Leticia que gracias a la formación que ellos les dieron a mis padres nosotros somos profesionales y hemos alcanzado nuestros sueños.

AGRADECIMIENTO

A la gloriosa Universidad Técnica del Norte por brindarme la oportunidad de prepararme profesionalmente dentro de sus aulas.

Al Economista Hernán Valencia Gerente de la Empresa Pública de Faenamiento y productos cárnicos EP-FYPROCAI, por brindarnos todas las facilidades para el desarrollo de nuestro trabajo de grado.

A nuestro amigo y director del Proyecto el Ing. Mauricio Rea, que con sus conocimientos ha guiado nuestro trabajo de grado y también a formado grandes profesionales.

A mis compañeros y amigos de la Universidad por los momentos agradables que compartimos dentro y fuera de las aulas, por las malas y buenas noches que tuvimos durante nuestra formación profesional.

A mi familia por permanecer siempre dándome fuerza en todas las actividades que realizo diariamente.

A todas las personas que me ayudaron de una u otra manera a culminar esta etapa de la vida.

TABLA DE CONTENIDOS

INDICE DE FIGURAS

RESUMEN

Actualmente en nuestro medio el desarrollo de los sistemas informáticos para las empresas ha constituido un importante avance en cuanto a la optimización de procesos. El uso de nuevas tecnologías y de la web ha generado un cambio importante y significativo para el sector público y privado dentro del Ecuador.

Los desarrolladores de software dedicados sobre la plataforma JEE buscamos la mejor arquitectura de implementar la aplicación ya que el éxito del proyecto depende de la elección de la arquitectura que brinde la escalabilidad, mantenibilidad y sobre todo la rapidez de las transacciones de la base de datos.

En la Empresa Pública de Faenamiento y Productos Cárnicos de Ibarra (EP-FYPROCAI) se están generando cambios importantes tanto a nivel administrativo como a nivel tecnológico. Es por este motivo que la empresa conjuntamente con el Ilustre Municipio de Ibarra han visto la necesidad de generar proyectos en beneficio de la comunidad, uno de los más importantes es el proyecto de la Subasta ganadera; dicho proyecto ayudará a los pequeños y grandes productores a comercializar su ganado en igualdad de condiciones y con precios justos y competitivos.

Para contribuir con tan importante proyecto se desarrolla un software que permita llevar la parte financiera de una forma organizada y segura, dicho software ayudará a la empresa a registrar la información contable, de inventarios y cuentas por cobrar, permitiendo así la optimización de procesos dentro de la feria ganadera.

SUMMARY

Currently in our development of computer systems for businesses has been a major advance in terms of process optimization. The use of new technologies and the Web has created an important and meaningful to the public and private sector in Ecuador.

Dedicated software developers on JEE platform we implement the best architecture of the application since the project's success depends on the choice of architecture that provides scalability, maintainability, and especially the speed of transactions in the database.

In the Empresa Pública de Faenamiento y Productos cárnicos de Ibarra (EP-FYPROCAI) are generating important changes both at the administrative and technological level. That is why the company together with the Illustrious Municipality of Ibarra have seen the need to generate projects that benefit the community, one of the most important is the project of the livestock auction, the project will help small and large producers market their cattle under the same conditions and with fair and competitive prices.

To contribute to this important project is a software that will promote the financial part of an organized and safe, this software will help the company to record the accounting, inventory and accounts receivable, thus allowing the optimization of processes within the cattle fair.

CAPITULO I
INTRODUCCION

1.1. Antecedentes.

Los procesos de Faenamiento de animales, en los comúnmente denominados camales, datan de hace más de dos décadas atrás, siendo sus instalaciones obsoletas, su equipamiento anticuado, y en general todos sus elementos no han sido modernizados y en muchos casos son los principales generadores de focos infecciosos.

La venta directa en finca ha sido el canal de comercialización más empleado por los pequeños productores antes de la aparición de las subastas, por las dificultades para transportar el ganado a las plantas.

1.2. Problema.

Existe ineficiencia en los procesos de Faenamiento, debido a muchos factores, siendo los más importantes la carencia de instalaciones adecuadas falta de equipamiento y capacitación a los operadores, políticas de control sanitario, políticas de procedencia legal de los animales y control de camales clandestinos.

Esta problemática da como resultado pérdidas o no genera ingresos necesarios para mantener y actualizar las instalaciones de un Centro de Faenamiento.

Además no existe un sistema informático que permita llevar los registros de las transacciones financieras, lo que ocasiona que en este proceso se pueda manipular la información de forma manual y no restringida.

1.3. Objetivos.

1.3.1. Objetivo General.

Desarrollar el Módulo Financiero del Sistema Informático (SUBGANA[1]) para la Empresa Pública Municipal de Faenamiento y Productos Cárnicos de Ibarra (EP – FYPROCAI[2]).

[1] **Subgana.-** Nombre del sistema Subasta Ganadera
[2] **EP – FYPROCAI.-** Empresa Pública de Faenamiento y Productos Cárnicos de Ibarra

1.3.2. Objetivos Específicos.

- Automatizar mediante un Sistema Informático el proceso financiero de la venta de ganado.

- Almacenar los registros contables de las transacciones financieras que se realicen durante el proceso de subasta y en los locales comerciales de forma permanente.

- Generar mensualmente los Balances y Reportes Financieros para ver la situación financiera actual de la EP – FYPROCAI.

- Llevar un registro de los pagos mensuales que realizan los dueños de los locales comerciales que existen en la Feria Ganadera.

1.4. Justificación.

El Ilustre Municipio de Ibarra, ha decidido crear una estructura de manejo de productos cárnicos que preste sus servicios tanto local como regionalmente denominado como Sistema Integrado de Manejo de Productos Cárnicos (SIMPC[3]), el mismo que está conformado por tres grandes ejes que comienza por la selección y obtención de materia prima de buena calidad, a costos competitivos, de procedencia legal y en buenas condiciones de salubridad por medio de la SUBASTA GANADERA, posteriormente, esta materia prima entra en el proceso de faenado utilizando las instalaciones de un nuevo centro de Faenamiento, el mismo que trabaja bajo normas y especificaciones rigurosas tanto en su proceso productivo, así como en su funcionamiento. Posteriormente la carne ya faenada sigue la etapa de procesamiento y obtención de productos mediante la Planta Procesadora de Cárnicos, que cuenta con equipos e instalaciones modernas, que garanticen un adecuado funcionamiento, siempre atendiendo las normativas y exigencias que se requieren para el efecto.

Los tres componentes de Sistema Integrado de Manejo de Productos Cárnicos, por su concepción, dimensión y estructuración estarán en capacidad de brindar sus servicios al Cantón Ibarra y a la Región, convirtiendo así a esta iniciativa en un referente a nivel Nacional. La implementación de un Nuevo Centro de Faenamiento

[3] **SIMPC.-** Sistema Integrado de Productos Cárnicos

Regional en la ciudad de Ibarra, obedece a varios factores como los estratégicos, de infraestructura, climáticos, ubicación geográfica, de mercado, entre otros.

Para que haya mayor transparencia en cuanto a la definición del precio por medio de la libre oferta y demanda del ganado, es necesario emplear el mecanismo que consiste en la comercialización de ganado en pie a través de la subasta ganadera.

1.5. Alcance.

El Módulo Financiero para el correcto funcionamiento del sistema, será desarrollado como proyecto innovador, creativo, investigativo y productivo por un ex-alumno de la Facultad de Ingeniería en Ciencias Aplicadas de la Universidad Técnica del Norte.

En el sistema financiero encontramos los siguientes módulos:

- Contabilidad General
- Inventarios de Bodega
- Ventas
- Cuentas por cobrar
- Cuentas por pagar

Dichos submódulos se integran de una forma natural para lograr la máxima completitud y consistencia de los datos provenientes de diferentes fuentes, pero vistos por el usuario como una sola base de datos. Lo anterior permite a la empresa, disponer de información veraz y en el momento oportuno, para la toma de decisiones a los más altos niveles.

Las principales características de este sistema son:

- Seguridad de manejo de la información.
- Procesos transaccionales.

- Interfaz totalmente gráfica y amigable al usuario.
- Escalable.
- Al ser aplicación web no importa el sistema operativo por parte del Usuario.
- Integración total entre módulos.
- Estándar en el manejo de las pantallas y reportes.
- Facilidad de Integración de nuevos módulos, en caso que requiera la empresa.
- Generación automática de asientos contables.
- Desarrollado utilizando tecnología de vanguardia.

CAPÍTULO II
MARCO TEÓRICO

2.1. Servidor de Aplicaciones.

Un servidor de aplicaciones es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general a través de Internet y utilizando el protocolo http. Los servidores de aplicación se distinguen de los servidores web por el uso extensivo del contenido dinámico y por su frecuente integración con bases de datos.

Además, un servidor de aplicaciones es un producto basado en un componente que se encuentra en el plano medio de la arquitectura central de un servidor. Proporciona servicios de middleware[4], es decir, trabaja como un intermediario para la seguridad y el mantenimiento, además de proveer acceso a los datos.

Un servidor de aplicación maneja la mayoría de las transacciones relacionadas con la lógica y el acceso a los datos de la aplicación. La ventaja principal de un servidor de aplicaciones es la facilidad para desarrollarlas, puesto que éstas no necesitan ser programadas y en cambio, se arman a partir de módulos provistos por el servidor de aplicaciones.

2.1.1. Servidores J2EE.

Es una plataforma de programación, parte de la Plataforma Java, para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java con arquitectura de N capas distribuidas y que se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones.

La plataforma Java EE está definida por una especificación. Similar a otras especificaciones del Java Community Process, Java EE es también considerada informalmente como un estándar debido a que los proveedores deben cumplir ciertos requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por The Java Community Process / JCP[5].

[4] **Middleware.-** Middleware es un software de computadora que conecta componentes de software o aplicaciones para que puedan intercambiar datos entre éstas
[5] **Java Community Process**.- Proceso de la Comunidad JAVA

Java EE incluye varias especificaciones de API[6], tales como JDBC[7], RMI[8], e-mail, JMS[9], Servicios Web, XML [10]y define cómo coordinarlos. Java EE también configura algunas especificaciones únicas para Java EE para componentes. Estas incluyen Enterprise JavaBeans, servlets, portlets, JavaServerPages y varias tecnologías de servicios web. Ello permite al desarrollador crear una Aplicación de Empresa portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel.

Tipos de Servidores J2EE.

JBOSS.- Es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE 1.4, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de e-business. Combinando una arquitectura orientada a servicios revolucionaria con una licencia de código abierto, JBoss AS puede ser descargado, utilizado, incrustado y distribuido sin restricciones por la licencia. Por este motivo es la plataforma más popular de middleware para desarrolladores, vendedores independientes de software y, también, para grandes empresas.

WEBSPHERE.- Es una familia de productos de software propietario de IBM, aunque el término se refiere de manera popular a uno de sus productos específicos: WebSphere Application Server (WAS). WebSphere ayudó a definir la categoría de software middleware y está diseñado para configurar, operar e integrar aplicaciones de e-business a través de varias plataformas de red usando las tecnologías del Web. Esto incluye componentes de run-time (como el WAS) y las herramientas para desarrollar aplicaciones que se ejecutarán sobre el WAS.

La familia de productos WebSphere además incluye herramientas para diseñar procesos de negocio (WebSphere Business Modeler), para integrarlos en las aplicaciones

[6]**API.-** Interfaz de Programación de Aplicaciones
[7]**JDBC.-** Java Database Conectivity. Api que permite la ejecución de operaciones sobre base datos desde el lenguaje de programación Java.
[8]**RMI**.- Es un mecanismo ofrecido por java para invocar un método de manera remota.
[9]**JMS.-** Forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java.
[10]**XML.-**Lenguaje de Marcas Extensible. Permite definir la gramática de lenguajes específicos.

existentes (WebSphere Designer) y para ejecutar y monitorizar dichos procesos (WebSphere Process Server, WebSphere Monitor)

ORACLE WEBLOGIC.- Es un servidor de aplicaciones Java EE y también un servidor web HTTP desarrollado por BEA Systems posteriormente adquirida por Oracle Corporation. Se ejecuta en Unix, Linux, Microsoft Windows, y otras plataformas.

WebLogic puede utilizar Oracle, DB2, Microsoft SQL Server, y otras bases de datos que se ajusten al estándar JDBC. El servidor WebLogic es compatible con WS-Security y cumple con los estándares de J2EE 1.3 desde su versión 7 y con la J2EE 1.4 desde su versión 9 y Java EE para las versiones 9.2 y 10.x.

2.1.2. Otros Servidores.

Internet Information Services (IIS).- Internet Information Services o IIS es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS.

Este servicio convierte a una PC en un servidor web para Internet o una intranet, es decir que en las computadoras que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente.

Los servicios de Internet Information Services proporcionan las herramientas y funciones necesarias para administrar de forma sencilla un servidor web seguro.

El servidor web se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas. Por ejemplo, Microsoft incluye los de Active Server Pages (ASP) y ASP.NET. También pueden ser incluidos los de otros fabricantes, como PHP o Perl.

2.2. Lenguaje de Programación.

Un lenguaje de programación es un idioma artificial diseñado para expresar instrucciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse

para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.

Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación.

2.2.1. Tipos de Lenguajes de Programación.

Existen dos tipos de lenguajes claramente diferenciados; los lenguajes de bajo nivel y los de alto nivel.

El ordenador sólo entiende un lenguaje conocido como código binario o código máquina, consistente en ceros y unos. Es decir, sólo utiliza cero y uno para codificar cualquier acción.

Los lenguajes más próximos a la arquitectura hardware se denominan lenguajes de bajo nivel y los que se encuentran más cercanos a los programadores y usuarios se denominan lenguajes de alto nivel.

LENGUAJES DE BAJO NIVEL.- Son lenguajes totalmente dependientes de la máquina, es decir, que el programa que se realiza con este tipo de lenguajes no se puede migrar o utilizar en otras máquinas.

Al estar prácticamente diseñados a medida del hardware, aprovechan al máximo las características del mismo.

Dentro de este grupo se encuentran:

El lenguaje máquina: este lenguaje ordena a la máquina las operaciones fundamentales para su funcionamiento. Consiste en la combinación de ceros y unos para formar las órdenes entendibles por el hardware de la máquina.

Este lenguaje es mucho más rápido que los lenguajes de alto nivel.

La desventaja es que son bastantes difíciles de manejar y usar, además de tener códigos fuente enormes donde encontrar un fallo es casi imposible.

El lenguaje ensamblador es un derivado del lenguaje máquina y está formado por abreviaturas de letras y números llamadas mnemotécnicos. Con la aparición de este lenguaje se crearon los programas traductores para poder pasar los programas escritos en lenguaje ensamblador a lenguaje máquina. Como ventaja con respecto al código máquina es que los códigos fuentes eran más cortos y los programas creados ocupaban menos memoria. Las desventajas de este lenguaje siguen siendo prácticamente las mismas que las del lenguaje ensamblador, añadiendo la dificultad de tener que aprender un nuevo lenguaje difícil de probar y mantener.

LENGUAJES DE ALTO NIVEL.- Son aquellos que se encuentran más cercanos al lenguaje natural que al lenguaje máquina.

Se tratan de lenguajes independientes de la arquitectura del ordenador. Por lo que, en principio, un programa escrito en un lenguaje de alto nivel, lo puedes migrar de una máquina a otra sin ningún tipo de problema.

Estos lenguajes permiten al programador olvidarse por completo del funcionamiento interno de la maquina/s para la que están diseñando el programa. Tan solo necesitan un traductor que entiendan el código fuente como las características de la máquina.

Suelen usar tipos de datos para la programación y hay lenguajes de propósito general (cualquier tipo de aplicación) y de propósito específico (como FORTRAN para trabajos científicos).

2.2.2. Lenguajes de Programación más Utilizados.

JAVA.- Es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Con respecto a la memoria, su gestión no es un problema ya que ésta es gestionada por el propio lenguaje y no por el programador.

C.- Es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL.

Al igual que B, es un lenguaje orientado a la implementación de Sistemas Operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje débilmente tipificado de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

C++.- Es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.

Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.

Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes, tales como ROOT.

Una particularidad del C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales.

C#.- Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un

estándar por la ECMA [11](ECMA-334) e ISO [12](ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Ya existe un compilador implementado que provee el marco Mono - DotGNU, el cual genera programas para distintas plataformas como Windows, Unix y GNU/Linux.

Cabe recalcar que existen muchos lenguajes de Programación por tal motivo hemos citado algunos de los más utilizados por desarrolladores en todo el mundo.

En la figura siguiente mostramos el ranking de los lenguajes de Programación más utilizados durante el año 2011 hasta el mes de Abril.

Position Apr 2011	Position Apr 2010	Delta in Position	Programming Language	Ratings Apr 2011	Delta Apr 2010	Status
1	2	↑	Java	19.043%	+0.99%	A
2	1	↓	C	16.162%	-1.90%	A
3	3	=	C++	9.225%	-0.48%	A
4	6	↑↑	C#	7.185%	+2.75%	A
5	4	↓	PHP	6.584%	-3.08%	A
6	7	↑	Python	4.931%	+0.73%	A
7	5	↓↓	(Visual) Basic	4.682%	-1.71%	A
8	11	↑↑↑	Objective-C	4.386%	+2.10%	A
9	8	↓	Perl	1.991%	-1.56%	A
10	10	=	JavaScript	1.513%	-0.96%	A
11	12	↑	Ruby	1.482%	-0.74%	A
12	20	↑↑↑↑↑↑↑↑	Lua	1.035%	+0.51%	A
13	9	↓↓↓↓	Delphi	1.034%	-1.68%	A
14	-	=	Assembly	0.967%	0.00%	A
15	23	↑↑↑↑↑↑↑↑	Lisp	0.934%	+0.45%	A
16	32	↑↑↑↑↑↑↑↑↑↑	Ada	0.768%	+0.41%	A
17	16	↓	Pascal	0.713%	+0.06%	A
18	21	↑↑↑	Transact-SQL	0.583%	+0.08%	B
19	-	=	Scheme	0.581%	0.00%	B
20	15	↓↓↓↓↓	Go	0.557%	-0.15%	A-

Ilustración 1. Ranking de Lenguajes de Programación (Fuente: http://www.dosbit.com)

[11]**ECMA.-** Es una organización internacional basada en membrecías de estándares para la comunicación y la información.
[12]**ISO.-** Organización internacional de Normalización.

2.3. Métricas del Software.

2.3.1. Introducción.

- Se aplica las métricas para valorar la calidad de los productos de ingeniería o los sistemas que se construyen.

- Proporcionan una manera sistemática de valorar la calidad basándose en un conjunto de reglas claramente definidas.

- Se aplican a todo el ciclo de vida permitiendo descubrir y corregir problemas potenciales.

2.3.2. Factores de Calidad del Software.

Los requisitos del Software son la base de las medidas de calidad. La falta de concordancia con los requisitos es una falta de calidad.

Unos estándares específicos definen un conjunto de criterios de desarrollo que guían la manera en que se hace la ingeniería del Software. Si no se siguen los criterios, habrá seguramente poca calidad.

Existe un conjunto de requisitos implícitos que a menudo no se nombran. Si el software cumple con sus requisitos explícitos pero falla en los implícitos, la calidad del software no será fiable.

2.3.3. Factores de calidad McCall

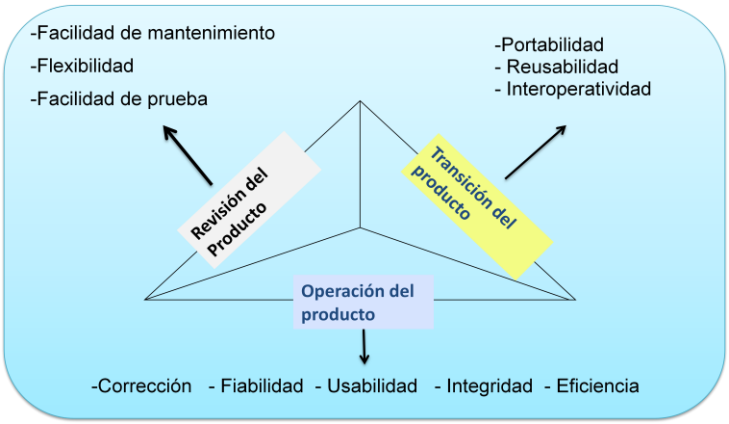

Ilustración 2. Factores de Calidad McCall (Fuente: export.writer.zoho.com)

Operación del Producto.

Corrección: Hasta donde satisface un programa su especificación y logra los objetivos del cliente.

Fiabilidad: Hasta dónde se puede esperar que un programa lleve a cabo de su función con la exactitud requerida.

Eficiencia: La cantidad de recursos informáticos y de código necesarios para que un programa realice su función.

Integridad: Hasta dónde se puede controlar el acceso al software o a los datos por personas no autorizadas.

Usabilidad (facilidad de manejo): El esfuerzo necesario para aprender a operar los datos de entrada e interpretar las salidas de un programa.

Revisión del Producto.

Facilidad de mantenimiento: El esfuerzo necesario para localizar y arreglar un error en un programa.

Flexibilidad: El esfuerzo necesario para modificar un programa operativo.

Facilidad de prueba: El esfuerzo necesario para probar un programa para asegurarse de que realiza su función pretendida.

Transición del Producto.

Portabilidad: El esfuerzo necesario para transferir el programa de un entorno de sistema hardware y/o software a otro entorno diferente.

Reusabilidad (capacidad de reutilización): Hasta donde se puede volver a emplear un programa (o partes de un programa) en otras aplicaciones.

Interoperatividad: El esfuerzo necesario para acoplar un sistema con otro.

2.4. Framework del Desarrollo Web.

La palabra inglesa framework define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

En el desarrollo de software, un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, con base a la cual otro proyecto de software puede ser más fácilmente organizado y desarrollado.

Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio, y provee una estructura y una especial metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio.

2.4.1. Objetivos del Framework.

- Desarrollo rápido.
- Desarrollo estructurado.
- Reutilización de código.
- Disminuir el esfuerzo en el desarrollo.
- Aprovechamos las funcionalidades ya implementadas.
- No debemos reinventar la rueda.
- Nos concentramos directamente en la solución del problema.
- Tener como aliado a las metodologías de desarrollo ágiles.

2.4.2. Arquitectura.

Dentro de este aspecto, podemos basarnos en el modelo MVC [13](Controlador-Modelo-Vista), ya que debemos fragmentar nuestra programación. Tenemos que contemplar estos aspectos básicos en cuanto a la implementación de nuestro sistema:

[13]**MVC.-**Modelo Vista Controlador

Controlador.

Con este apartado podemos controlar el acceso a nuestra aplicación, y esto puede incluir: archivos, scripts, y/o programas; cualquier tipo de información que permita la interfaz. Así, podremos diversificar nuestro contenido de forma dinámica, y estática; sólo debemos controlar ciertos aspectos.

Modelo.

Esta parte del MVC es la capa para acceder a datos desde la BDD.

Vista.

Al final, a este miembro de la familia le corresponde dibujar, o expresar la última forma de los datos: la interfaz gráfica que interactúa con el usuario final del programa. Después de todo, a esta parte le toca evidenciar la información obtenida hasta hacerla llegar al controlador.

2.4.3. Listado de Principales Frameworks JavaScript y Ajax.

- Mootools
- JQuery
- Prototype
- YUI
- Dojo
- Qooxdoo
- GWT Google Web Toolkit
- Rico
- Ext JS

2.4.4. Listado de Principales Frameworks PHP.

- Zend
- Symfony
- Seagull
- Prado
- CakePHP

2.4.5. Listado de Principales Frameworks JAVA.

- Spring Framework
- Struts
- Hibernate
- JSF
- JavaFX
- Flex
- DOJO.
- Seam.

2.5. Metodología RUP [14](RATIONAL UNIFIED PROCESS).

Es un proceso de Ingeniería del Software. Proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo. Su propósito es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costos y calendario predecibles. En definitiva el RUP es una metodología de desarrollo de software que intenta integrar todos los aspectos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos de software.

2.5.1. Características.

Guiado/Manejado por casos de uso: La razón de ser de un software es servir a usuarios ya sean humanos u otros sistemas; un caso de uso es una facilidad que el software debe proveer a sus usuarios. Los casos de uso reemplazan la antigua especificación funcional tradicional y constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema.

Centrado en Arquitectura: La arquitectura involucra los elementos más significativos del sistema y está influenciada entre otros por plataformas de software, sistemas operativos, manejadores de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados y requerimientos no funcionales. Es como una radiografía del sistema que estamos desarrollando, lo suficientemente completa como para que todos

[14] **RUP.** Rational Unified Process

los implicados en el desarrollo tengan una idea clara de qué es lo que están construyendo, pero lo suficientemente simple como para que si se quita algo, una parte importante del sistema quede sin especificar. Se representa mediante varias vistas que se centran en aspectos concretos.

Iterativo e Incremental: Para hacer más manejable un proyecto se recomienda dividirlo en ciclos. Para cada ciclo se establecen fases de referencia, cada una de las cuales debe ser considerada como un miniproyecto cuyo núcleo fundamental está constituido por una o más iteraciones de las actividades principales básicas de cualquier proceso de desarrollo. En concreto RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades.

2.5.2. Fases de la Metodología.

Inicio.- Antes de iniciar un proyecto es conveniente plantearse algunas cuestiones: ¿Cuál es el objetivo? ¿Es factible? ¿Lo construimos o lo compramos? ¿Cuánto va a costar? La fase de inicio trata de responder a estas preguntas y a otras más. Sin embargo no pretendemos una estimación precisa o la captura de todos los requisitos. Más bien se trata de explorar el problema lo justo para decidir si vamos a continuar o a dejarlo. Generalmente no debe durar mucho más de una semana.

Los objetivos son:

- Establecer el ámbito del proyecto y sus límites.
- Encontrar los casos de uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
- Mostrar al menos una arquitectura candidata para los escenarios principales.
- Estimar el coste en recursos y tiempo de todo el proyecto.
- Estimar los riesgos, las fuentes de incertidumbre.

Al terminar la fase de inicio se deben comprobar los criterios de evaluación para continuar:

- Todos los interesados en el proyecto coinciden en la definición del ámbito del sistema y las estimaciones de agenda.

- Entendimiento de los requisitos, evidenciado por la fidelidad de los casos de uso principales.
- Las estimaciones de tiempo, coste y riesgo son creíbles.
- Comprensión total de cualquier prototipo de la arquitectura desarrollado.
- Los gastos hasta el momento se asemejan a los planeados.
- Si el proyecto no pasa estos criterios hay que plantearse abandonarlo o repensarlo profundamente.

Elaboración.- El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos.

Cuando termina esta fase se llega al punto de no retorno del proyecto: a partir de ese momento pasamos de las relativamente ligeras y de poco riesgo dos primeras fases, a afrontar la fase de construcción, costosa y arriesgada. Es por esto que la fase de elaboración es de gran importancia.

En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final. Este prototipo debe contener los casos de uso críticos identificados en la fase de inicio. También debe demostrarse que se han evitado los riesgos más graves, bien con este prototipo, bien con otros de usar y tirar.

Los objetivos de esta fase son:

- Definir, validar y cimentar la arquitectura.
- Completar la visión.
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones. Debe incluir los costes si procede.
- Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en un tiempo razonable.

Al terminar deben obtenerse los siguientes productos:

- Un modelo de casos de uso completa al menos hasta el 80%: todos los casos y actores identificados, la mayoría de los casos desarrollados.

- Requisitos adicionales.
- Descripción de la arquitectura software.
- Un prototipo ejecutable de la arquitectura.
- Lista de riesgos y caso de negocio revisados.
- Plan de desarrollo para el proyecto.
- Un caso de desarrollo actualizado que especifica el proceso a seguir.
- Posiblemente un manual de usuario preliminar.
- La forma de aproximarse a esta fase debe ser tratar de abarcar todo el proyecto con la profundidad mínima. Sólo se profundiza en los puntos críticos de la arquitectura o riesgos importantes.

En la fase de elaboración se actualizan todos los productos de la fase de inicio el glosario, el caso de negocio.

Los criterios de evaluación de esta fase son los siguientes:

- La visión del producto es estable.
- La arquitectura es estable.
- Se ha demostrado mediante la ejecución del prototipo que los principales elementos de riesgo han sido abordados y resueltos.
- El plan para la fase de construcción es detallado y preciso. Las estimaciones son creíbles.
- Todos los interesados coinciden en que la visión actual será alcanzada si se siguen los planes actuales en el contexto de la arquitectura actual.
- Los gastos hasta ahora son aceptables, comparados con los previstos.
- Si no se superan los criterios de evaluación quizá sea necesario abandonar el proyecto o replanteárselo considerablemente.

Construcción.- La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todos los componentes, características y requisitos, que no lo hayan sido hecho hasta ahora, han de ser implementados, integrados y testeados, obteniéndose una versión del producto que se pueda poner en manos de los usuarios (una versión beta).

El énfasis en esta fase se puede controlar las operaciones realizadas, administrando los

recursos eficientemente, de tal forma que se optimicen los costes, los calendarios y la calidad.

Los objetivos concretos incluyen:

- Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
- Conseguir una calidad adecuada tan rápido como sea práctico.
- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) tan rápido como sea práctico.

Los productos de la fase de construcción según deben ser:

- Modelos Completos (Casos de Uso, Análisis, Diseño, Despliegue e Implementación).
- Arquitectura íntegra (mantenida y mínimamente actualizada).
- Riesgos Presentados Mitigados.
- Plan del Proyecto para la fase de Transición.
- Manual Inicial de Usuario (con suficiente detalle).
- Prototipo Operacional – beta.
- Caso del Negocio Actualizado.

Transición.- La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que típicamente se requerirá desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y usabilidad del producto.

En concreto se citan algunas de las cosas que puede incluir esta fase:

- Testeo de la versión Beta para validar el nuevo sistema frente a las expectativas de los usuarios.
- Funcionamiento paralelo con los sistemas legados que están siendo sustituidos por nuestro proyecto.
- Conversión de las bases de datos operacionales.
- Entrenamiento de los usuarios y técnicos de mantenimiento.
- Traspaso del producto a los equipos de marketing, distribución y venta.

Los principales objetivos de esta fase son:

- Conseguir que el usuario se valga por sí mismo.
- Un producto final que cumpla los requisitos esperados, que funcione y satisfaga suficientemente al usuario.

Los productos de la fase de transición son:

- Prototipo Operacional
- Documentos Legales
- Caso del Negocio Completo
- Línea de Base del Producto completa y corregida que incluye todos los modelos del sistema
- Descripción de la Arquitectura completa y corregida

Las iteraciones de esta fase irán dirigidas normalmente a conseguir una nueva versión.
Las actividades a realizar durante las iteraciones dependerán de su finalidad, si es corregir algún error detectado, normalmente será suficiente con llevar a cabo los flujos de trabajo de implementación y test, sin embargo, si se deben añadir nuevas características, la iteración será similar a la de una iteración de la fase de construcción.

La complejidad de esta fase depende totalmente de la naturaleza del proyecto, de su alcance y de la organización en la que deba implantarse.

2.5.3. Ciclo de Vida.

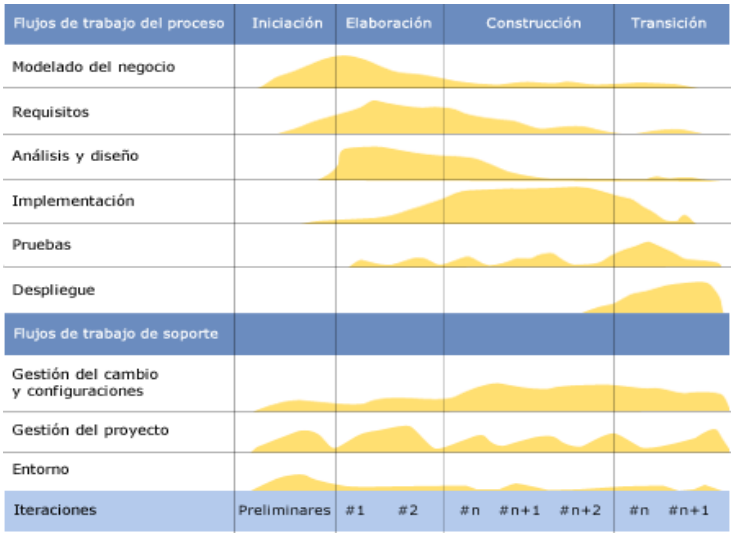

Ilustración 3. Ciclo de Vida de RUP (Fuente: http://www.scribd.com/doc/297224/RUP)

En el ciclo de vida del RUP se observa una implementación de desarrollo en espiral. Con el ciclo de vida se establecen tareas en fases e iteraciones. El RUP maneja el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable.

Las primeras iteraciones (en las fases de Inicio y Elaboración) se enfocan hacia la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, la eliminación de los riesgos críticos, y al establecimiento de una base de inicio.

2.5.4. Artefactos[15].

El RUP en cada una de sus fases realiza una serie de artefactos que sirven para comprender mejor tanto el análisis como el diseño del sistema estos artefactos son los siguientes:

Inicio:

- Documento Visión.
- Especificación de Requerimientos.

Elaboración:

- Diagramas de caso de uso.

Construcción:

Documento Arquitectura que trabaja con las siguientes vistas:

- Vista Lógica:
  - Diagrama de clases.
  - Modelo E-R (Si el sistema así lo requiere).
- Vista de Implementación:
  - Diagrama de Secuencia.
  - Diagrama de estados.
  - Diagrama de Colaboración.

[15] **Artefactos.-** Documentos que se realiza durante las fases de la metodología RUP.

- Vista Conceptual:
    - Modelo de dominio.
- Vista física:
    - Mapa de comportamiento a nivel de hardware.

2.6. Contabilidad.

La Contabilidad es la ciencia social, que se encarga de estudiar, medir y analizar el patrimonio de las empresas y de los individuos, con el fin de servir en la toma de decisiones y control, presentando la información, previamente registrada, de manera sistemática y útil para las distintas partes interesadas. Posee además una técnica que produce sistemáticamente y estructuradamente información cuantitativa y valiosa, expresada en unidades monetarias acerca de las transacciones que efectúan las Entidades económicas y de ciertos eventos económicos identificables y cuantificables que la afectan, con la finalidad de facilitarla a los diversos públicos interesados.

El producto final de la contabilidad son todos los estados contables o estados financieros que son los que resumen la situación económica y financiera de la empresa. Esta información resulta útil para gestores, reguladores y otros tipos de interesados como los accionistas, acreedores o propietarios.

Es la utilización de ciertos principios al registrar, clasificar y sumar, en términos monetarios, datos financieros y económicos, para informar en forma oportuna y fehaciente las operaciones de la vida de una empresa.

<u>Características.</u>

- Rendición de informes a terceras personas sobre el movimiento financiero de la empresa.
- Cubrir la totalidad de las operaciones del negocio en forma sistemática, histórica y cronológica.

- Debe implantarse necesariamente en la compañía para informar oportunamente de los hechos desarrollados.

- Se utiliza el lenguaje de los negocios.

- Se basa en reglas, principios y procedimientos contables para el registro de las operaciones financieras de un negocio.

- Describe las operaciones en el engranaje analítico de la teneduría de la partida doble.

Importancia de la contabilidad.

La contabilidad es de gran importancia porque todas las empresas tienen la necesidad de llevar un control de sus negociaciones mercantiles y financieras. Así obtendrá mayor productividad y aprovechamiento de su patrimonio. Por otra parte, los servicios aportados por la contabilidad son imprescindibles para obtener información de carácter legal.

Naturaleza de la contabilidad. Ciencia, técnica o tecnología.

Ciencia: Puesto que es un conocimiento verdadero. No es una suposición de hechos sin relevancia alguna, al contrario, analiza cada hecho económico y en todos aplica un conocimiento adquirido. Es un conocimiento sistemático, verificable y falible. Busca, a través de la formulación de hipótesis, la construcción de conjuntos de ideas lógicas que sirvan para predecir y explicar los fenómenos relativos a su objeto de estudio. Con el propósito de identificar fenómenos o sucesos que aporten gran información para su mejor desempeño.

Técnica: Porque trabaja con base en un conjunto de procedimientos o sistemas para acumular, procesar e informar datos útiles referentes al patrimonio. Es una serie de pasos para realizar una tarea y en contabilidad la tarea es el registro, la teneduría de libros.

Sistema de información: De acuerdo con las opiniones y enfoques profesionales que intentan dejar de lado el debate acerca de la naturaleza de lo contable, al definir la contabilidad recurren a un término que no implica asumir un carácter científico, técnico o tecnológico. Dicen, por lo tanto, que se trata de "un subsistema dentro del sistema de

información del ente" (dado que no solamente se refiere a empresas), toma toda la información del ente referente a los elementos que definen el patrimonio, la procesa y la resume de tal forma que cumpla con los criterios básicos que uniforman la interpretación de la Información Financiera (contable), de esta manera analistas financieros y no financieros usan la información contable, de ahí se concluye que independientemente de las definiciones anteriores, la contabilidad es en sí un sistema de información.

Metodología contable.

La contabilidad, como ciencia, utiliza un método, denominado método contable, que se compone de cuatro pasos:

- Captación de los hechos de contenido económico susceptibles de ser contabilizados.

- Cuantificación de los hechos contables.

- Representación mediante registro de los hechos en cuentas contables (instrumentos conceptuales) anotados en libros de contabilidad (instrumentos materiales).

- Agregación de la información registrada que se presenta de forma sintetizada en los estados financieros (cuentas anuales).

2.6.1. Marco legal y técnico del ciclo contable.

Para el desarrollo de un sistema contable se debe tener en cuenta algunos cuerpos legales vigentes como son: leyes, reglamentos y estatutos como Leyes tributarias, societarias etc. Así mismo las disposiciones establecidas en los Principios de Contabilidad Generalmente Aceptados (PCGA[16]) y en normas técnicas de contabilidad (NEC[17]) y (NIIF[18]); relacionados específicamente con el desarrollo de un sistema contable. A continuación se hará énfasis en algunos de los PCGA, que establecen la forma correcta de realizar la contabilidad en una empresa: Partimos diciendo que los

[16]**PCGA.-** Principios Contables generalmente aceptados
[17]**NEC.-** Normas Ecuatorianas Contables
[18]**NIIF.-** Normas Internacionales de información financiera

PCGA, son bases de cuantificación de las operaciones y presentación de la información económica y financiera de la empresa a través de los estados financieros.

2.6.2. Conceptos de contabilidad generalmente aceptados.

Conocidos como (PCGA) son un conjunto de reglas generales y normas que sirven de guía contable para formular criterios referidos a la medición del patrimonio y a la información de los elementos patrimoniales y económicos de un ente.

Los PCGA constituyen parámetros para que la confección de los estados financieros sea sobre la base de métodos uniformes de técnica contable.

PRINCIPIOS BASICOS:

Son aquellos que se consideran fundamentales por cuanto orientan la acción de la profesión contable.

Ente contable.

El Ente contable lo constituye la empresa como entidad que desarrolla la actividad económica. El campo de acción de la contabilidad financiera, es la actividad económica de la empresa.

Equidad.

La contabilidad y su información deben basarse en el principio de equidad, de tal manera que el registro de los hechos económicos y su información se basen en la igualdad para todos los sectores sin preferencia para ninguno en particular.

Medición de Recursos.

La contabilidad y la información financiera se fundamentan en los bienes materiales e inmateriales que poseen valores económicos y por tanto susceptibles de ser valuados en términos monetarios.

La contabilidad financiera se ocupa por tanto, en forma especial, de la medición de recursos y obligaciones económicas y los cambios operados en ellos.

Período de tiempo.

La contabilidad financiera provee información acerca de las actividades económicas de una empresa por períodos específicos, los que en comparación con la vida misma de la empresa, son cortos.

Normalmente los períodos de tiempo de un ejercicio y otros son iguales, con la finalidad de poder establecer comparaciones y realizar análisis que permitan una adecuada toma de decisiones.

Las actividades continuas de la empresa son segmentadas con el fin de que la correspondiente información pueda ser preparada y presentada periódicamente.

Esencia sobre la forma.

La contabilidad y la información financiera se basan en la realidad económica de las transacciones.

La contabilidad financiera enfatiza la sustancia o esencia económica del evento, aún cuando la forma legal pueda diferir de la sustancia económica y sugiera diferentes tratamientos.

Continuidad del ente contable

Los principios contables parten del supuesto de la continuidad de las operaciones del ente contable, empresa en marcha, a menos que se indique lo contrario, en cuyo caso se aplicarán técnicas contables de reconocido valor, en atención a las particulares circunstancias del momento.

Si la liquidación de una empresa es inminente, no puede ser considerada como empresa en marcha.

Medición en términos monetarios.

La contabilidad financiera cuantifica en términos monetarios los recursos, las obligaciones y los cambios que se producen en ellos.

La unidad monetaria de medida para la contabilidad y para la información financiera, en el Ecuador es el dólar.

Estimaciones.

Debido a que la contabilidad financiera involucra asignaciones o distribuciones de ciertas partidas, entre períodos de tiempo relativamente cortos de actividades complejas y conjuntas, es necesario utilizar estimaciones o aproximaciones.

La continuidad, complejidad, incertidumbre y naturaleza común de los resultados inherentes a la actividad económica imposibilitan, en algunos casos, el poder cuantificar con exactitud ciertos rubros, razón por la cual se hace necesario el uso de estimaciones.

Acumulación.

La determinación de los ingresos periódicos y de la posición financiera depende de la medición de Recursos y obligaciones económicas. y sus cambios a medida que estos ocurren, en lugar de simplemente limitarse al registro de ingresos y pagos en efectivo.

Para la determinación de la utilidad neta periódica y de la situación financiera, es imprescindible el registro de estos cambios. Esta es la esencia de la contabilidad en base al método de acumulación.

Precio de intercambio.

Las mediciones de la contabilidad financiera están principalmente basadas en precios a los cuales los recursos y obligaciones económicas son intercambiados. La medición en términos monetarios está basada primordialmente en los precios de intercambio.

Juicio o Criterio.

Las estimaciones, imprescindiblemente usadas en la contabilidad, involucran una importante participación del juicio o criterio del profesional contable.

Uniformidad.

Los principios de contabilidad deben ser aplicados uniformemente de un periodo a otro. Cuando por circunstancias especiales se presenten cambios en los principios técnicos y en sus métodos de aplicación deberá dejarse constancia expresa de tal situación, a la vez que informar sobre los efectos que causen en la información contable.

No hay que olvidar que el concepto de la uniformidad permite una mejor utilización de la información y de la presentación de los estados financieros.

Clasificación y contabilización.

Las fuentes de registro de los recursos, de las obligaciones y de los resultados son hechos económicos cuantificables que deben ser convenientemente clasificados y contabilizados en forma regular y ordenada, esto facilita el que puedan ser comprobables o verificables.

Significatividad.

Los informes financieros se interesan únicamente en la información suficiente significativa que pueda afectar las evaluaciones sobre los datos presentados.

Valuación al costo.

Este principio establece que los activos de una empresa deben ser valuados al costo de adquisición o producción, como concepto básico de valuación; asimismo, las fluctuaciones de la moneda común denominador, no deben incidir en alteraciones al principio expresado, sino que se harán los ajustes necesarios a la expresión numeraria de los respectivos costos, por ejemplo ante un fenómeno inflacionario.

Objetividad.

Los cambios en el activo, pasivo y en la expresión contable del patrimonio neto, se deben conocer formalmente en los registros contables, tan pronto como sea posible medirlos objetivamente y expresar dicha medida en términos monetarios.

Las modificaciones en el inventario se deben registrar tal cual es la operación en los libros de contabilidad, para medirlos objetivamente en términos monetarios y así no hacer distorsiones en la realidad de los registros contables.

Prudencia.

Ante la circunstancia de tener que elegir entre dos valores, el contador debe optar por el más bajo, minimizando de esta manera la participación del propietario en las operaciones contables. Este principio general se puede expresar diciendo: “Contabilizar todas las pérdidas cuando se conocen, y las ganancias solamente cuando se hayan percibido”.

Este principio es también llamado criterio conservador.

Ante el hecho que el contador se encuentre entre dos o más caminos razonables a seguir, deberá optar por el que muestre la menor cifra de dos valores de activos relativos a un partida determinada; o ante el caso de registrar una operación éste la hará de modo que la participación del propietario sea la menor posible.

Revelación suficiente.

La información contable debe ser clara y comprensible para juzgar e interpretar los resultados de operación y la situación de la empresa. La información financiera debe ser la correcta y exacta.

Partida Doble.

Es un principio que exige para todo asiento contable uno o varios cargos y uno o varios abonos, siendo la suma de los cargos igual a la suma de los abonos, es decir se debe mantener un equilibrio matemático.

2.6.3. Las Cuentas Contables y El Plan de Cuentas.

Cuenta contable.

Es un término o denominación objetiva, usado en Contabilidad para registrar, clasificar y resumir los incrementos y disminuciones de naturaleza similar, correspondientes a los rubros integrantes del Activo, Pasivo, Patrimonio, Ingresos y Gastos.

Activo: Es un recurso controlado por la entidad como resultado de sucesos pasados, del que la entidad espera obtener, en el futuro, beneficios económicos.

Pasivo: Es una obligación presente de la entidad, surgida a raíz de sucesos pasados, al vencimiento de la cual, espera desprenderse de recursos que incorporan beneficios económicos.

Patrimonio: Es la parte residual de los activos de la entidad, una vez deducidos todos sus pasivos.

Ingresos: Son los incrementos en los beneficios económicos, producidos a lo largo del periodo sobre el que se informa, en forma de entradas o incrementos de valor de los activos, o bien como decrementos de las obligaciones, que dan como resultado aumentos del patrimonio, distintas de las relacionadas con las aportaciones de inversores de patrimonio.

Gastos: Son los decrementos en los beneficios económicos, producidos a lo largo del periodo sobre el que se informa, en forma de salidas o disminuciones del valor de los activos, o bien por la generación o aumento de los pasivos, que dan como resultado decrementos en el patrimonio, distintos de los relacionados con las distribuciones realizadas a los inversores de patrimonio.

Plan General de Cuentas.

Es un listado de cuentas ordenadas metódicamente, ideada de manera específica para una empresa o ente facilitando el desarrollo del proceso contable y la elaboración de estados financieros.

Estructura: El plan de cuentas debe ser estructurado de acuerdo con las necesidades de información de la empresa presentes y futuras. Además debe ser específico y particularizado; sistemático (ordenamiento y presentación), flexible, homogéneo y claro. La estructura de un plan de cuentas se lo realiza en función a grupos, subgrupos y las cuentas en sí.

Contenido: Un plan de Cuentas debe tener un código para cada cuenta principal, y para sus respectivas subcuentas.

Ilustración 4. Ejemplo de Plan de Cuentas

Este es el primer paso y el más importante, si se ingresa correctamente el plan de cuentas a un sistema contable de la empresa, no tendrá dificultades en presentar los informes financieros al cierre de un período o ejercicio económico.

La estructura debe ser agrupada y jerárquica por niveles.

Primer nivel.

Este grupo está dado por los términos de la situación financiera, económica y potencial.

Situación financiera.

- Activos.
- Pasivo.
- Patrimonio.

Situación económica.

- Ingresos.

- Gastos.

Situación potencial.

- Cuentas de orden.

Segundo Nivel.

El subgrupo de las cuentas anteriores está dado por la división racional de los grupos, efectuado bajo algún criterio de uso generalizado.

El activo se ordena bajo el criterio de liquidez

- Activo corriente.
- Activo fijo o propiedad, planta y equipo.
- Diferido y otros activos.

El pasivo se ordena bajo el criterio de exigibilidad

- Pasivo corriente (corto plazo).
- Pasivo fijo (largo plazo).
- Diferidos y otros pasivos.

El patrimonio se ordena bajo el criterio de inmovilidad

- Capital.
- Reservas.
- Superávit de capital.
- Resultados.

El Ingreso se ordena de acuerdo a la actividad económica de la empresa.

- Ingresos operacionales.
- Ingresos no operacionales.
- Ingresos extraordinarios (otros ingresos).

Los Gastos se ordenan así:

- Gastos operacionales
- Gastos no operacionales
- Gastos Extraordinarias

Las cuentas de orden se desagregan así:

- Deudoras
- Acreedoras

Código de Cuentas.

Son números que sirven para identificar a las cuentas antes mencionadas. Estos códigos facilita el archivo, para la ubicación de las fichas o registros que representan las cuentas.

Permite añadir nuevas cuentas dentro del plan, como consecuencia de nuevas operaciones.

1.	Activo.
1.1.	Activo corriente.
1.1.1.	Disponible.
1.1.1.1.	Caja.
1.1.1.2.	Bancos.
1.1.1.2.01	Banco Pichincha.
1.1.1.2.02	Banco de Guayaquil.
1.2.	Activo fijo.
1.3.	Activo diferido.
2.	Pasivo.
2.1.	Pasivo corriente.
2.2.	Pasivo largo plazo.
2.3.	Pasivo diferido.
3.	Patrimonio.
3.1.	Capital.
3.2.	Reservas.
3.3.	Superávit de capital.
3.4.	Resultados.

4. Ingresos.
4.1. Ingresos operacionales.
4.2. Ingresos no operacionales.
4.3. Ingresos extraordinarios.
5. Gastos.
5.1. Gastos operacionales.
5.2. Gastos no operacionales.
5.3. Gastos extraordinarios.

2.6.4. Ciclo Contable.

El ciclo contable es el período de tiempo en el que se registran todas las transacciones que ocurren en una empresa ya sea mensual, trimestral, semestral o anualmente; el más usado es el anual.

Los procedimientos del ciclo contable son aquellos pasos que se realizan para mostrar finalmente la información financiera de una Empresa.

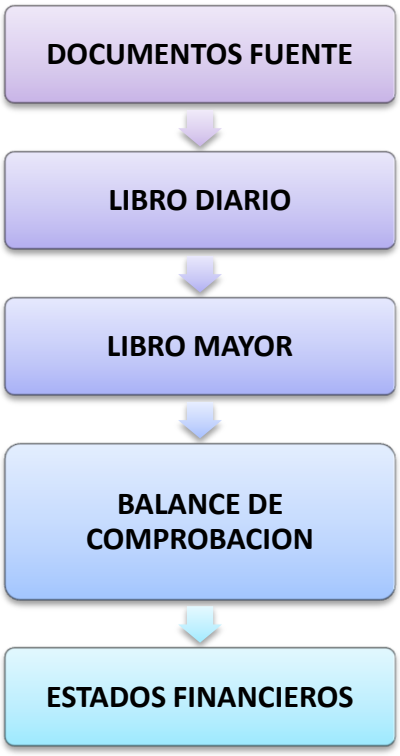

Ilustración 5. Proceso Contable

El Libro Diario.

Es aquel en el que se anotan o registran las operaciones de las empresas al momento de realizar una transacción. Este es conocido también como el libro de primera anotación.
Entre las características de un diario podemos mencionar que posee una primera columna indicando la fecha, luego otra donde se anota el detalle, una tercera que se llama referencia que es donde se anota el número o código que corresponde a la cuenta que se carga; tiene tres columnas seguidas donde aparecerá un auxiliar, el débito y el crédito.

El mayor.

Es un libro de segunda anotación el cual recibe la información del diario indicando los débitos y créditos que se realizaron en el diario. Los pases al mayor debe empezar con el pase del asiento débito, se registran los datos en el lado izquierdo de la cuenta débito; en la columna de la fecha, la fecha, en la columna de referencia, el número de la página donde se tomó el asiento, en la columna de valores el importe del débito.

Los Estados Financieros.

Son documentos esencialmente numéricos que a una fecha o por un período determinado presentan la situación financiera de una empresa, los resultados obtenidos en un período determinado y el comportamiento del efectivo. La importancia de los Estados Financieros viene dada por la necesidad de las empresas conocer y dar a conocer su situación determinada generalmente en un período contable de 1 año o menos.

Los Estados Financieros deben cumplir con los requisitos de: universalidad, al expresar la información clara y accesible; continuidad, en períodos regulares; periodicidad, que se lleven a cabo en forma periódica; oportunidad, que la información que consiguen sea rendida oportunamente.

Los Estados Financieros son:

- Estado de Situación o Balance General.
- Estado de Ganancias y Pérdidas.

- Estado Flujo de Efectivo.

Balance General.- Documento contable que refleja la situación patrimonial de una empresa en un momento del tiempo. Consta de dos partes, activo y pasivo. El activo muestra los elementos patrimoniales de la empresa, mientras que el pasivo detalla su origen financiero. La legislación exige que este documento sea imagen fiel del estado patrimonial de la empresa.

El activo suele subdividirse en inmovilizado y activo circulante. El primero incluye los bienes muebles e inmuebles que constituyen la estructura física de la empresa, y el segundo la tesorería, los derechos de cobro y las mercaderías. En el pasivo se distingue entre recursos propios, pasivo a largo plazo y pasivo circulante. Los primeros son los fondos de la sociedad (capital social, reservas); el pasivo a largo plazo lo constituyen las deudas a largo plazo (empréstitos, obligaciones), y el pasivo circulante son capitales ajenos a corto plazo (crédito comercial, deudas a corto). Existen diversos tipos de balance según el momento y la finalidad. Es el estado básico demostrativo de la situación financiera de una empresa, a una fecha determinada, preparado de acuerdo con los principios básicos de contabilidad gubernamental que incluye el activo, el pasivo y el capital contable.

Es un documento contable que refleja la situación financiera de un ente económico, ya sea de una organización pública o privada, a una fecha determinada y que permite efectuar un análisis comparativo de la misma; incluye el activo, el pasivo y el capital contable.

Se formula de acuerdo con un formato y un criterio estándar para que la información básica de la empresa pueda obtenerse uniformemente como por ejemplo: posición financiera, capacidad de lucro y fuentes de fondeo.

Estado de Ganancias y Pérdidas.- Es un documento complementario donde se informa detallada y ordenadamente como se obtuvo la utilidad del ejercicio contable.

El estado de resultados está compuesto por las cuentas nominales, transitorias o de resultados, o sea las cuentas de ingresos, gastos y costos. Los valores deben corresponder exactamente a los valores que aparecen en el libro mayor y sus auxiliares, o a los valores que aparecen en la sección de ganancias y pérdidas de la hoja de trabajo.

Los estados financieros son los informes sobre la situación financiera y económica de una empresa en un periodo determinado.

Estado Flujo de Efectivo.- La información sobre los flujos de efectivo de una empresa es útil para proporcionar a los usuarios de estados financieros una base para evaluar la habilidad de la empresa para generar efectivo y sus equivalentes y las necesidades de la empresa en las que fueron utilizados dichos flujos de efectivo. Las decisiones económicas que toman los usuarios requieren una evaluación de la habilidad de una empresa para generar efectivo y sus equivalentes, así como la oportunidad y certidumbre de su generación.

El objetivo de esta Norma es requerir la presentación de información acerca de los cambios históricos en el efectivo y sus equivalentes de una empresa, por medio de un estado de flujos de efectivo que clasifica los flujos de efectivo por las actividades operativas, de inversión y de financiamiento durante el período.

2.7. Cuentas por Cobrar o Cuenta Clientes.

Están constituidas por créditos a favor de las empresas, correspondientes a las ventas, prestación de servicios y demás operaciones normales, incluyendo cuentas de clientes no garantizadas, efectos o documentos por cobrar, aceptaciones de clientes y montos acumulados o no facturados por los cuales pueden expedirse o no facturadas con posterioridad.

En la mayoría de las empresas, las cantidades por ventas de contado se acumulan todos los días en una o más cajas registradoras y la suma se registra por medio de un asiento de diario al final del día.

El crédito se ha extendido en los negocios modernos, y ya es casi imposible sostener un volumen razonable de ventas sin concederlo. El vendedor tiene que financiar a su comprador y la deuda debe permanecer en los libros hasta que se cobre o se cancele por incobrable. Las cuentas a cobrar se incluyen en el activo circulante solamente los montos a cobrar nacidos de las operaciones del negocio, los cuales se espera convertir en dinero durante el ciclo operativo.

El origen de esta partida es la venta a crédito de mercancías, las empresas mercantiles y manufactureras, o de servicios en las empresas de servicios públicos y profesionales, hoteles, e instituciones similares. Se carga a la cuenta del cliente el precio de las mercancías despachadas o de los servicios prestados, y se le abonan los pagos recibidos. El saldo representa un derecho legítimo de la empresa a percibir el dinero por él expresado.

Son derechos legítimamente adquiridos por la empresa que, llegado el momento de ejecutar o ejercer ese derecho, recibirá a cambio efectivo o cualquier otra clase de bienes y servicios.

2.7.1. Clasificación de las Cuentas por Cobrar.

Atendiendo a su origen, las cuentas por cobrar pueden ser clasificadas en:

- Provenientes de ventas de bienes o servicios
- No provenientes de venta de bienes o servicios.

Cuentas por Cobrar Provenientes de Ventas de Bienes o Servicios: Este grupo de cuentas por cobrar está formado por aquellas cuyo origen es la venta a crédito de bienes o servicios y que, generalmente están respaldadas por la aceptación de una "factura" por parte del cliente.

Las cuentas por cobrar provenientes de ventas a crédito son comúnmente conocidas como "cuentas por cobrar comerciales" o "cuentas por cobrar a clientes" y deben ser presentadas en el balance general en el grupo de activo circulante o corriente, excepto aquellas cuyo vencimiento sea mayor que el ciclo normal de operaciones de la empresa, el cual, en la mayoría de los casos, es de doce meses. En aquellas empresas donde el ciclo normal de operaciones sea superior a un año, pueden incluirse dentro del activo circulante, aun cuando su vencimiento sea mayor de doce meses, siempre y cuando no sobrepase ese ciclo normal de operaciones, en cuyo caso deberán ser clasificadas fuera del activo circulante, en el grupo de activos a largo plazo.

Cuentas por Cobrar No Provenientes de Ventas a Crédito: Como el título indica, se refiere a derechos por cobrar que la empresa posee originados por transacciones diferentes a ventas de bienes y servicios a crédito.

Este tipo de cuentas por cobrar deberán aparecer clasificadas en el balance general en el grupo de activo circulante, siempre que se espere que van a ser cobradas dentro del ciclo normal de operaciones de la empresa, el cual, como se ha comentado, generalmente es de doce meses.

De acuerdo con la naturaleza de la transacción que las origina, las cuentas por cobrar no provenientes de ventas de bienes o servicios, pueden ser clasificadas a su vez en dos grupos: Cuentas por cobrar que representen derechos por cobrar en efectivos y Cuentas por cobrar que representan derechos por cobrar en bienes diferentes a efectivo.

2.8. Cuentas por Pagar o Cuenta Proveedores.

Las cuentas por pagar son deudas que tiene una empresa por concepto de bienes y servicios que compra a crédito. Llevar un registro de lo que debe y cuándo son los vencimientos le permitirá gozar de una buena situación crediticia y retener su dinero el mayor tiempo posible.

En su registro de cuentas por pagar deberá registrar la siguiente información: fecha de la factura, número de factura, monto de la factura, plazos, fecha de pago, importe pagado, saldo (si corresponde) y nombre y dirección del proveedor.

Las Cuentas por Pagar surgen por operaciones de compra de bienes materiales (Inventarios), servicios recibidos, gastos incurridos y adquisición de activos fijos o contratación de inversiones en proceso.

Si son pagaderas a menor de doce meses se registran como Cuentas por Pagar a Corto Plazo y si su vencimiento es a más de doce meses, en Cuentas por Pagar a Largo plazo.
Es preciso analizar estos pasivos por cada acreedor y en cada uno de éstos por cada documento de origen (fecha, número del documento e importe) y por cada pago efectuado. También deben analizarse por edades para evitar el pago de moras o indemnizaciones.

Las Cuentas por Pagar a Largo Plazo al finalizar cada período económico, deben reclasificarse a Corto Plazo.

2.8.1. Manejo y Control de Proveedores.

- Saldo al momento global, por proveedor y por factura.
- Antigüedad de saldos.
- Funcionamiento en red.
- Adaptable a sus necesidades.

2.8.2. Procedimientos de Comprobación Interna.

Cuentas por Pagar Corto Plazo.

Cuadre contable de las partidas pendientes en el mayor por deudores. Verificación de los documentos en los expedientes de pago por acreedores (proveedores).Comprobar si existen partidas o saldos deudores (contrario a la naturaleza de estas cuentas).Análisis por edades determinando las deudas vencidas (más de treinta días).Verificar los convenios de pagos suscritos.

Cuentas por Pagar Diversas.

Analizar las partidas que integran el saldo de esta cuenta, clasificarlas de acuerdo a su contenido, comprobando su cuadre contable así como analizar por edades para determinar las envejecidas. Verificar los documentos justificantes de las obligaciones pendientes de pago, así como las conciliaciones, confirmaciones y convenios de pago.

Efectos, Cuentas y Partidas por Pagar a Largo Plazo.

Verificar el cuadre contable de los saldos y partidas que integran esta cuenta en cada una de las subcuentas y mayores auxiliares por acreedores. Análisis por edades comprobando que en esta cuenta se incluyan exclusivamente los que exceden de un año. Comprobar los documentos en los expedientes de pago (contratos, convenios). Comprobar las confirmaciones y conciliaciones con los acreedores. Analizar las obligaciones vencidas y pendientes de pago, así como las partidas deudoras (contrarias a la naturaleza de esta cuenta).

Documentos por Pagar.

De igual forma que en las ventas, cuando la empresa exige la firma de pagarés o letras de cambio a los clientes para garantizar el pago de los valores a crédito, igualmente cuando se realiza compras, los proveedores también van a exigir la firma de pagarés o letras de cambio para asegurarse de que se cumplan con los pagos de los valores por los cuales nos han concedido el crédito.

Letra de cambio.

Es una orden escrita de una persona (girador) a otra (girado) para que pague una determinada cantidad de dinero en un tiempo futuro (determinado o determinable) a un tercero (beneficiario). Las personas que intervienen son: El Girador o librador: Da la orden de pago y elabora el documento. El Girado: Acepta la orden de pago firmando el documento comprometiéndose a pagar. Por lo tanto responsabilizándose, indicando en el mismo, el lugar o domicilio de pago para que el acreedor haga efectivo su cobro. El Beneficiario o tomador, recibe la suma de dinero en el tiempo señalado.

2.9. Inventarios.

Las Empresas requieren de un estricto control sobre sus existencias de artículos ya sea de materia prima, consumo interno o productos para la venta. Integra todos los movimientos transaccionales que tienen relación con consumos, producción, ventas, ingresos, devoluciones, traslados entre bodegas, entre otros.

Los procesos de compras e inventario se realizan de la siguiente forma:

El ALMACEN se encarga de controlar y administrar el inventario de consumo interno y el inventario para la venta. Ambos tipos de inventario es almacenado por separado.

Cada departamento de las distintas sucursales envía al ALMACEN las solicitudes de materiales que les son necesarias para su funcionamiento (ya sea de su inventario de consumo o de su inventario para la venta), utilizando un documento de Solicitud de Mercancía e identificando que se solicita mercancía del inventario de consumo interno. Este documento es registrado por el personal del almacén para su posterior despacho.

El ALMACEN realiza el despacho de las Solicitudes de Mercancía dependiendo de si hay las existencias de la mercancía solicitada por los departamentos de la empresa.

Basado en un punto de reorden (de cada uno de los artículos), se realizan reportes trimestrales de los artículos del inventario que deben ser comprados por la compañía. Cuando se agota cualquiera de los inventarios (de consumo o para la venta), el almacén genera un documento llamado Requisición de Mercancía, donde colocan todos aquellos artículos que es necesario comprar. Se generan requisiciones separadas para los dos tipos de inventario, por lo que la requisición identifica el tipo de inventario que se está solicitando comprar.

El departamento de COMPRAS recibe y registra todas las Requisiciones de Mercancía identificando si es una compra de consumo interno o para las ventas de la compañía. Se compra a los mismos proveedores, pero es necesario para la empresa, hacer la distinción del tipo de mercancía para determinar su almacenamiento y posterior distribución.

Los artículos de cada solicitud se clasifican o agrupan de acuerdo a las compras que se vallan a realizar para así obtener mejores precios de los proveedores.

El personal de COMPRAS genera un documento llamado Solicitud de Cotización de Mercancía donde se colocan todos los artículos que se desea sean cotizados por los distintos proveedores, los artículos que aparecen en este documento representan la suma de todos los artículos similares que han sido registrados en las distintas requisiciones de mercancía, es así que una solicitud de cotización puede estar asociada con diferentes requisiciones de mercancía. Es importante saber a qué Requisición pertenece cada artículo que aparece en las solicitudes de cotización para poder almacenarla de forma correcta (inventario de consumo interno, inventario para la venta) o para poder entregarlo al departamento que hizo la Requisición de Mercancía.

La Orden de Compra generada es enviada al proveedor el cual se encarga de despacharla según los términos y condiciones de la misma. La mercancía es entregada en el ALMACEN. La mayoría de las órdenes de Compras deben indicar el periodo de entrega, que puede ser días, meses o años y debe indicar la fecha probable de entrega.

Luego de recibida la Orden de Compra el proveedor realiza el despacho de la mercancía solicitada, este lleva la mercancía al almacén donde es revisada y verificada por el

personal del mismo. Si los bienes comprados satisfacen las exigencias especificadas en la Orden de Compra esta es recibida y se emite un Comprobante de Recepción de Mercancía que le permitirá al proveedor hacer la solicitud de cobro correspondiente (al ser generado este comprobante se incrementan las existencias de mercancía, dependiendo del tipo de inventario).

El personal del Almacén identifica el tipo de inventario del que se trata (inventario de consumo interno o para la venta) y lo almacena de acuerdo a esta condición.

El almacén distribuye la mercancía, dependiendo del tipo, a cada una de las unidades solicitantes, para ello realiza el proceso de entrega de mercancía. Durante este proceso se revisan las Solicitudes de Mercancía pendientes de entrega y se van supliendo dependiendo de la existencia de los artículos solicitados en el inventario y de la prioridad que se le haya dado a las solicitudes. La prioridad la decide el almacenista y este puede basar su decisión en notas o solicitudes especiales de la Gerencia. Estas notas no son registradas en el Sistema, pero si la prioridad de la solicitud de mercancía.

CAPITULO III
ANÁLISIS DE LA IMPLEMENTACIÓN DEL SISTEMA INFORMÁTICO SUBGANA

3.1. Elección de Herramientas de Desarrollo y Gestor de Base de Datos.

Para realizar el estudio de las herramientas de desarrollo de software, se evalúan los 5 elementos de software requeridos para la construcción de aplicaciones web:

- Plataforma Operativa.
- Servidor Web.
- Lenguaje de Desarrollo.
- Estándares de Desarrollo.
- Herramienta de la Base de Datos a utilizar.

Una vez cumplido el proceso anterior, analizamos las características técnicas que se tomarán en cuenta en la elaboración del sistema, comparando las alternativas del uso del software (libre o propietario) que son las siguientes:

- Requerimientos de Hardware y Software tanto para el servidor como para el cliente.
- Costos operativos de tecnología.
- Compatibilidad con la plataforma tecnológica existente o proyectada.
- Estabilidad, fiabilidad y facilidad de uso.
- Seguridad.
- Documentación.
- Existencia de implementaciones de software de éxito comprobadas en el ámbito local y nacional.
- Disponibilidad de las actualizaciones del software.
- Los costos de las herramientas a utilizar.

De acuerdo al decreto 1014 emitido por parte de la presidencia del Ec. Rafael Correa Delgado que promueve el uso de software libre en las instituciones públicas del Ecuador, se establece utilizar las siguientes tecnologías:

- Framework de Desarrollo JSF 2.0

- Componentes EJB.
- API para las interfaces Primefaces 2.0.
- Motor de persistencia JPA.
- Servidor de Aplicaciones JBOSS 6.0
- Motor de Base de Datos POSTGRES 8.3
- El sistema operativo del servidor quedará abierto entre Linux y Windows ya que no afectaría el funcionamiento de la aplicación.
- Navegador Mozilla Firefox 3.0 en adelante.

A continuación, se explica detalladamente cada uno de los anteriores elementos.

3.1.1. Framework de Desarrollo JSF 2.0.

Es un framework web MVC [19]que se centran en simplificar la creación de interfaces de usuario para aplicaciones Java web y hacer que los componentes de interfaz de usuario sean reutilizables fáciles de implementar. A diferencia de 1.x JSF, casi todo lo que se declaran en faces-config.xml, con JSF[20] 2.0, se le permite utilizar la anotación a la navegación declarada, que hacen que su desarrollo sea más fácil y más rápido.

Otra definición de JavaServer Faces, es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE. JSF usa JavaServer Pages (JSP) como la tecnología que permite hacer el despliegue de las páginas, pero también se puede acomodar a otras tecnologías como XUL[21].

JSF incluye:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.

- Un conjunto por defecto de componentes para la interfaz de usuario.

[19]**MVC.-** Modelo Vista Controlador
[20]**JSF.-** Java Server Faces
[21]**XUL.-**Lenguaje basado en XML para la interfaz de usuario

- Dos bibliotecas de etiquetas personalizadas para JavaServer Pages que permiten expresar una interfaz JavaServer Faces dentro de una página JSP.

- Un modelo de eventos en el lado del servidor.

- Administración de estados.

- Beans[22] administrados.

Objetivos de JSF:

- Definir un conjunto simple de clases base de Java para componentes de la interfaz de usuario, estado de los componentes y eventos de entrada. Estas clases tratarán los aspectos del ciclo de vida de la interfaz de usuario, controlando el estado de un componente durante el ciclo de vida de su página.

- Proporcionar un conjunto de componentes para la interfaz de usuario, incluyendo los elementos estándares de HTML[23] para representar un formulario. Estos componentes se obtendrán de un conjunto básico de clases base que se pueden utilizar para definir componentes nuevos.

- Proporcionar un modelo de JavaBeans para enviar eventos desde los controles de la interfaz de usuario del lado del cliente a la aplicación del servidor.

- Definir APIs para la validación de entrada, incluyendo soporte para la validación en el lado del cliente.

- Especificar un modelo para la internacionalización y localización de la interfaz de usuario.

- Automatizar la generación de salidas apropiadas para el objetivo del cliente, teniendo en cuenta todos los datos de configuración disponibles del cliente, como versión del navegador.

[22] **Beans**.- Un Bean es un componente software que tiene la particularidad de ser reutilizable y así evitar la tediosa tarea de programar los distintos componentes uno a uno.
[23] **HTML.-** Lenguaje de Marcado de Hipertexto.

3.1.2. Enterprise Java Beans (EJB).

Enterprise Java Beans (EJB) es una plataforma para construir aplicaciones de negocio portables, reusables y escalables usando el lenguaje de programación Java. Desde el punto de vista del desarrollador, un EJB es una porción de código que se ejecuta en un contenedor EJB, que no es más que un ambiente especializado (runtime[24]) que provee determinados componentes de servicio.

Los EJBs pueden ser vistos como componentes, desde el punto de vista que encapsulan comportamiento y permite reutilizar porciones de código, pero también pueden ser vistos como un framework, ya que, desplegados en un contenedor, proveen servicios para el desarrollo de aplicaciones enterprise, servicios que son invisibles para el programador y no ensucian la lógica de negocio con funcionalidades trasversales al modelo de dominio (a menudo requerimientos no funcionales o aspectos). En la especificación 3.0, los EJB no son más que POJOs[25] (clases planas comunes y corrientes de Java) con algunos poderes especiales implícitos, que se activan en runtime cuando son ejecutados en un contenedor de EJBs.

Los servicios que debe proveer el contenedor de EJBs deben ser especificados por el programador a través de metadata de configuración que puede escribirse como:

- Anotaciones de Java5 intercaladas en el código de las clases.
- Descriptores XML (archivos XML separados).

A partir de EJB 3 se puede usar cualquiera de estas dos técnicas. Las técnicas no son exclusivas, pueden coexistir anotaciones con descriptores XML y, en el caso de superponerse la metadata, los XML tendrán prioridad y podrán sobreescribir las anotaciones.

Una anotación transforma un simple POJO en un EJB.

Ilustración 6. Estructura de un EJB

[24]**Runtime.-**Se encarga de integrar java con el sistema.
[25]**Pojo.-** Utilizada por programadores Java para enfatizar el uso de clases simples y que no dependen de un framework en especial

Tipos de EJBs.

Existen tres tipos de EJBs:

- **Session Beans (Bean de Sesión):** en una aplicación Enterprise típica, dividida en cuatro grandes capas o layers (presentación, lógica de, persistencia y base de datos), los Session Beans viven en la lógica de negocio. Hay dos grandes tipos de Session Beans: Stateless[26] y Stateful[27], el primero no conserva el estado de ninguno de sus atributos de la invocación de un método a otro y el segundo conserva el estado a lo largo de toda una sesión. Los Session Beans Stateless son los únicos que pueden exponerse como web services.

- **Message-Driven Beans (MDBs):** también viven en la lógica de negocio y los servicios que proveen son parecidos a los Session Beans, con la diferencia de que los MDBs[28] son usados para invocar métodos de forma asincrónica. Cuando se produce la invocación de un método de un MDB desde un cliente, la llamada no bloquea el código del cliente y el mismo puede seguir con su ejecución, sin tener que esperar indefinidamente por la respuesta del servidor. Los MDBs encapsulan el popular servicio de mensajería de Java, JMS.

- **Entities(Bean de Entidad):** los entities viven en la capa de persistencia y son los EJBs que manejan la Java Persistence API (JPA), también parte de la especificación de EJB 3.0. Los entities son POJOs con cierta metadata que permite a la capa de persistencia mapear los atributos de la clase a las tablas de la base de datos y sus relaciones.

Los Session Beans son invocados por el cliente con el propósito de ejecutar operaciones de negocio específicas, como por ejemplo podría ser chequear la historia crediticia del cliente de un banco. El nombre sesión implica que la instancia del bean estará disponible durante una unidad de y no sobrevivirá a una caída del servidor. Un bean de sesión sirve para modelar cualquier funcionalidad lógica de una aplicación.

Los MDBs también procesan lógica de negocio, pero un cliente nunca invoca a un método de un MDB directamente. El sistema de mensajería asincrónica propone la

[26] **Stateless**.- Se caracteriza por no conservar los atributos durante la sesión.
[27] **Stateful.-** Conserva el estado del EJB durante toda la sesión.
[28] **MBDs.-** Message Drive Beans

utilización de una capa intermedia en la comunicación entre el productor y el consumidor del mensaje. En EJB 3, esta capa se llama MOM (Message-oriented middleware). Básicamente la MOM es un software que permite funcionar como servidor de mensajería, reteniendo los mensajes del productor y enviándolos posteriormente al consumidor en el momento en que esté disponible para recibirlo. (Es un funcionamiento similar al de un servidor de correo electrónico.)

Entities y la Java Persistence API: Debido al auge de los frameworks ORM (Hibernate, TopLink, etc), Sun tuvo que replantear su complicada y antinatural especificación de persistencia, que tanto dolores de cabeza le daba a los programadores que usaban EJB 2, al extremo que optó por reescribirla casi por completo. Así nació JPA.

Contenedor de EJBs.

Así como cuando compilamos una clase simple de Java, necesitamos una Java Virtual Machine (JVM) para ejecutarla, necesitamos un contenedor de EJBs para ejecutar los Session Beans y los MDBs.

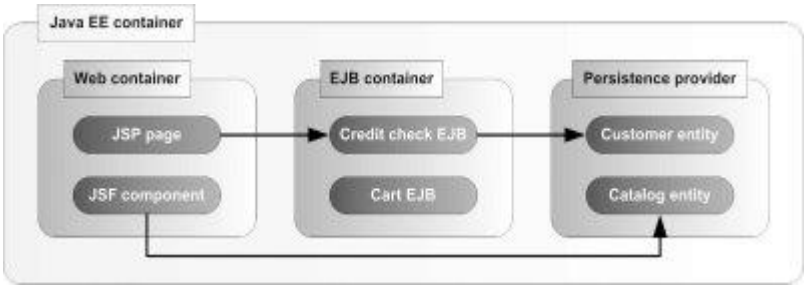

Ilustración 7. Contenedor de EJBs

Un contenedor Java EE es un servidor de aplicaciones que es capaz de ejecutar EJBs, puede servir como web container y además puede incluir otras APIS y servicios, como por ejemplo el de persistencia. Algunos servidores de aplicaciones pueden proveer solamente un contenedor web, como es el caso de Apache Tomcat, o sólo proveer servicios de persistencia, como es el caso de Hibernate. Un servidor de aplicaciones como JBoss trae un servidor Apache Tomcat y un servidor Hibernate, que se ejecutan dentro de forma transparente.

Qué servicios proveen los EJBs?

- **Integración:** Proveen una forma de acoplar en tiempo de ejecución diferentes componentes, mediante simple configuración de anotaciones o XMLs. El acoplamiento se puede hacer mediante Inyección de Dependencia (DI) o usando JNDI, como se hacía en EJB 2 (explicaré el concepto de Inyección de Dependencia en detalle en el próximo post). La integración es un servicio que proveen los beans de sesión y los MDBs.

- **Pooling**: El contenedor de EJBs crea para componentes EJB un pool de instancias que es compartido por los diferentes clientes. Aunque cada cliente ve como si recibiera siempre instancias diferentes de los EJB, el contenedor está constantemente rehusando objetos para optimizar memoria. El pooling es un servicio que se aplica a los Stateless Session Beans y a los MDBs.

- **Thread-safely:** El programador puede escribir componentes del lado del servidor como si estuviera trabajando en una aplicación sencilla con un solo thread (hilo). El contenedor se encarga de que los EJBs tengan el soporte adecuado para una aplicación multiusuario (como son en general las aplicaciones enterprise) de forma transparente, asegurando el acceso seguro y consistente.

  Aplica a los beans de sesión y a los MDBs.

- **Administración de Estados:** El contenedor de EJBs almacena y maneja el estado de un Stateful Session Bean de forma transparente, lo que significa que el programador puede mantener el estado de los miembros de una clase como si estuviera desarrollando una aplicación desktop ordinaria. El contenedor maneja los detalles de las sesiones.

- **Mensajería:** Mediante los MDBs es posible desacoplar por completo dos componentes para que se comuniquen de forma asincrónica.

- **Transacciones:** EJB soporta el manejo de transacciones declarativas que permiten agregar comportamiento transaccional a un componente simplemente usando anotaciones o XMLs de configuración. Esto significa que cuando un método de un EJB (Session Bean o MDB) se completa normalmente, el

contenedor se encargará de realizar el commit[29] de la transacción y efectivizar los cambios que se realizaron en los datos de forma permanente. Si algo fallara durante la ejecución del método (una excepción o cualquier otro problema), la transacción haría un rollback[30] y es como si el método jamás se hubiera invocado.

- **Seguridad:** EJB soporta integración con la Java Authentication and Authorization Service (JAAS) API, haciendo casi transparente el manejo transversal de la seguridad. Aplica a todos los Session Beans.

- **Interceptors:** EJB introduce un framework liviano y simple para AOP (programación orientada a aspectos). No es tan robusto y completo como otros, pero es lo suficientemente útil para que sea utilizado por los demás servicios del contenedor para brindarnos de forma invisible los crosscutting concerns de seguridad, transacciones, thread-safely. Además, nosotros, como programadores, podemos agregar nuevos aspectos como logging o auditoria y demás de forma configurable.

- **Acceso Remoto:** Es posible acceder de forma remota a distintos EJBs de forma sencilla, simplemente mediante la Inyección de Dependencia. El procedimiento para inyectar un componente local o uno remoto es exactamente el mismo, abstrayéndonos de las complicaciones específicas de RMI o similares. Este servicio aplica únicamente a los Session Beans.

- **Web Services:** Un Stateless Session Bean puede publicar sus métodos como web services mediante una sencilla anotación.

- **Persistencia:** EJB 3 provee la especificación JPA para el mapeo de objetos (Entities) a tablas.

- **Catching and Performance:** JPA provee de forma transparente un importante número de servicios que permiten usar un caché de entidades en memoria y una lectura y escritura sobre la base de datos.

[29] **Commit.-** Proceso que se realiza en una base de datos para terminar satisfactoriamente una transacción.
[30] **Rollback.-** Proceso que se realiza en una base de datos cuando ha ocurrido algún problema durante la transacción.

3.1.3. Primefaces.

PrimeFaces es un componente para JavaServer Faces (JSF) de código abierto que cuenta con un conjunto de componentes ricos que facilitan la creación de las aplicaciones web. Primefaces está bajo la licencia de Apache License V2. Una de las ventajas de utilizar Primefaces, es que permite la integración con otros componentes como por ejemplo RichFaces[31].

Propiedades.

- Conjunto de componentes ricos (Editor de HTML, autocompletar, cartas, gráficas o paneles, entre otros).
- Soporte de Ajax con despliegue parcial, lo que permite controlar cuáles componentes de la página actual se actualizarán y cuáles no.
- Veinte y cinco temas prediseñados.
- Componente para desarrollar aplicaciones web para móviles-celulares, especiales para Iphones, Palm, Android y teléfonos móviles Nokia.

Versiones.

- Primefaces 1: Trabaja con JSF 1.2
- Primefaces 2 y 3: Trabaja con JSF 2

3.1.4. JPA.

El Java Persistence API (JPA) es una especificación de Sun Microsystems para la persistencia de objetos Java a cualquier base de datos relacional.

[31]**RichFaces.-** Es una biblioteca de código abierto basada en Java para crear aplicaciones web con Ajax.

Arquitectura.

El siguiente diagrama muestra la relación entre los componentes principales de la arquitectura de JPA.

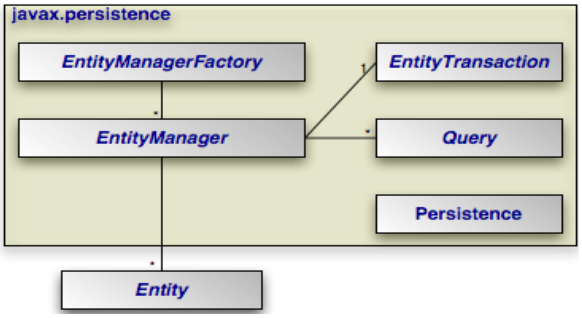

Ilustración 8. Arquitectura de JPA

Varias de las interfaces del diagrama anterior son solo necesarias para su utilización fuera de un servidor de aplicaciones que soporte EJB 3, como es el caso del EntityManagerFactory que es ampliamente usado en desarrollo de aplicaciones de escritorio. En un servidor de aplicaciones, una instancia de EntityManager típicamente suele ser inyectada, haciendo así innecesario el uso de un EntityManagerFactory. Por otra parte, las transacciones dentro de un servidor de aplicaciones se controlan mediante un mecanismo estándar de controles de, por lo tanto la interfaz EntityTransaction también no es utilizada en este ambiente.

Persistence: La clase javax.persistence.Persistence contiene métodos estáticos de ayuda para obtener una instancia de EntityManagerFactory de una forma independiente al vendedor de la implementación de JPA.

EntityManagerFactory: La clase javax.persistence.EntityManagerFactory nos ayuda a crear objetos de EntityManager utilizando el patrón de diseño del Factory (fábrica).

EntityManager: La clase javax.persistence.EntityManager es la interfaz principal de JPA utilizada para la persistencia de las aplicaciones. Cada EntityManager puede realizar operaciones CRUD (Create, Read, Update, Delete) sobre un conjunto de objetos persistentes.

Entity: La clase javax.persistence.Entity es una anotación Java que se coloca a nivel de clases Java serializables y que cada objeto de una de estas clases anotadas representa un registro de una base de datos.

EntityTransaction: Cada instancia de EntityManager tiene una relación de uno a uno con una instancia de javax.persistence.EntityTransaction, permite operaciones sobre datos persistentes de manera que agrupados formen una unidad de trabajo transaccional, en el que todo el grupo sincroniza su estado de persistencia en la base de datos o todos fallan en el intento, en caso de fallo, la base de datos quedará con su estado original. Maneja el concepto de todos o ninguno para mantener la integridad de los datos.

Query: La interface javax.persistence.Query está implementada por cada vendedor de JPA para encontrar objetos persistentes manejando cierto criterio de búsqueda. JPA estandariza el soporte para consultas utilizando Java Persistence Query Language (JPQL) y Structured Query Language (SQL). Podemos obtener una instancia de Query desde una instancia de un EntityManager.

Excepciones.

A continuación se muestra la arquitectura que JPA tiene para el control de las excepciones:

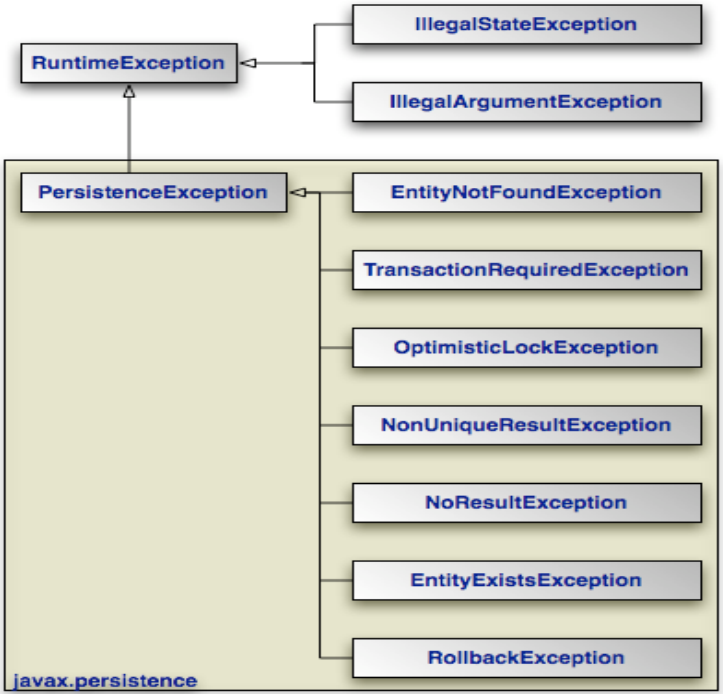

Ilustración 9. Excepciones en JPA

Mapeo con anotaciones EJB3/JPA.

Las entidades de JPA no son más que simples POJOs (Plain Old Java Objects) que tienen definidos sobre su clase, propiedad o métodos unas anotaciones especiales de EJB3.0. Las anotaciones de JPA se clasifican en dos categorías:

Anotaciones de mapeo lógico que permiten describir modelo de objeto, asociaciones de clase.

Anotaciones de mapeo físico que describen esquemas físicos de base de datos, tablas, columnas, índices, etc.

JPA reconoce dos tipos de clases persistentes: las clases entidad y las clases inmersas o embebidas (embebidas).

Las anotaciones JPA, conocidas también como anotaciones EJB 3.0, se encuentran en el paquete javax.persistence.*. Muchos IDEs que soportan a JDK5 como Eclipse, Netbeans, IntelliJ IDEA, poseen herramientas y plugins para generar clases de entidad con anotaciones de JPA a partir de un esquema de base de datos.

Declaración un Bean de Entidad.

Para declarar un bean de entidad se agrega la anotación @Entity a la clase que va a ser persistente, hay que tener en cuenta que una clase persistente además de tener esta anotación, debe implementar la interfaz Serializable.

Para declarar el identificador de la clase que representará la clave primaria en la tabla de base de datos, se utiliza la anotación @Id. Más adelante se conocerá cómo declarar identificadores más complejos con la anotación @EmbeddedId.

Ejemplo:

```
@Entity
public class Carro implements Serializable {
Long id;
@Id
```

```
public Long getId() { return id; }
public void setId(Long id) { this.id = id; }
}
```

Dependiendo de donde se declare la anotación, sea en propiedades o métodos, el acceso para el motor de persistencia JPA será correspondientemente accediendo a una propiedad o a un método. Lo más aconsejable es que las anotaciones se las declare a nivel de métodos getters, ya que son los más visitados. Se debe evitar tener mezclado anotaciones tanto en propiedades como en métodos.

Declaración de Tablas.

Para definir los datos de una tabla de base de datos, se utiliza la anotación @Table a nivel de la declaración de la clase, esta anotación permite definir el nombre de la tabla con la que se está mapeando la clase, el esquema, el catálogo, constraints de unicidad. Si no se utiliza esta anotación y se usa únicamente la anotación Entity, el nombre de la tabla corresponderá al nombre de la clase.

```
@Entity
@Table(name="tbl_carro")
public class Carro implements Serializable {
...
```

La anotación Table contiene los atributos de catalog y schema, en caso de que necesitemos definirlos. También puede definir constraints de unicidad utilizando la anotación @UniqueConstraint en conjunción con la anotación @Table

```
@Table(name="tbl_carro",
uniqueConstraints = {@UniqueConstraint(columnNames={"placa", "dia"})}
)
```

Anotaciones.

Una anotación o metadato proporciona un recurso adicional al elemento de código al que va asociado en el momento de la compilación. Cuando la máquina virtual de Java (JVM)

ejecuta la clase busca estos metadatos y determina el comportamiento a seguir con el código al que va unido la anotación.

@**Entity:** Indica que es una Entidad.
name: Por defecto el nombre de la clase pero se puede especificar otra diferente.

@**Table:** Específica la tabla principal relacionada con la entidad.
name: Nombre de la tabla, por defecto el de la entidad si no se especifica.
catalog: Nombre del catálogo.
schema: Nombre del esquema.
uniqueConstraints: Constrains entre tablas relacionadas con la anotación
@Column y @JoinColumn.

@**SecondaryTable:** Especifica una tabla secundaria relacionada con la entidad si éste englobara a más de una. Tiene los mismos atributos que @Table.

@**SecondaryTables:** Indica otras tablas asociadas a la entidad.

@**UniqueConstraints:** Específica que una única restricción se incluya para la tabla principal y la secundaria.

@**Column:** Especifica una columna de la tabla a mapear con un campo de la entidad.
name: Nombre de la columna.
unique: Si el campo tiene un único valor.
nullable: Si permite nulos.
insertable: Si la columna se incluirá; en la sentencia INSERT generada.
updatable: Si la columna se incluirá; en la sentencia UPDATE generada.
table: Nombre de la tabla que contiene la columna.
length: Longitud de la columna.
precision: Número de dígitos decimales.
scale: Escala decimal.

@**JoinColumn:** Especifica una campo de la tabla que es foreign key de otra tabla definiendo la relación del lado propietario.
name: Nombre de la columna de la foreign key.
referenced: Nombre de la columna referencia.

unique: Si el campo tiene un único valor.
nullable: Si permite nulos.
insertable: Si la columna se incluirá; en la sentencia INSERT generada.
updatable: Si la columna se incluirá; en la sentencia UPDATE generada.
table: Nombre de la tabla que contiene la columna.

@**JoinColumns:** Anotación para agrupar varias JoinColumn.

@**Id:** Indica la clave primaria de la tabla.

@**GeneratedValue:** Asociado con la clave primaria, indica que ésta se debe generar por ejemplo con una secuencia de la base de datos.
strategy: Estrategia a seguir para la generación de la clave: AUTO (valor por defecto, el contenedor decide la estrategia en función de la base de datos),IDENTITY (utiliza un contador, Ej.: MySQL), SEQUENCE (utiliza una secuencia, Ej.: Oracle, PostgreSQL) y TABLE (utiliza una tabla de identificadores).
generator: Forma en la que genera la clave.

@**SequenceGenerator:** Define un generador de claves primarias utilizado junto con la anotación @GeneratedValue. Se debe especificar la secuencia en la entidad junto a la clave primaria.
name: Nombre del generador de la clave.
sequence: Nombre de la secuencia de la base de datos del que se va a obtener la clave.
initialValue: Valor inicial de la secuencia.
allocationSize: Cantidad a incrementar de la secuencia cuando se llegue al máximo.

@**TableGenerator:** Define una tabla de claves primarias generadas. Se debe especificar en la anotación @GeneratedValue con strategy = GenerationType.TABLE.
name: Nombre de la secuencia.
table: Nombre de la tabla que guarda los valores generados.
catalog: Catalogo de la tabla.
schema: Esquema de la tabla.
pkColumn Name(): Nombre de la clave primaria de la tabla.
valueColumn Name(): Nombre de la columna que guarda el último valor generado.
pkColumn Value(): Valor de la clave primaria.
initialValue: Valor inicial de la secuencia.

allocationSize: Cantidad a incrementar de la secuencia.
uniqueConstraints: Constrains entre tablas relacionadas.

@AttributeOverride: Indica que sobrescriba el campo con el de la base de datos asociado.
name: Nombre del campo
column: Columna del campo

@AttributeOverrides: Mapeo de varios campos.

@EmbeddedId: Se utiliza para formar la clave primaria con múltiples campos.

@IdClass: Se aplica en la clase entidad para especificar una composición de la clave primaria mapeada a varios campos o propiedades de la entidad.

@Transient: Indica que el campo no se debe persistir.

@Version: Se utiliza a la hora de persistir la entidad en base de datos para identificar las entidades según su versión. Se actualiza automáticamente cuando el objeto es mapeado en la base de datos.

@Basic: Mapeo por defecto para tipos básicos: tipos primitivos, wrappers de los tipos primitivos, String, BigInteger, BigDecimal, Date, Calendar, Time, Timestamp, byte[],Byte[], char[], Character[], enumerados y cualquier otra clase serializable.

@OneToOne: (1:1) Indica que un campo está; en relación con otro (Ej: dentro de una empresa, un empleado tiene un contrato de trabajo).
cascade: Forma en que se deben actualizar los campos: ALL, PERSIST, MERGE,REMOVE y REFRESH.
fetch: Determina la forma en que se cargan los datos: FetchType.LAZY (carga de la entidad únicamente cuando se utiliza), FetchType.EAGER (carga de todas las entidades relacionadas con ella).
optional: Si la asociación es opcional.
mappedBy: El campo que posee la relación, únicamente se especifica en un lado de la relación.

@**ManyToOne:** (N:1) Indica que un campo está; asociado con varios campos de otra entidad (ej: las cuentas que posee un banco o los empleados que pertenecen a un departamento).
cascade, fetch y optional: Igual que la anterior anotación.

@**OneToMany:** (1:N) Asocia varios campos con uno (Ej: varios departamentos de una empresa).
cascade, fetch y optional: Igual que la anterior anotación.
mappedBy: El campo que posee la relación. Es obligatorio que la relación sea unidireccional.

@**ManyToMany:** (N:M) Asociación de varios campos con otros con multiplicidad muchos a muchos (Ej.: relaciones entre empresas).
cascade, fetch y mappedBy: Igual que la anterior anotación.

@**Lob:** Se utiliza junto con la anotación @Basic para indicar que un campo se debe persistir como un campo de texto largo si la base de datos soporta este tipo.

@**Temporal:** Se utiliza junto con la anotación @Basic para especificar que un campo fecha debe guardarse con el tipo java.util.Date o java.util.Calendar. Si no se especificar a ninguno de estos por defecto se utiliza java.util.Timestamp.

@**Enumerated:** Se utiliza junto con la anotación @Basic e indica que el campo es un tipo enumerado (STRING), por defecto ORDINAL.

@**JoinTable:** Se utiliza en el mapeo de una relación ManyToMany o en una relación unidireccional OneToMany.
name: Nombre de la tabla join a donde enviar la foreing key.
catalog: Catalog de la tabla.
schema: Esquema de la tabla.
joinColumns: Columna de la foreign key de la tabla join que referencia a la tabla primaria de la entidad que posee la asociación (sólo en un lado de la relación).
inverseJoinColumns: Columnas de la foreign key de la tabla join que referencia a la tabla primaria de la entidad que no posee (lado inverso de la relación)
uniqueConstraints: Constraints de la tabla.

@MapKey: Específica la clave de una clase de tipo java.util.Map.

@OrderBy: Indica el orden de los elementos de una colección por un ítem específico deforma ascendente o descendente.

@Inheritance: Define la forma de herencia de una jerarquía de clases entidad, es decirla relación entre las tablas relacionales con las entidades.

@NamedQuery: Especifica el nombre del objeto query utilizado junto a EntityManager.
name: Nombre del objeto query.
query: Especifica la query a la base de datos mediante lenguaje Java Persistence Query Language (JPQL).

@NamedQueries: Específica varias queries como la anterior.

@NamedNativeQuery: Específica el nombre de una query SQL normal.
name: Nombre del objeto query.
query: Especifica la query a la base de datos.
resultClass: Clase del objeto resultado de la ejecución de la query.
resultSetMapping: Nombre del SQLResultSetMapping definido.

@NamedNaviteQueries: Específica varias queries SQL.

3.1.5. JBOSS.

Como se explicó en el Capítulo II, JBoss es un servidor de aplicaciones J2EE de código abierto implementado en Java puro. Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible Java. Los principales desarrolladores trabajan para una empresa de servicios, JBoss Inc., adquirida por Red Hat en abril del 2006, fundada por Marc Fleury, el creador de la primera versión de JBoss. El proyecto está apoyado por una red mundial de colaboradores. Los ingresos de la empresa están basados en un modelo de negocio de servicios. JBoss implementa todo el paquete de servicios de J2EE.

En el siguiente diagrama se explica la funcionalidad de JBOSS:

Ilustración 10. Arquitectura de JBOSS.

¿Qué ofrece?

Ofrece un camino sencillo, abierto y económicamente rentable para modernizar su infraestructura de aplicaciones heredadas. Debido a que es de código abierto, JBoss brinda una flexibilidad incomparable y un costo total de propiedad considerablemente menor que sus principales competidores. JBoss provee un valor superior.

Aspecto Financiero.

Recupere hasta el 70% del presupuesto asignado al middleware empresarial, concentre la mayor parte de su presupuesto informático en la gente y en las aplicaciones que hacen que su negocio se distinga del resto. Comience sus proyectos con anterioridad con muchos menos riesgos.

Debido a que JBoss es de código abierto, aprovecha la capacidad colectiva de la investigación y el desarrollo de una comunidad compuesta por miles de desarrolladores. Esto no sólo acelera la maduración del código e impulsa una mayor innovación, sino que también le presenta a Red Hat una estructura de costos superior, que le permite ofrecer mayor valor a los usuarios.

JBoss le permite desarrollar, desplegar e integrar aplicaciones y servicios Web.

3.1.6. Base de Datos PostgreSQL.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD[32] y con su código fuente disponible libremente. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. A continuación tenemos un gráfico que ilustra de manera general los componentes más importantes en un sistema PostgreSQL.

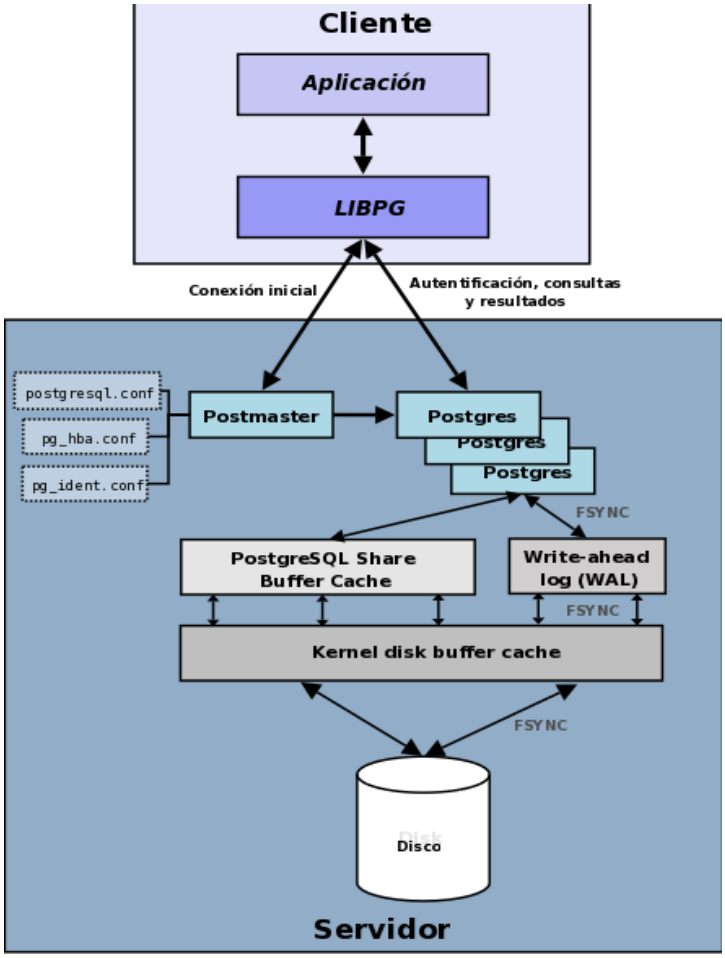

Ilustración 11. Arquitectura PostgreSQL.

- *Aplicación cliente*: Esta es la aplicación cliente que utiliza PostgreSQL como administrador de bases de datos. La conexión puede ocurrir vía TCP/IP[33] o sockets[34] locales.

[32] **BSD**.- Es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution).
[33] **TCP/IP.-** Es un modelo de descripción de protocolos de red.

- *Demonio postmaster*: Este es el proceso principal de PostgreSQL. Es el encargado de escuchar por un puerto/socket por conexiones entrantes de clientes. También es el encargado de crear los procesos hijos que se encargaran de autentificar estas peticiones, gestionar las consultas y mandar los resultados a las aplicaciones clientes

- *Ficheros de configuración*: Los 3 ficheros principales de configuración utilizados por PostgreSQL, postgresql.conf, pg_hba.conf y pg_ident.conf

- *Procesos hijos postgres*: Procesos hijos que se encargan de autentificar a los clientes, de gestionar las consultas y mandar los resultados a las aplicaciones clientes.

- *PostgreSQL share buffer cache*: Memoria compartida usada por PostgreSQL para almacenar datos en caché.

- *Write-Ahead Log (WAL):* Componente del sistema encargado de asegurar la integridad de los datos (recuperación de tipo REDO).

- *Kernel disk buffer cache*: Caché de disco del sistema operativo.

- *Disco*: Disco físico donde se almacenan los datos y toda la información necesaria para que PostgreSQL funcione.

Características.

La última serie de producción es la 9.1. Sus características técnicas la hacen una de las bases de datos más potentes y robustas del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

[34]**Socket.-** Designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.

A continuación, se presenta algunas de las características más importantes y soportadas por PostgreSQL:

- Integridad referencial.
- Tablespaces.
- Nested transactions (savepoints).
- Replicación asincrónica/sincrónica / Streaming replication - Hot Standby.
- Copias de seguridad en caliente (Online/hot backups).
- Unicode.
- Juegos de caracteres internacionales.
- Regionalización por columna.
- Multi-Version Concurrency Control (MVCC).
- Multiples métodos de autentificación.
- Acceso encriptado vía SSL.
- Actualización in-situ integrada (pg_upgrade).
- SE-postgres.
- Completa documentación.
- Licencia BSD.

Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.

Programación y Desarrollo.

- Funciones/procedimientos almacenados (stored procedures[35]) en numerosos lenguajes de programación, entre otros PL/pgSQL[36], PL/Perl[37], etc.

- Bloques anónimos de código de procedimientos (sentencias DO[38])

- Numerosos tipos de datos y posibilidad de definir nuevos tipos. Además de los tipos estándares en cualquier base de datos, están disponibles, entre otros, tipos geométricos, de direcciones de red, de cadenas binarias, UUID[39], XML, matrices.

[35]**Stored Procedures.-** Procedimientos Almacenados
[36]**PL/pgSQL.-** (Procedural Language/PostgreSQL Structured Query Language) es un lenguaje imperativo provisto por el gestor de base de datos PostgreSQL.
[37]**PL/Perl.-** Es un lenguaje de procedimientos con el apoyo del PostgreSQL
[38]**DO.-** Bloques anónimos de código de procedimientos

- Soporta el almacenamiento de objetos binarios grandes (gráficos, videos, sonido)

- APIs para programar en C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, PHP, Lisp, Scheme, Qt y muchos otros.

SQL.

- SQL92, SQL99, SQL2003, SQL2008.
- Llaves primarias (primary keys) y foráneas (foreign keys).
- Check, Unique y Not null constraints.
- Restricciones de unicidad.
- Columnas auto-incrementales.
- Índices compuestos, únicos, parciales y funcionales en cualquiera de los métodos de almacenamiento disponibles, B-tree, R-tree, hash ó GiST.
- Sub-selects.
- Consultas recursivas.
- Funciones 'Windows'.
- Joins.
- Vistas (views).
- Disparadores (triggers) comunes, por columna, condicionales.
- Reglas (Rules).
- Herencia de tablas (Inheritance).
- Eventos LISTEN/NOTIFY.

Límites.

Entre los principales límites están:

Límite	Valor
Máximo tamaño base de dato	Ilimitado (Depende de tu sistema de almacenamiento)
Máximo tamaño de tabla	32 TB
Máximo tamaño de fila	1.6 TB
Máximo tamaño de campo	1 GB
Máximo numero de filas por tabla	Ilimitado
Máximo numero de columnas por tabla	250 - 1600 (dependiendo del tipo)
Máximo numero de indices por tabla	Ilimitado

Ilustración 12. Límites físicos de PostgreSQL.

[39]**UUID.-** La intención de los UUID es habilitar, a los sistemas distribuidos, un identificador de información único sin una importante coordinación central

3.2. Análisis de submódulos del Sistema.

Una vez definido la tecnología de desarrollo, se analizarán los módulos a desarrollar detenidamente con un enfoque en la parte financiera de la Subasta Ganadera.

A continuación, se muestra un diagrama[40] generalizado del sistema financiero de la Subasta Ganadera.

Ilustración 13. Análisis de los módulos del Sistema.

3.2.1. Submódulo Contable.

El Sistema de Contabilidad es parte de la organización financiera que comprende el proceso de las operaciones patrimoniales de la Subasta Ganadera, expresadas en términos financieros, desde la entrada original a los registros de contabilidad hasta el informe sobre ellas, la interpretación y consolidación contable; comprende, además, los documentos, los registros y los archivos de las transacciones.

El objetivo principal de este módulo es, registrar todos los eventos contables generados por las operaciones comerciales y productivas que realiza la Subasta apoyando la gestión

[40]**Diagrama.-** Esquema de información que representa datos y o procesos.

financiera, la gestión ejecutiva, y manteniendo los libros legales requeridos por las entidades externas de acuerdo a la normativa legal vigente.

A continuación, se presenta un esquema del Módulo Contable.

Ilustración 14. Estructura del Módulo de Contabilidad.

Plan de Cuentas.- Para el desarrollo de esta parte del sistema contable hemos analizado el plan general de cuentas de instituciones estatales y se ha realizado un plan de cuentas que se apegue a las necesidades de la institución.

Dentro del sistema utilizaremos la siguiente estructura como ejemplo:

CODIGO	NOMBRE	TIPO	CODIGO PADRE
1.	ACTIVOS	CONTENEDORA	0
1.1.	CAJA	DETALLE	1

Libro Diario.- Es un documento numerado, que le permite registrar en forma cronológica todas las transacciones realizadas por la empresa. El libro diario es el registro contable principal en cualquier sistema contable, en el cual se anotan todas las operaciones. El primer registro de una operación se hace en el diario.

Es el libro en el cual se registran todos los ingresos y egresos efectuados por la empresa, en el orden que se vayan realizando durante el período (compra, ventas, pagos, cobros, gastos, etc.). Éste libro consta de dos columnas: la del Debe y la del Haber. Para que los registros sean válidos deben asentarse en el libro debidamente autorizado.

A esta operación se le llama asentar en el diario. El diario es un libro de registro original o de primera anotación.

El asiento de cada transacción indica qué cuentas deben cargarse y cuáles deben acreditarse posteriormente en el mayor.

La estructura del sistema se divide en dos partes sumamente importantes: Maestro – Detalle.

- Maestro*:* Este elemento permite registrar los siguientes campos:

  a) Nro. Asiento contable.- Es el numero automático que genera al ingresar un asiento contable.
  b) Fecha.- Es la fecha de ingreso del asiento diario. Ejemplo: 27/Enero/2009.
  c) Motivo o detalle.- Se describe rápidamente el porqué del asiento contable, además debe existir un documento físico que verifique el asiento.

- Detalle:

  d) Cuenta.- Aquí se ingresa el código de la cuenta contable anteriormente registrada en el plan general de cuentas de la empresa. Ejemplo: 1.1.
  e) Concepto.- Una vez ingresado el código de la cuenta se desplegará en esta opción la descripción de la cuenta. Ejemplo Caja
  f) Debe.- Se registran los datos de lo que obtendrá la empresa.
  g) Haber.- Se registran los datos de lo que entregará la empresa.

A continuación, un ejemplo de un asiento contable:

		DEBE	HABER
Asiento No. 1	**Fecha: 1/11/11 0:00**		
3231	SUELDOS Y SALARIOS	0.00	100.00
1111	CAJA	100.00	0.00
	Estado de situacion Inicial		

Ilustración 15. Ejemplo de Asiento Contable

Mayorización: La Mayorización es el proceso mediante el cual conforme van apareciendo las cuentas en el Diario General se utiliza una T para en ella registrar los valores debitados o acreditados previamente en el Libro Diario, como es lógico este proceso no se puede dar sin previamente disponer de un Libro Diario en el que ya existan registros.

Se debe tener presente lo siguiente:

1) Las cuentas de ACTIVO tienen Saldo Deudor.
2) Las cuentas de PASIVO tienen Saldo Acreedor.
3) Las cuentas de CAPITAL tienen Saldo Acreedor.
4) Las cuentas de EGRESOS tienen Saldo Deudor.
5) Las cuentas de INGRESOS tienen Saldo Acreedor.

La Mayorización se hace de cada una de las cuentas contables de detalle.

Cierre de Período: En este proceso consistente en cerrar o cancelar las cuentas de resultados y llevar su resultado a las cuentas de balance respectivas.

Al finalizar un periodo contable, se debe proceder a cerrar las cuentas de resultado para determinar el resultado económico del ejercicio o del periodo que bien puede ser una pérdida o una utilidad.

En el sistema una vez cerrado el período contable se procederá a generar el asiento de cierre.

Balance de Comprobación:

Es un instrumento financiero que se utiliza para visualizar la lista del total de los débitos y de los créditos de las cuentas, junto al saldo de cada una de ellas (ya sea deudor o acreedor). De esta forma, permite establecer un resumen básico de un estado financiero.

3.2.2. Submódulo de Inventarios de Bodega.

Este módulo permite llevar un control exacto de todos los instrumentos y equipos que se encuentran dentro de la bodega.

Es importante señalar que no es un sistema de inventario de puntos de venta.

A continuación, se presenta el esquema del Sub-módulo de Inventarios:

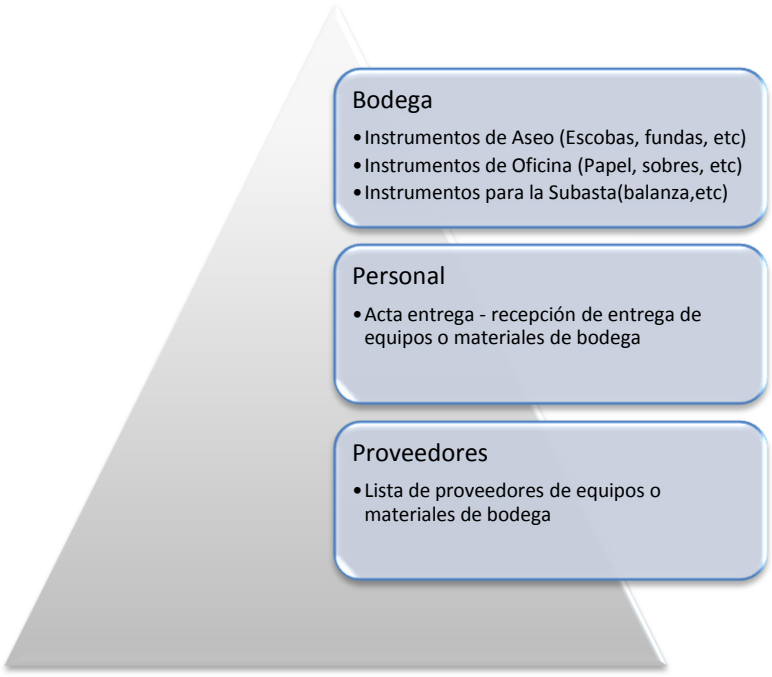

Ilustración 16. Estructura del Módulo de Inventarios.

Bodega: Esta parte del sistema ayudará a llevar un control sobre la entrada y salida del material de la misma, mediante el sistema se podrá registrar pedidos por parte del empleado.

Este módulo tiene como finalidad manejar el stock de los materiales, incluso ayudará a verificar el stock físico y del sistema en caso de una Auditoria a la Institución.

Personal: Se deberá registrar a todas las personas encargadas de realizar operaciones con bodega, además se emitirá un acta Entrega-Recepción por parte de la persona encargada del sistema y la persona solicitante de materiales.

Proveedores: En este elemento del sistema se especifica los proveedores de los materiales y equipos de bodega.

3.6.3. Submódulo de Ventas.

El Submódulo de ventas ayudará a la Institución a generar facturas o comprobantes de pago de la subasta ganadera para los compradores de ganado.

Este sistema es el que permite la integración entre el módulo de subasta y el módulo financiero.

A continuación, se muestra un diagrama del Submódulo de Ventas:

Ilustración 17. Estructura del Módulo de Ventas.

Compradores.

En esta parte del sistema se muestra todas las personas que se registraron como compradores, además cada comprador está asignado un código que ayuda al proceso de

Licitación y adjudicación de los animales ofertados en la subasta.

Vendedores.

Se muestra el listado de los vendedores previamente ingresados al momento de ingresar con los animales al proceso de subasta.

Venta.

La venta comprende un proceso específico dentro del sistema, a continuación se detalla dicho proceso.

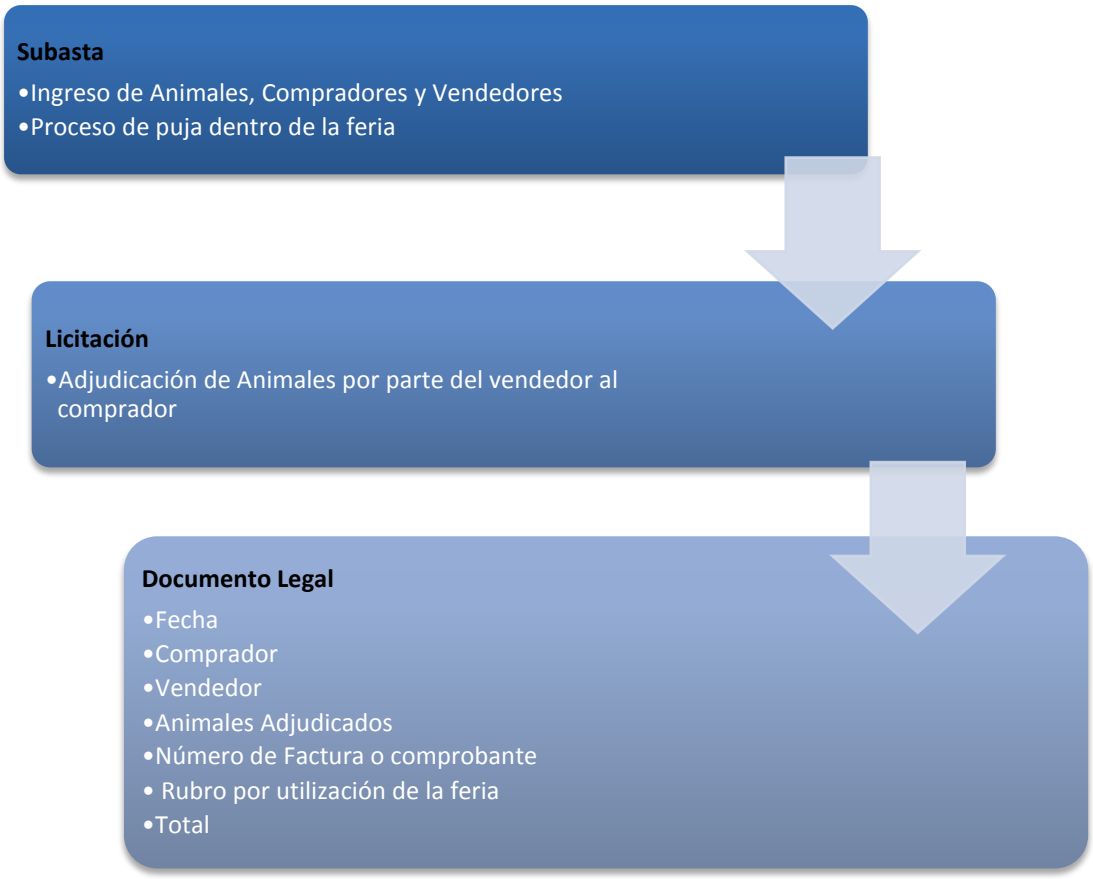

Ilustración 18. Diagrama del Proceso de Venta.

Toda transacción se ingresará automáticamente en el sistema financiero. En el Capítulo IV se analizará y explicará detenidamente los casos de uso de este módulo.

3.2.4. Submódulo Cuentas por Cobrar.

Este módulo es encargado de registrar las cuentas por cobrar a los deudores, arrendatarios (locales de comida, comerciales) de las instalaciones de esta empresa.

A continuación, un diagrama descriptivo de las entidades del sistema:

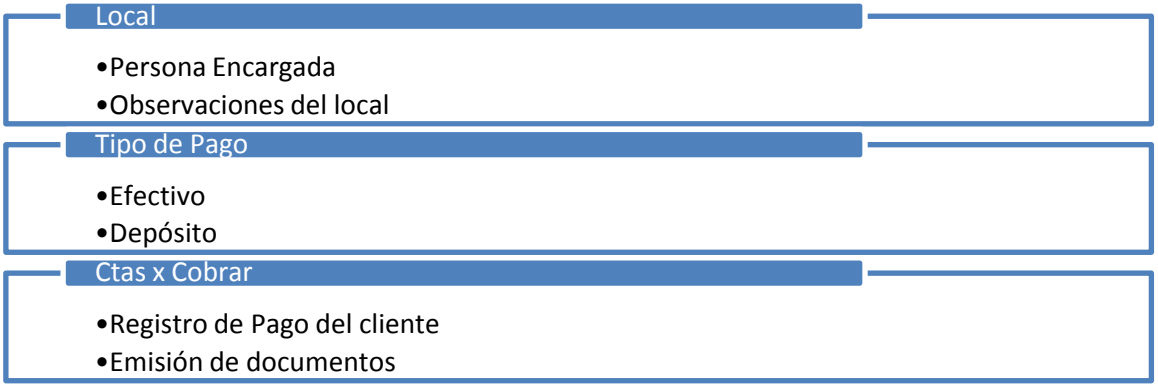

Ilustración 19. Estructura del Módulo de Cuentas por Cobrar.

Local.- Aquí se realizará la gestión y administración de Locales que se arriendan en la feria, este módulo ayudará a llevar organizadamente los datos de dicho local tales como: Número de local, Actividad comercial, Persona encargada del local.

Tipos de Pago.- Se almacenará la forma de pago que realiza el cliente, puede hacerse de dos maneras:

- Efectivo.- Se registra automáticamente en libro diario como caja.
- Depósitos.- Se registra automáticamente en libro diario como bancos.

Cuentas por Cobrar.- Aquí es la parte donde el usuario interactúa con la interfaz, se mostrará tres partes fundamentales dentro del sistema:

- Registro del pago del Cliente.
- Emisión de comprobante de pago, e ingreso al módulo contable.

3.2.5. Submódulo Cuentas por Pagar.

En éste módulo constarán los registros de las cuentas por pagar a los proveedores de equipos (informáticos, de limpieza), suministros de oficina.

A continuación, se presenta un diagrama de este módulo:

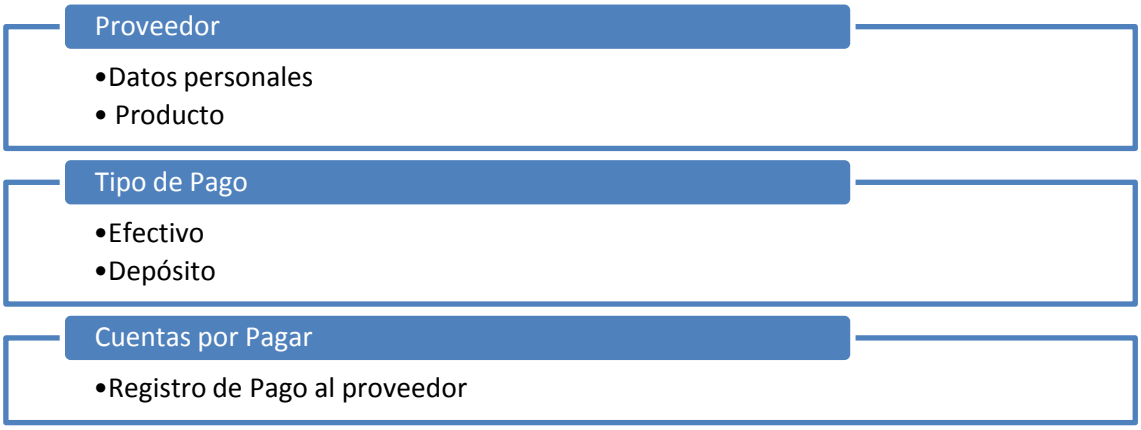

Ilustración 20. Estructura del Módulo de Cuentas por Pagar.

Proveedor.- Se administrará todos los proveedores de productos y equipos que se ingresará a la bodega.

Tipos de Pago.- Se almacenará la forma de pago que realiza el cliente, puede hacerse de dos maneras:

- Efectivo.- Se registra automáticamente en libro diario como caja.
- Depósitos.- Se registra automáticamente en libro diario como bancos.

Cuentas por Pagar.- En esta parte del sistema se procederá a realizar el siguiente proceso:

- Se ingresará el pago al proveedor, el número de factura o el comprobante de pago respectivo, y la cantidad cancelada.
- Para créditos de más de un mes se anunciará la próxima fecha de pago y la respectiva cantidad.

CAPITULO IV
DESARROLLO DEL APLICATIVO

4.1. Planificación del Sistema.

El objetivo de la Planificación del proyecto de este software es proporcionar un marco de trabajo que nos permita a los programadores hacer estimaciones razonables de recursos, costos y planificación temporal.

Estas estimaciones se hacen dentro de un marco de tiempo limitado al comienzo del proyecto de software, y deberían actualizarse regularmente a medida que progresa el proyecto. Además las estimaciones deberían definir los escenarios del mejor caso, y peor caso, de modo que los resultados del proyecto pueden limitarse.

La aplicación SUBGANA será desarrollada utilizando la metodología de unificación de procesos, la misma que permitirá utilizar sus normas para definir el proyecto y de la misma manera permitirá organizar de mejor manera tanto para desarrollo como para documentación.

4.1.1. Propósito.

El propósito para este proyecto es definir, planificar y controlar el desarrollo de un producto razonable con un costo mínimo y un período de tiempo específico.

Además se debe analizar detenidamente las variables del desarrollo del proyecto para que en este proceso se fortalezca las bases del sistema y no se deje de lado este importante proyecto.

4.1.2. Alcance.

Utilizando la metodología de RUP se ha generado el plan de desarrollo de este proyecto, para así poder tener la documentación necesaria en el momento que así se lo requiera.

Para definir bien los requisitos y objetivo de este proyecto, la empresa EP-FYPROCAI nos ha colaborado con la información necesaria para el desarrollo del proyecto.

4.1.3. Vista General del Proyecto.

La información que a continuación se incluye ha sido extraída de ciertas reuniones que se han realizado con la persona encargada de brindarnos la ayuda necesaria para realizar la aplicación.

Dependiendo de la necesidad de implementar un sistema financiero en la Feria Ganadera que permita llevar el registro diario de las transacciones que se realizarán en dicha feria, se determina crear un software financiero que tendrá como finalidad optimizar el proceso de manejo y administración de los recursos empresariales de la institución antes mencionada.

El proyecto se crea con la finalidad de satisfacer necesidades específicas de la empresa.

En el Capítulo III se ha analizado a detalle el sistema, lo que permite distinguir los siguientes módulos:

- Módulo de Contabilidad.
- Módulo de Inventarios de bodega.
- Módulo de Cuentas por Pagar.
- Módulo de Cuentas por Cobrar.
- Módulo de Ventas.

4.1.4. Suposiciones y Restricciones.

Las suposiciones y restricciones respecto a la aplicación, y que se derivan directamente de las entrevistas con la persona encargada por parte de la empresa.

Puntos críticos:

- Seguridad en la aplicación.
- Velocidad de transferencia de información en los procesos del sistema.
- Manejo y adaptación del usuario directo del sistema.
- Integración de la aplicación de la subasta con nuestro módulo financiero.

4.1.5. Entregables del Proyecto.

A continuación, se indican y describen cada uno de los artefactos que serán generados y utilizados por el proyecto**,** estos constituyen los artefactos entregables. Esta lista constituye la configuración o esquema del RUP desde la perspectiva de artefactos el mismo que será utilizado para el desarrollo de este proyecto.

Es preciso destacar que de acuerdo a la filosofía de RUP (todo proceso es iterativo e incremental), todos los artefactos son objeto de modificaciones a lo largo del proceso de desarrollo, con lo cual, sólo al término del proceso podríamos tener una versión definitiva y completa de cada uno de ellos. Sin embargo, el resultado de cada iteración y los hitos del proyecto están enfocados a conseguir un cierto grado de completitud y estabilidad de los artefactos.

Plan de desarrollo del software.

Es el presente documento.

Visión.

En este documento se define la perspectiva de los usuarios con respecto al proyecto, especificando las necesidades y características del mismo.

Glosario.

Permite establecer una terminología consensuada.

Especificaciones de casos de uso.

Es una descripción detallada utilizando una plantilla de documento, donde se incluyen: precondiciones, post-condiciones, flujo de eventos, requisitos no-funcionales asociados. También, para casos de uso, cuyo flujo de eventos sea complejo podrá adjuntarse una representación gráfica mediante un diagrama de actividad.

Segmenta el conjunto de necesidades atendiendo a la categoría del usuario que participa en el mismo, está basado en lenguaje natural, es decir, es accesible por los usuarios.

Prototipos de interfaces de usuario.

Se trata de prototipos que permiten al usuario hacerse una idea más o menos precisa de la interfaz que proveerá la aplicación y así, conseguir retroalimentación de su parte respecto a los requisitos del proyecto. Estos prototipos se realizarán con las mismas herramientas con las que se realizará todo el proyecto obteniendo un producto en el que se pueda navegar y ejecutar la aplicación. Así mismo, este artefacto, será desechado en la fase de construcción en la medida que el resultado de las iteraciones vayan desarrollando el producto final.

Modelo de análisis y diseño.

Este modelo establece la realización de los casos de uso en clases y pasando desde 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.

Modelo de datos.

Pronosticando que la persistencia de la información del sistema será soportada por una base de datos relacional, este modelo describe la representación lógica de los datos persistentes, de acuerdo con el enfoque para modelado relacional de datos. Para expresar este modelo se utiliza un diagrama de clases (donde se utiliza un profile UML para modelado de datos, para conseguir la representación de tablas, vistas, claves).

Modelo de implementación.

Este modelo es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros ejecutables, ficheros de código fuente, y todo otro tipo de ficheros necesarios para la implantación y despliegue del sistema. (Este modelo es sólo una versión preliminar al final de la fase de elaboración, posteriormente tiene bastante refinamiento).

Casos de prueba.

Cada prueba es especificada mediante un documento que establece las condiciones de

ejecución, las entradas de la prueba, y los resultados esperados. Estos casos de prueba son aplicados como pruebas de regresión en cada iteración.

Lista de riesgos.

Este documento incluye una lista de los riesgos conocidos y vigentes en el proyecto, ubicados en orden decreciente de importancia y con acciones específicas de contingencia o para su mitigación.

Manual de instalación.

Este documento incluye las instrucciones para realizar la instalación del producto.

Material de apoyo al usuario final.

Corresponde a un conjunto de documentos y facilidades de uso del sistema, incluyendo: guías del usuario, guías de operación, guías de mantenimiento.

Producto.

La estructura del sistema SUBGANA será empaquetada y almacenada en un CD con los mecanismos apropiados para facilitar su instalación. El producto, a partir de la primera iteración de la fase de construcción es desarrollado incremental e iterativamente, obteniéndose una nueva versión al final de cada iteración.

4.1.6. Organización del Proyecto.

Participantes en el proyecto:

El personal de apoyo que guiará el desarrollo y ejecución del proyecto, por parte de la institución.

***Tutor*.-** Es la persona encargada de verificar la veracidad del sistema, además de colaborar en la construcción de la arquitectura del sistema.

***Programador*.-** Los conocimientos que poseen y con el que debe cumplir este perfil es de: conocimiento amplio en manejo de herramientas J2EE, Java, JPA, Primefaces.

***Administrador*.-** Es la persona encargada de gestionar los procesos como también de administrar la base de datos.

Usuarios Externos.- Son aquellas personas que utilizarán el sistema.

Roles y Responsabilidades.

PUESTO	RESPONSABILIDAD
Tutor	Tiene una importancia vital ya que es él, quien generalmente asume la tutoría del proyecto encargándose así de proveer del personal necesario, los recursos imprescindibles y de tomar las decisiones que ayuden a que el proyecto cumpla con los objetivos propuestos. Es por esta razón que es muy importante que el tutor esté familiarizado con la gestión de proyectos y todas las técnicas y herramientas que la componen y que le facilitarán su labor al trabajar en un proyecto.
Programador	Contribuir con los conocimientos adquiridos durante su fase de aprendizaje en la universidad. Esta persona será la encargada de desarrollar todo el sistema, modelo de datos, modelo del sistema, seguridad y accesibilidad.
Administrador	Gestiona y administra el sistema, además de realizar diariamente respaldos de la base de datos.
Usuarios Externos	Encargados del uso y manejo adecuado del sistema

4.1.7. Plan del Proyecto.

En esta sección se presenta la organización en fases, iteraciones y el calendario del proyecto.

Plan de Fases.

El desarrollo se llevará a cabo en base a fases con una o más iteraciones en cada una de ellas. La siguiente tabla muestra la distribución de tiempos y el número de iteraciones de cada fase.

Fase	Nro. Iteraciones	Duración
Fase de Inicio	1	5 semanas
Fase de Elaboración	1	8 semanas
Fase de Construcción	5	48 semanas
Fase de Transición	1	4 semanas

Los hitos que marcan el final de cada fase se describen en la siguiente tabla:

Descripción	Hito
Fase de Especificación	En esta fase se desarrollarán los requisitos del modelo desde la perspectiva del usuario. Se tiene que especificar claramente los requisitos o funcionalidades específicas que desea para la aplicación que está solicitando. En un proyecto J2EE se presentan algunas particularidades importantes frente a este aspecto: ➢ Diseño ➢ Tiempo-costo ➢ Satisfacción del usuario ➢ Compatibilidad del modelo ➢ Forma de acceso ➢ Precisión y calidad ➢ Características específicas de cada módulo ➢ Usuarios y clientes ➢ Recursos necesarios

Fase de Elaboración	En esta fase se analizan los requisitos y se desarrolla un prototipo de arquitectura (incluyendo las partes más relevantes y/o críticas del sistema). Al final de esta fase, todos los casos de uso correspondientes a requisitos que serán implementados en la primera release de la fase de Construcción o desarrollo. La iteración tendrá como objetivo la identificación y especificación de los principales casos de uso.
Fase de Desarrollo	Durante la fase de desarrollo se terminan de analizar y diseñar todos los casos de uso, refinando el modelo de análisis / diseño, el producto se construye en base a dos iteraciones, cada una produciendo una release a la cual se le aplican las pruebas y se valida con el usuario. El hito que marca el fin de esta fase es la versión de la release 2.0, con la capacidad operacional parcial del producto que se haya considerado como crítica, lista para ser entregada a los usuarios para pruebas beta.
Fase de Transmisión de Tecnología y Puesta en Marcha	En esta fase se presentará el release final para su distribución, asegurando una implantación y cambio del sistema previo de manera adecuada, incluyendo el entrenamiento de los usuarios. El hito que marca el fin de esta fase, incluye la entrega de toda la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario, la finalización del entrenamiento de los usuarios y el empaquetamiento del producto.

Calendario del Proyecto.

A continuación, se presenta un calendario de las principales tareas del proyecto incluyendo sólo las fases de inicio y elaboración. Como se ha comentado, el proceso iterativo e incremental de RUP está caracterizado por la realización en paralelo de todas las disciplinas de desarrollo a lo largo del proyecto, con lo cual la mayoría de los artefactos son generados muy tempranamente en el proyecto pero van desarrollándose en mayor o menor grado de acuerdo a la fase e iteración del proyecto.
La siguiente figura ilustra este enfoque, en ella lo ensombrecido, marca el énfasis de cada disciplina (Flujo de Trabajo) en un momento determinado del desarrollo.

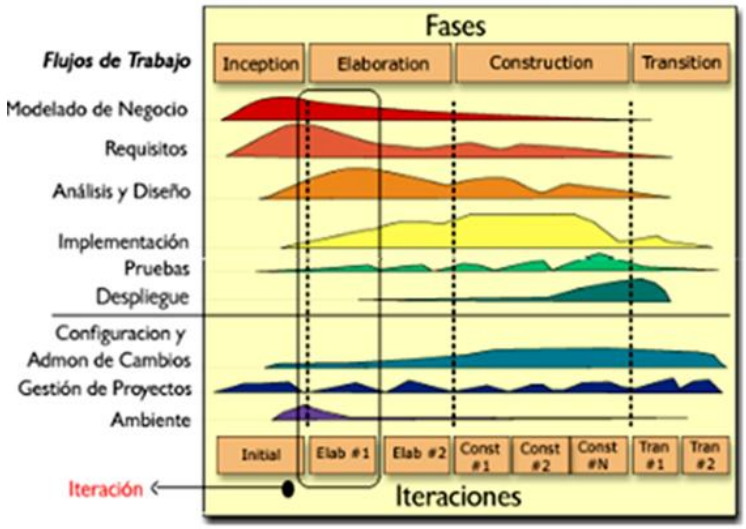

Ilustración 21. Calendario del Proyecto.

Para este proyecto se ha establecido el siguiente calendario. La fecha de aprobación indica cuándo el artefacto en cuestión tiene un estado de completitud suficiente para someterse a revisión y aprobación, pero esto no quita la posibilidad de su posterior refinamiento y cambios.

Disciplinas / Artefactos generados o modificados durante la FASE DE INICIO	Comienzo	Aprobación
Modelado del Negocio		
➢ Modelo de Casos de Uso del Negocio ➢ Modelo de Objetos del Negocio	Semana 2	Semana 3

Requisitos		
Modelo de Casos de Uso	Semana 3	siguiente fase
Especificación de Casos de Uso	Semana 4	siguiente fase
Especificaciones Adicionales	Semana 4	siguiente fase
Análisis / Diseño		
Modelo de Análisis / Diseño	Semana 4	siguiente fase
Modelo de Datos	Semana 4	siguiente fase
Implementación		
Prototipos de Interfaces de Usuario	Semana 3	Semana 5
Modelo de Implementación	Semana 4	siguiente fase
Pruebas		
Casos de Pruebas Funcionales	Semana 5	Semana 5
Despliegue		
Modelo de Despliegue	Semana 4	Semana 5
Gestión de Cambios y Configuración	Durante todo el proyecto	
Gestión del proyecto		
Plan de Desarrollo del Software en su versión 1.0 y planes de las Iteraciones	Semana 1	Semana 5
Ambiente	Durante todo el proyecto	

Disciplinas / Artefactos generados o modificados durante la FASE DE ELABORACIÓN	**Comienzo**	**Aprobación**
Modelado del Negocio		
➢ Modelo de Casos de Uso del Negocio ➢ Modelo de Objetos del Negocio	Semana 5	aprobado
Requisitos.		
Modelo de Casos de Uso	Semana 3	Semana 5
Especificación de Casos de Uso	Semana 4	Semana 5
Especificaciones Adicionales	Semana 4	Semana 5
Análisis / Diseño		
Modelo de Análisis / Diseño	Semana 6	Revisar en cada iteración
Modelo de Datos	Semana 7	Revisar en cada

		iteración
Implementación		
Prototipos de Interfaces de Usuario	Semana 8	Revisar en cada iteración
Modelo de Implementación	Semana 10	Revisar en cada iteración
Pruebas		
Casos de Pruebas Funcionales	Semana 11	Revisar en cada iteración
Despliegue		
Modelo de Despliegue	Semana 11	Revisar en cada iteración
Gestión de Cambios y Configuración	Durante todo el proyecto	
Gestión del proyecto		
Plan de Desarrollo del Software en su versión 1 y planes de las Iteraciones	Semana 1	Revisar en cada iteración
Ambiente	Durante todo el proyecto	

Seguimiento y control del proyecto.

Gestión de requisitos.

Los requisitos del sistema son especificados en el Capítulo III de este documento. Cada requisito tendrá una serie de atributos tales como importancia, estado, iteración donde se implementa, etc. Estos atributos permitirán realizar un efectivo seguimiento de cada requisito. Los cambios en los requisitos serán gestionados mediante una solicitud de cambio, las cuales serán evaluadas y distribuidas para asegurar la integridad del sistema y el correcto proceso de gestión de configuración y cambios.

Control de plazos.

El calendario del proyecto tiene un seguimiento semanal por las personas encargadas de revisión del proyecto.

Control de calidad.

Los defectos detectados en las revisiones y formalizados también en una solicitud de cambio tendrán un seguimiento para asegurar la conformidad respecto de la solución de dichas deficiencias.

Gestión de riesgos

A partir de la fase de inicio se mantendrá una lista de riesgos asociados al proyecto y de las acciones establecidas como estrategia para mitigarlos o acciones de contingencia.

4.2. Análisis y Elaboración del Sistema.

Una vez definida la solución a desarrollar, es necesario realizar una especificación detallada de la misma con el objetivo de preparar su diseño y su arquitectura. Esta especificación se realiza durante esta etapa del desarrollo, donde es muy importante la interacción con los usuarios de éste para descartar la posibilidad de omisión y/o errores que lleven a diseños no adecuados.

4.2.1. Definición del Sistema.

Para realizar la definición de la aplicación fue importante realizar trabajo de campo en donde se definieron los requerimientos del sistema SUBGANA. Además, es importante mencionar que la aplicación es desarrollada en función a las especificaciones de la empresa beneficiada.

El módulo financiero nos permite llevar un control adecuado de las transacciones comerciales que se realizan por parte de la empresa.

Cabe considerar que el análisis hecho en primeras reuniones nos ayudó, durante el proceso de desarrollo, a realizar el diseño de la solución adoptada.

4.2.2. Establecimiento de Requisitos

Requisitos legales.

Dado que la empresa procesará datos relativos a vendedores y compradores de ganado, se debe cumplir todas las normativas vigentes en protección de datos y acceso a la información. Por ello, el diseño deberá contemplar los métodos de identificación y control de acceso a la información de acuerdo a la ley y con los mecanismos de certificación, control/cifrado adecuados para cumplir lo establecido legalmente.

Requisitos de propiedad intelectual y licencias.

El proyecto debe basarse en software libre y con licencias lo menos restrictivas posible, teniendo en cuenta las inversiones necesarias, tanto en la adquisición inicial como en el servicio postventa de actualizaciones y mantenimiento.

Requisitos de acceso único.

Los datos de la empresa se centralizarán en un servicio distribuido (interno). Esta opción deberá contemplar los requisitos de seguridad para la aceptación del servicio de las máquinas clientes y con los criterios adecuados en cuanto a los permisos de lectura-escritura para cada máquina cliente. También los datos de usuario de la red deberán estar centralizados por un servicio de información distribuida (NIS), de modo que cada usuario tenga en la feria una única cuenta y una palabra clave para el acceso a los servicios desde todas las máquinas. Dada la existencia de máquinas Windows, será necesaria la instalación de Mozilla Firefox para la ejecución de la aplicación.

Requisitos sobre la base de datos.

La base de datos de la empresa deberá ser sólo accesible a las máquinas de la Intranet, ya que contendrá información sensible sobre contabilidad, ventas y la información personal de los empleados, compradores y vendedores. Esta base de datos podrá accederse a través de peticiones SQL desde el aplicativo web.

Requisitos del sistema de seguridad.

El sistema informático de la empresa (servidores y máquinas clientes) deberá tener un diseño basado en un cortafuegos institucional con una definición clara de zonas (DMZ), con direcciones privadas y traslación. El sistema deberá contar con las herramientas de detección y análisis adecuado que permitan a los administradores realizar un control

adecuado del sistema informático. La organización interna de la red se basará en un sistema de distribución de direcciones IP automático (DHCP) y un servidor de nombres secundario interno (DNS).

Requisitos de impresión en red.

Desde cada máquina de la empresa se podrá acceder a un servicio de impresión centralizado mediante colas, donde cada máquina cliente deberá ser identificada y contabilizado el trabajo de impresión realizado.

Requisitos del aplicativo web.

La aplicación web deberá admitir servicios complementarios como puede ser la subasta a través del internet.

Además es importante señalar que el sistema deberá ser escalable, es decir, tendrá la habilidad para extender el margen de operaciones sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos.

Debido al gran impulso del uso del software libre en las instituciones públicas, nuestra aplicación deberá ser multiplataforma.

Requisitos tecnológicos, mantenimiento y administración.

Dada la magnitud del proyecto, es crucial para el buen funcionamiento posterior a la implantación/integración de la formación de los administradores y de los usuarios. Esta formación podrá realizarse durante los períodos de prueba para generar confianza y para que acepten la nueva tecnología sin preconceptos ni falsas expectativas.

Al ser el sistema web no limita el número de usuarios para acceder al sistema, sin embargo el sistema inicialmente tendrá las siguientes necesidades en cuanto a equipos e infraestructura física:

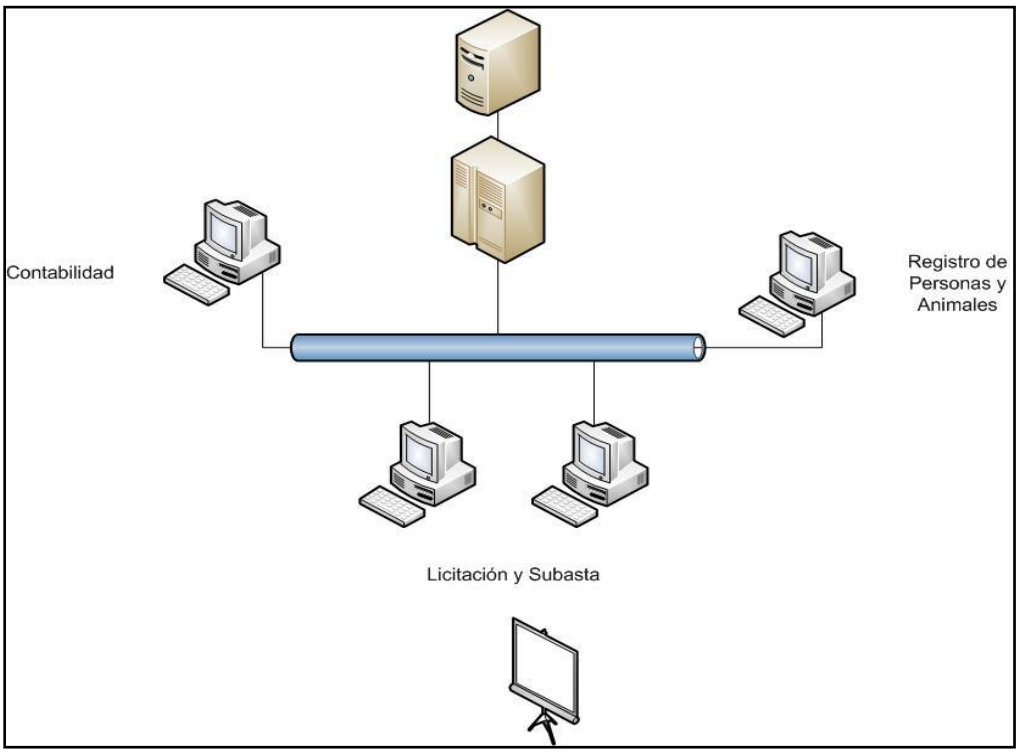

Ilustración 22. Requisitos tecnológicos del Sistema.

Requisitos de organización.

La empresa deberá realizar un organigrama de recursos humanos para redefinir las tareas y las responsabilidades en el nuevo sistema. Esta reorganización tendrá por objetivo asignar las nuevas responsabilidades entre los empleados de la misma teniendo en cuenta los nuevos servicios y productos de la empresa. De la misma manera, es conveniente que la empresa genere un “Libro Blanco” que recoja no sólo la funcionalidad de la empresa, sus servicios y productos, sino los procedimientos y las responsabilidades de cada empleado responsable en forma funcional. Esta información la conocerán todos los empleados y será donde quedará anotado por escrito las responsabilidades de cada uno de los integrantes de la plantilla de la empresa.

Requisitos de seguridad.

La empresa, además, deberá contar con un documento de seguridad que describa tanto las responsabilidades de los usuarios que trabajan con información sensible, como los procedimientos que hay que seguir en caso de que se descubran problemas relativos a la seguridad (intrusos, pérdida de confianza, vulnerabilidades, etc.) con el fin de incluir todos los pasos que deben seguir los responsables ante situaciones de este tipo.

4.2.3. Análisis mediante Casos de Uso.

Caso de Uso: Control de Acceso al Sistema.

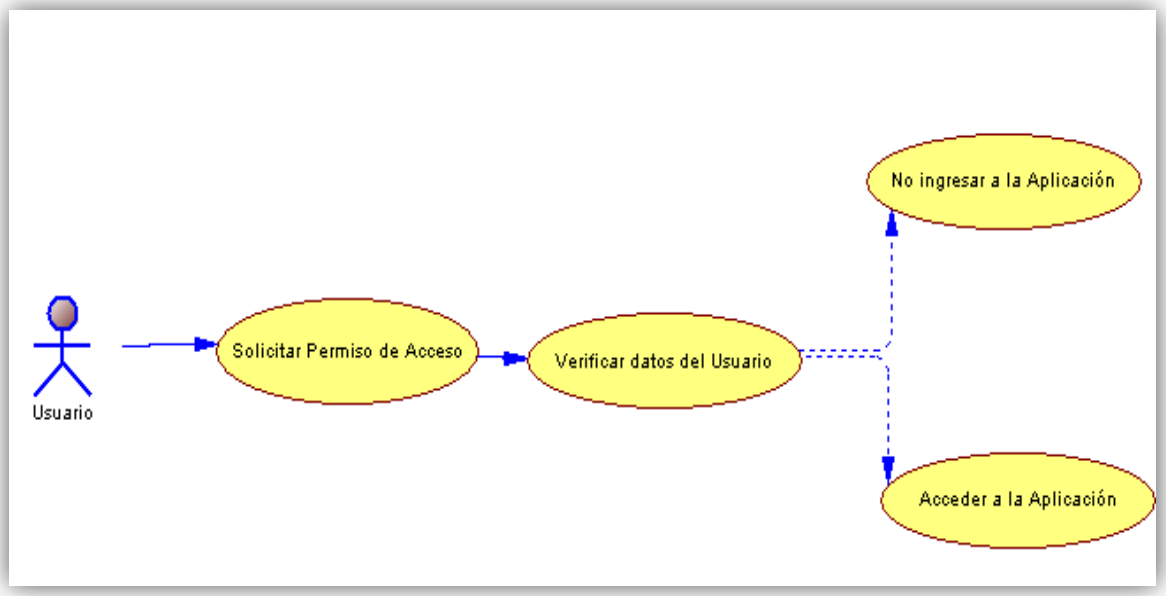

Ilustración 23. Caso de Uso: Control de Acceso al Sistema.

Descripción breve.

Es el proceso en el cual el usuario ingresa sus datos y se valida dicha información para poder acceder a la aplicación

Flujo básico de eventos.

- El usuario debe ingresar con su nombre de usuario y clave para poder acceder al sistema.
- El sistema verificará la existencia del usuario en la base de datos.
- En caso de que los datos sean correctos la aplicación activará a las opciones que tiene permiso cada perfil de este usuario.
- Si los datos son incorrectos se mostrará un mensaje bloqueando el acceso al sistema.

Flujos alternativos.

- El usuario abandona el sistema al cerrar sesión en cualquier momento.

Precondiciones.

- Tener acceso al sistema.
- Tener creado el usuario y además estar activo.

Post condiciones.

- Ninguna.

Contabilidad.

Caso de Uso: Gestión y Administración del Plan de Cuentas.

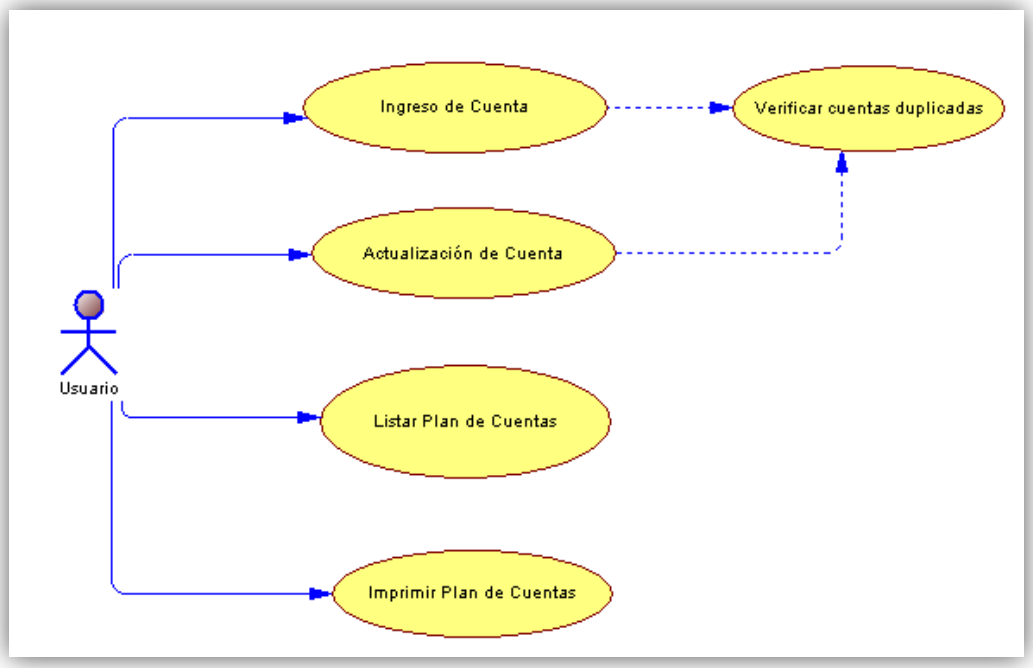

Ilustración 24. Caso de Uso: Gestión y Administración del Plan de Cuentas.

Descripción breve.

Es el proceso con el cual administra todas las cuentas contables que se utilizarán en todo el sistema financiero.

Flujo básico de eventos.

- El usuario podrá ingresar las cuentas contables necesarias para el correcto

funcionamiento del sistema. Además cada las cuentas se validan al momento de guardar para que no exista registros duplicados.

- El usuario podrá actualizar las cuentas contables.
- Todas las cuentas se mostrarán en una lista correctamente ordenada.
- El usuario podrá imprimir el plan de cuentas.

Flujos alternativos

- El usuario abandona el sistema al cerrar sesión en cualquier momento.

Precondiciones.

- Tener acceso al sistema.
- Estar logueado en el sistema.

Post condiciones.

- Ninguna.

Caso de Uso: Gestión y Administración del Libro Diario.

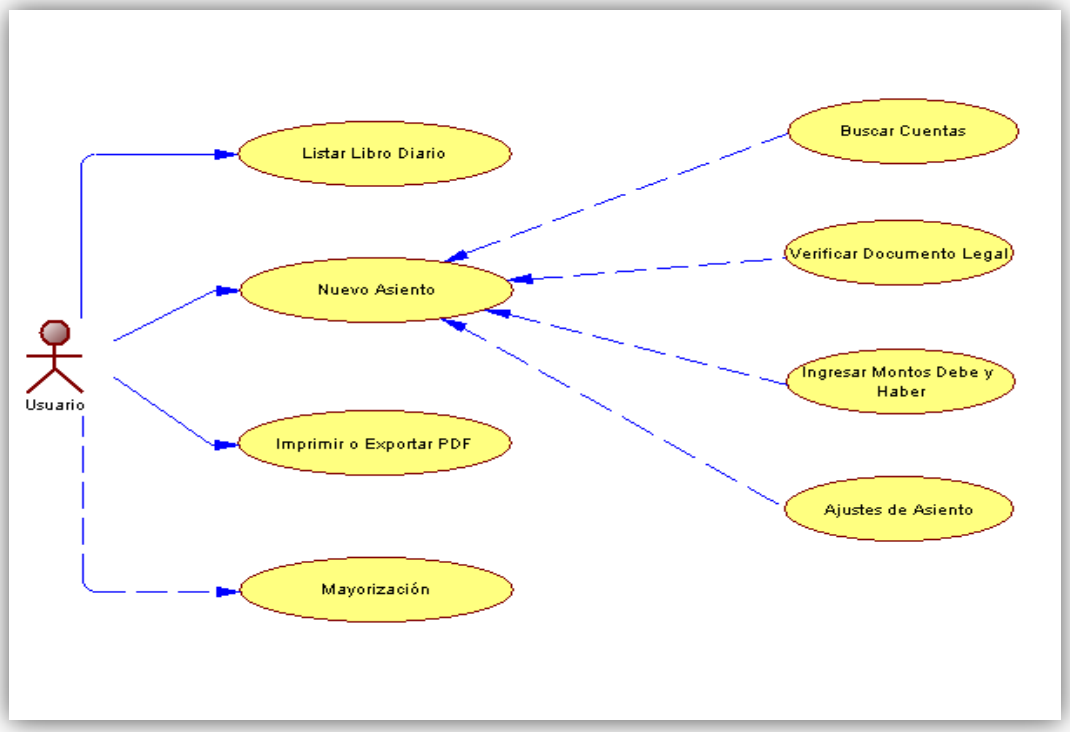

Ilustración 25. Caso de Uso: Gestión y Administración del Libro Diario.

Descripción breve.

Es el proceso con el cual se registran las transacciones o hechos económicos de la Empresa.

Este proceso es permanente y automático para los otros módulos de este sistema.

Flujo básico de eventos.

- El usuario tendrá acceso al libro diario de la empresa, sin embargo no podrá modificar ningún registro.
- Verificar el documento legal que valide el ingreso del asiento.
- Sistema valida que el total del debe sea igual al total del haber, en caso de no cumplir con este requerimiento no procederá a guardar la información.
- Consultar y seleccionar mediante búsquedas inteligentes las cuentas necesarias para ingresar un asiento contable.
- Realizar ajustes contables mediante el ingreso de un nuevo asiento.

Flujos alternativos.

- El usuario abandona el sistema al cerrar sesión en cualquier momento.

Precondiciones.

- Tener acceso al sistema.
- Estar logueado en el sistema.

Post condiciones.

- Ninguna.

Inventario.

Caso de Uso: Gestión y Administración de Productos.

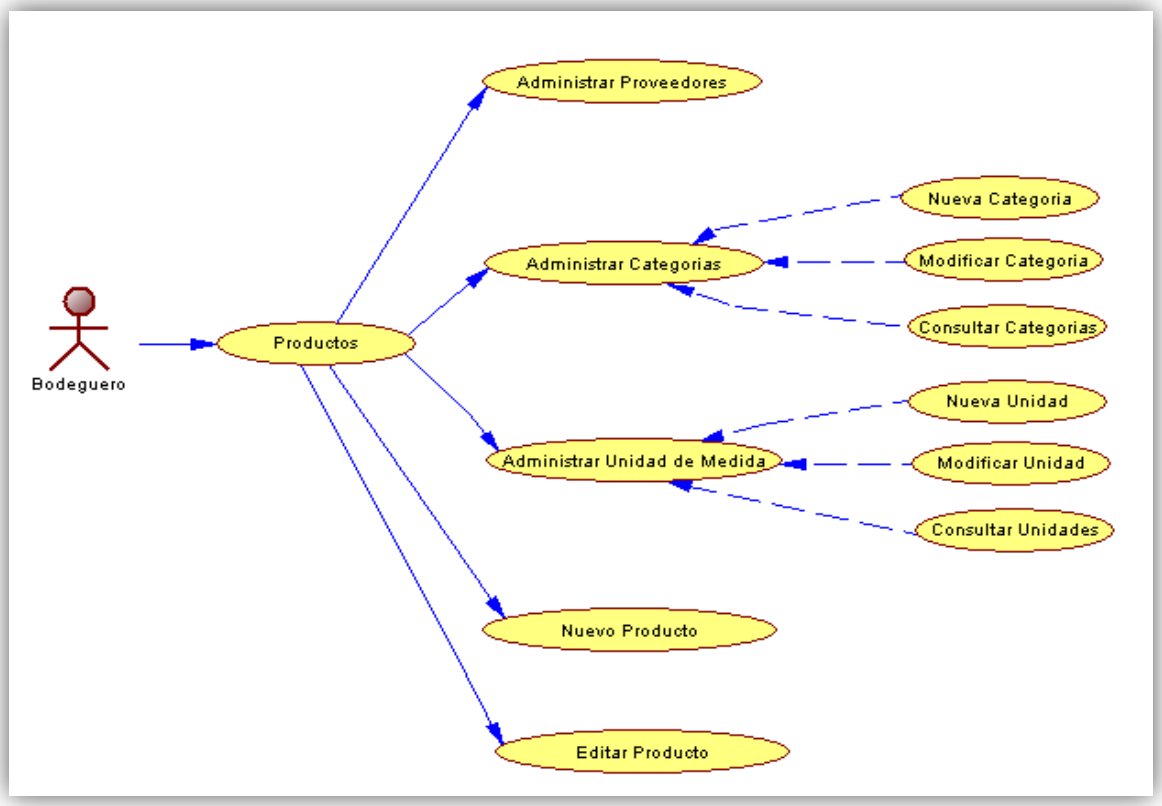

Ilustración 26. Caso de Uso: Gestión y Administración de Productos.

Descripción breve.

Es el proceso con el cual se administra los productos que serán utilizados por la empresa como útiles de oficina, de aseo entre otros.

Además es importante señalar que estos productos no serán comercializados ya que solo funcionarán como recursos para utilizarse dentro de la empresa.

Flujo básico de eventos.

- Ingresar nuevos productos.
- Consultar mediante búsquedas inteligentes cualquier producto.
- Administrar unidades de medida de los productos.
- Administrar categorías de los productos.
- Administrar proveedores.
- Imprimir los productos existentes en bodega.

Flujos alternativos.

- El usuario abandona el sistema al cerrar sesión en cualquier momento.

Precondiciones.

- Tener acceso al sistema
- Estar logueado en el sistema.

Post condiciones.

- Ninguna.

Caso de Uso: Kardex

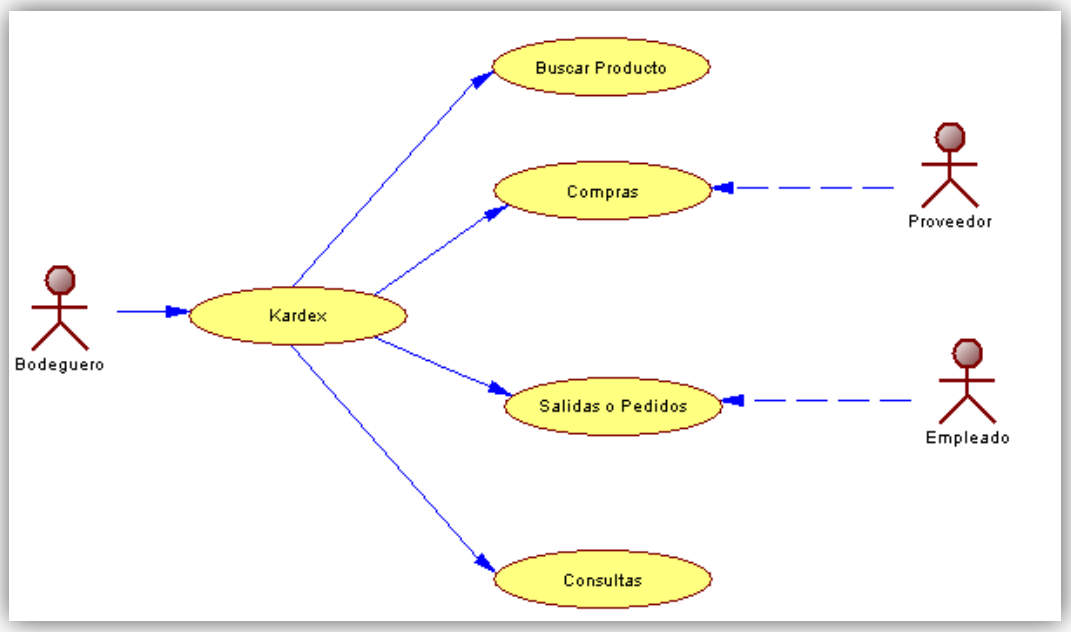

Ilustración 27. Caso de Uso: Kardex.

Descripción breve.

Es el proceso con el cual se administran las compras y salidas de cada uno de los productos existentes en bodega.

Además es un registro de manera organizada de la mercadería que se tiene en una bodega.

Flujo básico de eventos.

- Realizar compras de productos y administrar el stock del producto.
- Consultar el detalle del Kardex de cada producto.
- Administrar pedidos de los productos por parte de los empleados de la empresa.
- Imprimir el Kardex del producto.

Flujos alternativos.

- El usuario abandona el sistema al cerrar sesión en cualquier momento.

Precondiciones.

- Tener acceso al sistema.
- Estar logueado en el sistema.

Post condiciones.

- Ninguna.

Cuentas x Cobrar.

Caso de Uso: Administración de Cuentas por Cobrar

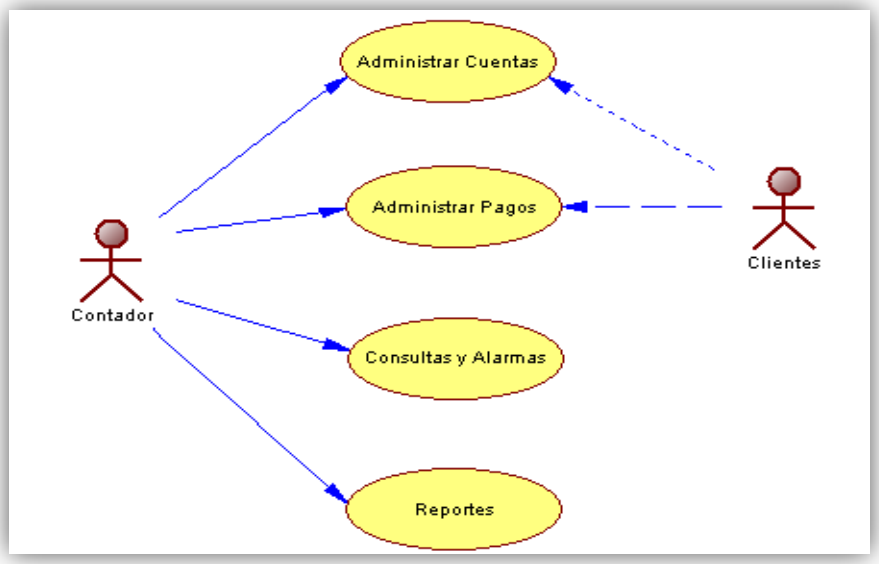

Ilustración 28. Caso de Uso: Administración de Cuentas por Cobrar.

Descripción breve.

Es el proceso con el cual se administran las cuentas y deudas de los clientes, en este caso especial los arrendatarios de los locales comerciales y locales de comida.

Flujo básico de eventos.

- Administrar cuentas, es decir cada local tendrá una cuenta.
- Administrar pagos, este proceso permitirá realizar el cobro mensual por cada uno de los locales comerciales.
- Consultar mediante búsquedas inteligentes el estado de cuenta de cada uno de los locales y también de los clientes.
- Se emitirá alarmas mensuales de las personas retrasadas del pago del mes.

Flujos alternativos.

- El usuario abandona el sistema al cerrar sesión en cualquier momento.

Precondiciones.

- Tener acceso al sistema.
- Estar logueado en el sistema.

Post condiciones.

- Ninguna.

Cuentas x Pagar.

Caso de Uso: Administración de Cuentas por Pagar.

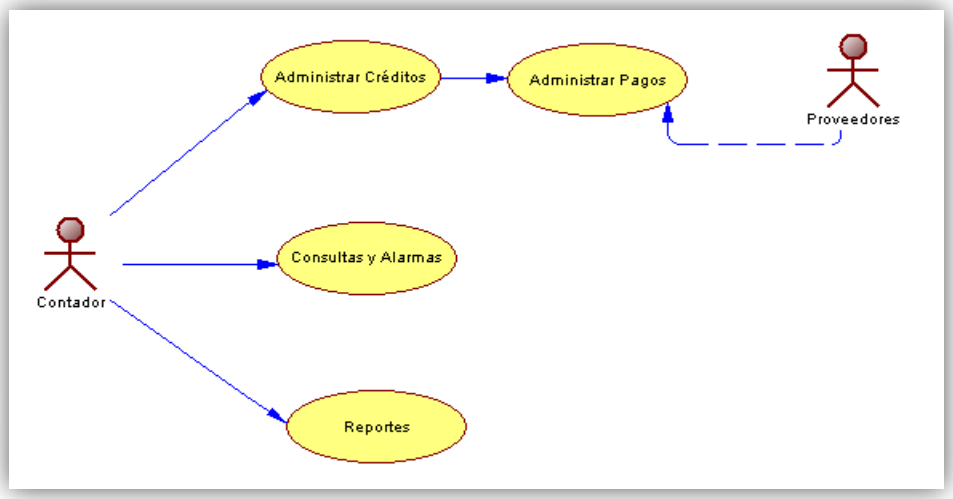

Ilustración 29. Caso de Uso: Administración de Cuentas por Pagar.

Descripción breve.

Es el proceso con el cual se administran los créditos y deudas que tiene la empresa a los proveedores, en este caso especial los proveedores de productos de oficina y de aseo.

Flujo básico de eventos.

- Administrar créditos.
- Administrar pagos, este proceso permitirá realizar el pago de acuerdo al crédito que tiene la empresa con los proveedores.
- Consultar mediante búsquedas inteligentes el estado de los créditos.
- Se emitirá alarmas mensuales de las deudas vencidas de pago.

Flujos alternativos.

- El usuario abandona el sistema al cerrar sesión en cualquier momento.

Precondiciones.

- Tener acceso al sistema
- Estar logueado en el sistema.

Post condiciones.

- Ninguna.

4.3. Diseño Lógico del Sistema.

El objetivo de la fase de diseño de un proyecto de estas características es obtener los componentes definidos en la etapa de análisis. Básicamente, se debe tener en cuenta que en esta etapa se inicia el proceso de selección de los servicios y su distribución física, y se analiza cómo serán ejecutados estos servicios en una arquitectura hardware. Además, se deberán determinar las especificaciones de desarrollo e integración, así como definir el entorno de pruebas y seleccionar qué criterios se utilizarán para que éstas sean representativas del correcto funcionamiento del sistema.

Se pueden resumir las tareas en esta fase como:

- Definición de la arquitectura del sistema: identificación de los componentes hardware, su interconexión, jerarquía software, seguridad y privilegios. Es decir, todo lo que es necesario para que el sistema pueda ser configurado y puesto en marcha con garantías.
- Diseño y estructuración de la base de datos.

4.3.1. Arquitectura.

La arquitectura del sistema es el primer paso para identificar los elementos hardware y dónde se ejecutarán los servicios. El objetivo es disponer de un conjunto de documentos y diagramas completos (que contengan todo el nivel de detalle necesario y suficiente), que sean comprensibles para personal no técnico, como por ejemplo la dirección de la empresa, y a la vez, que puedan ser utilizados como base para profundizar en el diseño del sistema.

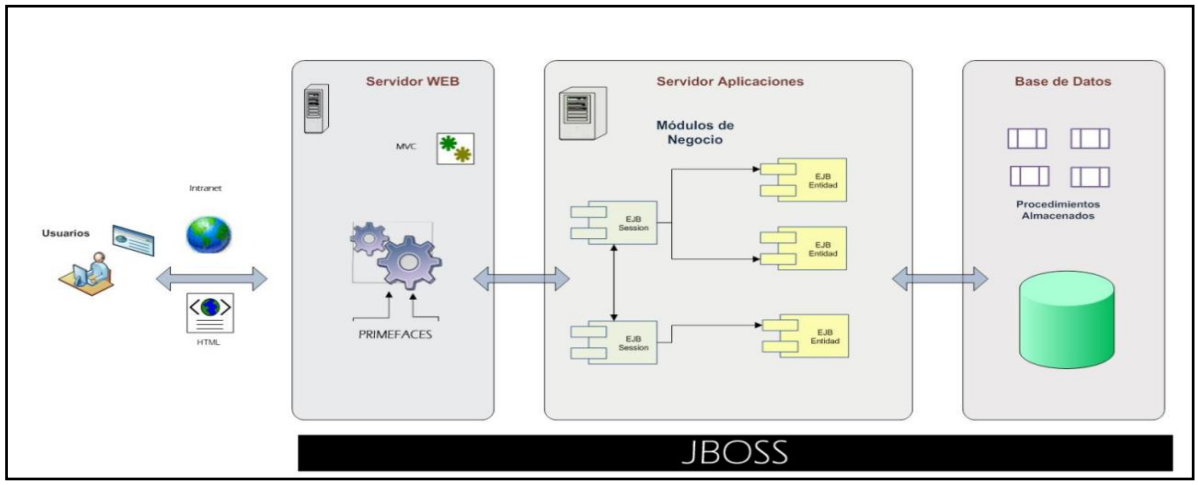

Ilustración 30. Arquitectura del Sistema.

4.3.2. Modelo Físico de Datos.

La base de datos está integrada con el sistema de licitación y subasta ganadera y comparte tablas con dicho sistema, además de manejar tablas que forman parte del área Financiera.

Diagrama Entidad – Relación.

Debido a la magnitud de la aplicación se ha dividido por módulos para mejorar el entendimiento.

Cuentas por Cobrar.

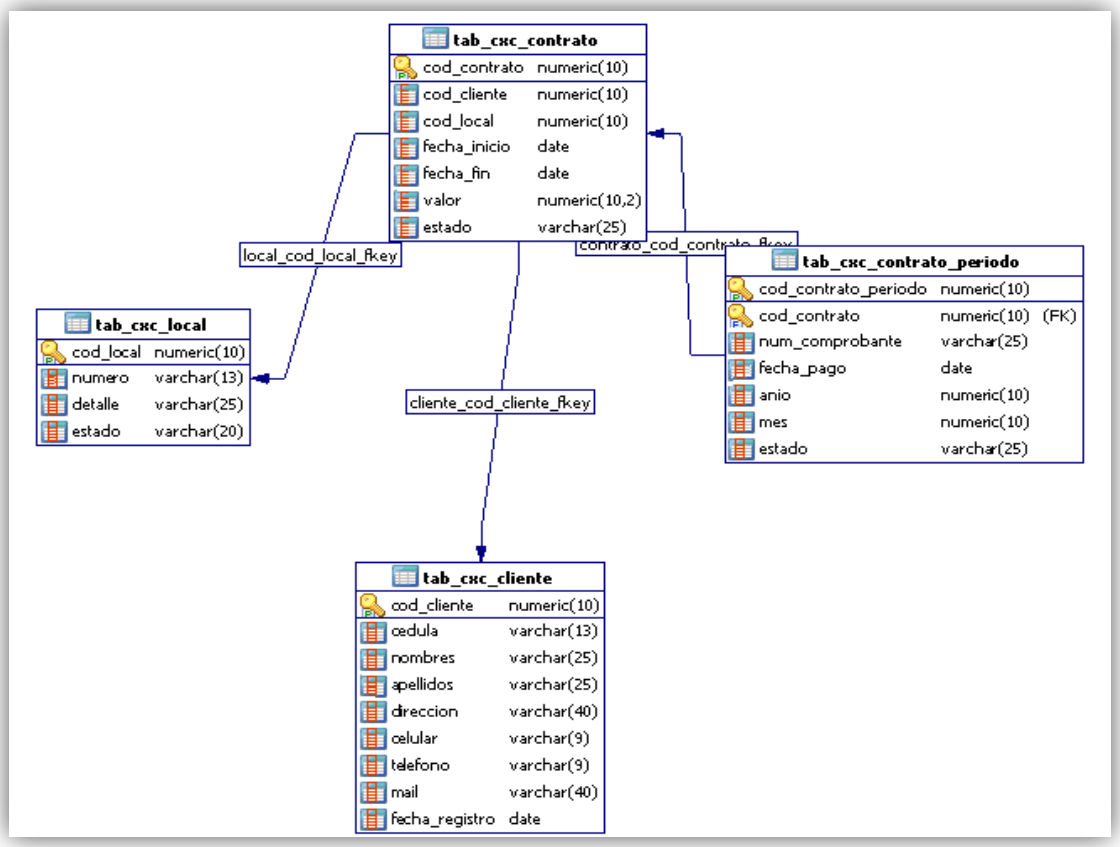

Ilustración 31. Entidad-Relación (Cuentas por Cobrar).

Licitación y Subasta Ganadera.

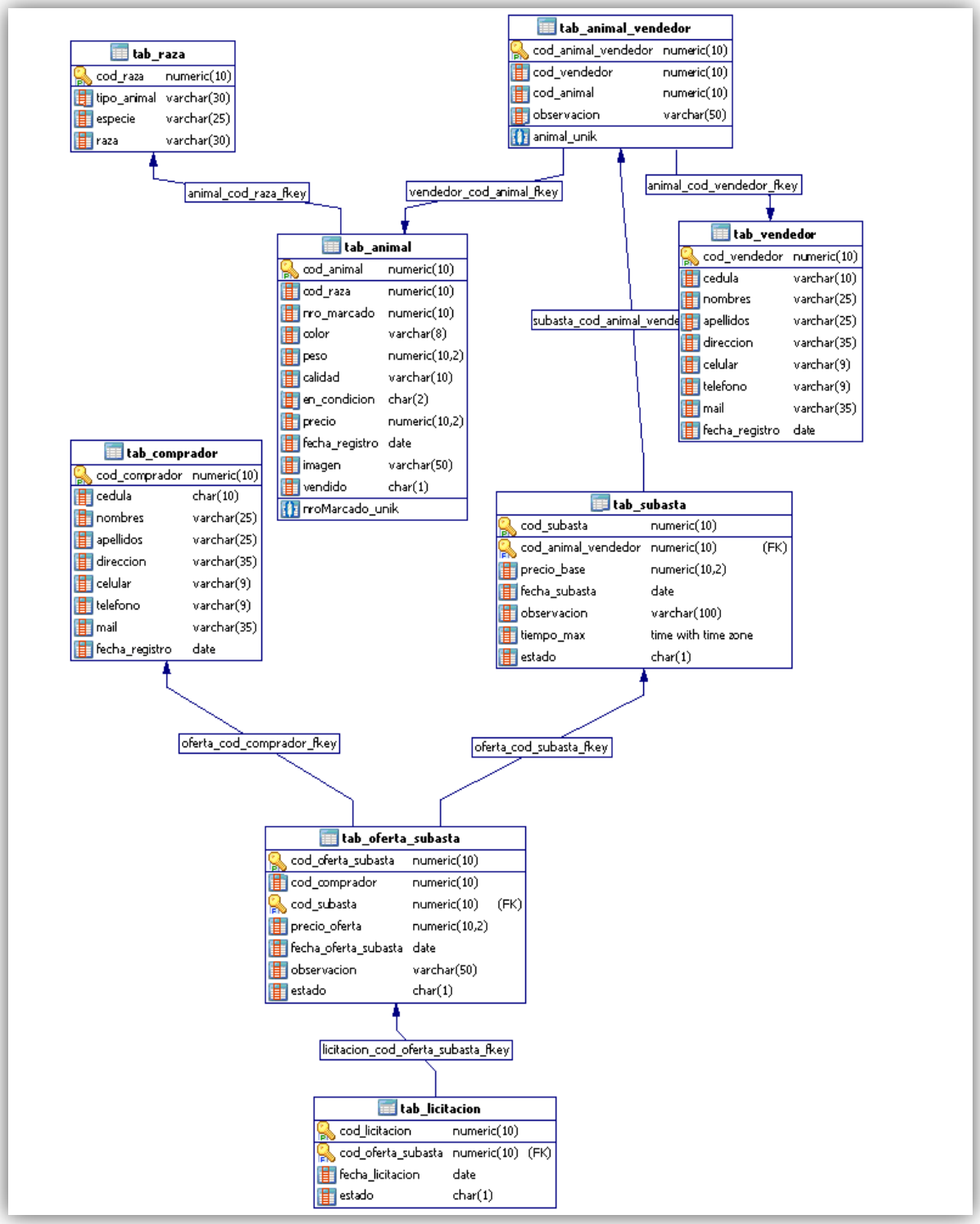

Ilustración 32. Entidad-Relación (Subasta Ganadera).

Contabilidad y Seguridad.

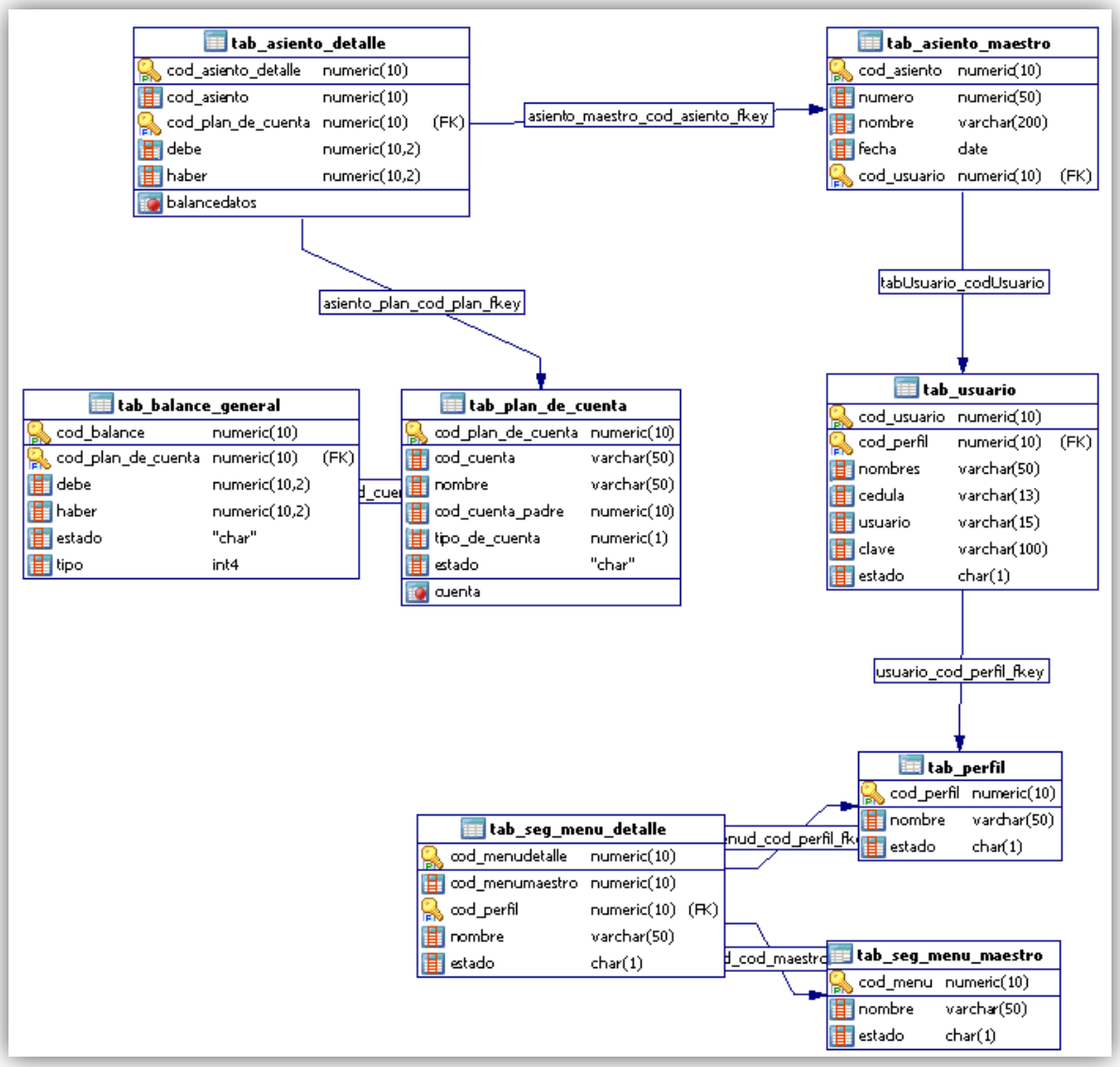

Ilustración 33. Entidad-Relación (Contabilidad).

Inventarios y Cuentas por Pagar.

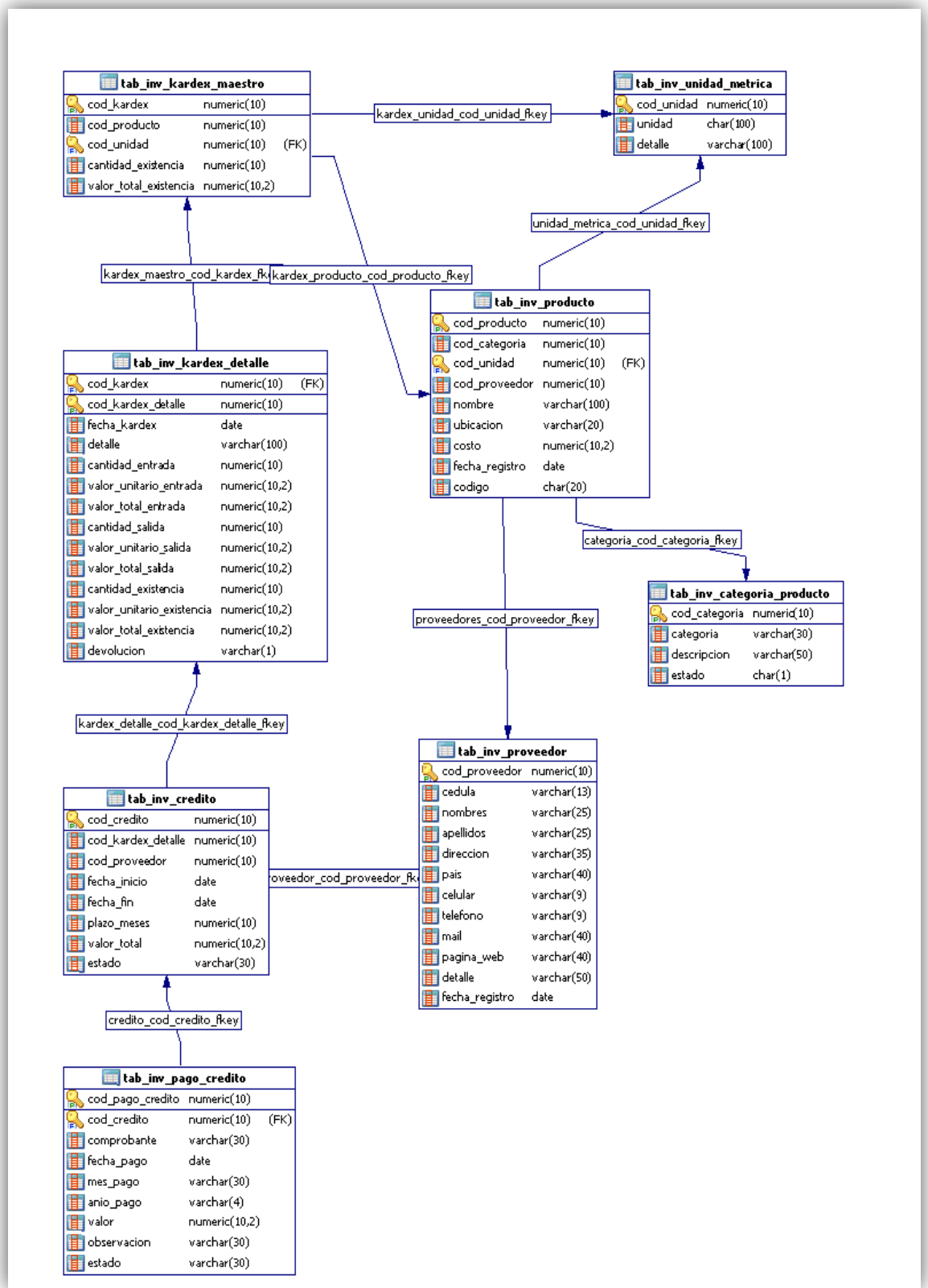

Ilustración 34. Entidad-Relación (Inventarios y Cuentas por Pagar).

Diccionario de Datos.

A continuación, se presenta el diccionario de datos para el sistema financiero.

TAB_SEG_MENU_MAESTRO		En esta tabla almacenamos los nombres de cada módulo como menú maestro.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_menu	numeric(10)	Registra un número que se irá incrementando mediante una secuencia	-
nombre	varchar(50)	Registra el nombre del menú.	-
estado	char(1)	Almacena el estado del menú si está activo o no	-

TAB_SEG_MENU_DETALLE		En esta tabla almacenamos los menús que tendrá acceso cada uno de los perfiles.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_menudetalle	numeric(10)	Registra un número que se irá incrementando mediante una secuencia	-
cod_menumaestro	numeric(10)	Código del menú maestro.	tab_seg_menu_maestro
cod_perfil	numeric(10)	Código del Perfil de Usuario.	tab_perfil
nombre	varchar(50)	Registra el nombre del menú.	-
estado	char(1)	Almacena el estado del menú si está activo o no	-

TAB_PERFIL		En esta tabla almacenamos los perfiles que se le asignará a cada uno de los usuarios.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_perfil	numeric(10)	Registra un número que se irá incrementando mediante una secuencia	-

nombre	varchar(50)	Registra el nombre del perfil.	-
estado	char(1)	Almacena el estado del perfil si está activo o no	-

TAB_USUARIO		En esta tabla almacenamos todos los usuarios de la aplicación, nadie podrá acceder a la aplicación sino tiene cuenta.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_usuario	numeric(10)	Registra un número que se irá incrementando mediante una secuencia	-
cod_perfil	numeric(10)	Registra el código del perfil al cual pertenece el usuario.	tab_perfil
nombres	varchar(50)	Registra los nombres de cada uno de los usuarios.	-
cedula	varchar(13)	La cédula o RUC del usuario	-
usuario	varchar(15)	El nombre de usuario con el cual accede a la aplicación	-
password	varchar(100)	Clave del usuario encriptada en la aplicación	-
estado	char(1)	Almacena el estado del usuario si está activo o no	-

Módulo de Contabilidad.

TAB_PLAN_DE_CUENTA		En esta tabla almacenamos todas las cuentas contables que tiene la empresa para llevar la contabilidad.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_plan_de_cuenta	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
cod_cuenta	varchar(50)	Registra el código de la cuenta que se utilizará en el	-

		sistema financiero.	
nombre	varchar(50)	Se almacena el nombre de la cuenta contable.	-
cod_cuenta_padre	numeric(10)	Se almacena el código de la cuenta contable del nodo raíz.	-
tipo_de_cuenta	numeric(1)	Cada cuenta podrá ser de tipo contenedora 0, o de tipo detalle 1.	-
estado	char	Almacena el estado de la cuenta si está activo o no	-

TAB_ASIENTO_MAESTRO		En esta tabla almacenamos el encabezado de cada asiento contable.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_asiento	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
numero	numeric(50)	Registra el número del asiento contable.	-
nombre	varchar(200)	Se almacena el detalle del asiento contable, el documento físico que debe existir para cada transacción.	-
Fecha	date	Almacena la fecha actual del registro del asiento contable.	-

TAB_ASIENTO_DETALLE		En esta tabla almacenamos el detalle de cada uno de los asientos contables. Es importante señalar que la suma total del debe será igual a la suma del haber.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_asiento_detalle	numeric(10)	Registra un número que se irá incrementando	-

		mediante una secuencia.	
cod_asiento	numeric(10)	Registra el código de la tabla maestro.	tab_asiento_maestro
cod_plan_de_cuenta	numeric(10)	Registra el código de la cuenta a cuál se le realiza la transacción.	tab_plan_de_cuenta
debe	numeric(10,2)	Registra los valores correspondientes al debe.	-
haber	numeric(10,2)	Registra los valores correspondientes al haber.	-

TAB_BALANCE_GENERAL		En esta tabla almacenamos los totales de cada una de las cuentas.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_balance	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
cod_plan_de_cuenta	numeric(10)	Registra el código de la cuenta a cuál se le realiza la transacción.	tab_plan_de_cuenta
debe	numeric(10,2)	Registra los valores correspondientes al total del debe mediante un trigger de actualización.	-
haber	numeric(10,2)	Registra los valores correspondientes al haber mediante un trigger de actualización.	-
estado	char	Almacena el estado de la cuenta si esta activo o no	-
tipo	Int4	Nos muestra si la cuenta es de tipo contenedora o detalle	-

Módulo de Inventarios.

TAB_INV_UNIDAD_METRICA		En esta tabla almacenamos todas las unidades métricas de cada uno de nuestros productos.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_unidad	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
unidad	char(100)	Registra el nombre de la unidad de medida.	-
detalle	varchar(100)	Registra el detalle de la unidad métrica.	-

TAB_INV_CATEGORIA_PRODUCTO		En esta tabla almacenamos todas las categorías de los productos como por ejemplo equipos de oficina, etc.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_categoria	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
categoria	char(100)	Registra el nombre de la categoría.	-
detalle	varchar(100)	Registra el detalle de la categoría del producto.	-
estado	char	Almacena el estado de la categoría del producto si está activa o no	-

TAB_INV_PROVEEDOR		En esta tabla almacenamos todos los datos de los proveedores de productos y equipos de oficina.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_proveedor	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
nombres	varchar(50)	Registra los nombres de cada uno de los proveedores.	-
apellidos	varchar(50)	Registra los apellidos de cada uno de los proveedores.	-
cedula	varchar(13)	La cédula o RUC del proveedor	-
direccion	varchar(35)	La dirección completa del proveedor.	-
celular	varchar(9)	Número del Celular del proveedor.	-
telefono	varchar(9)	Número del teléfono del proveedor o número de la oficina.	-
mail	varchar(40)	Correo personal o de la empresa proveedora de productos.	-
pagina_web	varchar(40)	Dirección web del proveedor	-
detalle	varchar(50)	Se registrará algún dato adicional del proveedor en caso de que exista.	
estado	char(1)	Almacena el estado del usuario si está activo o no	-

TAB_INV_PRODUCTO		En esta tabla almacenamos todos los datos de los productos que estarán en la bodega.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_producto	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
cod_unidad	numeric(10)	Almacena el código de la unidad métrica del producto.	tab_inv_unidad_metrica
cod_categoria	numeric(10)	Almacena el código de la categoría del producto previamente creada en la anterior tabla.	tab_inv_categoria_producto
cod_proveedor	Numeric(10)	Almacena el código del proveedor como referencia.	tab_inv_proveedor
nombre	varchar(100)	Registra el nombre del producto.	-
ubicacion	varchar(20)	Almacena el lugar físico donde estará el producto	-
costo	numeric(10,2)	Almacena el costo del producto, se actualizará automáticamente cuando se ingrese adquisiciones.	-
fecha_registro	date	Fecha de registro del producto.	-
codigo	varchar(20)	Código de señalización del producto dentro de la bodega.	-

TAB_INV_KARDEX_MAESTRO		En esta tabla almacenamos el encabezado del kardex del producto.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_kardex	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
cod_producto	numeric(10)	Almacena el código del producto.	tab_inv_producto
cantidad_existencia	numeric(10)	Almacena el valor real de existencias físicas en la bodega de cada producto.	-
valor_total_existencia	numeric(10,2)	Almacena el valor total en dinero mediante el método promedio.	-

TAB_INV_KARDEX_DETALLE		En esta tabla se almacenan todas las entradas y salidas de los productos, a la vez actualiza la cantidad total, el costo de cada uno de los productos.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_kardex_detalle	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
cod_kardex	numeric(10)	Almacena el código delkardex maestro o encabezado del kardex.	tab_inv_kardex_maestro
fecha_kardex	date	Almacena la fecha del	-

		registro de la transacción del producto.	
detalle	varchar(100)	Almacena el detalle o el número de documento físico de cada transacción.	-
cantidad_entrada	numeric(10)	Almacena el número de unidades del producto que ingresan a bodega cuando se realiza una compra, en caso de ser entrega permanecerá con el valor 0.	-
Valor_unitario_entrada	numeric (10,2)	Registra el costo de compra por unidad del producto.	-
Valor_total_entrada	Numeric(10,2)	Registra el cálculo de de la multiplicación de cantidad de entrada por el costo unitario por producto.	-
cantidad_salida	numeric(10)	Almacena el número de unidades del producto que salen de bodega hacia el personal de la empresa.	-
Valor_unitario_salida	numeric (10,2)	Registra el costo de salida del producto.	-
Valor_total_salida	Numeric (10,2)	Registra el cálculo de de la multiplicación de cantidad de salida por el costo unitario de salida del producto.	-

cantidad_existencia	numeric(10)	Almacena el número de unidades del producto que existen en bodega.	-
Valor_unitario_existencia	numeric (10,2)	Registra el costo unitario total del producto.	-
Valor_total_existencia	Numeric (10,2)	Registra el valor total en existencias de productos.	-
devolucion	varchar(1)	En este campo almacenamos una bandera que nos indica si la transacción ha sido devuelta o anulada.	-

Módulo Cuentas por Pagar.

TAB_INV_CREDITO		En esta tabla almacenamos todos los créditos que la empresa realiza a los proveedores de productos.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_credito	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
cod_kardex_detalle	numeric(10)	Almacena el código del kardex detallecon el cual se realizó el crédito.	tab_inv_kardex_detalle
cod_proveedor	numeric(10)	Almacena el código del proveedor, quien nos emite el crédito.	-
fecha_inicio	date	Almacena la fecha de inicio del crédito.	-
fecha_fin	date	Almacena la fecha final	-

		del crédito.	
plazo_meses	numeric(10)	Almacena el número de meses del crédito.	-
valor_total	numeric(10, 2)	Almacena el valor total del crédito.	-
estado	varchar(30)	Registra el estado del crédito, puede ser pagado, anulado o activo.	-

TAB_INV_PAGO_CREDITO		En esta tabla se almacena todos los pagos realizados por cada uno de los créditos realizados a los proveedores.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_pago_credito	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
cod_credito	numeric(10)	Registra el código del crédito al cuál se realizará el pago.	tab_inv_credito
comprobante	varchar(30)	Almacena el número del comprobante de pago emitido por el proveedor	-
mes_pago	varchar(30)	Almacena el nombre del mes que se paga del crédito.	-
anio_pago	varchar(30)	Almacena el año que se paga del crédito.	-
valor	numeric (10,2)	Almacena el valor cancelado del crédito.	-
observacion	varchar(30)	En caso de existir alguna observación se almacenará en este campo	-
estado	varchar(30)	Registra el estado del pago.	-

Modulo Cuentas x Cobrar.

TAB_CXC_CLIENTE		En esta tabla se almacena la información de los clientes o arrendatarios de los locales comerciales.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_cliente	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
nombres	varchar(25)	Registra los nombres del cliente.	-
apellidos	varchar(25)	Registra los apellidos del cliente	-
direccion	varchar(40)	Almacena la dirección del domicilio del cliente.	-
celular	varchar(9)	Almacena el número del celular del cliente.	-
telefono	varchar(9)	Almacena el número del teléfono del cliente.	-
mail	varchar(40)	En caso de que el cliente tenga correo electrónico se registrará en este campo	-
fecha_registro	date	Almacena la fecha de Ingreso del cliente a la base.	-

TAB_CXC_LOCAL		En esta tabla se almacena la información de los locales comerciales que existen en la feria.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_cliente	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
numero	varchar(13)	Registra el número asignado a cada local.	-
detalle	varchar(25)	Almacena el detalle del	-

		local, es decir si es local comercial o de comida.	
estado	varchar(20)	Almacena el estado del local que puede ser arrendado o listo para arrendar.	-

TAB_CXC_CONTRATO		En esta tabla se registran los datos del contrato que suscribe la empresa con cada uno de los clientes para arrendar los diferentes locales comerciales.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_contrato	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	
cod_cliente	numeric(10)	Registra el código del cliente que va a suscribir el contrato.	tab_cxc_cliente
cod_local	numeric(10)	Registra el código del local que será arrendado por el cliente.	tab_cxc_local
fecha_inicio	date	Registra la fecha inicial de suscripción del contrato.	-
fecha_fin	date	Registra la fecha final de suscripción del contrato	-
valor	numeric (10,2)	Es el valor que se establece por concepto de arrendamiento del local.	-
estado	varchar(20)	Almacena el estado del contrato, puede ser en curso, terminado, anulado.	-

TAB_CXC_CONTRATO_PERIODO		En esta tabla se registran los datos de cancelación mensual de cada local por parte de los clientes.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**

cod_contrato_periodo	numeric(10)	Registra un número que se irá incrementando mediante una secuencia.	-
cod_contrato	numeric(10)	Registra el código del contrato entre el cliente y la empresa.	tab_cxc_contrato
num_comprobante	varchar(25)	Registra el número del comprobante de pago emitido el momento de pago por parte del cliente.	-
fecha_pago	date	Registra la fecha en que se realiza el pago.	-
anio	numeric (10)	Registra el año del período de pago.	-
mes	numeric (10)	Registra el mes del período de pago.	-
estado	varchar(25)	Almacena el estado del periodo, puede ser pagado, sin pagar y anulado.	-

4.3.3. Diagrama Global de Paquetes.

Contabilidad.

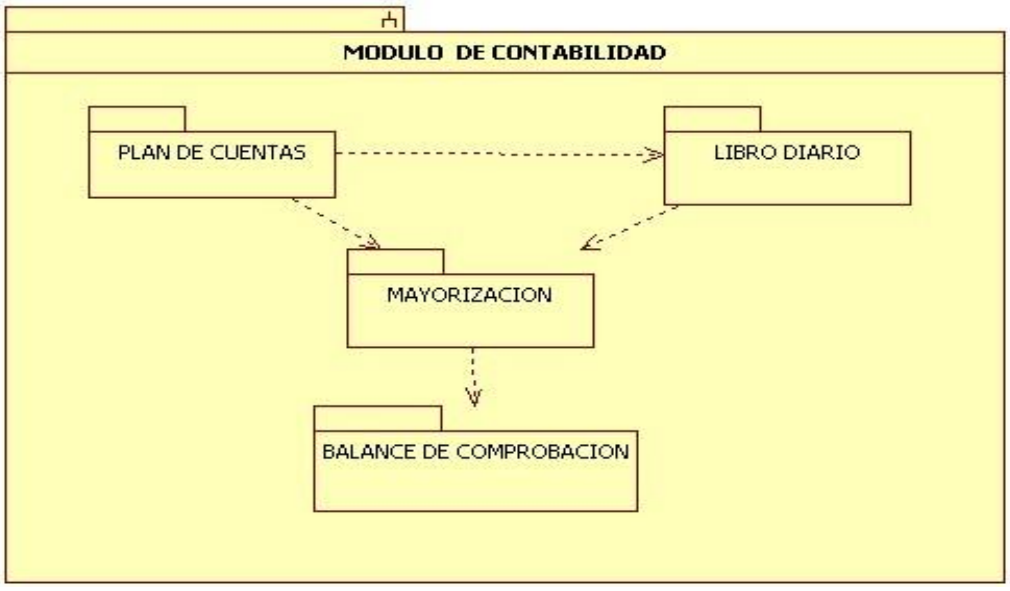

Ilustración 35. Diagrama de Paquetes: Contabilidad.

Inventarios.

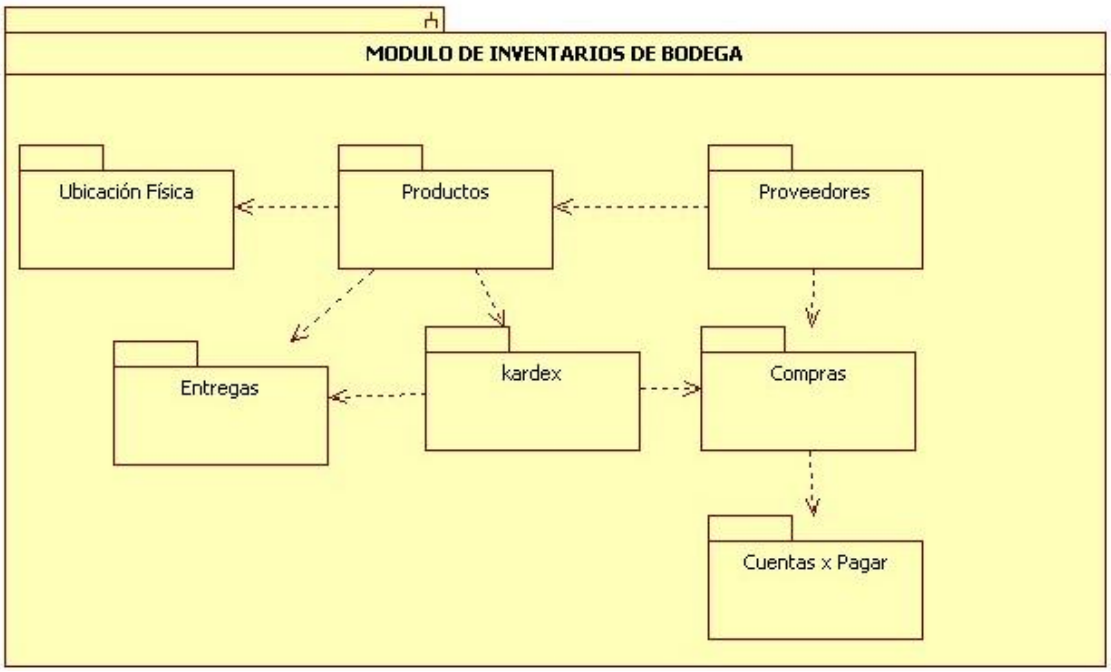

Ilustración 36. Diagrama de Paquetes: Inventarios.

Ventas.

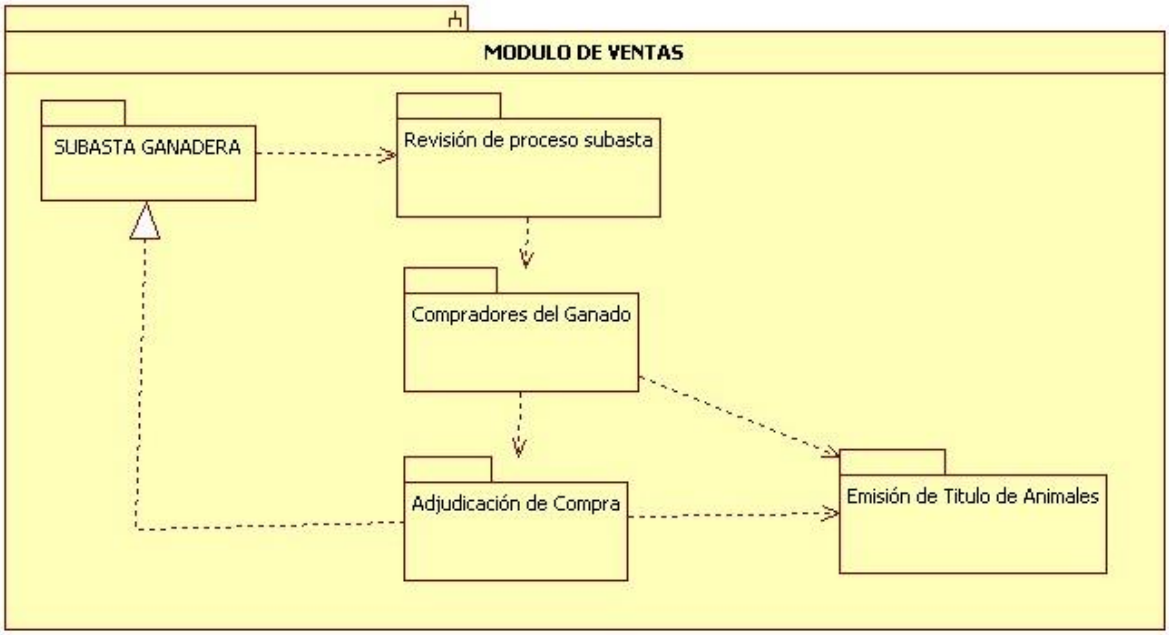

Ilustración 37. Diagrama de Paquetes: Ventas.

Cuentas por Cobrar.

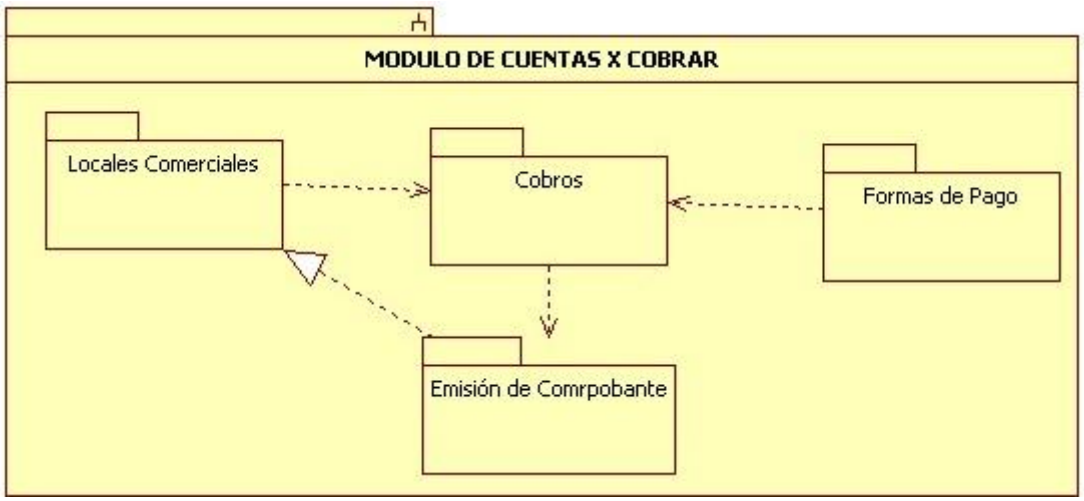

Ilustración 38. Diagrama de Paquetes: Cuentas por Cobrar.

4.4. Desarrollo del Sistema.

4.4.1. Planificación de las Actividades de Desarrollo.

Al momento de realizar la planificación se ha tenido en cuenta los siguientes puntos:

Análisis del Panorama. Donde realizamos la descripción general del proyecto, detalle de la organización del plan y resumen del resto de la documentación.

Análisis de Fases. Se analiza el ciclo de desarrollo del proyecto como es: análisis de requisitos, fase de diseño de alto nivel, fase de diseño de bajo nivel, etc. Asociada con cada fase debe de haber una fecha que especifique cuando se debe terminar estas fases y una indicación de cómo se pueden solapar las distintas fases del proyecto.

Análisis de organización. Se definen las responsabilidades específicas de los grupos que intervienen en el proyecto.

Análisis de pruebas. Se hace un esbozo general de las pruebas y de las herramientas, procedimientos y responsabilidades para realizar las pruebas del sistema.

Análisis de control de modificaciones. Se establece un mecanismo para aplicar las modificaciones que se requieran a medida que se desarrolle el sistema.

Análisis de documentación. Su función es definir y controlar la documentación asociada con el proyecto.

Análisis de capacitación. Se describe la preparación por parte del desarrollador que participa en el proyecto y las instrucciones a los usuarios para la utilización del sistema que se les entregue.

Análisis de revisión e informes. Se analiza cómo se informa del estado del proyecto y se definen las revisiones formales asociadas con el avance de proyecto.

Análisis de instalación y operación. Se describe el procedimiento para instalar el sistema en la localidad de la empresa de rastro.

Análisis de mantenimiento. Se establece un bosquejo de los posibles tipos de mantenimiento que se tienen que dar para futuras versiones del sistema.

4.4.2. Desarrollo y codificación.

4.4.2.1. Diseño de la Interfaz de Usuario.

Para diseñar una interfaz de usuario para un sistema Web, es indispensable y necesario analizar en los enlaces que debe tener la aplicación. Se debe pensar primeramente en las características principales que tendrá el sistema y luego a partir de esto se debe ir construyendo tantos enlaces a otras páginas Web como sea necesario.

Sin embargo al ser una aplicación J2EE se facilita el proceso de diseño de la aplicación ya que existen librerías que ayudan para una mejor presentación.

A continuación, se presenta una estructura estandarizada para toda la aplicación:

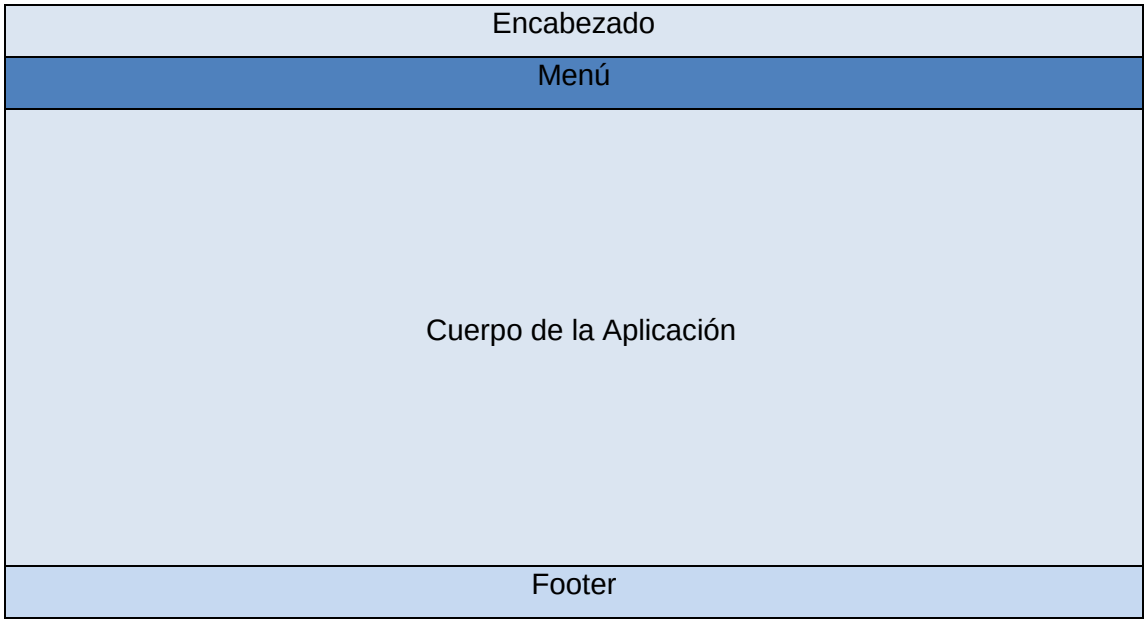

Ilustración 39. Estructura de la presentación Web.

- ***Encabezado*:** En esta parte de la estructura de la aplicación va el logo del sistema, junto con el nombre del sistema.

- ***Menú*:** En esta ubicación se pondrá los menús de nuestra aplicación, los cuales nos permitirán navegar por la aplicación de una manera rápida y eficiente.

- ***Cuerpo de la Aplicación*:** Es esta parte de la estructura se realizarán todas las operaciones de la aplicación, es decir: Crear, modificar, imprimir, eliminar entre otras.

- ***Footer*:** En este lugar se desplegará los derechos de Autor como también la licencia del sistema

4.4.2.2. Detalle de la Arquitectura de la Aplicación.

De acuerdo a las capas de la Arquitectura Orientada a Servicios mi aplicación consta de 3 capas:

Capa de Presentación.

Esta capa es aquella que se muestra al usuario final, además es importante señalar que los archivos de esta capa se clasifican por cada uno de los submódulos:

Capa de Datos.

Esta capa está organizada de la siguiente forma:

- Persistencia.
- DTO.- Son las entidades generadas para la interacción de la base de datos con la aplicación.
- DAO.- Aquí encontramos las interfaces para realizar la conexión entre la capa de Negocio y la Implementación de los Servicios.
- DAOImpl.- Se realiza la implementación de cada una de las funciones especificadas en el DAO.

Persistencia.

La Unidad de Persistencia consiste en la declaración de entidades o a su vez se especifica el origen de conexión a una base de datos relacional. Es definida por el archivo persistence.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd">
        <persistence-unit name="finansoftEJB">
        <jta-data-source>java:/conexionDS</jta-data-source>
        </persistence-unit>
</persistence>
```

Los elementos más importantes del archivo son los siguientes:

persistence-unit name: Especifica un nombre para el contexto o persistente. Si únicamente se especifica uno, no habrá; que incluir su nombre cuando se recupere el EntityManager (con la anotación @PersistenceContext o @PersistenceUnit).

jta-data-source: El servidor se extiende a la aplicación que le permite hacer referencia a las fuentes de datos. En este caso debemos crear un archivo en el servidor JBOSS con

los archivos de conexión. A continuación el archivo Conexión-DS.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE datasources
    PUBLIC "-//JBoss//DTD JBOSS JCA Config 1.5//EN"
    "http://www.jboss.org/j2ee/dtd/jboss-ds_1_5.dtd">
<datasources>
<local-tx-datasource>
<jndi-name>ConexionDS</jndi-name>
<use-java-context>true</use-java-context>
<connection-url>jdbc:postgresql://localhost:5432/bdd</connection-url>
<driver-class>org.postgresql.Driver</driver-class>
<user-name>postgres</user-name>
<password>root</password>
</local-tx-datasource>
</datasources>
```

Un nombre JNDI es un nombre que la gente debe usar para un objeto. Estos nombres están vinculados a su objeto por el nombramiento y servicio de directorio que es proporcionada por el servidor J2EE. Dado que los componentes J2EE deben acceder a este servicio a través de la API JNDI, por lo general se refieren a una de los objetos como su nombre JNDI.

El nombre JNDI de la base de datos Cloudscape es jdbc/Cloudscape . Cuando se inicia, el servidor J2EE lee la información de un archivo de configuración y agrega automáticamente los nombres JNDI base de datos como jdbc/Cloudscape al espacio de nombres.

Especificación:

Persistence.xml	conexionDS.xml
<jta-data-source> java:/**ConexionDS** </jta-data-source>	<jndi-name> **ConexionDS** </jndi-name>
Es obligatorio que lleven el mismo nombre caso contrario el proceso de mapeo fallara y no permitirá la conexión a la base.	

Una vez terminado la configuración de la unidad de persistencia en la capa de datos DTO, se puede apreciar el uso de JPA (Java Persistence Api), tal y como se explica en el Capítulo III.

Entidades DTO.-

- Las entidades son generadas por cada tabla de la base de datos
- Cada instancia de la entidad corresponde a un campo de la base de datos.
- La parte principal de la entidad es la clase Entity, aunque se pueden usar clases auxiliares.
- La clase debe llevar la anotación javax.persistence.Entity.
- Tener un constructor por omisión público o protegido. Sin argumentos.
- La clase, los métodos o las variables de instancia persistentes no deben ser declarados tipo final.
- Si una instancia de un entity es pasada por valor como un objeto debe implementar la interfaz java.io.Serializable.
- Los entities pueden extender tanto clases entity como cualquier otra, incluso clases no entity pueden extender clases entity.
- Las variables persistentes deben ser declaradas private o protected y solamente pueden ser accedidas a través de los correspondientes métodos de acceso: getters o setters.
- Un entity debe declarar una llave primaria. Esto se hace marcando el campo con la anotación @Id. Al igual que en las bases de datos relacionales, la llave primaria hace del entity un objeto único. Cuando se requiere de una llave compuesta se puede usar una clase como llave primaria compuesta. Para este tipo de claves, se utilizan las anotaciones javax.persistence.EmbededId y javax.persistence.IdClass.
- Los entities no pueden ser accedidos directamente, deben ser desplegados y usados localmente en un contenedor (desde beans de sesión, o de mensajería, o incluso desde cualquier componente web). De cualquier manera, el código cliente debe primero recuperar una instancia particular del entity desde el contexto de persistencia o crear una y adicionarla a dicho contexto.

Ejemplo: Entidad Perfil.

```
package com.tesis.modelo.dto;
import java.io.Serializable;
import javax.persistence.*;
import java.util.Set;
@Entity
@Table(name="tab_perfil")
public class TabPerfil implements Serializable {
	private static final long serialVersionUID = 1L;
	@Id
	@SequenceGenerator(name="TAB_PERFIL_CODPERFIL_GENERATOR",sequenceName="SEQ_TAB
_PERFIL)
	@GeneratedValue(strategy=GenerationType.SEQUENCE,generator="TAB_PERFIL_CODPERFIL_GE
NERATOR")
	@Column(name="cod_perfil")
	private long codPerfil;
	private String estado;
	private String nombre;
	//bi-directional many-to-one association to TabUsuario
	@OneToMany(mappedBy="tabPerfil")
	private Set<TabUsuario> tabUsuarios;
	public TabPerfil() {
	}
	public long getCodPerfil() {
		return this.codPerfil;
	}
	public void setCodPerfil(long codPerfil) {
		this.codPerfil = codPerfil;
	}
	public String getEstado() {
		return this.estado;
	}
	public void setEstado(String estado) {
		this.estado = estado;
	}
	public String getNombre() {
		return this.nombre;
	}
	public void setNombre(String nombre) {
		this.nombre = nombre;
	}
	public Set<TabUsuario> getTabUsuarios() {
		return this.tabUsuarios;
	}
	public void setTabUsuarios(Set<TabUsuario> tabUsuarios) {
		this.tabUsuarios = tabUsuarios;
	}
}
```

DAO.-

Consiste en utilizar un objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos. El DAO maneja la conexión con la fuente de datos para obtener y almacenar datos.

DAO	EXPLICACIÓN

package com.tesis.modelo.dao;
import java.util.List;
import javax.ejb.Local;
import com.tesis.modelo.dto.TabPerfil;
@Local
public interface PerfilDAO {
List<TabPerfil> listarPerfil();
TabPerfil insertarPerfil(TabPerfil perfil);
TabPerfil editarPerfil(TabPerfil perfil);
}
	En la Interfaz se colecciona los métodos y propiedades para implementar en la clase.
	
	En esta estructura JAVA se específica que se debe hacer pero no se implementa los métodos.

DAOImpl	EXPLICACIÓN

//NOMBRE DEL PAQUETE
package com.tesis.modelo.dao.impl;
//LIBRERIAS NECESARIAS
import java.util.ArrayList;
import java.util.List;
import com.tesis.modelo.dao.PerfilDAO;
import com.tesis.modelo.dto.TabPerfil;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;
import org.apache.log4j.Logger;
 	
//IMPLEMENTACION DE LA CLASE
@Stateless
public class PerfilDAOImpl implements PerfilDAO
	Esta clase es el medio de interacción entre la Base de Datos y la aplicación.
	
	Además se implementa todos los métodos que han sido especificados en la interfaz, es decir, no puede ser creado en esta clase un método que no esté en la interfaz y viceversa.

```
 {
//UNIDAD DE PERSISTENCIA
@PersistenceContext(unitName="finansoftEJB")
//ENTIDAD
private EntityManager em;
   public PerfilDAOImpl() {
    }
//METODO LISTAR PERFILES
@SuppressWarnings("unchecked")
public List<TabPerfil> listarPerfil() {
List<TabPerfil> per=new ArrayList<TabPerfil>();
-----
-----
return per;
}
//METODO INGRESAR
@Override
public TabPerfil insertarPerfil(TabPerfil perfil) {
em.persist(perfil);
return perfil;
  }
//METODO EDITAR PERFIL
@Override
public TabPerfil editarPerfil(TabPerfil perfil) {
em.merge(perfil);
   return perfil;
}
}
```

Capa de Negocio.

Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.

A continuación, mediante un gráfico se explica cómo actúa la capa de negocio:

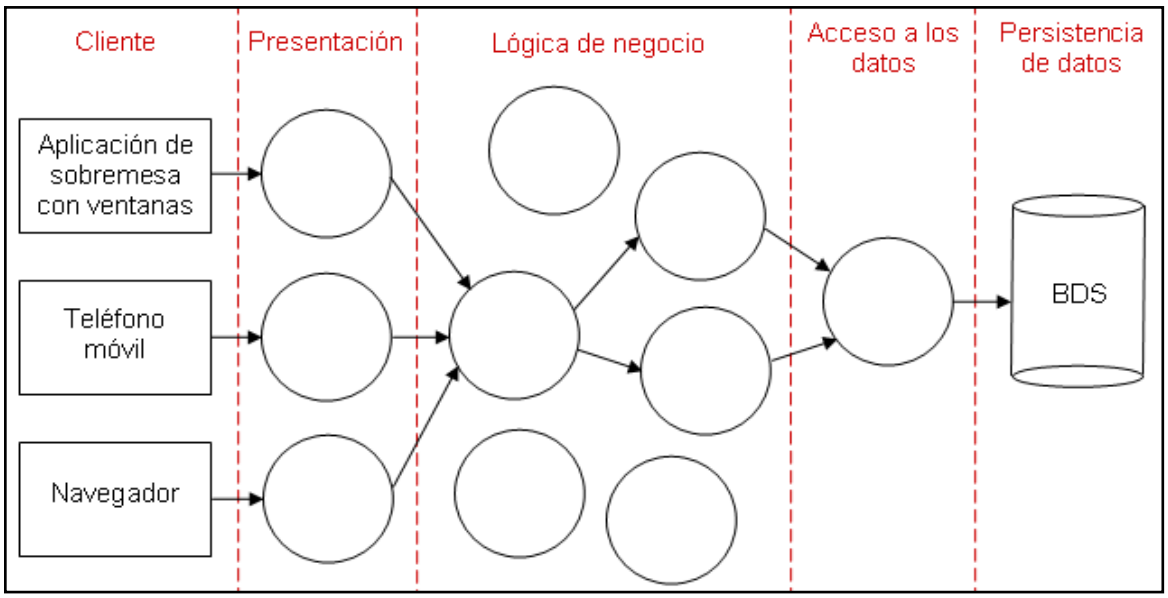

Ilustración 40. Análisis de la Capa de Negocio.

4.4.3. Factores de Calidad.

Para el desarrollo de nuestra aplicación nos basamos en los factores de calidad McCall, a continuación, se describen las iteraciones aplicadas en el proyecto.

Operación del Producto.

Corrección: La aplicación satisface las necesidades de la empresa de acuerdo a los objetivos anunciados en el inicio del proyecto.

Fiabilidad: Los módulos del sistema funcionan correctamente lo que permite cumplir con la exactitud requerida.

Eficiencia: Debido a la tecnología JAVA se ha desarrollado una aplicación eficiente a la hora de utilizar los recursos informáticos como también se optimiza el código al manejar framework claramente especificados y definidos.

Integridad: El módulo de seguridad permite mantener los datos de manera segura mediante encriptación como también el acceso a datos es muy seguro mediante la tecnología usada en el desarrollo.

Usabilidad (facilidad de manejo): La aplicación se desarrolló pensando en los usuarios finales de tal forma que el sistema es de fácil manejo.

Revisión del Producto.

Facilidad de mantenimiento: El proyecto se desarrolló bajo una estructura tecnológica lo que permite la facilidad para realizar mantenimientos e incluso agregar nuevos módulos del sistema.

Flexibilidad: El esfuerzo necesario para modificar un programa operativo. La flexibilidad para el cambio de procesos del sistema es adaptable en esta aplicación. La persona que está encargada deberá tener conocimientos previos de la tecnología utilizada.

Facilidad de prueba: Al ser una aplicación web se puede montar sobre cualquier máquina que cumpla con los requisitos de instalación del producto.

Transición del Producto.

Portabilidad: La aplicación es muy portable debido a la tecnología que se utiliza en el desarrollo del proyecto. Además java es independiente de la plataforma sobre la cual está desarrollándose la aplicación.

Reusabilidad (capacidad de reutilización): Todos los módulos se pueden volver a utilizar en cualquier otra aplicación que exista dentro de la empresa.

Interoperatividad: Es uno de los puntos más fuertes de la aplicación ya que se puede enlazar con otras tecnologías mediante el desarrollo de servicios web.

4.5. Implementación.

Una vez terminado el desarrollo de la aplicación se procedió a implementar en un servidor de aplicaciones de la empresa beneficiada.

El sistema consiste en una aplicación web modular, accesible a través de la Intranet de la Empresa para todos los usuarios legalmente autorizados. Además es importante señalar que se integra automáticamente con el sistema de Subasta Ganadera.

La aplicación financiera es de uso exclusivo para la Subasta Ganadera que se realiza en la feria de la ciudad de Ibarra. La aplicación ha sido personalizada de acuerdo a las necesidades de la empresa.

4.5.1. Diagrama de Secuencias.

***Diagrama de Secuencia*.-** Acceso al Sistema.

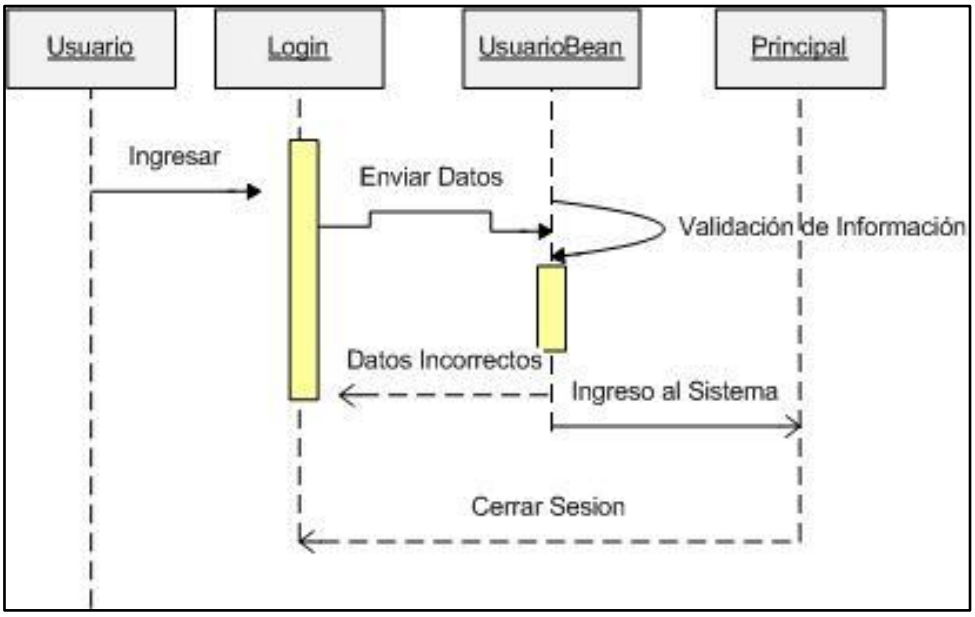

Ilustración 41. Diagrama de Secuencia.- Acceso al sistema.

Diagrama de Secuencia.- Compras a Proveedores.

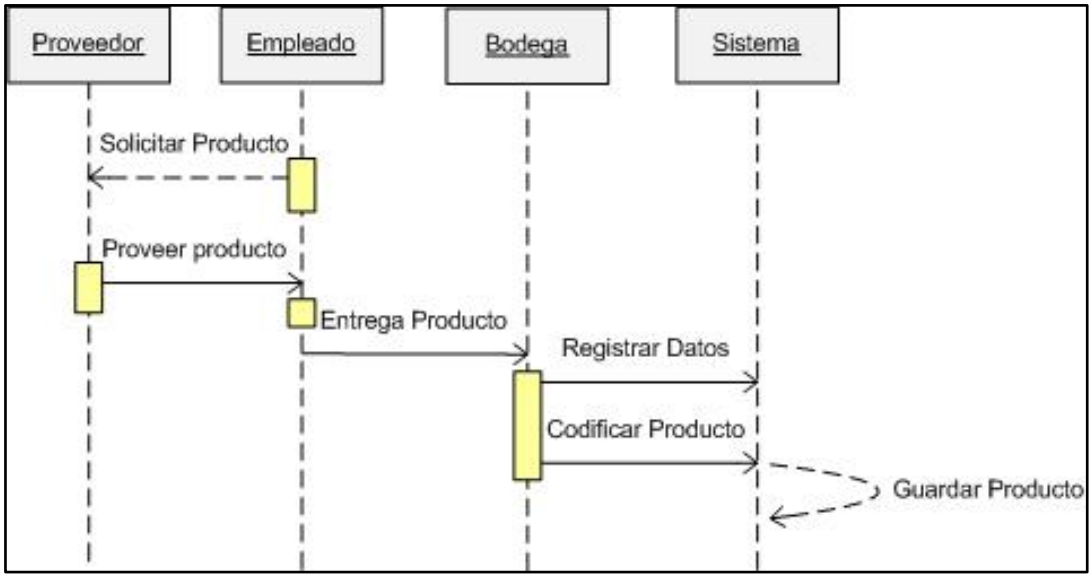

Ilustración 42. Diagrama de Secuencia.- Compras a Proveedores.

Diagrama de Secuencia.- Entrega de Productos de Bodega a empleados.

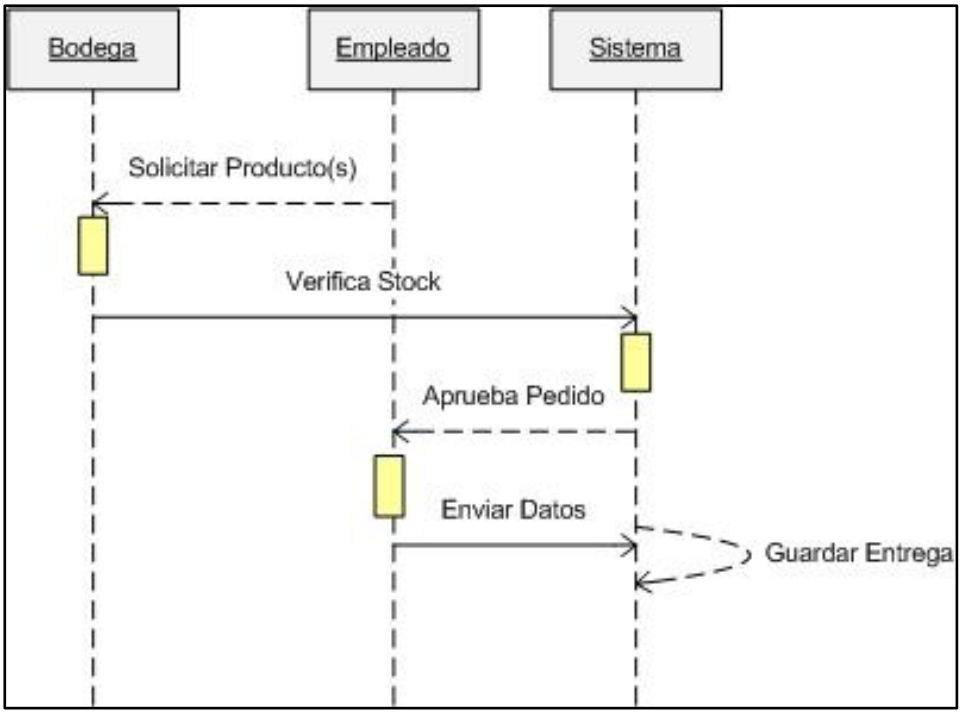

Ilustración 43. Diagrama de Secuencia.- Entrega de Productos de Bodega.

4.5.2. Diagrama de Actividades.

Diagrama de Actividad.- Acceso al Sistema.

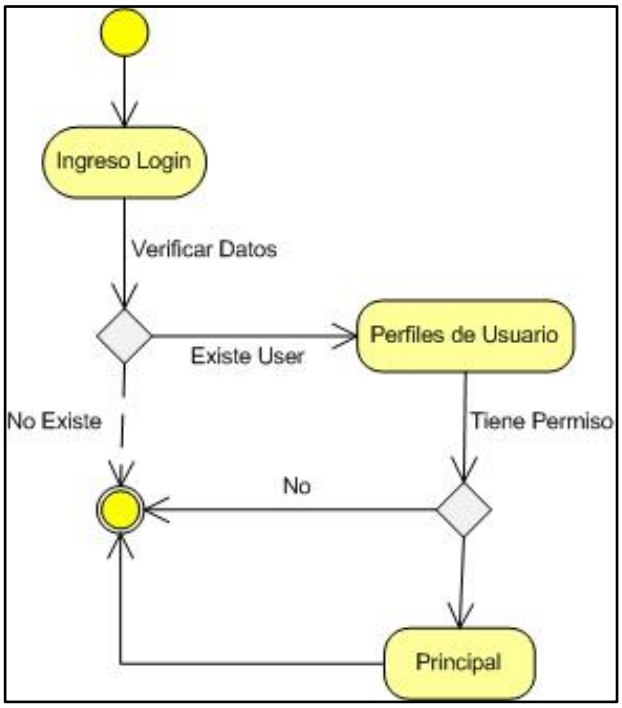

Ilustración 44. Diagrama de Actividad - Acceso al Sistema.

Diagrama de Actividad.- Ingreso Plan de Cuentas.

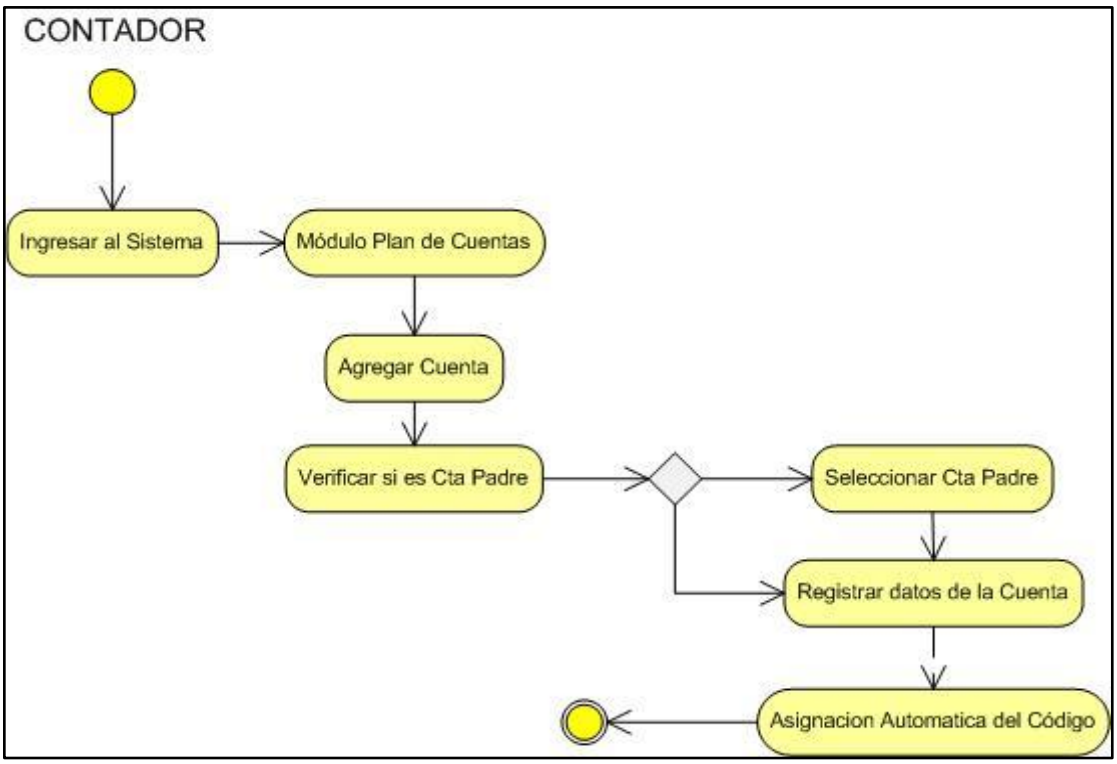

Ilustración 45. Diagrama de Actividades.- Ingreso de una Cuenta.

Diagrama de Actividad.- Pedidos de bodega.

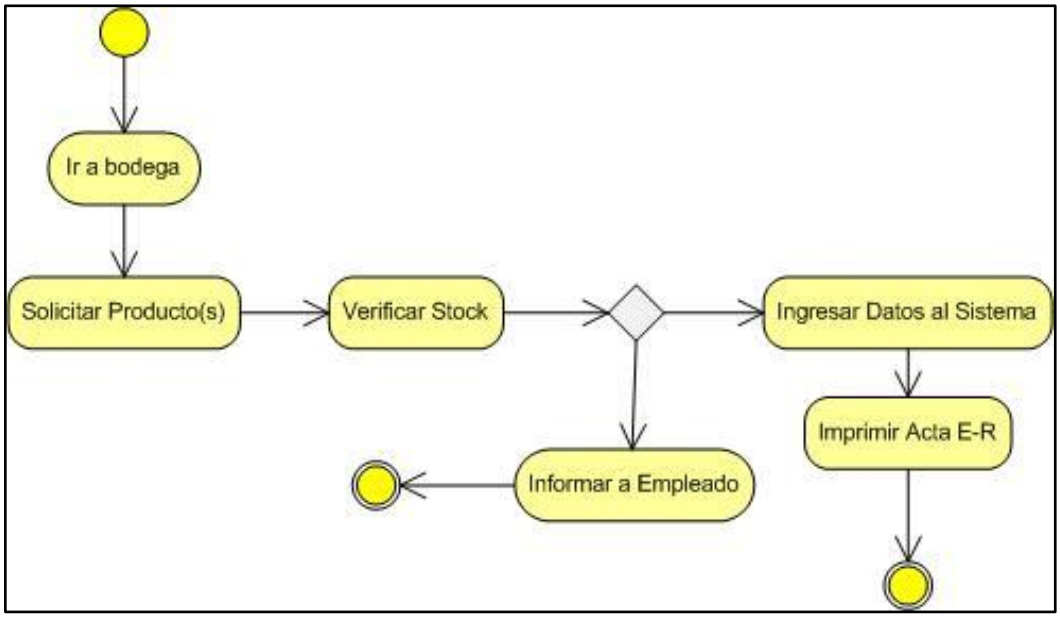

Ilustración 46. Diagrama de Actividades.- Pedidos Bodega.

Diagrama de Actividad.- Adjudicación de la Subasta.

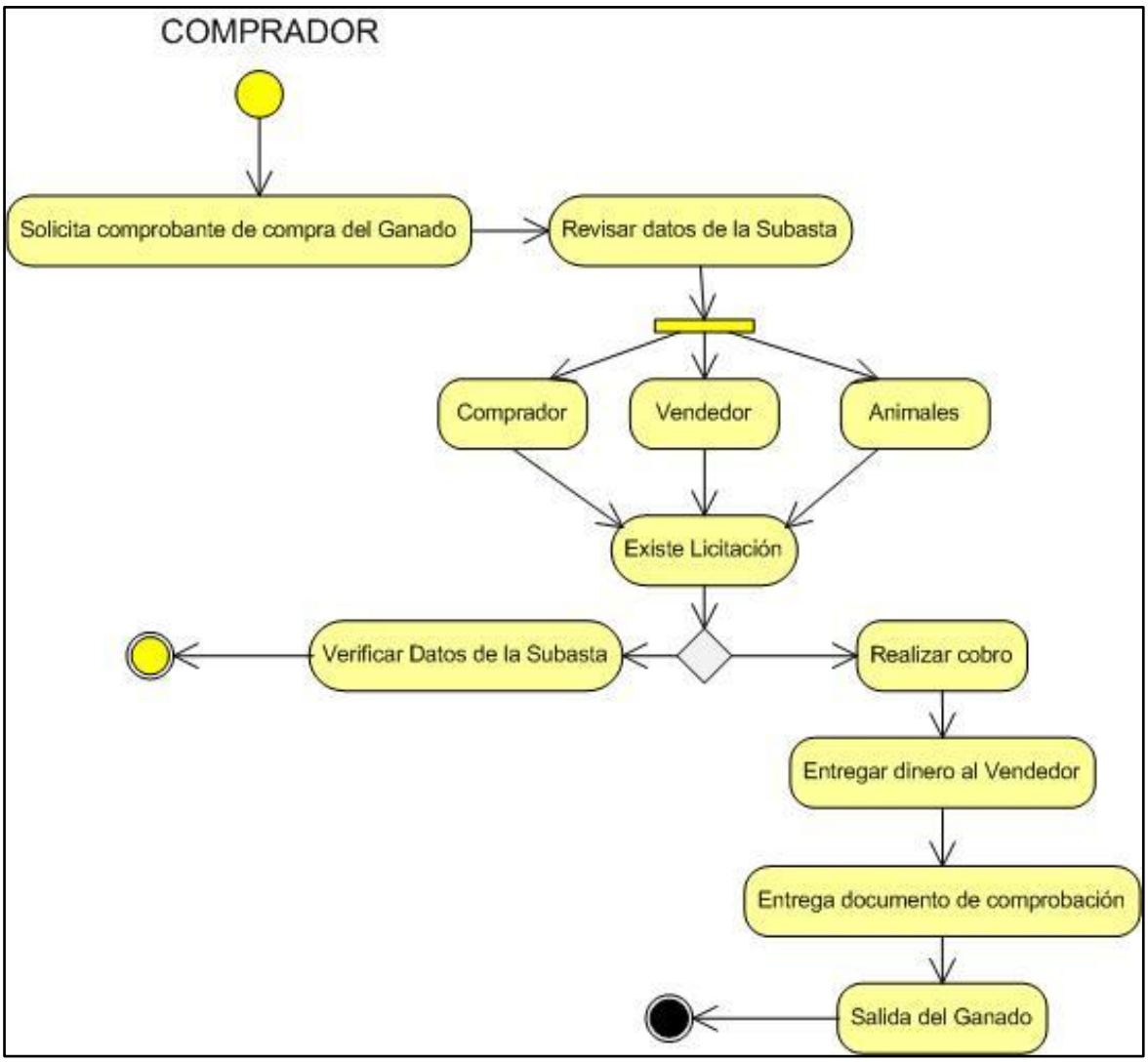

Ilustración 47. Diagrama de Actividad.- Adjudicación de la Subasta.

Diagrama de Actividad.- Cobros a clientes.

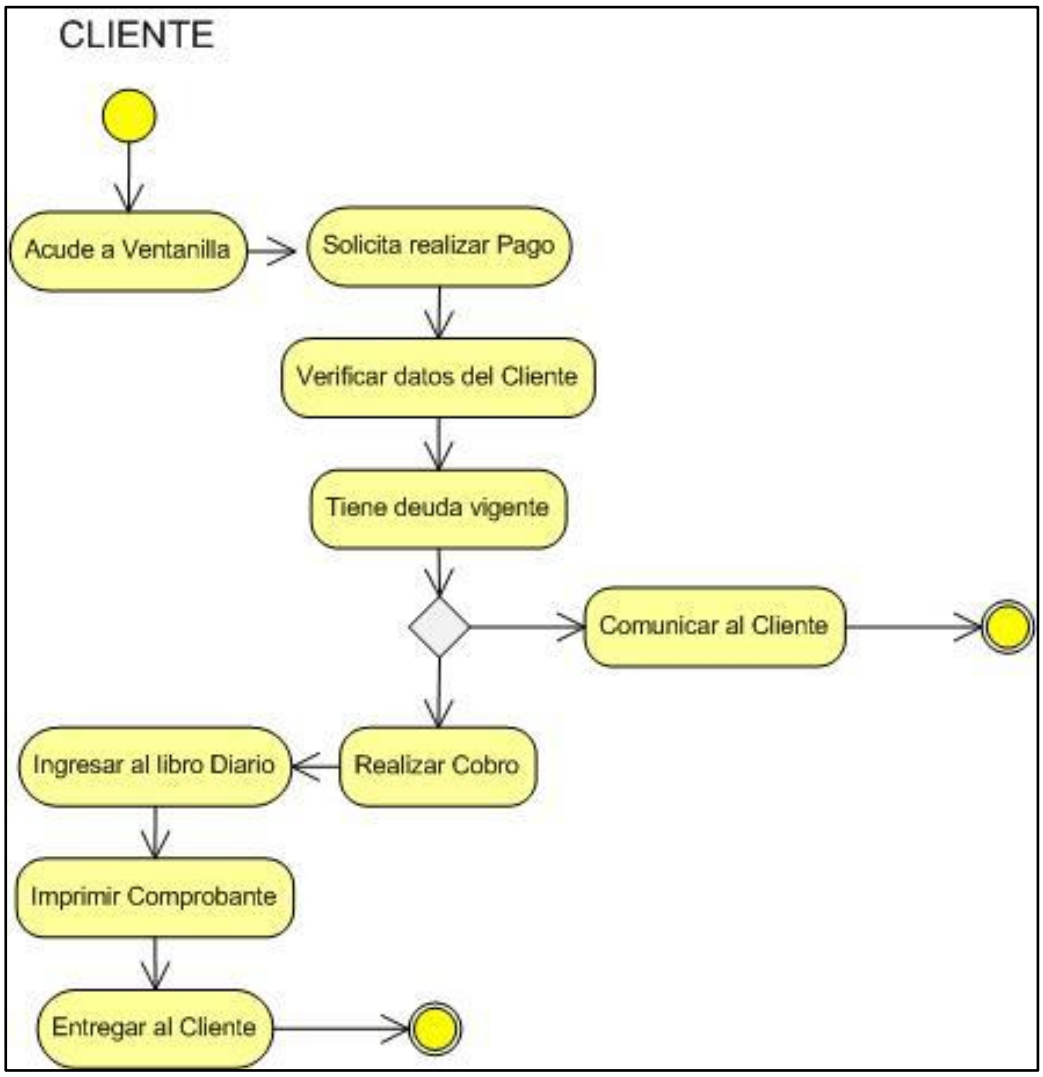

Ilustración 48. Diagrama de Actividad.- Cobro a Clientes.

4.6. Pruebas.

4.6.1. Especificación de Caso de Prueba: Seguridad de la Aplicación.

Descripción:

Antes de iniciar la aplicación aparecerá un formulario de acceso al sistema, el usuario que desee ingresar a la aplicación deberá tener creada una cuenta. Además, el sistema al ser una aplicación web puede facilitarse para que pueda acceder a las pages directamente por ingreso de la url por lo que hemos desarrollado un módulo de seguridad, el mismo que empaqueta la aplicación para que no se pueda acceder de manera aleatoria.

Comprobación:

Ingresar a la página inicial del sistema luego ingresar datos de usuario no registrados, la aplicación misma se encarga de emitir un mensaje de usuario no autorizado. Además también se puede ingresar directamente la url hacia donde quiere irse ejemplo: http://localhost:8080/subganaWeb/index.jsf, el módulo de seguridad hace que la página no se abra y salga una página como mensaje que no tiene permiso de acceso a ese recurso.

Condiciones de ejecución:

Para realizar esta prueba del cliente el usuario podrá realizarla en cualquier momento.

Entrada:

- Ingresar a la página inicial de la aplicación http://localhost:8080/subganaWeb/index.jsf.
- Ingresar datos de un usuario que no existe en la base de datos.
- Ingresar a la página http://localhost:8080/subganaWeb/principal.jsf directamente escribiendo en la barra de navegación la dirección antes mencionada.

Resultado esperado:

- No permite ingresar a usuarios que no existen en la base de datos.
- Al tratar de ingresar directamente a una página del sistema sin estar logueado, no permite ingresar y le expulsa de la aplicación.

Evaluación de prueba:

- Prueba satisfactoria.

4.6.2. Especificación de Caso de Prueba: Registro Asiento Contable

Descripción:

Para registrar los asientos contables el usuario deberá tener un comprobante físico, es

decir, un documento que verifique los datos del asiento contable. Luego deberá ingresar los datos del asiento, seguidamente ingresar el detalle del asiento contable, finalmente al momento de guardar los registros se deberá verificar que los valores del debe sean exactamente iguales a los datos del haber, en caso de cumplir con este requisito se procederá a guardar el asiento contable.

Comprobación:

Ingresar un asiento contable desde el administrador del libro diario, por ejemplo ingresar los datos de adquisición de computadoras, entonces ingresa al debe con la cuenta contable respectiva el valor de $1000, y en el haber ingresar el dato con $800, entonces el sistema deberá verificar la suma total de los valores del debe con la suma total de los valores del haber, en caso de no ser iguales la aplicación generará un mensaje de error, y que permita nuevamente ingresar los datos del asiento.

Condiciones de ejecución:

- El usuario deberá estar logueado en el sistema y tener permiso para el módulo de Contabilidad.
- El asiento debe cumplir con los principios de contabilidad.

Entrada:

- Cabecera del asiento contable.
- Detalle del asiento contable.

Resultado esperado:

- Registro normal de los asientos contables.
- Mensaje de error, en caso de no cumplir con los requisitos de este módulo.

Evaluación de prueba:

- Prueba satisfactoria.

4.6.3. Especificación de Caso de Prueba: Adquisición y compra de Productos.

Descripción:

En este caso de pruebas el usuario administrador de Inventarios tendrá la capacidad de realizar compras y posteriormente ingresarles a Inventarios. El proceso que se realiza para ingresar productos o agregar unidades al stock, parte desde la verificación en la base de datos si existe el producto o no, en caso de existir se registra primero al producto y posteriormente el stock del producto; en caso de existir el documento se selecciona y luego se ingresa el Kardex del producto, con los datos de la compra: proveedor, precio, fecha de compra.

Comprobación:

En esta prueba se registraron 50 productos que existirían enla bodega de la feria ganadera, cada producto se ingresó el stock a través del Kardex. Además todo tipo de compra se debe registrar en un asiento contable con el fin de llevar un control contable de las compras realizadas.

Condiciones de ejecución:

- El usuario deberá estar logueado en el sistema y tener permiso para el módulo de Inventarios.

Entrada:

- Proveedores.
- Productos.
- Kardex.

Resultado esperado:

- Registro normal de las compras realizadas.
- Mensaje de error, en caso de no cumplir con los requisitos de este módulo.

Evaluación de prueba:

- Prueba satisfactoria.

4.6.4. Especificación de Caso de Prueba: Entregas de Productos a Empleados.

Descripción:

Este proceso se utilizará para entregar productos a los empleados, ya que esta ayudará a realizar el trabajo de manera organizada. Para realizar una entrega de un producto, el empleado que necesita el producto deberá acercarse a la bodega de la feria ganadera y solicitar con su número de cédula el producto, entonces luego de eso el sistema verifica que exista este producto en stock para poder entregar, en caso de no existir se anulará la petición del empleado, en caso de existir se registrarán los datos de la salida del producto y además se entregará un comprobante al empleado.

Comprobación:

En esta prueba se estableció entregar tres escobas y tres recogedores de basura al empleado XYZ, dicha entrega emitió el documento según el formato establecido por la empresa. Además se imprime en dos ejemplares uno para el empleado y otro para la empresa.

Condiciones de ejecución:

- El usuario deberá estar logueado en el sistema y tener permiso para el módulo de Inventarios.
- Debe existir el producto.
- El producto debe tener stock para poder entregar, caso contrario deberá salir un mensaje de error indicando al usuario que no existe stock para este producto.

Entrada:

- Datos Empleados.
- Producto.
- Kardex.

Resultado esperado:

- Registro normal de las entregas de productos realizadas.
- Mensaje de error, en caso de no cumplir con los requisitos de este módulo.

Evaluación de prueba:

- Prueba satisfactoria.

4.6.5. Especificación de Caso de Prueba: Cuentas por Pagar Proveedores.

Descripción:

Esta prueba se realiza con los Proveedores de productos para bodega, este proceso es sencillo ya que cuando se realiza la compra de algún producto se debe especificar la forma de pago, y en caso de ser a crédito se especifica el número y el valor de las cuotas que se debe cancelar por la compra.

Comprobación:

Se realizó una compra de útiles de aseo para inventarios de bodega, con un total de la compra de $1000 de los cuales se especificó cancelar 12 cuotas mensuales de $98.00, entonces en nuestra aplicación ingresamos los datos, el cual nos permitirá llevar un registro financiero de esta cuenta por pagar a los proveedores.

Condiciones de ejecución:

- El usuario deberá estar logueado en el sistema y tener permiso para el módulo de Inventarios.
- La cuenta por pagar debe ser asignada a un proveedor.

Entrada:

- Datos Proveedor.
- Producto.
- Kardex.

- Datos del crédito de la compra.

Resultado esperado:

- Registro normal de las compras realizadas.
- Mensaje de error, en caso de no cumplir con los requisitos de este módulo.

Evaluación de prueba:

- Prueba satisfactoria.

4.6.6. Especificación de Caso de Prueba: Cuentas por Cobrar.

Descripción:

Esta prueba se realiza para los locales comerciales que existen dentro de la feria ganadera, ya que ellos deben pagar por el alquiler de su espacio físico mensualmente, además es importante señalar que la deuda será para la persona encargada del local.

Comprobación:

Para verificar el correcto funcionamiento de este módulo el administrador de cuentas por cobrar deberá ingresar el número del local, e inmediatamente se despliega los datos del cliente y de la deuda. Además todos los pagos realizados por los clientes se ingresarán en el libro diario y luego se emitirá un comprobante de pago.

Condiciones de ejecución:

- El usuario deberá estar logueado en el sistema y tener permiso para el módulo de Cuentas x Cobrar.
- La cuenta por cobrar debe ser asignada al dueño del local comercial.
- Se registrará pagos mensuales por cada local comercial.

Entrada:

- Datos Local Comercial.
- Datos del cobro realizado.

Resultado esperado:

- Registro normal de los cobros realizados.
- Asignación de nuevas cuentas por cada local creado.

Evaluación de prueba:

Prueba satisfactoria.

CAPITULO V
CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones.

- El desarrollo de esta aplicación ayudará a la Empresa Pública Municipal de Faenamiento y Productos Cárnicos de Ibarra (EP – FYPROCAI) a llevar de manera ordenada y sistematizada las transacciones comerciales y económicas que se llevan dentro de la empresa.

- Las aplicaciones web desarrolladas en java se pueden visualizar en cualquier navegador, sin embargo el rendimiento que se obtiene en Internet Explorer no refleja el rendimiento de la aplicación ya que se vuelve lenta y tarda demasiado en la obtención de resultados.

- La tecnología que se utilizó en el desarrollo de este proyecto facilita mucho a los programadores para llevar el código fuente de una manera ordenada, cómoda y segura.

- Se comprobó que la utilización de la librería Primefaces ayuda mucho para manejar Ajax, lo que permite que la aplicación sea mucho más eficiente como también ayuda en el tiempo de desarrollo ya que existen los componentes predefinidos.

- Se logró integrar la aplicación de la Subasta Ganadera que también fue desarrollada con la misma tecnología para mantener un estándar entre aplicaciones dentro de la empresa.

- Se diseñó una aplicación de fácil manejo para el usuario para evitar cualquier tipo de error humano que pueda ocasionar pérdida de información.

5.2. Recomendaciones.

- ✓ Para el correcto funcionamiento de las aplicaciones web desarrolladas con Java se recomienda ejecutarlas en los siguientes navegadores: Internet Explorer la versión 7.0 o superior, Mozilla Firefox, Google Chrome y cualquier navegador que tenga compatibilidad con JavaScript.

- ✓ Es recomendable para los desarrolladores utilizar frameworks y arquitectura de n capas que permitan la optimización del tiempo de construcción y también faciliten el mantenimiento de las aplicaciones.

- ✓ Se debe usar las herramientas de validación de código que tienen los entornos de desarrollo de software, estas herramientas ayudan a mejorar el estilo de programación.

- ✓ Es recomendable utilizar la tecnología Java para la construcción de aplicaciones web empresariales ya que con esta tecnología se puede cumplir con los factores de calidad del software.

- ✓ Se recomienda a las Autoridades de la Universidad y de la Facultad buscar convenios con otras instituciones para que los estudiantes y futuros profesionales apliquen sus conocimientos y al mismo tiempo tengan más oportunidades en el ambiente laboral.

- ✓ A nuestros docentes, es importante fomentar la investigación de nuevas tecnologías ya que con estas se puede llegar a soluciones más óptimas y rápidas.

5.3. Análisis de Impacto.

Para realizar el análisis de impacto sobre la implantación de la aplicación se estableció realizar en base a los tiempos de ejecución de los procesos como también se realiza un detenido análisis cuantitativo antes y después de la implantación del sistema.

5.3.1. Contabilidad.

Anteriormente se llevaba los registros contables de forma manual y en archivos de hojas de cálculo, por lo que se dificultaba realizar un análisis en tiempo real.

Es importante señalar que la mayoría de la información anteriormente era versátil e insegura por lo que generaba desconfianza al momento de generar un reporte contable entre fechas. Ahora mediante la aplicación se genera de manera mas rápida y segura.

A continuación, se muestra un gráfico aproximado del proceso de guardado de registros contables durante la venta del ganado.

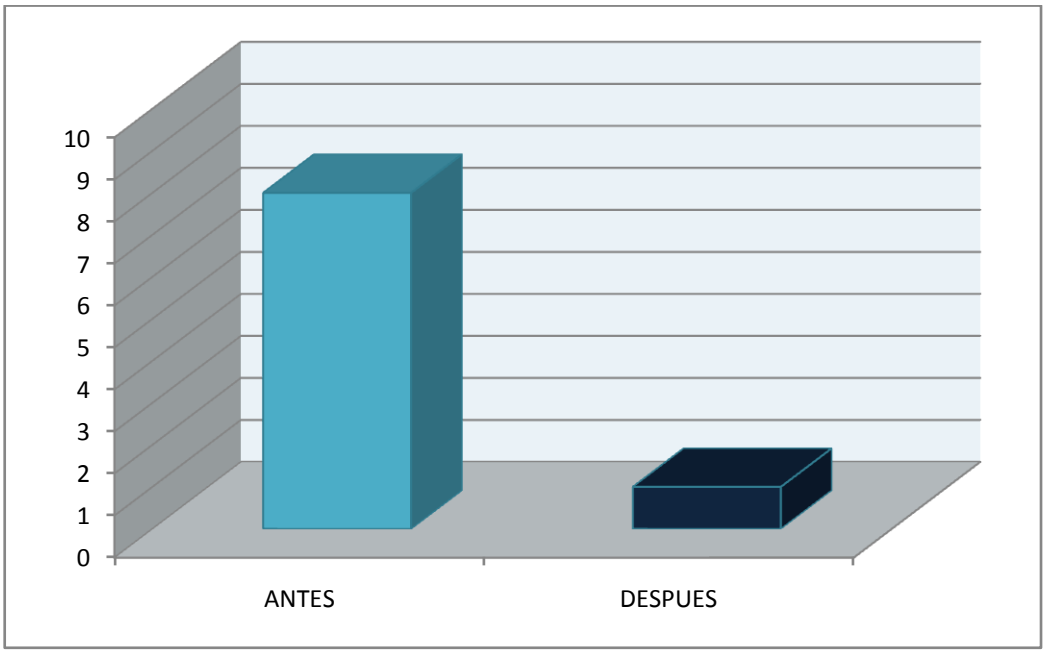

Ilustración 49. Análisis de Tiempo durante la Venta de Ganado.

En el anterior gráfico, se describe el tiempo de ejecución del proceso de guardado de un registro contable durante la venta de ganado. Antes de la implantación del sistema se demoraba aproximadamente cinco minutos en guardar ya que no existe un sistema informático que permite realizar el ingreso automático en el libro diario de la empresa, en

cambio durante las pruebas del sistema se registra toda la información, tanto de la venta del ganado como también del registro en el libro diario en aproximadamente un minuto. Por lo que se estableció conjuntamente con las personas encargadas del manejo de la aplicación que el sistema optimiza el tiempo de trabajo del empleado.

De la misma forma cuando se desea realizar un balance general de la empresa, es necesario seguir muchos procesos que dificultan tener los datos de forma rápida.

A continuación, se muestra un gráfico aproximado durante el tiempo de generación del balance general.

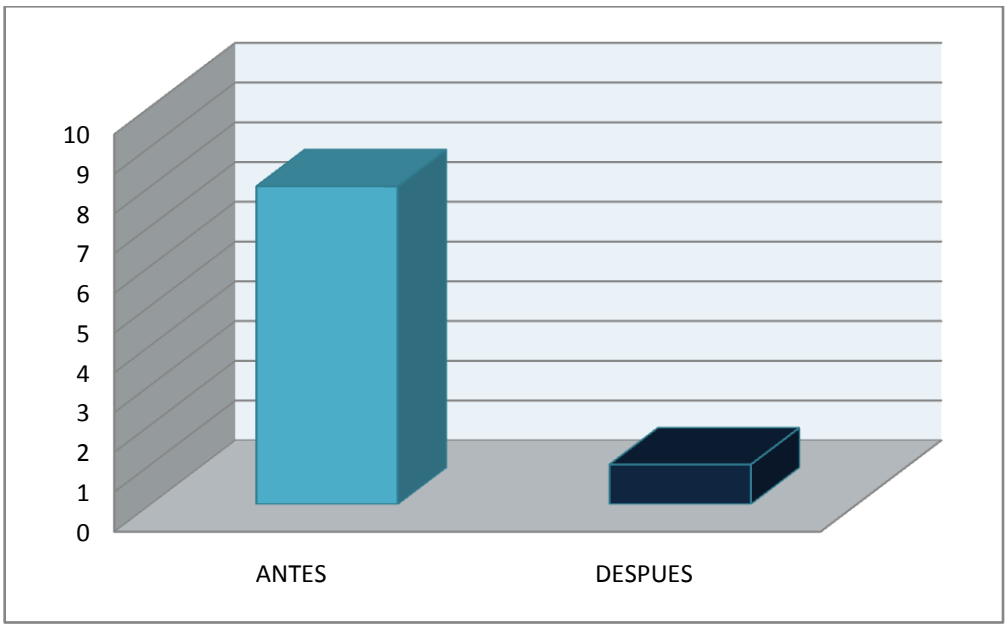

Ilustración 50. Análisis de Tiempo durante la generación del Balance General.

En la ilustración 50, se muestra un gráfico estadístico que permite visualizar la generación de un balance general en función del tiempo en horas. Este gráfico permite a observar que anteriormente la creación y visualización del balance general demoraba entre siete y ocho horas, con la implementación del sistema informático se mejora el tiempo ya que para generar el Balance General se demora menos de una hora.

5.3.2. Inventarios.

Debido a que por el momento no se encuentra disponible la infraestructura física para realizar la Subasta Ganadera se realizó un análisis en base a proyecciones por parte de los técnicos encargados de la empresa.

Mediante la aplicación se permitirá optimizar ciertos procesos en el manejo de los productos, uno de los más importantes es llevar un control de los productos al momento de realizar las compras y las entregas a los empleados para que puedan utilizar dentro de la empresa.

Con la implementación de la aplicación se regularizará el uso de los productos y equipos dentro de las instalaciones de la feria ganadera. También se logrará un ahorro sustancial en cuanto a los instrumentos de aseo ya que dentro del sistema se establece los datos del empleado al que ha sido asignado dicho producto y por tal motivo no se podrá utilizar para fines personales.

Otro punto importante es manejar el stock de productos, dicho proceso ayudará a establecer la disponibilidad de los productos, como también facilitará saber de que productos se deberá realizar la compra inmediata ya que durante el ingreso de personas y animales no deberá faltar ningún producto importante, como por ejemplo el papel, útiles de aseo para los establos donde estarán los animales durante el proceso de la subasta.

5.3.3. Cuentas por Cobrar.

Este módulo es de vital importancia para realizar el cobro a las personas que mantienen un local comercial dentro de la feria ganadera. Anteriormente el proceso de cobro se realizaba de forma manual y el empleado tenía que pasar por los locales comerciales revisando el pago de cada mes, con la implementación de esta aplicación se optimizará el tiempo de trabajo del empleado ya que dicha aplicación emitirá reportes mensuales de los locales que aún no han cancelado su cuota mensual y de esa forma el empleado se evita comprobar por cada local si ha cancelado o no.

Otro impacto significativo es en cuanto a los comerciantes, ya que ellos podrán acceder a los reportes de pagos que han realizado y así podrán evitarse las multas y sanciones que significa la falta de pago o incumplimiento de contrato de arriendo.

A continuación, se presenta un gráfico que representa el tiempo de cobro de un empleado antes y después de la implementación de la aplicación.

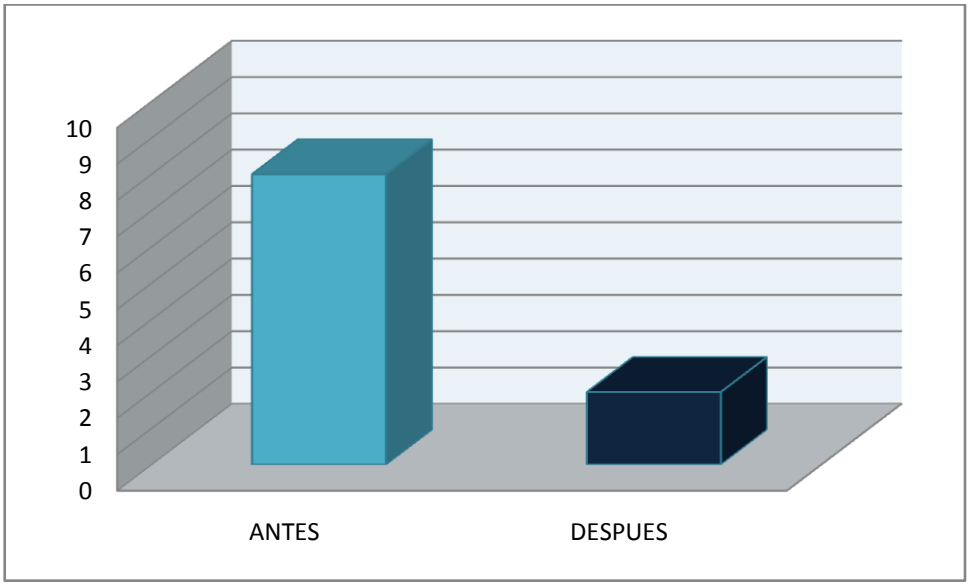

Ilustración 51. Análisis del Tiempo durante el Proceso de Cobro.

En la Ilustración 51, se muestra el tiempo que se demora durante el proceso de cobro a los locales comerciales, antes de la implementación se demoraba entre siete y ocho horas en realizar el cobro a todos los locales comerciales ya que era un proceso manual, con la implementación del sistema las personas encargadas del local comercial deberá acercarse a la ventanilla de cobro a realizar el respectivo pago mensual. Con esto se optimiza el tiempo de trabajo de los empleados.

Además las personas encargadas de los locales comerciales podrán solicitar un reporte general de los pagos realizados, como también podrán cancelar de algunos meses ya sea por adelanto o por retraso.

5.3.4. Ventas.

Este proceso tiene relación directa con la subasta, lo que implica que a este módulo se le ha realizado las pruebas necesarias para la implementación una vez que termine de construirse la infraestructura física se procederá a utilizar este módulo.

Sin embargo los resultados obtenidos durante la fase de pruebas muestran grandes ventajas sobre el actual proceso de venta de ganado.

Uno de los impactos más importantes es que la venta de ganado anteriormente no tenía control, lo que implicaba que los productores podían vender su ganado, incluso existe en muchas de las veces ganado robado que podía ser comercializado. Ahora con la

implementación de esta aplicación se podrá establecer los precios reales del ganado, también se emitirá documentos de propiedad del ganado para que este pueda ser faenado sin ningún problema, además todo el ganado portará la identificación que emite el Ministerio de Agricultura y Ganado, con esto se desvincula la comercialización ilícita de ganado.

Finalmente es importante mencionar que sin la ayuda de este software se dificulta manejar procesos delicados dentro de la empresa, como también se optimiza en muchos de los casos el tiempo de trabajo de los empleados.

BIBLIOGRAFIA

LIBROS.

Martin, A. (2008). *Programador Certificado Java 2.* Mexico: Alfaomega Grupo Editor S.A.

Zapata, P. (2005). *Contabilidad General.* Mc Graw Hill.

Deitel, H. (2003). *Como programar en Java.* Mexico: Pearson Education Inc.

PUBLICACIONES EN LINEA.

Enterprise Java Beans EJB. (Enero de 2005). Recuperado el Abril de 2012, de http://www.proactiva-calidad.com/java/ejb/introduccion.html

Alvarez, M. (Febrero de 2008). *Listado de Distintos Framworks Javascript.* Recuperado el Mayo de 2012, de http://www.desarrolloweb.com/articulos/listado-distintos-framework-javascript.html

Alvarez, S. (Febrero de 2006). *Tipos de Lenguajes de Programación.* Obtenido de http://www.desarrolloweb.com/articulos/2358.php

Coplec. (2010). *Java Persistence Api (JPA).* Recuperado el Abril de 2012, de http://www.coplec.org/?q=book/export/html/240

Diego, R. (25 de Diciembre de 2007). *¿Que es un servidor de aplicaciones?* Recuperado el 1 de 12 de 2011, de http://www.editum.org/Que-Es-Un-Servidor-De-Aplicaciones-p-473.html

Editor. (2007). *PHP Frameworks.* Recuperado el Febrero de 2012, de http://www.phpframeworks.com/

Grajeda, P. (Agosto de 2008). *Tecnologías Java: Introducción a EJB 3.0.* Recuperado el Abril de 2012, de http://tecnologiasjava.blogspot.com/2008/08/introduccin-ejb-30-parte-1.html

Java. (2011). *¿Qué es java y porqué lo necesito?* Recuperado el Mayo de 2012, de http://www.java.com/es/download/faq/whatis_java.xml

López, E. S. (Noviembre de 2008). *Framework para Desarrolladores de Aplicaciones Web.* Recuperado el Mayo de 2012, de http://www.slideshare.net/estebansaavedra/frameworks-para-desarrollo-de-aplicaciones-web-presentation

PostgreSQL. (s.f.). Recuperado el Febrero de 2012, de http://www.postgresql.org/

Quasar, C. (2011). *JBOSS Enterprise Middleware.*Recuperado el Febrero de 2012, de http://www.quasarbi.com/mostrarpagina.php?codpage=620

Wikipedia, E. (12 de Mayo de 2010). *Servidor de Aplicaciones*. Recuperado el 2011, de Fundación Wikipedia Inc.: http://es.wikipedia.org/wiki/Servidor_de_aplicaciones

Wikipedia, E. (Mayo de 2012). *Framework.* Recuperado el Mayo de 2012, de http://es.wikipedia.org/wiki/Framework

Wikipedia, E. (Marzo de 2012). *Java Server Faces JSF.* Recuperado el Mayo de 2012, de http://es.wikipedia.org/wiki/JavaServer_Faces

Wikipedia, E. (Mayo de 2012). *JBOSS.* Recuperado el Mayo de 2012, de http://es.wikipedia.org/wiki/JBoss

Wikipedia, E. (Marzo de 2012). *WebSphere.* Recuperado el Abril de 2012, de http://es.wikipedia.org/wiki/WebSphere

ANEXOS

ANEXO 1: MANUAL DE CONFIGURACION.

Este documento está elaborado en formato digital y se encuentra en el CD.

ANEXO 2: MANUAL DE USUARIO.

Este documento está elaborado en formato digital y se encuentra en el CD.

ANEXO 3: MANUAL TÉCNICO.

Este documento está elaborado en formato digital y se encuentra en el CD.