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 DE LICITACIÓN Y SUBASTA GANADERA"

Autor: Verónica Marcela Cuasapaz Vela

Director: Ing. Xavier Mauricio Rea Peñafiel

Ibarra – Ecuador

2012

CERTIFICACIÓN

Certifico que la Tesis **"SISTEMA INFORMÁTICO DE SUBASTA GANADERA (SUBGANA)"** con el aplicativo **"MÓDULO DE LICITACIÓN Y SUBASTA GANADERA"** ha sido realizada en su totalidad por la señora: Verónica Marcela Cuasapaz Vela portadora de la cédula de identidad número: 100286968-1.

……

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 de la Egresada VERÓNICA MARCELA CUASAPAZ VELA con CI: 100286968-1 quien desarrolló su trabajo con el tema "SISTEMA INFORMÁTICO DE SUBASTA GANADERA (SUBGANA)" con el aplicativo "MÓDULO DE LICITACIÓN Y SUBASTA GANADERA", 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 de la egresada VERÓNICA MARCELA CUASAPAZ VELA. Una vez que hemos recibido la capacitación y documentación respectiva, nos comprometemos a continuar utilizando el mencionado aplicativo en beneficio de nuestra empresa.

La egresada VERÓNICA MARCELA CUASAPAZ VELA 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, VERÓNICA MARCELA CUASAPAZ VELA, con cedula de identidad Nro. 100286968-1, 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 DE LICITACIÓN Y SUBASTA GANADERA",** 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: VERÓNICA MARCELA CUASAPAZ VELA
Cédula: 100286968-1

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	100286968-1
APELLIDOS Y NOMBRES	VERÓNICA MARCELA CUASAPAZ VELA
DIRECCIÓN	LA VICTORIA
EMAIL	vero_mcv@hotmail.com
TELÉFONO FIJO	062959592
TELÉFONO MOVIL	085146705

DATOS DE LA OBRA	
TITULO	"SISTEMA INFORMÁTICO DE SUBASTA GANADERA (SUBGANA - MÓDULO DE LICITACIÓN Y SUBASTA GANADERA"
AUTOR	VERÓNICA MARCELA CUASAPAZ VELA
FECHA	14 DE JUNIO 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, VERÓNICA MARCELA CUASAPAZ VELA, con cedula de identidad Nro. 100286968-1, 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: VERÓNICA MARCELA CUASAPAZ VELA
Cédula: 100286968-1
Ibarra a los 23 días del mes de julio del 2012

DEDICATORIA

Dedicó mi tesis en primer lugar a mis abuelitos que han sido como unos verdaderos padres para mí, gracias por estar a mi lado apoyándome y aconsejándome siempre, gracias a su ejemplo y palabras de aliento crecí como persona.

A mi madre, por hacer de mí una mejor persona a través de sus consejos, enseñanzas y amor, y porque siempre ha estado pendiente de mis estudios.

Quiero agradecer a mi esposo, el cual me ha apoyado e impulsado a alcanzar la meta que hoy logro, que se ha sacrificado junto a mí y ha sido mi soporte para no darme por vencida, que ha compartido conmigo los sacrificios y noches de desvelo, pero sobre todo ha sido mi compañero incondicional.

A mi bebit@ que es un regalo que Dios ha mandado a nuestra familia para bendecirnos, es señal de que hay esperanza para continuar adelante, esperamos con cariño y regocijo tu llegada.

A mis hermanos mayores y menores, por estar siempre presentes, cuidándome y por ayudarme a cumplir mis objetivos como persona y estudiante.

AGRADECIMIENTO

A ti Dios, por ayudarme a terminar este proyecto, gracias por darme la fuerza y el coraje para hacer este sueño realidad, por estar conmigo en cada momento de mi vida.

A mi amado Claudio, gracias por apoyarme y creer en mí. Gracias por amarme a pesar de cómo soy y porque desde que te conocí supe que eras el hombre de mi vida, junto a ti me han pasado las cosas más increíbles, lo único que te puedo decir es que Te Amo.

A mi familia por darme su amor, apoyo, confianza y compartir únicos e inolvidables momentos en mi vida.

Un agradecimiento especial al Ing. Mauricio Rea, Director de tesis; que con su profunda calidad humana, el apoyo brindado y los conocimientos que supo compartir, fue posible realizar esta tesis.

A la prestigiosa Universidad Técnica del Norte por acogerme en sus aulas y ayudarme a contribuir en mi desarrollo profesional.

A mis profesores quienes me han forjado como una profesional en esta etapa universitaria, tanto dentro como fuera de las aulas de clases.

A mis compañeros de clases quienes me acompañaron en esta trayectoria de aprendizaje y conocimientos.

TABLA DE CONTENIDOS

INDICE DE FIGURAS

RESUMEN

El módulo de Subastas y Licitación del sistema de Subasta Ganadera (SUBGANA) fue desarrollado con la finalidad de brindar competitividad, precios justos, sistematización de procesos, etc.

El módulo funciona de la siguiente manera, primero es necesario el registro de vendedores, luego de animales. Es importante también el registro de cada comprador candidato, el mismo que proporcionará sus datos personales.

El submódulo de Subasta funciona tanto de manera local como en la web, pero no ambos simultáneamente. Este submódulo consiste en publicar una subasta seleccionando un animal disponible, registrando el peso del animal medido en la báscula, también se registra el precio del animal que se obtiene como resultado de la multiplicación del peso por el valor del kg o lb, mismo que es proporcionado por el MAGAV; también es necesario registrar la Hora Inicial y Final de la subasta.

La subasta local consiste en que en el recinto donde se realiza la subasta, se encuentran como participantes: los vendedores, animales, compradores, personal de la empresa. El proceso funciona de la siguiente manera: el vocero que dirige la subasta nombra cada uno de los animales en un determinado orden, vale recalcar que cada animal será subastado uno a uno, al animal que será subastado se le procede a pesar en la bascula y se muestra en la pantalla tanto el Número de Marcado como también el peso y el precio base, estos datos son mostrados a todos los compradores, para que den sus ofertas. Cada oferta dada para cada uno de los animales es registrada de forma manual, ya que luego de terminada la subastase registra en el sistema sólo la mejor oferta.

Para el proceso de subasta web, cada comprador anteriormente registrado puede acceder al sistema web y proporcionar una o varias ofertas para cada una de las subastas que se encuentran en estado “abierta”.

Una vez finalizado el tiempo para cada subasta, el administrador del sistema, es la única persona encargada de aceptar una oferta de todas las que se encuentran registradas para cada subasta.

Al aceptar una oferta automáticamente, la subasta pasa a estado “cerrada”, el estado del animal que pertenece a dicha subasta se modifica a estado “subastado”.

El proceso de Licitación se realiza de forma automática para cada subasta una vez que ésta se halla en estado “cerrada”, dicho proceso asigna a cada comprador el animal por el que ha hecho la mejor oferta resultando como dueño absoluto.

SUMMARY

The module Bid Auctions Livestock Auction System (SUBGANA) was developed in order to provide competitive, fair prices, systematization of processes, etc.

The module operates as follows, first registration is required vendors, after animals. It is also important record of each candidate buyer; it will provide your personal data.

The Auction submodule works both locally and on the web, but not both simultaneously. This submodule is to publish an auction selecting an animal available, recording the animal's weight measured on the scale, also recorded the price of the animal that occurs as a result of multiplying the weight by the value of kg or lb, it is provided by the Ministry of Agriculture and Livestock (MAGAV) but it is necessary to record the start and end of the auction.
The local auction is that in the site where the auction takes place, are as participants: vendors, animals, buyers, personnel of the company. The process works as follows: spokesman who runs the auction appoints each of the animals in a particular order, is worth noting that each animal will be auctioned off one by one, the animal to be auctioned will be weighed on the scale and screen shown in both the Dial Number as well as the weight and the base price, these data are shown to all buyers, to give their offerings. Each offer given for each animal is recorded manually, because after the auction ended is recorded in the system only the best deal.

For the process of web auction, each buyer can access previously recorded the web system and provide one or more offers for each of the auctions that are in state "open".

After the time for each auction, the system administrator is the only person responsible for accepting an offer of all who are registered for each auction.

To accept a bid automatically, the auction goes to state "closed" state of the animal that belongs to the auction shall be amended to state "auctioned".

The bidding process is performed automatically for each auction once it is in state "closed", this process assigns to each buyer that the animal has the best resulting as absolute master.

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.

1.3. Objetivos.

1.3.1. Objetivo General.

Desarrollar e implementar el Módulo de Licitación y Subasta Ganadera 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.

- Diseñar un módulo que se pueda integrar con nuevos desarrollos.
- Mejorar el proceso de venta de ganado mediante un Sistema Informático, con la finalidad de poder establecer un precio justo, transparente y competitivo.
- Diseñar una base de datos que abarque el sistema a desarrollarse.
- Generar reportes estadísticos de tal manera que permita ayudar a tomar decisiones a las autoridades de la EP – FYPROCAI.

1.4. Justificación del Proyecto.

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

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

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 de Licitación y Subasta Ganadera para el correcto funcionamiento de la misma, será desarrollado como proyecto innovador, creativo, investigativo y productivo por una alumna de la Facultad de Ingeniería en Ciencias Aplicadas de la Universidad Técnica del Norte.

El Módulo Licitación y Subasta Ganadera está divido en tres submódulos: Registro, Licitación y Subasta, Reportes y Estadísticas.
Para facilitar la comprensión de la tarea que cumple cada uno de estos, se presenta la relación entre cada uno de ellos con los componentes que debe tener todo sistema de este tipo.

Submódulo de Registro: este submódulo realiza las siguientes funciones:
Ingresar los datos del vendedor.
Ingresar los datos de los animales.

Submódulo de Licitación y Subasta: es el encargado de manejar la interacción directa entre compradores y vendedores. Entre las principales funciones están:
Ingresar los datos de los compradores.
Llevar a cabo la Subasta de ganado, permitiendo a los vendedores ofrecer sus animales y que varios compradores hagan sus propuestas. La elección de la subasta ganadora será en base a la mejor propuesta ofrecida por el comprador, es decir, la de mayor precio.
Automatizar el proceso de compra y entrega de documentación por parte del comprador.
Este submódulo funcionará tanto a nivel local como también en la web.

Submódulo de Reportes y Estadísticas: permitirá generar los reportes del sistema a nivel estratégico, es decir, para promover el cambio de los objetivos, procesos, productos, servicios y relaciones con el entorno para ayudar a la organización a obtener ventajas competitivas.

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[5].

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 distrib.uidas 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[6], Java EE es también considerada informalmente como un estándar debido a que los proveedores deben cumplir ciertos

[4] **Middleware.-** Middleware es un software de computadora que conecta componentes de software o aplicaciones para que puedan intercambiar datos entre éstas.
[5] **J2EE.-** Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java.
[6] **Java Community Process(JCP).-** Proceso de la Comunidad JAVA.

requisitos de conformidad para declarar que sus productos son conformes a Java EE; estandarizado por The Java Community Process / JCP.

Java EE incluye varias especificaciones de API[7], tales como JDBC[8], RMI[9], e-mail, JMS[10], Servicios Web, XML[11] 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[12]. 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[13], aunque el término se refiere de manera popular a uno de sus productos específicos: WebSphere Application Server (WAS[14]). 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

[7] **API.-** Interfaz de Programación de Aplicaciones
[8] **JDBC.-** Java Database Conectivity. Api que permite la ejecución de operaciones sobre base datos desde el lenguaje de programación Java.
[9] **RMI**.- Es un mecanismo ofrecido por java para invocar un método de manera remota.
[10] **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.
[11] **XML.-** Lenguaje de Marcas Extensible. Permite definir la gramática de lenguajes específicos.
[12] **E-business**.- Es la integración del negocio de una empresa incluyendo productos, procesos y servicios por medio del Internet.
[13] **IBM.-** Empresa que fabrica y comercializa hardware, software y servicios relacionados con la informática.
[14] **WAS.-** Es un servidor de aplicaciones de software.

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 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 0 y 1 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 maquina: este lenguaje ordena a la máquina las operaciones fundamentales para su funcionamiento. Consiste en la combinación de 0's y 1's para formar las ordenes 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[15](ECMA-334) e ISO[16](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)

[15]**ECMA.-** Es una organización internacional basada en membrecías de estándares para la comunicación y la información.
[16]**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.

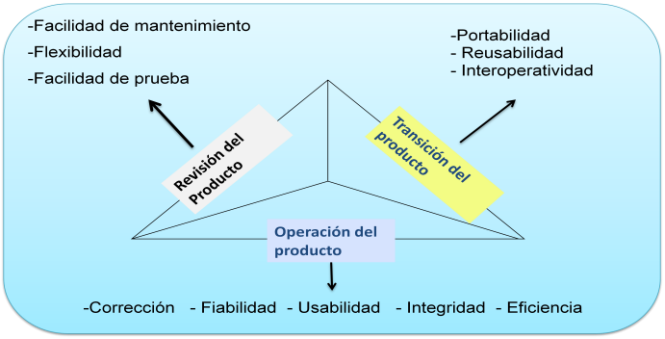


Ilustración 2. Factores de Calidad McCall (Fuente: export.writer.zoho.com)

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[17] (Controlador-Modelo-Vista), ya que debemos fragmentar nuestra programación. Tenemos que contemplar estos aspectos básicos en cuanto a la implementación de nuestro sistema:

[17] **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[18](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

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

sistemas heredados y requerimientos no funcionales. Es como una radiografía del sistema que estamos desarrollando, lo suficientemente completa como para que todos los implicados en el desarrollo tengan una idea clara de qué es lo que están construyendo, pero lo suficientemente simple como para que si quitamos 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.

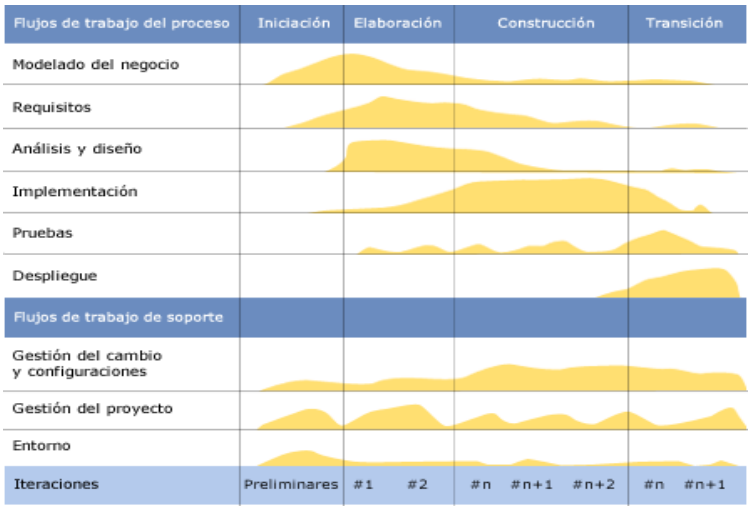


Ilustración 3. Ciclo de Vida de RUP (Fuente: http://www.scribd.com/doc/297224/RUP)

En el ciclo de vida RUP veremos una implementación del 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[19].

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

[19] **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. Subasta Ganadera.

Una **subasta** o **remate** es una venta organizada de un producto basado en la competencia directa, y generalmente pública, es decir, a aquel comprador (postor) que pague la mayor cantidad de dinero o de bienes a cambio del producto. El bien subastado se adjudica al postor que más dinero haya ofrecido por él, aunque si la subasta es en sobre cerrado, el bien se adjudica a la mejor oferta sin posibilidad de mejorarla una vez conocida.

Historia.

La palabra subasta tiene raíces históricas lejanas y viene originalmente del latín *sub asta*, bajo lanza, debido a que el reparto de tierras conquistadas entre los soldados participantes se señalaba hincando una lanza en la parcela ocupada en suerte. Asimismo la venta del botín de la guerra se anunciaba con una lanza y la venta se realizaba ante la misma.

Uno de los ejemplos históricos más famosos era el de la subasta de la esposa durante el imperio babilónico la cual se llevaba a cabo anualmente. La operación comenzaba con la subasta de la mujer más bella y luego se procedía una a una con las demás. Era de hecho un acto ilegal "obtener" una esposa fuera de dicho proceso de compra.

Igualmente famosas eran las subastas de esclavos durante el imperio romano los cuales eran capturados en campañas militares para luego ser subastados en el foro. Los fondos recaudados en dichas subastas servían a su vez para financiar los esfuerzos bélicos del imperio.

A pesar de que transacciones como las anteriormente descritas habían tenido auge en sus respectivas sociedades, el sistema de venta basado en subastas había sido relativamente raro hasta el siglo XVII. Posiblemente, la más antigua casa de subastas al mundo, la Estocolmo Auction House (Estocolmo Auktionsverk), se estableció en 1674 en Suecia.

A finales del siglo XVIII, poco después de la Revolución Francesa, las subastas llegaron a celebrarse en las tabernas para vender artículos de arte. Dichas subastas se celebraban diariamente y los catálogos eran imprimidos para anunciar elementos disponibles que generalmente eran artículos de colección raros. En algunos casos estos catálogos terminaron por convertirse en obras de arte que contenían infinidad de detalles sobre los artículos en subasta. Las dos casas de subasta más importantes, Sotheby's[20] e Christie's[21], llevaron a cabo su primera subasta oficial en 1744 y 1766 respectivamente.

En el caso de las subasta dinámicas, el que dirige y adjudica públicamente al ganador de la mejor puja (oferta) se denomina subastador o martillero, en referencia al uso de un martillo madera que golpea sobre un mesón para indicar la finalización de la subasta.

Subasta Ganadera.

Es una estructura de comercialización donde se realizan transacciones entre vendedores y compradores directos mediante pujas públicas reguladas por el Reglamento de operaciones conocido por todos los asistentes, mismo que busca transparencia en las transacciones, formación de precio, generación de información actualizada de las variables de Negociación. Tiene distintos modelos organizacionales.

2.6.1. Características.

Una Subasta Ganadera permite:

- Crear un mercado especializado conformado por productores, agroindustriales, comerciantes, compradores, exportadores, etc.
- Definir un lugar y día de reunión a donde acudan los interesados a realizar sus operaciones.

[20] **Sotheby's.-** Es una empresa de subastas, fundada en el Reino Unido y que actualmente tiene perfil multinacional, con sedes en las principales capitales del mundo.
[21] **Christie's.-** Es una de las más famosas casas de subasta del mundo. Fundada en Londres, Christie's llevó a cabo las mayores subastas de los siglos XVIII, XVIIII y XX, y hoy sigue siendo una vitrina popular para lo único y bello.

- Facilitar el contacto entre oferentes y demandantes y apoyar la realización de negocios entre las partes.
- Crear un centro de registro de información sobre ofertas, demandas y precios al cual pueden acudir los socios del mercado en forma permanente.
- Identificar, seleccionar y registrar socios del mercado para crear confianza entre las partes.
- Implantar el sistema de subasta pública usando normas de calidad o muestras representativas para darle transparencia al mercado y obtener precios de competencia.
- Estrechar los vínculos entre el sector público y privado para desarrollar el sistema de comercialización.

2.6.2. Tipos de Subasta.

Subasta Electrónica.

Es un proceso dinámico de negociación de precios on-line entre proveedores. Es una manifestación del B2B[22]. Se trata de un nuevo sistema de negociación en el que, en cierto modo, desaparece la relación personal con el cliente para convertirla en una negociación a través de internet.

Subasta Inglesa.

La Subasta Inglesa comienza con precio base y va subiendo. Los participantes van poniendo posturas uno tras otro, donde cada uno ofrece una cantidad mayor a la anterior. La subasta termina cuando nadie ofrece algo mayor a la última postura, o cuando se termina el tiempo de la subasta.

Subasta Inglesa Inversa.

La Subasta Inglesa Inversa descendente inicia con un precio mayor (tope) y los postores comienzan a ofrecer un precio menor, uno tras otro, hasta que termine el tiempo de la subasta. El ganador será el precio menor. En este tipo de subasta el precio deberá de ser menor o igual al precio de reserva.

[22] **B2B.-** (Busissnes to Busissnes) Negocio a negocio. Consiste en el comercio electrónico entre empresas a través de Internet.

Subasta Holandesa.

En la Subasta Holandesa el sistema lanza las pujas iniciando con un precio mayor al esperado y de ahí comienza a bajar, hasta que un participante acepte un precio y es el que se declara ganador, siempre y cuando supere el precio de reserva. En este tipo de subasta el precio ganador debe ser mayor o igual al precio de reserva.

Subasta de Lotes Parciales.

La Subasta de Lotes Parciales consta de múltiples lotes que se asignan a las mejores posturas de la subasta. Cada postor puede dar un precio por lote y elegir el número de lotes que quiere adquirir. El sistema buscará la asignación que le dé mejores beneficios al cliente.

Subasta Silenciosa.

La Subasta Silenciosa es donde todos los postores colocan una única puja sin conocer las pujas del resto de los participantes. El ganador de esta subasta será quien coloque la mejor postura antes del cierre de la misma, siempre y cuando mejore el precio de reserva.

Subasta Ranking.

En la Subasta Ranking se definen el número de ganadores para cada subasta, y los mejores precios son los ganadores. Cada postor puede ver el lugar que ocupa pero no puede ver las posturas de los demás. En este tipo de subasta serán ganadoras las mejores posturas y que sean mayores o iguales al precio de reserva.

Subasta en Sobre Cerrado.

La subasta en sobre cerrado es aquella en la que los postores presentan su oferta en una sola ocasión. Puede ser De primer precio si el ganador paga el precio que ofreció, o De segundo precio si el ganador paga el precio ofrecido por quien quedó en segundo lugar.

Subasta Dinámica.

Los postores conocen las ofertas de su competencia y pueden modificar la suya mientras la subasta está abierta. La subasta dinámica puede ser ascendente (inglesa), que parte de un precio de reserva y consiste en que los postores vayan presentando precios ascendentes, ganando quien ofrezca el precio mayor; descendente (holandesa) que se inicia con un precio determinado, superior a todas las ofertas, y que el subastador va bajando por etapas: gana el postor que primero acepta un precio. En la subasta americana todos los postores deben pagar la oferta que hacen, pero sólo el que realiza la mejor oferta obtiene el producto.

Subasta Round Robin.

Se trata de una variante de la subasta con oferta cerrada que viene utilizada primordialmente para la venta de inmuebles. Los interesados hacen sus ofertas en una bolsa cerrada y luego el subastador le comunica a los postores cual es la oferta que se está adjudicando la subasta en esos momentos. Después de dicha comunicación, los participantes pueden tratar de superar la oferta máxima o abandonar la subasta.

Subasta a la Baja.

También conocida con el nombre de subasta inversa, en la que el postor ganador es aquel que realiza la puja única más baja. La subasta permanece abierta durante un tiempo determinado durante el cual la gente puede pujar sin que el resto de pujadores conozcan el valor de las pujas. Una vez finalizado el tiempo de la subasta, esta se cierra y se da a conocer el ganador de la misma.

Subasta Céntimo a Céntimo.

Esta nueva variante de subastas fue inventada en Finlandia en 2008 y consiste en subir el precio del producto en un céntimo con cada puja realizada. Cada producto tiene un reloj, y la subasta sólo termina cuando finaliza la cuenta atrás y por tanto, cuando ninguna otra persona haya pujado por el producto durante el período marcado en su reloj. La última persona en pujar gana el producto por el importe que lo acompaña al finalizar la subasta.

Subasta y Concurso.

La subasta trata sobre un tipo expresado en dinero, con adjudicación al licitador que, sin exceder de aquel, ofrece el precio más bajo.

En el Concurso, la adjudicación recaerá en el licitador que, en su conjunto, haga la proposición más ventajosa, teniendo en cuenta los criterios que se hayan establecido en los pliegos, sin atender exclusivamente al precio de la misma y sin prejuicio del derecho de la Administración a declararlo desierto.

2.6.3. Tipos de Subasta Ganadera.

Subastas presenciales.

En este tipo de subasta el ganado ingresa y es ofertado a precio por kilo, los compradores pujan con paletas numeradas y son asignados por un "Martillo".

Video Subastas.

Son aquellas en las que con modernas tecnologías se le muestra al cliente el ganado a subastar. Adicionalmente, se ofrece un nuevo servicio de ventas a través de Internet, en donde la empresa de subastas coloca en su portal las fotografías de los lotes a vender, cada uno identificado con un código numérico y con la indicación de edades, pesos y características raciales.

2.6.4. Ventajas y Desventajas de la Subasta Ganadera.

Ventajas.

- Mecanismo moderno de comercialización que transparenta negociaciones, ayuda a formar precios, genera información a tiempo real.
- Estrategia política para manejo de Contingentes, abastos.
- Estrategia para la mejora de la sanidad agropecuaria.
- Mecanismo de apoyo para el fomento de la producción, de pequeños y medianos productores y comerciantes.

- Modernización Institucional de entidades de Comercialización
- Igualdad de oportunidades por optar por un mercado potencial.
- Conocimiento del precio del producto en el mercado.
- Competitividad por calidad, precio y diferenciación.
- Conocimiento claro de las reglas de negociación.
- Confirmación del pago.
- Libre competencia.
- Conocimiento de los proveedores, disminuir gastos por búsqueda de proveedores.
- Evitar gastos por “falsos fletes” y transportes.
- Libre competencia por acceso a productos.
- Oportunidad de generar alianzas con socios comerciales.

Desventajas.

- No contar con la infraestructura necesaria para el correcto funcionamiento de la Subasta Ganadera.
- Para poder realizar una subasta se necesita incurrir en una serie de gastos: salarios del personal, garantías sociales, incentivos, pólizas y seguros, servicios profesionales, teléfono, internet, electricidad, agua potable, depreciaciones, mantenimiento de los edificios, corrales, potreros, fotocopiadora, gastos de Asambleas, financieros, papelería, equipos, materiales de subasta y donaciones, entre otros.

2.6.5. Seguridad en la Subasta Ganadera.

Las instalaciones y equipos en las subastas, deberán cumplir con los siguientes requisitos:

- Los corrales deberán brindar seguridad a todos los usuarios en general y trabajadores de la subasta, así como para los animales, permitiendo un trato adecuado para éstos.
- El desembarcadero deberá ser apropiado, cómodo y seguro, tener rampa ajustable a la altura del camión, con bordes redondeados, tornillos ajustados, sin salientes ni con objetos punzo cortantes que lesionen a los animales. Los pisos deben ser antideslizantes, sin obstáculos, grietas ni orificios.

- Todos los corrales de contención deberán tener acceso seguro para que los ganaderos puedan observar los lotes de cerca, previos a su remate, si así lo desean.
- Deberán tener un cepo de contención seguro y adecuado con capacidad suficiente para cada animal, sin objetos punzo cortantes que lesionen los animales.
- Todas las instalaciones deben contar con adecuada iluminación y ventilación, principalmente la mesa de exhibición, la báscula y la gradería de subasta.
- Las áreas de básculas, gradería y mesa de exhibición deberán ser cubiertas.
- Las básculas deberán exhibir los pesos por medio de pantalla electrónica (luminosa), proporcionando en forma visible para el público, el peso en pie del animal. Esta información deberá ser visible en el momento de la pesa.
- Las instalaciones deberán contar con áreas de parqueo, amplias y seguras permitiendo un manejo adecuado de los camiones y evitando posibles accidentes. Terminantemente prohibido lavar los camiones y botar desechos en los predios de la subasta.
- Deberá existir un servicio eficiente de manejo de caja para la recepción y pago de las transacciones comerciales.
- La mesa de exhibición deberá brindar seguridad de contención, así como buena visibilidad.
- Los corrales, pisos e instalaciones destinados a los animales, deberán reunir todas las condiciones que permitan un eficaz control sanitario y desinfección, a fin de evitar la difusión de enfermedades transmisibles de los animales.
- El estiércol que se saque del recinto de la subasta deberá ser sometido previamente a un proceso de tratamiento.
- El personal de la subasta deberá estar visible y debidamente identificado, de la misma manera contar con formación adecuada para el manejo de los animales.
- Contar con una oficina para la ubicación del médico Veterinario Regente.

Pasos para la creación de Subastas.

- Motivación, Capacitación y Sensibilización.
- Análisis continúo del Entorno.
- Análisis continúo del Sector.
- Integración y Compromisos
- Simulaciones.

- Gestión gerencial y Mercadotécnica.
- Propuesta de Creación de la Subasta.
- Acción estratégica de Creación.

2.6.6. Análisis de la Subasta Ganadera en Ecuador.

Actualmente no se realizan subastas ganaderas en nuestro país, cabe recalcar que si se llevan a cabo ferias ganaderas en Tulcán, San Gabriel, Santo Domingo de los Tsachilas, El Carmen, Ambato, Ibarra.

En Santo Domingo de los Tsachilas y El Carmen se han realizado subastas virtuales de ganado, que consisten en mostrar en una pantalla gigante y a través de video los animales que van a ser subastados, mientras se da a conocer su genética, peso, edad y la base con la que empiezan a ser vendidos, hasta que se adjudica al mejor postor.

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 cinco 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 Econ. 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 explicaremos detalladamente cada uno de los anteriores elementos.

3.1.1. Framework de Desarrollo JSF 2.0.

Es un framework web MVC[23] 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[24] 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[25].

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.

[23] **MVC.-** Modelo Vista Controlador
[24] **JSF.-** Java Server Faces
[25] **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[26] 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[27] 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.

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

[26] **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.
[27] **HTML.-** Lenguaje de Marcado de Hipertexto.

contenedor EJB, que no es más que un ambiente especializado (runtime[28]) 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[29] (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[30] 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 sobrescribir las anotaciones.

Una anotación transforma un simple POJO en un EJB.

Ilustración 4. Estructura de un EJB

[28]**Runtime.-**Se encarga de integrar java con el sistema.
[29]**Pojo.-** Utilizada por programadores Java para enfatizar el uso de clases simples y que no dependen de un framework en especial
[30]**Metadata.-** Es información que describe el contenido de un archivo u objeto.

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[31] y Stateful[32], 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[33]):** 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 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 utilización de una capa intermedia en la comunicación entre el productor y el consumidor

[31] **Stateless.-** Se caracteriza por no conservar los atributos durante la sesión.
[32] **Stateful**.- Conserva el estado del EJB durante toda la sesión.
[33] **MDBs.-** Es un bean empresarial que permite a las aplicaciones J2EE procesar mensajes de forma asíncrona.

del mensaje. En EJB 3, esta capa se llama MOM[34] (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.

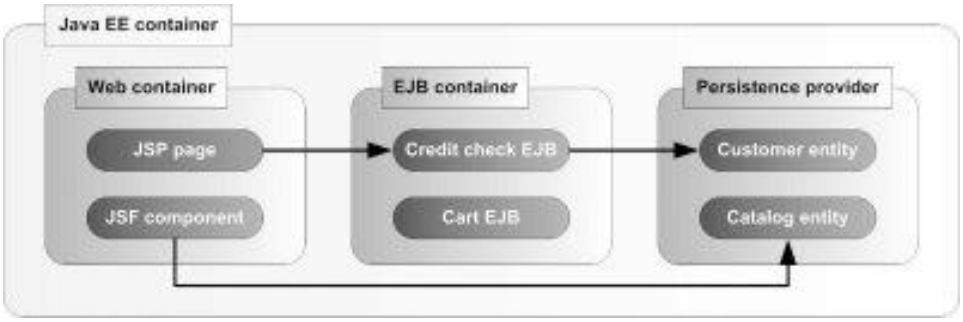


Ilustración 5. Contenedor de EJBs

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.

[34]**MOM.-** Es la infraestructura de software o hardware que soporta el envío y recepción de mensajes entre sistemas distribuidos.

¿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[35], como se hacía en EJB 2. 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 reusando 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[36] 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

[35] **JNDI.-** La Interfaz de Nombrado y Directorio Java (Java Naming and Directory Interface) es una Interfaz de Programación de Aplicaciones (API) de Java para servicios de directorio.
[36] **Commit.-** Proceso que se realiza en una base de datos para terminar satisfactoriamente una transacción.

transacción haría un rollback[37] 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.

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[38].

[37] **Rollback.-** Proceso que se realiza en una base de datos cuando ha ocurrido algún problema durante la transacción.
[38] **RichFaces.-** Es una biblioteca de código abierto basada en Java para crear aplicaciones web con Ajax.

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.

Arquitectura.

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

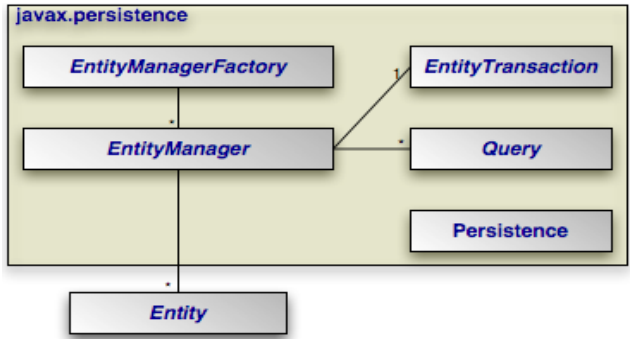


Ilustración 6. Arquitectura de JPA

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 mostraremos la arquitectura que JPA tiene para el control de las excepciones:

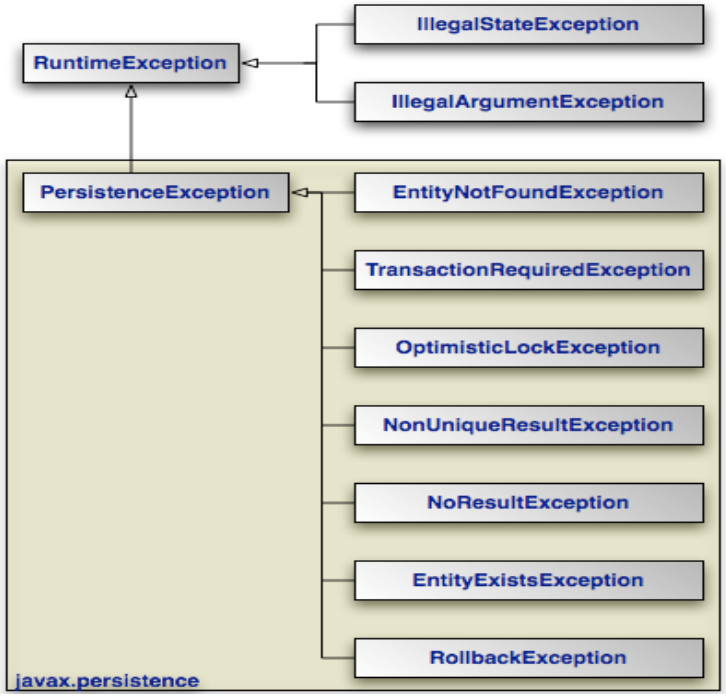


Ilustración 7. Excepciones en JPA

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 agregamos la anotación @Entity a la clase que va a ser persistente, hay 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, lo hacemos utilizando la anotación @Id. Más adelante veremos 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 declaremos 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. Debemos evitar tener mezclado anotaciones tanto en propiedades como en métodos.

Declaración de Tablas.

Para definir los datos de una tabla de base de datos, utilizamos la anotación @Table a nivel de la declaración de la clase, esta anotación nos permite definir el nombre de tabla con la que se está mapeando la clase, el esquema, el catálogo, constraints de unicidad.

Si no se utiliza esta anotación y se utiliza 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. Usted 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 queva 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 elcódigo 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 tablaprincipal 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 explicamos la funcionalidad del JBOSS:

Ilustración 8. 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[39] 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.

[39] **BSD**.- Es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution).

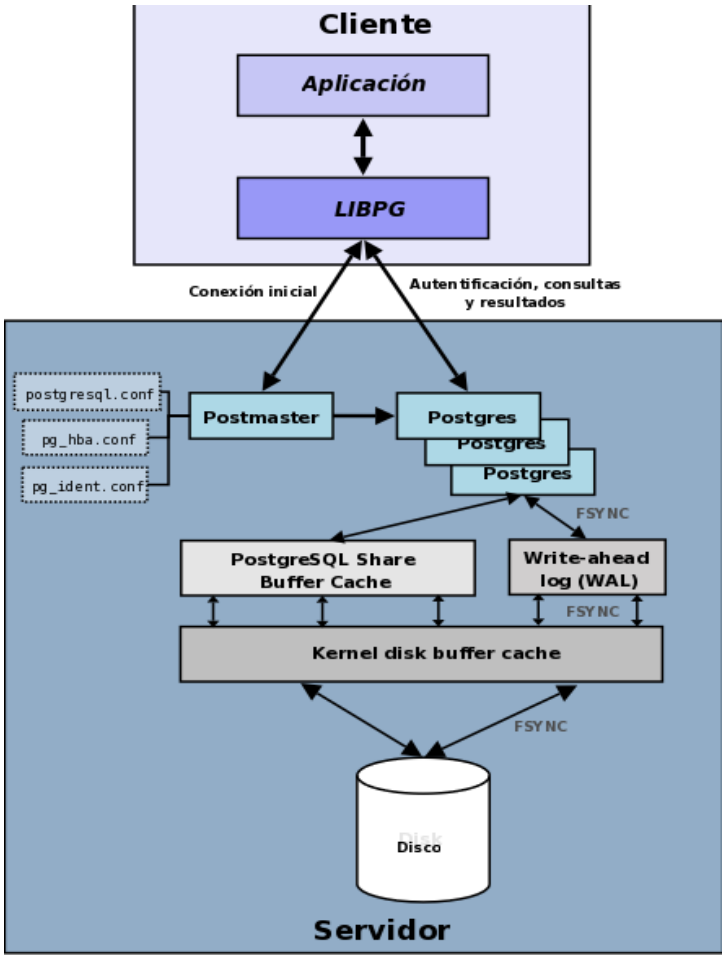


Ilustración 9. Arquitectura PostgreSQL

- *Aplicación cliente*: Esta es la aplicación cliente que utiliza PostgreSQL como administrador de bases de datos. La conexión puede ocurrir vía TCP/IP[40]o sockets[41] locales.
- *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 tres ficheros principales de configuración utilizados por PostgreSQL, postgresql.conf, pg_hba.conf y pg_ident.conf.

[40]**TCP/IP.-** Es un modelo de descripción de protocolos de red.
[41]**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.

- *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 diez y seis 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.

A continuación presentamos 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[42]) en numerosos lenguajes de programación, entre otros PL/pgSQL[43], PL/Perl[44], etc.
- Bloques anónimos de código de procedimientos (sentencias DO[45]).
- Numerosos tipos de datos y posibilidad de definir nuevos tipos. Además de los tipos estándares en cualquier base de datos, tenemos disponibles, entre otros, tipos geométricos, de direcciones de red, de cadenas binarias, UUID[46], XML, matrices.
- 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.

[42] **Stored Procedures.-** Procedimientos Almacenados.
[43] **PL/pgSQL.-** (Procedural Language/PostgreSQL Structured Query Language) es un lenguaje imperativo provisto por el gestor de base de datos PostgreSQL.
[44] **PL/Perl.-** Es un lenguaje de procedimientos con el apoyo del PostgreSQL.
[45] **DO.-** Bloques anónimos de código de procedimientos.
[46] **UUID.-** La intención de los UUID es habilitar, a los sistemas distribuidos, un identificador de información único sin una importante coordinación central.

- 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 tenemos:

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 10. Límites físicos de PostgreSQL

3.2. Análisis de submódulos del Sistema.

Una vez definida la tecnología de desarrollo, se analizará los módulos a desarrollar detenidamente enfocándonos en la parte de Licitación y Subasta de la Subasta ganadera.

A continuación se muestra un diagrama generalizado del módulo de Licitación y Subasta de la Subasta ganadera.

Ilustración 11. Análisis de los módulos del Sistema

3.2.1. Submódulo de Registro.

El submódulo deberá gestionar los datos de los:

- Vendedores.
- Animales.

Permitirá el registro, consulta y modificación de datos.

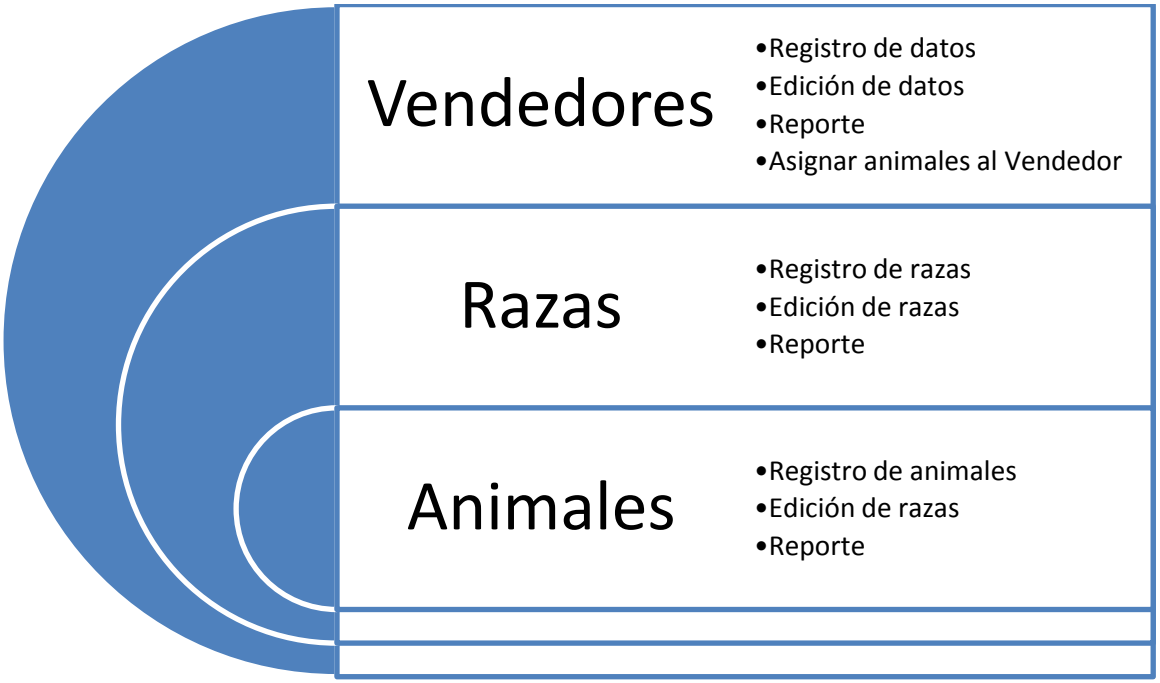


Ilustración 12. Estructura del Módulo de Registro

3.2.2. Submódulo de Licitación y Subasta.

Este submódulo maneja la interacción directa entre compradores y vendedores. Entre las principales funciones están:

- Ingresar los datos de los compradores.
- Llevar a cabo la Subasta de ganado, permitiendo que los compradores hagan sus propuestas respecto al animal que se esté subastando en ese momento. La elección de la oferta ganadora será en base a la mejor oferta ofrecida por el comprador, es decir, la de mayor precio.

A continuación se presenta un esquema del submódulo.

Ilustración 13. Estructura del Módulo de Licitación y Subasta

3.2.3. Submódulo de de Reportes y Estadísticas.

Permitirá generar los reportes del sistema a nivel estratégico, es decir, para promover el cambio de los objetivos, procesos, productos, servicios y relaciones con el entorno para ayudar a la institución a obtener ventajas competitivas.

La institución necesita los reportes que constan en el siguiente esquema:

Ilustración 14. Estructura del Módulo de Reportes

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 permita al programador 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.

El Objetivo de la planificación se logra mediante un proceso de descubrimiento de la información que lleve a estimaciones razonables.

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.

Se ha desarrollado 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 de Licitación y Subasta en la Feria Ganadera que permita sistematizar el proceso de subasta de ganado que se desea implementar en dicha feria, se determina crear un software que tendrá como finalidad optimizar el proceso de licitación y subasta del ganado.

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 nos permite ver los siguientes módulos:

- Módulo de Registro.
- Modulo de Licitación y Subasta.
- Módulo de Reportes y Estadísticas.

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 entre el módulo de Licitación y Subasta con el módulo Financiero.

4.1.5. 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 posee 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

4.1.6. 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 una la distribución de tiempos y el número de iteraciones de cada fase (para las fases de construcción y transición es solo una aproximación muy preliminar).

Fase	Nro. Iteraciones	Duración
Fase de Inicio	1	5 semanas
Fase de Elaboración	1	8 semanas
Fase de Construcción	5	24 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 desarrollará los requisitos del modelo desde la perspectiva del usuario. Se tiene que claramente especificar 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 reléase 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 reléase 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 presentarán el reléase final para distribución, asegurando una implantación y cambio del sistema previo de manera adecuada, incluyendo el entrenamiento de los usuarios.

	El hito que marca el fin de esta fase, incluye la entrega de toda la documentación del proyecto con los manuales de instalación y todo el material de apoyo al usuario, la finalización del entrenamiento de los usuarios y el empaquetamiento del producto.

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.

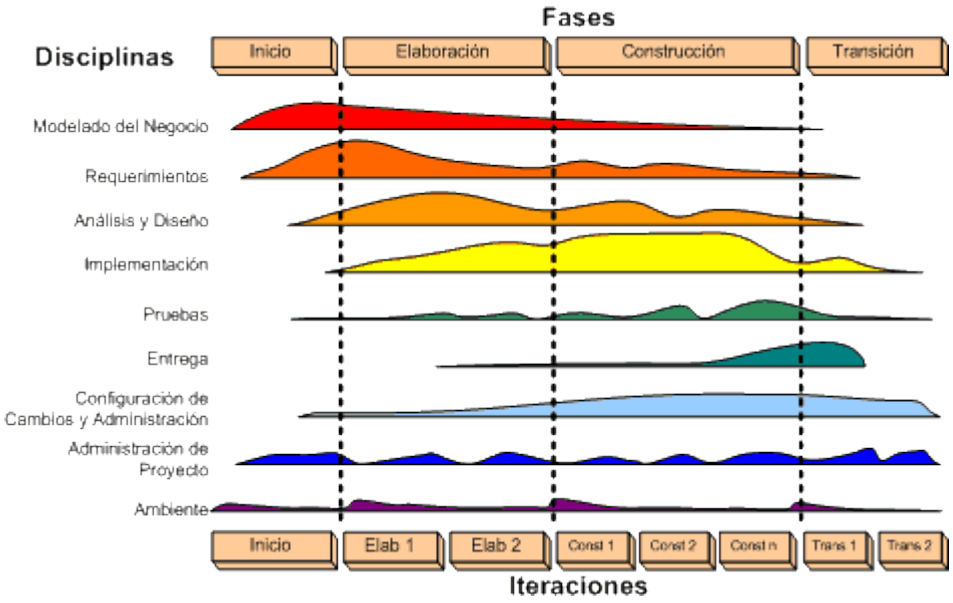


Ilustración 15. Las fases y disciplinas del RUP

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	

Fuente: Propia

Tabla 1: Roles y Responsabilidades – Fase de elaboración

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 tendrá 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 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.

Como se verá en los párrafos siguientes, se debe describir el sistema con un alto grado de detalle, especificando no sólo la función de cada elemento, sino su comunicación con los restantes componentes. También deberá ser necesario especificar qué usuario será el responsable de cada uno de estos componentes y hasta dónde llegará su responsabilidad. Cabe considerar que el análisis realizado debe servir para poder, más adelante, hacer el diseño de la solución adoptada. Es por ello por lo que dentro de este apartado se utilizará la enumeración de requisitos del estudio de viabilidad para establecer las necesidades exactas del mismo y cuál será el intercambio de información (si la hay) entre los diferentes componentes del sistema. Se debe remarcar la importancia de este apartado, ya que una definición inadecuada del sistema de intercambio de información puede dar lugar a un diseño que no contemple todas las necesidades de la empresa.

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, facturación 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 adecuadas 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 (CUPS), 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, la 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:

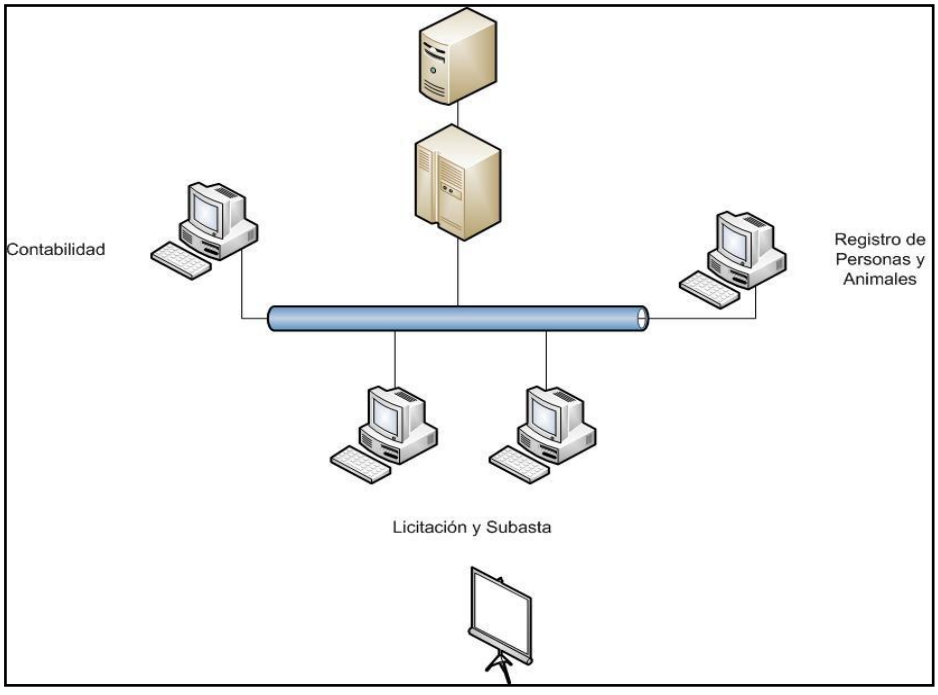


Ilustración 16. Requisitos tecnológicos del Sistema

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. Asimismo, 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 quedarán anotadas 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 con-fianza, 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

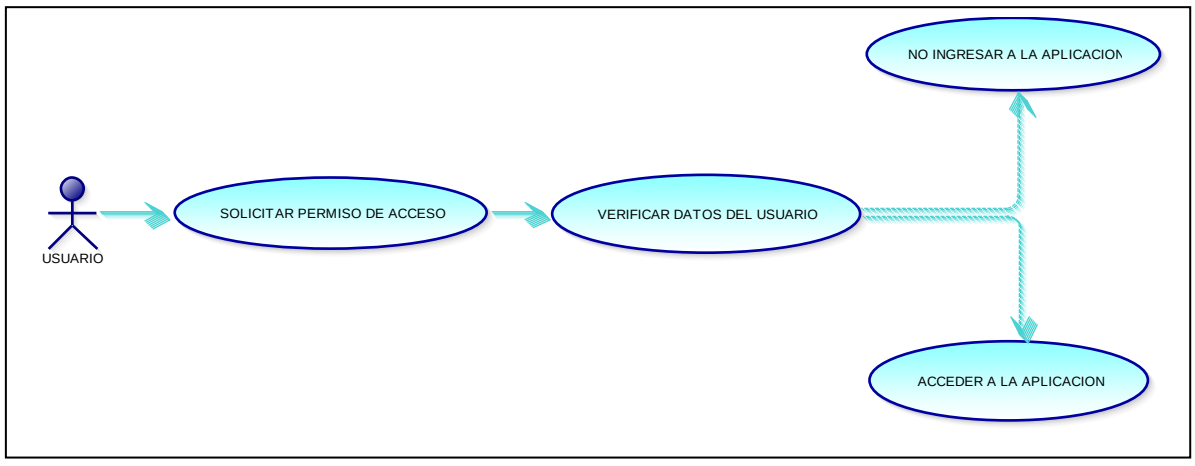


Ilustración 17. Caso de Uso: Control de Acceso al Sistema

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 login 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.

Pos condiciones.

- Ninguna.

Gestionar los datos de los vendedores.

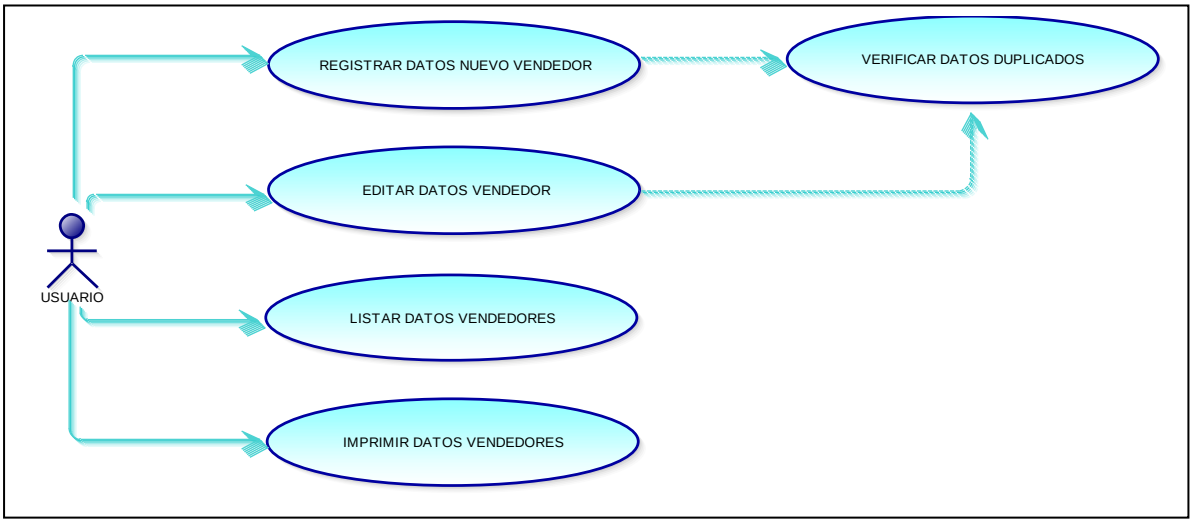


Ilustración 18. Caso de Uso: Gestionar los datos de los vendedores

Caso de Uso: Registro de los datos de un vendedor.

Descripción breve.

Es el proceso con el cual se registran los datos de un vendedor.

Flujo básico de eventos.

- El usuario solicita al sistema comenzar el proceso de registro de los datos de un nuevo vendedor
- El sistema permite al usuario registrar los siguientes datos del nuevo vendedor: Código del Vendedor, nº de la cédula, nombres, apellidos, fecha de nacimiento, género, dirección, correo, celular, teléfono, fecha de registro.
- El usuario comprueba que los datos del nuevo vendedor coincidan con los de su documento de identificación.
- El usuario proporciona los datos requeridos y solicita al sistema que los almacene.
- El sistema almacena los datos proporcionados e informa al usuario de que el proceso ha terminado con éxito

Flujos alternativos.

- El usuario abandona el sistema al cerrar sesión en cualquier momento.
- Si el sistema detecta que el nuevo vendedor ya está registrado, el sistema informa de la situación al usuario permitiéndole modificar los datos proporcionados, a continuación este caso de uso continúa.

Precondiciones.

- Tener acceso al sistema.
- Estar logueado en el sistema.
- El solicitante es una persona vendedora de animales de la feria y tiene su documento de identificación disponible.

Pos condiciones.

La información del vendedor está registrada.

Caso de Uso: Edición de los datos de un vendedor.

Descripción breve.

Es el proceso con el cual se editan los datos de un vendedor.

Flujo básico de eventos.

- El usuario solicita al sistema comenzar el proceso de modificación de los datos de un vendedor.
- El sistema muestra los siguientes datos correspondientes al vendedor a modificar: Código del Vendedor, nº de la cédula, nombres, apellidos, fecha de nacimiento, género, dirección, correo, celular, teléfono, fecha de registro, nacionalidad.
- El sistema permite al usuario modificar los siguientes datos: nº de la cédula, nombres, apellidos, fecha de nacimiento, género, dirección, correo, celular, teléfono.
- El usuario modifica los datos que el sistema le permite y solicita al sistema que los almacene.
- El sistema modifica los datos correspondientes al vendedor e informa al usuario de que el proceso ha terminado con éxito.

Flujos alternativos.

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

Precondiciones.

- Tener acceso al sistema
- Estar logueado en el sistema.
- El solicitante es una persona vendedora de animales de la feria y tiene su documento de identificación disponible.

Postcondiciones.

La información del vendedor está actualizada.

Caso de Uso: Listar los datos de un vendedor.

Descripción breve.

Es el proceso con el cual se listan los datos de un vendedor.

Flujo básico de eventos.

- El usuario solicita al sistema comenzar el proceso de consulta de los datos de un vendedor.
- El sistema solicita que se identifique al vendedor.
- El usuario proporciona los datos de identificación al sistema.
- El sistema muestra la siguiente información asociada al vendedor: nombres, apellidos, dirección, celular, teléfono, fecha de registro.
- Si el usuario solicita la impresión de los datos, el sistema imprime los datos del vendedor.

Flujos alternativos.

- El usuario abandona el sistema al cerrar sesión en cualquier momento.
- Si el sistema no tiene registrado ningún vendedor con la identificación proporcionada, el sistema comunica al usuario la situación, a continuación este caso de uso termina.

Precondiciones.

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

Pos condiciones.

Ninguna.

Gestionar los datos de los animales.

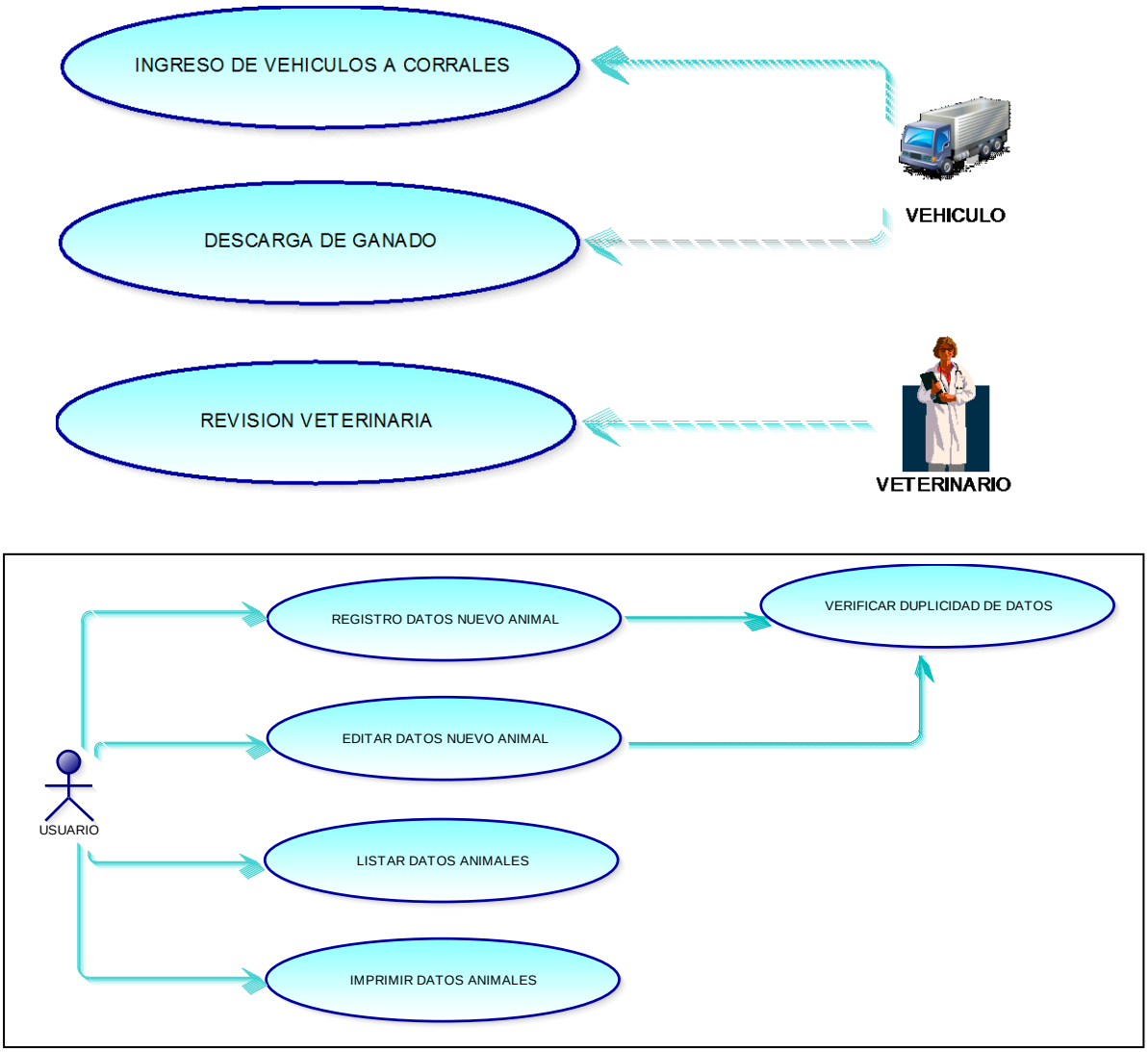


Ilustración 19. Caso de Uso: Gestionar los datos de los animales

Caso de Uso: Registro de los datos de un animal.

Descripción breve.

Es el proceso con el cual se registran los datos de un animal.

Flujo básico de eventos.

- El usuario solicita al sistema comenzar el proceso de registro de los datos de un nuevo animal.

- El sistema permite al usuario registrar los siguientes datos del nuevo animal: Código del Animal, raza, nro. de marcado del animal, color, calidad, condición, fecha de registro.
- El usuario comprueba que los datos del nuevo animal coincidan con los de de la ficha correspondiente a cada animal.
- El usuario proporciona los datos requeridos y solicita al sistema que los almacene.
- El sistema almacena los datos proporcionados e informa al usuario de que el proceso ha terminado con éxito.

Flujos alternativos.

- El usuario abandona el sistema al cerrar sesión en cualquier momento.
- Si el sistema detecta que el nuevo animal ya está registrado, el sistema informa de la situación al usuario permitiéndole modificar los datos proporcionados, a continuación este caso de uso continúa.

Precondiciones.

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

Pos condiciones.

La información del animal está registrada.

Caso de Uso: Editar datos de un Animal.

Descripción breve.

Es el proceso con el cual se editan los datos de un animal.

Flujo básico de eventos.

- El usuario solicita al sistema comenzar el proceso de modificación de los datos de un animal.

- El sistema muestra los siguientes datos correspondientes al animal a modificar: Código del Animal, raza, nro. de marcado del animal, color, peso, calidad, condición, precio, fecha de registro.
- El sistema permite al usuario modificar los siguientes datos: raza, nro. de marcado del animal, color, peso, calidad, condición, precio, fecha de registro.
- El usuario modifica los datos que el sistema le permite y solicita al sistema que los almacene.
- El sistema modifica los datos correspondientes al animal e informa al usuario de que el proceso ha terminado con éxito.

Flujos alternativos.

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

Precondiciones.

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

Pos condiciones.

La información del animal está actualizada.

Caso de Uso: Listar los datos de un Animal.

Descripción breve.

Es el proceso con el cual se listan los datos de un animal.

Flujo básico de eventos.

- El usuario solicita al sistema comenzar el proceso de listar los datos de un animal.
- El sistema muestra la siguiente información asociada al animal: raza, nro. de marcado del animal, color, calidad, condición, fecha de registro.
- Si el usuario solicita la impresión de los datos, el sistema imprime los datos del animal.

Flujos alternativos.

- El usuario abandona el sistema al cerrar sesión en cualquier momento.
- Si el sistema no tiene registrado ningún animal, el sistema comunica al usuario la situación, a continuación este caso de uso termina.

Precondiciones.

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

Pos condiciones.

Ninguna.

Gestionar los datos de los compradores.

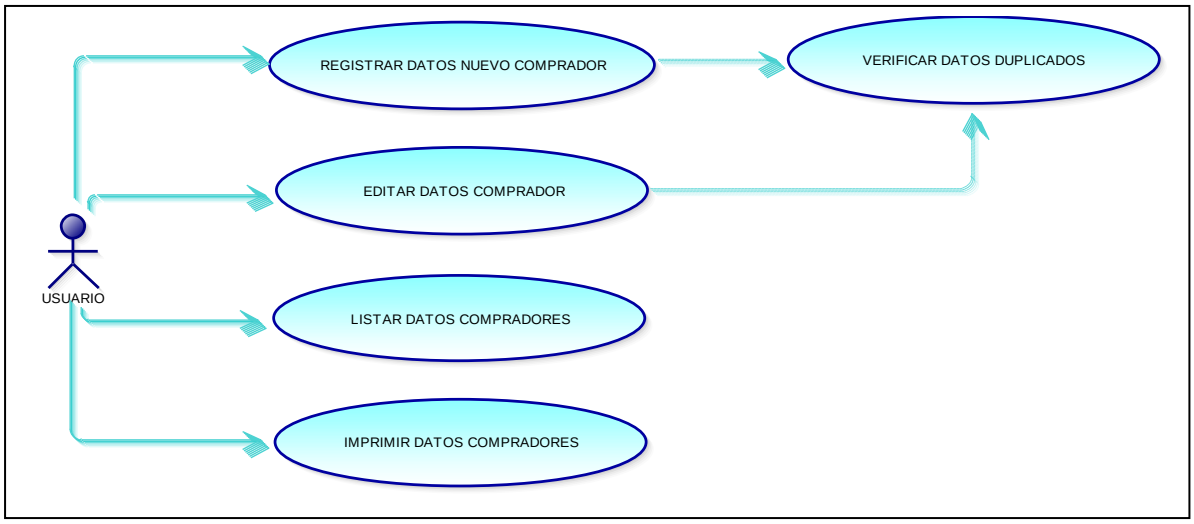


Ilustración 20. Caso de Uso: Gestionar los datos de los compradores.

Caso de Uso: Registro de los datos de un comprador.

Descripción breve.

Es el proceso con el cual se registran los datos de un comprador.

Flujo básico de eventos.

- El usuario solicita al sistema comenzar el proceso de registro de los datos de un nuevo comprador.
- El sistema permite al usuario registrar los siguientes datos del nuevo comprador: número de cédula, nombres, apellidos, dirección, correo, celular, teléfono, fecha de registro.
- El usuario comprueba que los datos del nuevo comprador coincidan con los de su documento de identificación.
- El usuario proporciona los datos requeridos y solicita al sistema que los almacene.
- El sistema almacena los datos proporcionados e informa al usuario de que el proceso ha terminado con éxito.

Flujos alternativos.

- El usuario abandona el sistema al cerrar sesión en cualquier momento.
- Si el sistema detecta que el nuevo comprador ya está registrado, el sistema informa de la situación al usuario permitiéndole modificar los datos proporcionados, a continuación este caso de uso continúa.

Precondiciones.

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

Pos condiciones.

La información del comprador está registrada.

Caso de Uso: Edición de los datos de un comprador.

Descripción breve.

Es el proceso con el cual se editan los datos de un comprador.

Flujo básico de eventos.

- El usuario solicita al sistema comenzar el proceso de edición de los datos de un comprador.
- El sistema muestra los siguientes datos correspondientes al comprador a modificar: número de cédula, nombres, apellidos, dirección, correo, celular, teléfono, fecha de registro.
- El sistema permite al usuario modificar los siguientes datos: número de cédula, nombres, apellidos, dirección, correo, celular, teléfono.
- El usuario modifica los datos que el sistema le permite y solicita al sistema que los almacene.
- El sistema modifica los datos correspondientes al vendedor e informa al usuario de que el proceso ha terminado con éxito.

Flujos alternativos.

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

Precondiciones.

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

Postcondiciones.

La información del comprador está actualizada.

Caso de Uso: Listar los datos de un comprador.

Descripción breve.

Es el proceso con el cual se listan los datos de un comprador.

Flujo básico de eventos.

- El usuario solicita al sistema comenzar el proceso de consulta de los datos de un comprador.
- El sistema solicita que se identifique al comprador.
- El usuario proporciona los datos de identificación al sistema.
- El sistema muestra la siguiente información asociada al comprador: nombres, apellidos, dirección, celular, teléfono.
- Si el usuario solicita la impresión de los datos, el sistema imprime los datos del comprador.

Flujos alternativos.

- El usuario abandona el sistema al cerrar sesión en cualquier momento.
- Si el sistema no tiene registrado ningún comprador con la identificación proporcionada, el sistema comunica al usuario la situación, a continuación este caso de uso termina.

Precondiciones.

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

Postcondiciones.

Ninguna.

Gestionar la Licitación y Subasta de Animales.

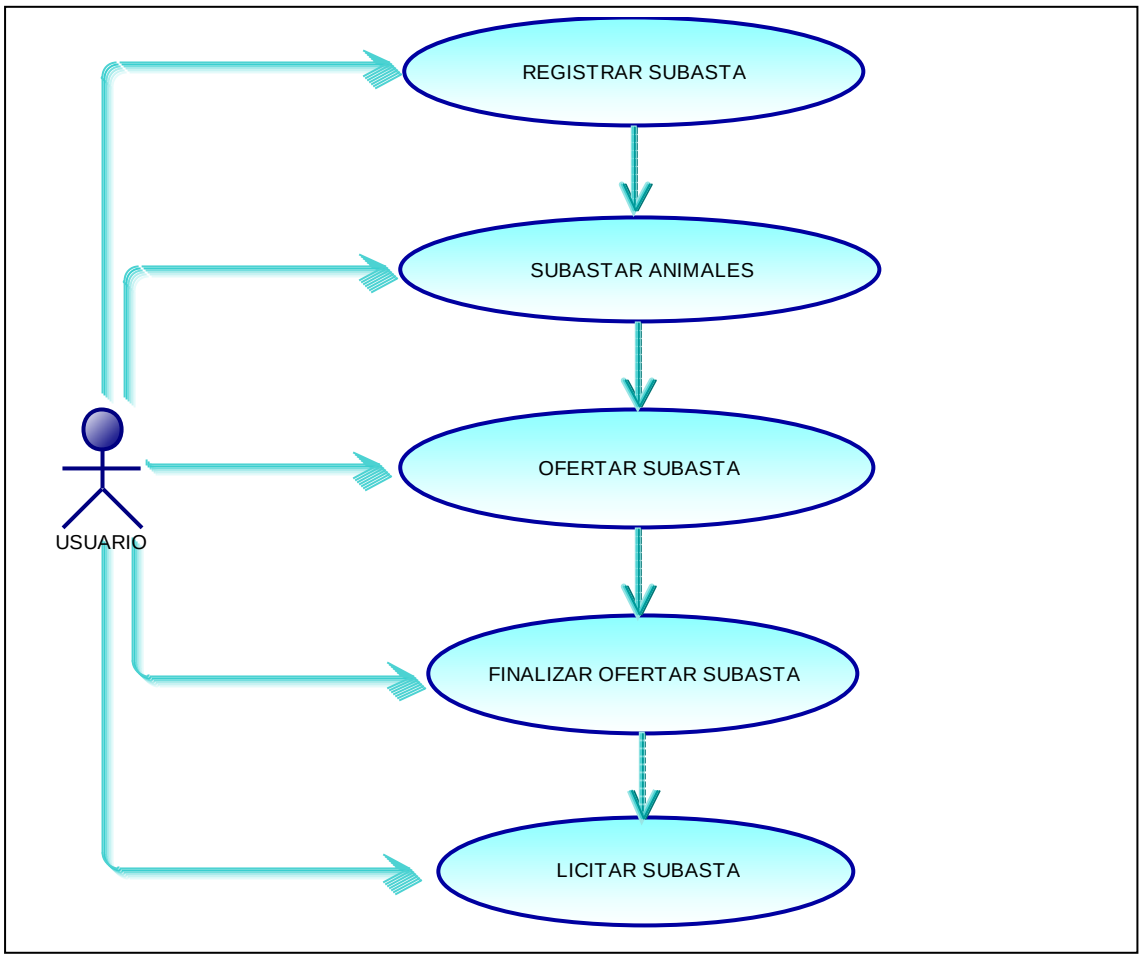


Ilustración 21. Caso de Uso: Gestionar la Licitación y Subasta de Animales

Caso de Uso: Subasta de Animales.

Descripción breve.

Es el proceso con el cual se realiza la subasta de animales.

Flujo básico de eventos.

- La persona encargada de subastar los animales inicia el proceso de subasta, da a conocer a los compradores las siguientes características del animal: número de marcado del animal, raza, color, peso, el precio.
- Se registra la subasta en el sistema. Ver caso de uso: **Registrar Subasta.**

- A continuación los compradores dan sus ofertas. Ver caso de uso: **Ofertar Subasta.**
- Se procede a licitar la subasta a cada comprador. Ver caso de uso: **Licitar Subasta.**

Flujos alternativos.

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

Precondiciones.

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

Postcondiciones.

Se realiza la subasta de animales.

Caso de Uso: Registrar Subasta.

Descripción breve.

Es el proceso con el cual se registra la subasta de animales.

Flujo básico de eventos.

- El usuario solicita al sistema comenzar el proceso de ingreso de subastas.
- El sistema permite al usuario registrar los siguientes datos de la nueva subasta: los datos del animal a subastar, el vendedor, el peso, el precio, el precio base, hora inicio, hora fin, observación.
- El precio base se utilizará para validar que la primera oferta sea mayor a este valor. Este precio es obligatorio ser ingresado, pero no puede ser igual a cero.
- El usuario proporciona los datos requeridos y solicita al sistema que los almacene.
- El sistema almacena los datos proporcionados e informa al usuario de que el proceso ha terminado con éxito.

Flujos alternativos.

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

Precondiciones.

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

Postcondiciones.

Se registra una nueva subasta en el sistema.

Caso de Uso: **Ofertar Subasta*.***

Descripción breve.

Es el proceso con el cual se registran manualmente las ofertas de la subasta de animales.

Flujo básico de eventos.

- Una vez que la subasta inicia el comprador procede a dar el precio a ofertar para el animal que desee. Las ofertas se realizarán en forma individual por cada animal.
- Cuando un comprador realiza una oferta escuchan los otros participantes de la subasta, tanto vendedores, compradores y administradores. Una persona encargada lleva el registro de estas ofertas.
- El comprador puede realizar otra oferta hasta que la subasta llegue a su fin. En este momento ya no se podrán realizar subastas. Ver caso de uso: ***Finalizar Ofertar Subasta*.**

Flujos alternativos.

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

Precondiciones.

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

Postcondiciones.

Se registra manualmente la oferta realizada.

Caso de Uso: Finalizar Ofertar Subasta.

Descripción breve.

Es el proceso con el cual se finaliza la oferta en la subasta.

Flujo básico de eventos.

- Se controlará durante el curso de la subasta el tiempo que falta para que termine la subasta. Cuando el tiempo finalice se informará a los compradores para que no continúen proporcionando sus ofertas.
- Se registran la mejor oferta de la subasta en el sistema.
- Se indicará que la subasta terminó y que está pendiente de licitarse.

Flujos alternativos.

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

Precondiciones.

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

Postcondiciones.

La subasta queda en estado pendiente de licitar y los compradores no pueden continuar realizando ofertas para la subasta de ese animal.

Caso de Uso: Licitar Subasta.

Descripción breve.

Es el proceso con el cual se realiza la licitación de la subasta luego de que termina el tiempo pautado para la misma.

Flujo básico de eventos.

- Luego de completarse el tiempo pautado para la subasta el usuario solicita al sistema comenzar el proceso licitar subastas, entonces se muestra al administrador un formulario para realizar la licitación.
- En el registro manual de ofertas de cada animal consta de las ofertas de los compradores ordenadas de mayor a menor para el animal en cuestión.
- El administrador completa para cada animal la licitación a cada comprador en el sistema.
- El sistema registra automáticamente la licitación al aceptar la mejor oferta.
- Es obligatorio tener que Licitar un animal al comprador que dio la mejor oferta. La licitación de cada animal es independiente del resto de los animales.
- El sistema valida la información antes de registrar una licitación.

Flujos alternativos.

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

Precondiciones.

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

Postcondiciones.

La subasta pasa a estado cerrado. La subasta se licita al comprador ganador.

Caso de Uso: **Ofertar Subasta Web*.***

Descripción breve.

Es el proceso con el cual se registran en el sistema las ofertas ingresadas por cada comprador.

Flujo básico de eventos.

- Una vez que la subasta inicia el comprador procede a ingresar en el sitio el precio de la oferta para el animal que desee. Las ofertas se realizarán en forma individual por cada animal.
- El sistema valida que la oferta ingresada sea mayor a la oferta anterior ingresada para ese animal. En caso que la oferta sea la primera, el precio base determinado será utilizado para tener en cuenta en la validación.
- Si la oferta se valida entonces se registra la oferta dada por el comprador. En la pantalla del comprador se visualizará el animal involucrado en la subasta con la mejor oferta hasta ese momento.
- El comprador puede realizar otra oferta hasta que la subasta llegue a su fin. En este momento ya no se podrán realizar subastas. Ver caso de uso: ***Finalizar Ofertar Subasta Web*.**

Flujos alternativos.

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

Precondiciones.

- Tener acceso al sitio de subastas web.
- El comprador debe estar registrado en sistema.

Postcondiciones.

Se registra en el sistema la oferta realizada.

Caso de Uso: Finalizar Ofertar Subasta Web.

Descripción breve.

Es el proceso con el cual se finaliza la oferta en la subasta.

Flujo básico de eventos.

- Se controlará durante el curso de la subasta el tiempo que falta para que termine la subasta. Cuando el tiempo finalice, los compradores ya no podrán continuar proporcionando sus ofertas.
- Se indicará que la subasta terminó y que está pendiente de licitarse.

Flujos alternativos.

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

Precondiciones.

- Tener acceso al sitio de subastas web.
- El comprador debe estar registrado en sistema.

Postcondiciones.

La subasta queda en estado pendiente de licitar y los compradores no pueden continuar realizando ofertas para la subasta de ese animal.

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.

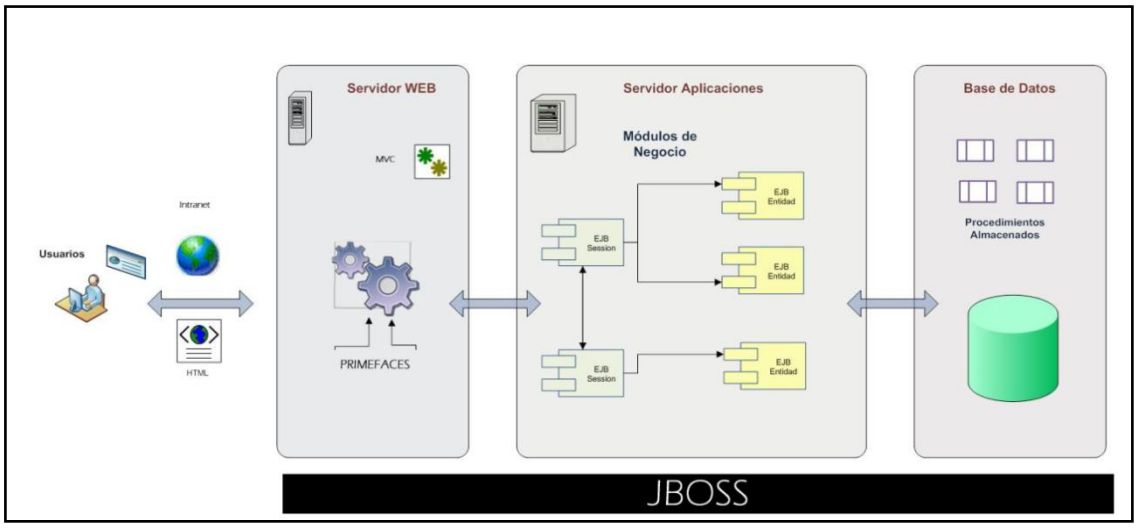


Ilustración 22. Arquitectura del Sistema

4.3.2. Modelo Físico de Datos.

La base de datos está integrada con el sistema de 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 su comprensión.

Módulo de Registro.

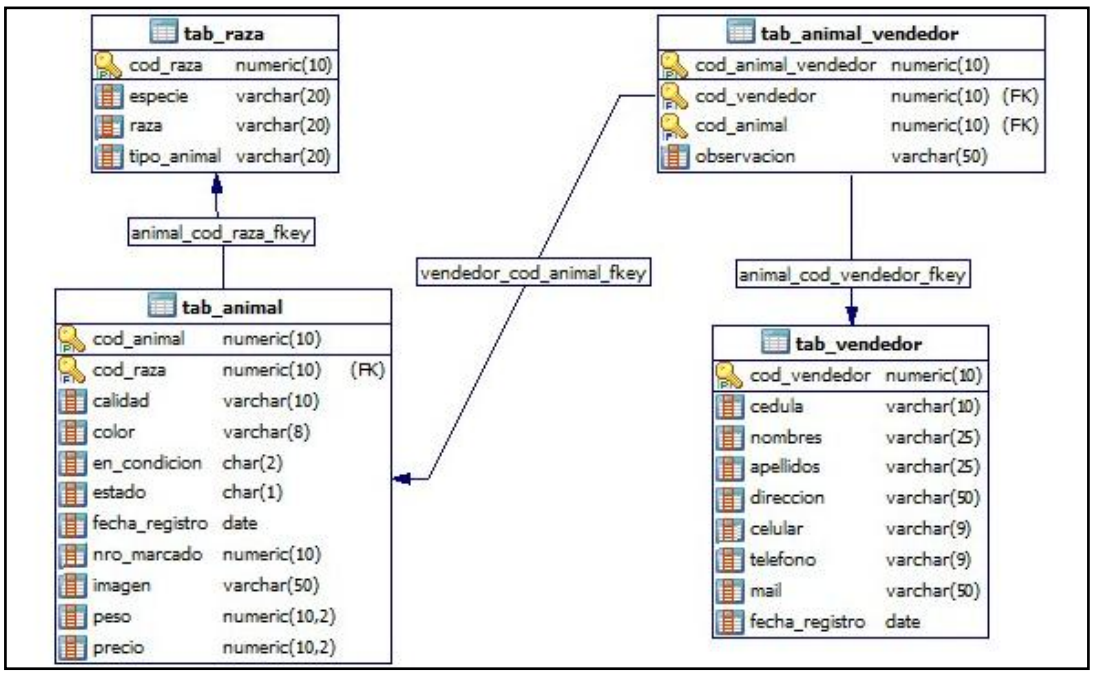


Ilustración 23. Entidad-Relación (Módulo de Registro)

Módulo de Licitación y Subastas.

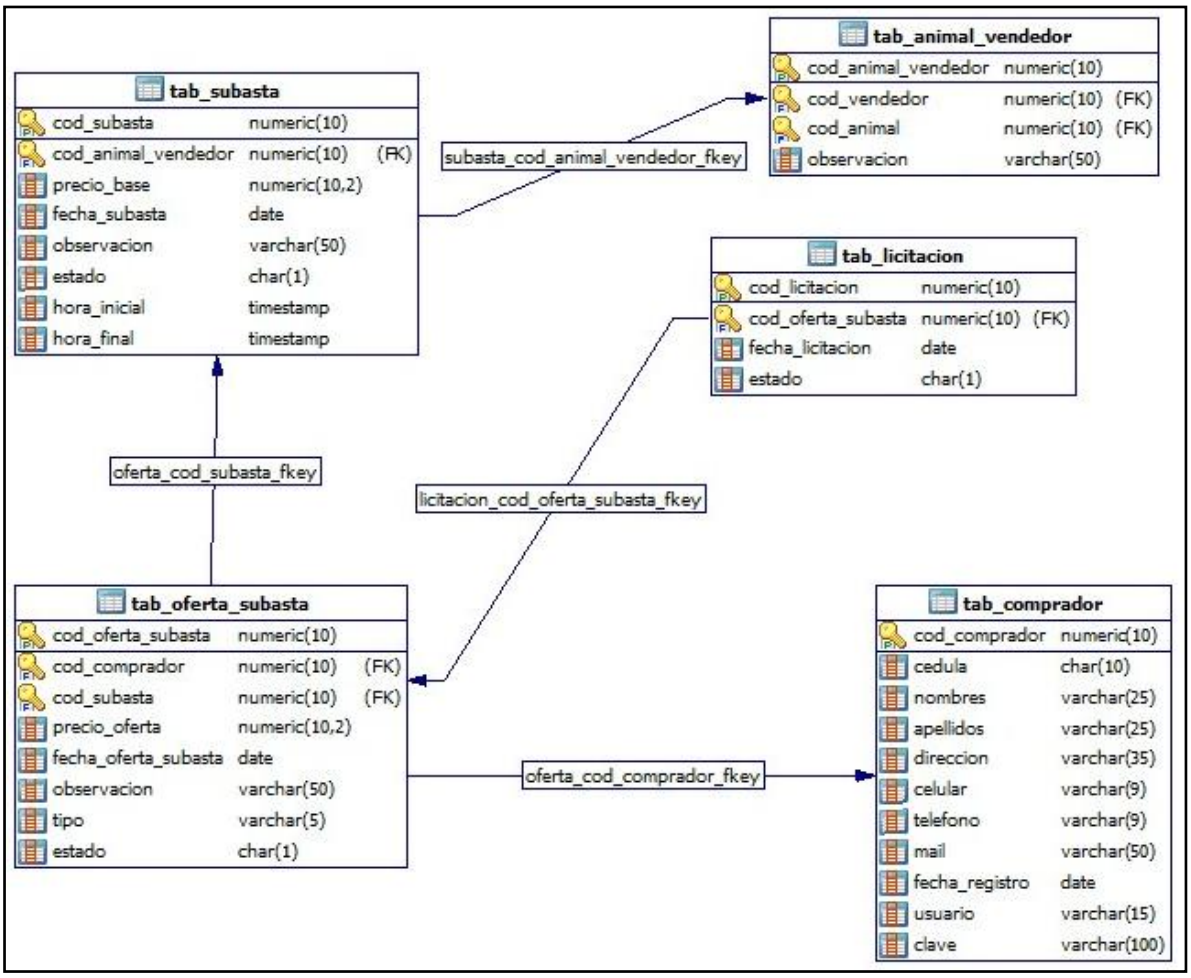


Ilustración 24. Entidad-Relación (Módulo de Licitación y Subastas)

Seguridad.

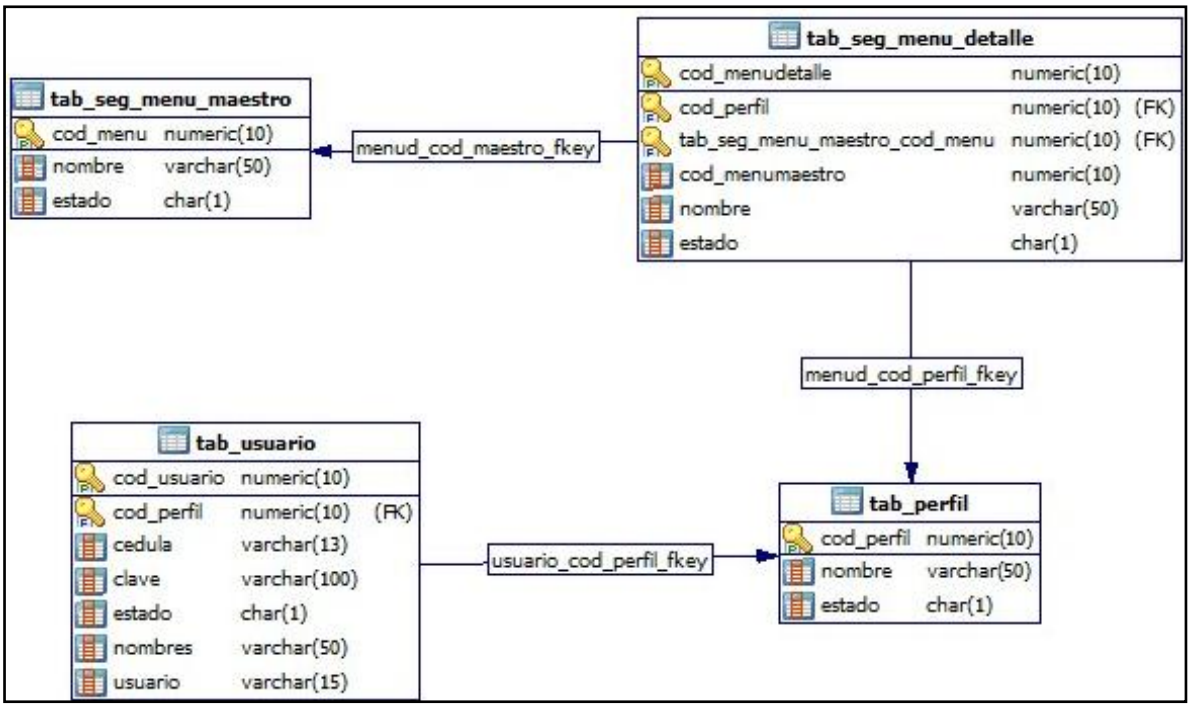


Ilustración 25. Entidad-Relación (Seguridad)

Diccionario de Datos.

A continuación presentamos el diccionario de datos para el sistema de licitación y subastas.

Módulo de Seguridad.

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	character varying (50)	Registra el nombre del menú.	-
estado	character(1)	Almacena el estado del menú si esta 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	character varying(50)	Registra el nombre del menú.	-
estado	character (1)	Almacena el estado del menú si esta 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	character varying(50)	Registra el nombre del perfil.	-
estado	character(1)	Almacena el estado del perfil si esta 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	character varying(13)	La cédula o RUC del usuario	-
usuario	character varying(15)	El nombre de usuario con el cual accede a la aplicación	-
clave	character varying(100)	Clave del usuario encriptada en la aplicación	-
estado	character (1)	Almacena el estado del usuario si esta activo o no	-

Módulo de Registro.

TAB_VENDEDOR		En esta tabla almacenamos los datos de los vendedores de animales.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_vendedor	numeric(10)	Registra un número que se irá incrementando mediante una secuencia	-
cedula	character varying(10)	Registra la cedula del vendedor.	-
nombres	character varying(25)	Registra los nombres del vendedor.	-
apellidos	character varying(25)	Registra los apellidos del vendedor.	-
direccion	character varying(50)	Registra la dirección del vendedor.	-
celular	character varying(9)	Registra el número de celular del vendedor.	-

telefono	character varying(9)	Registra el número de teléfono del vendedor.	-
mail	character varying(50)	Registra el correo electrónico del vendedor.	-
fecha_registro	date	Registra la fecha de registro del vendedor.	-

TAB_RAZA		En esta tabla almacenamos las razas de los animales.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_raza	numeric(10)	Registra un número que se irá incrementando mediante una secuencia	-
especie	character varying(20)	Registra la especie del animal.	-
raza	character varying(20)	Registra la raza del animal.	-
tipo_animal	character varying(20)	Registra el tipo de animal.	-

TAB_ANIMAL		En esta tabla almacenamos los datos de los animales.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_animal	numeric(10)	Registra un número que se irá incrementando mediante una secuencia	-
cod_raza	numeric(10)	Registra el código de la raza a la cual pertenece el animal.	tab_raza
calidad	character varying(10)	Registra la calidad del animal.	-
color	character varying(8)	Registra el color del animal.	-
en_condicion	character(2)	Registra la condición del animal (SI o NO).	-

estado	character(1)	Almacena el estado del animal.	-
fecha_registro	date	Registra la fecha de registro del animal.	-
nro_marcado	numeric(10)	Registra el número de marcado del animal.	-
imagen	character varying(50)	Registra la ruta de la imagen del animal	-
peso	numeric(10,2)	Registra el peso del animal	-
precio	numeric(10,2)	Registra el precio del animal	

TAB_ANIMAL_VENDEDOR		En esta tabla almacenamos los datos de los animales asignados a cada vendedor.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_animal_vendedor	numeric(10)	Registra un número que se irá incrementando mediante una secuencia	-
cod_vendedor	numeric(10)	Registra el código del vendedor.	tab_vendedor
cod_animal	numeric(10)	Registra el código del animal.	tab_animal
observacion	character varying(50)	Registra la observación.	-

Módulo de Licitación y Subasta.

TAB_SUBASTA		En esta tabla almacenamos los datos de la subasta.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_subasta	numeric(10)	Registra un número que se irá incrementando	-

		mediante una secuencia	
cod_animal_vendedor	numeric(10)	Registra el código de la tabla animal vendedor el cual pertenece a la subasta.	tab_animal_vendedor
precio_base	numeric(10,2)	Registra el precio base de la subasta.	-
fecha_subasta	date	Registra la fecha de la subasta.	-
hora_inicial	timestamp without time zone	Almacena la hora de inicio de la subasta.	-
hora_final	timestamp without time zone	Almacena la hora final de la subasta.	-
observacion	character varying(50)	Almacena la observación de la subasta.	-
estado	character (1)	Almacena el estado de la subasta (abierta, cerrada).	-

TAB_COMPRADOR		En esta tabla almacenamos los datos del comprador.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_comprador	numeric(10)	Registra un número que se irá incrementando mediante una secuencia	-
cedula	character varying(10)	Registra la cedula del comprador.	-
nombres	character varying(25)	Registra los nombres del comprador.	-
apellidos	character varying(25)	Registra los apellidos del comprador.	-

direccion	character varying(50)	Registra la dirección del comprador.	-
celular	character varying(9)	Registra el número de celular del comprador.	-
telefono	character varying(9)	Registra el número de teléfono del comprador.	-
mail	character varying(50)	Registra el correo electrónico del comprador.	-
fecha_registro	date	Registra la fecha de registro del comprador.	-
usuario	character varying(15)	Registra el usuario para la subasta web	-
clave	character varying(100)	Registra la clave para la subasta web	-

TAB_OFERTA_SUBASTA		En esta tabla almacenamos los datos de la oferta de la subasta.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_oferta_subasta	numeric(10)	Registra un número que se irá incrementando mediante una secuencia	-
cod_comprador	numeric(10)	Registra el código del comprador	tab_comprador
cod_subasta	numeric(10)	Registra el código de la subasta	tab_subasta
precio_oferta	numeric(10,2)	Registra el precio de la oferta de la subasta.	-
fecha_oferta_subasta	date	Registra la fecha de la oferta de la subasta.	-
observacion	character varying(50)	Almacena la observación de la subasta.	-

tipo	character varying(5)	Almacena si la oferta es la mejor de la subasta.	-
estado	character(1)	Almacena el estado de la subasta (abierta, cerrada).	-

TAB_LICITACION		En esta tabla almacenamos los datos de la oferta de la subasta.	
Columna	**Tipo de Dato**	**Descripción**	**Referencia**
cod_licitacion	numeric(10)	Registra un número que se irá incrementando mediante una secuencia	-
cod_oferta_subasta	numeric(10)	Registra el código de la oferta de la subasta.	tab_ oferta_subasta
fecha_licitacion	date	Registra la fecha de la licitación.	-
estado	character(1)	Almacena el estado de la licitación.	-

4.3.3. Diagrama Global de Paquetes.

Registro:

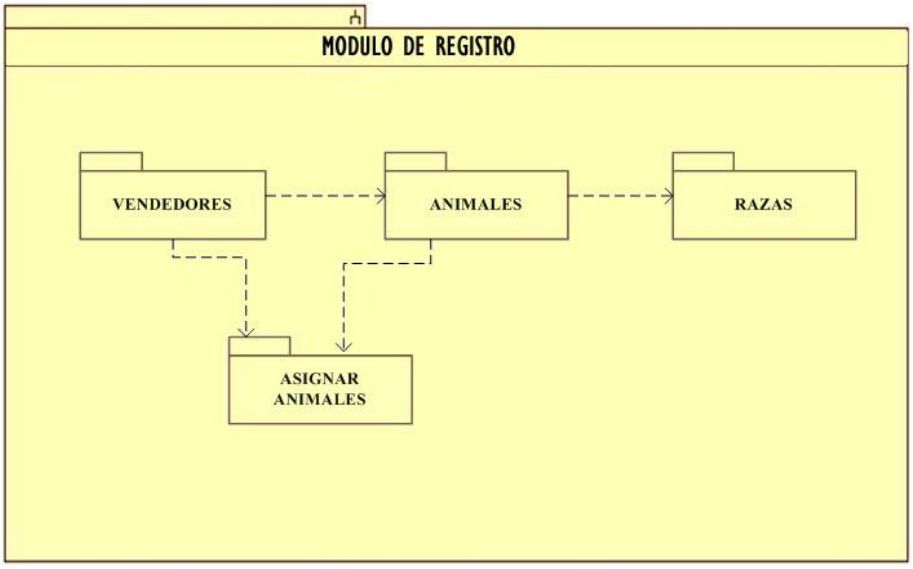


Ilustración 26. Diagrama de Paquetes: Módulo de Registro

Licitación y Subasta:

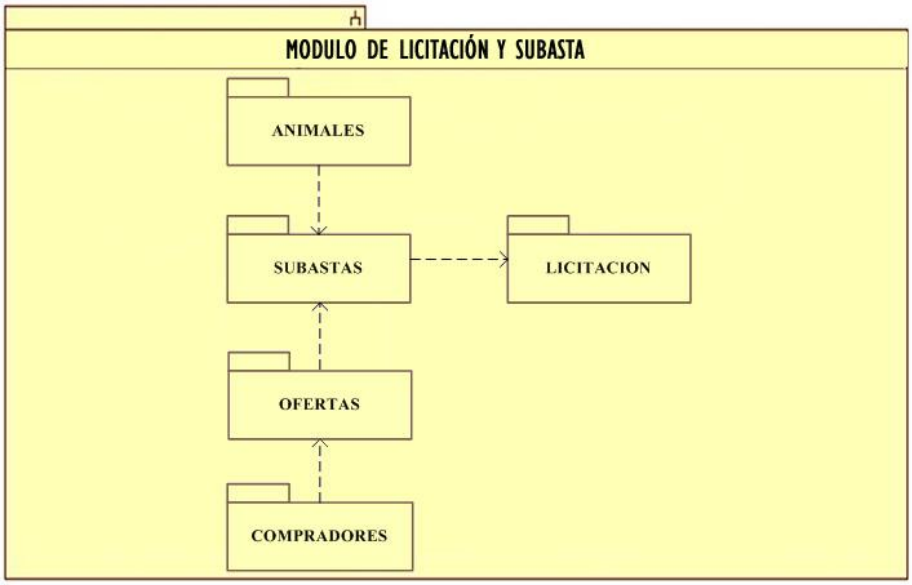


Ilustración 27. Diagrama de Paquetes: Módulo de Licitación y Subasta

4.4. Desarrollo del Sistema.

4.4.1. Planificación de las Actividades de Desarrollo.

Al momento de realizar la planificación tomamos 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.

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 presentamos una estructura estandarizada para toda la aplicación:

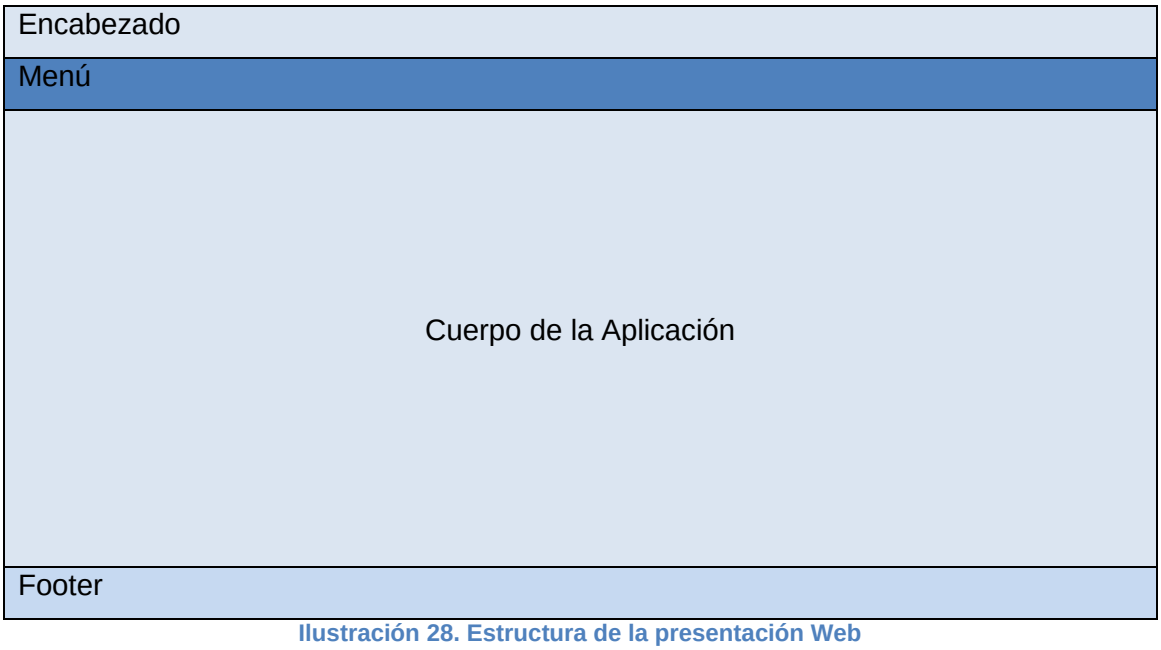


Ilustración 28. Estructura de la presentación Web

- ***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

Detalle de la Arquitectura de la Aplicación.

De acuerdo a las capas de la Arquitectura Orientada a Servicios mi aplicación consta de tres 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="subganaEJB">

       <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. Siúnicamente se especifica uno, no habrá; que incluir su nombre cuando se recupere elEntityManager (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>
```

Un nombre JNDI es un nombre que se 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 laanotación @Id. Al igual que en las bases de datos relacionales, la llave primaria hace delentity un objeto único. Cuando se requiere de una llave compuesta se puede usar unaclase como llave primaria compuesta. Para este tipo de claves, se utilizan las anotacionesjavax.persistence.EmbededId y javax.persistence.IdClass.
- Los entities no pueden ser accedidos directamente, deben ser desplegados y usadoslocalmente en un contenedor (desde beans de sesión, o de mensajería, oincluso desde cualquier componente web). De cualquier manera, el código cliente debeprimero recuperar una instancia particular del entity desde el contexto de persistencia ocrear una y adicionarla a dicho contexto.

Ejemplo: Entidad Perfil.

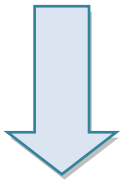
```
package com.tesis.modelo.dto;

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_GENERATOR")

	@Column(name="cod_perfil")

	private long codPerfil;


	private String estado;


	private String nombre;


	//bi-directional many-to-one association to TabSegMenuDetalle

	@OneToMany(mappedBy="tabPerfil")

	private Set<TabSegMenuDetalle> tabSegMenuDetalles;


	//bi-directional many-to-one association to TabUsuario

	@OneToMany(mappedBy="tabPerfil")

	private Set<TabUsuario> tabUsuarios;


	public TabPerfil() {

	}
```

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 especifíca 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;
	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.

```
//IMPLEMENTACION DE LA CLASE
@Stateless
public class PerfilDAOImpl implements PerfilDAO {
//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:

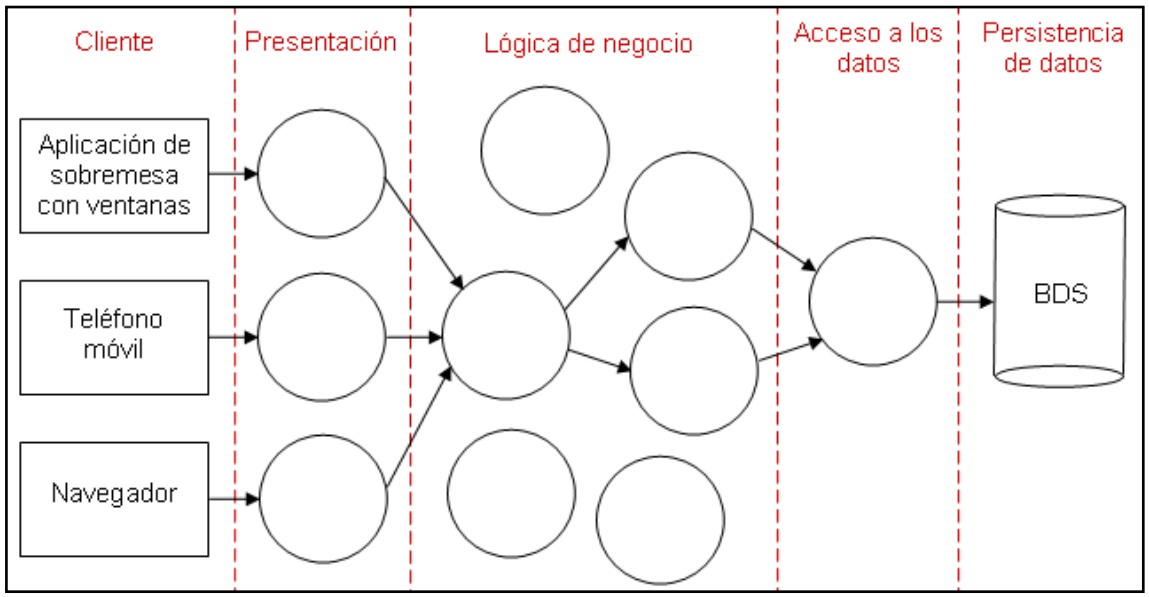


Ilustración 29. Análisis de la Capa de Negocio

4.4.3. Factores de Calidad.

Para el desarrollo de la aplicación nos basamos en los factores de calidad McCall, a continuación se describe las iteraciones aplicadas al 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 nos 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 al 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 utilizamos 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 procede 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 módulo financiero del Sistema Informático de Subasta ganadera.

La aplicación desarrollada 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.

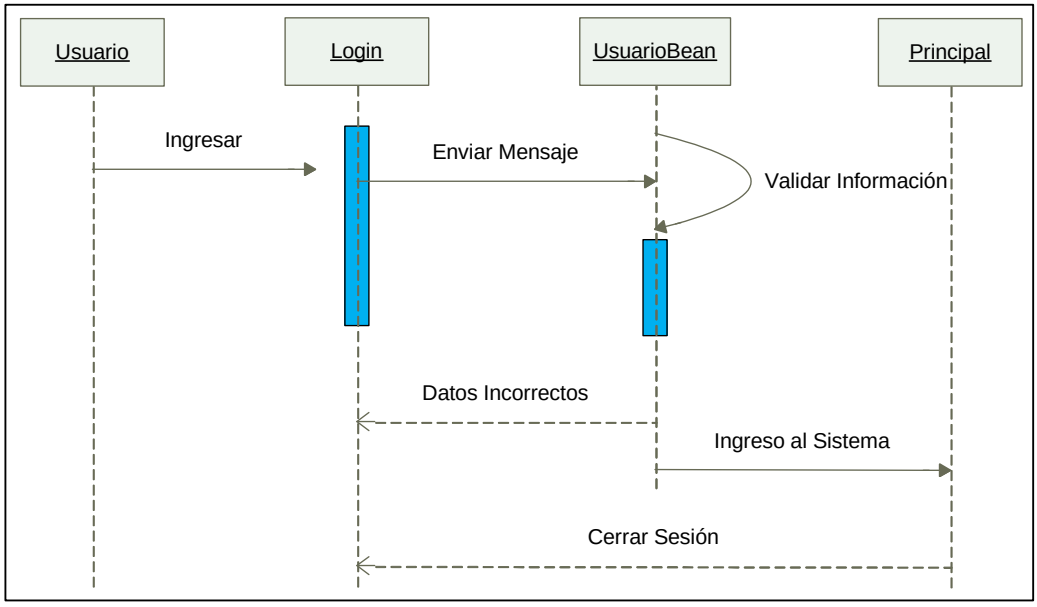


Ilustración 30. Diagrama de Secuencia.- Acceso al sistema

Diagrama de Secuencia.- Registro de Actores.

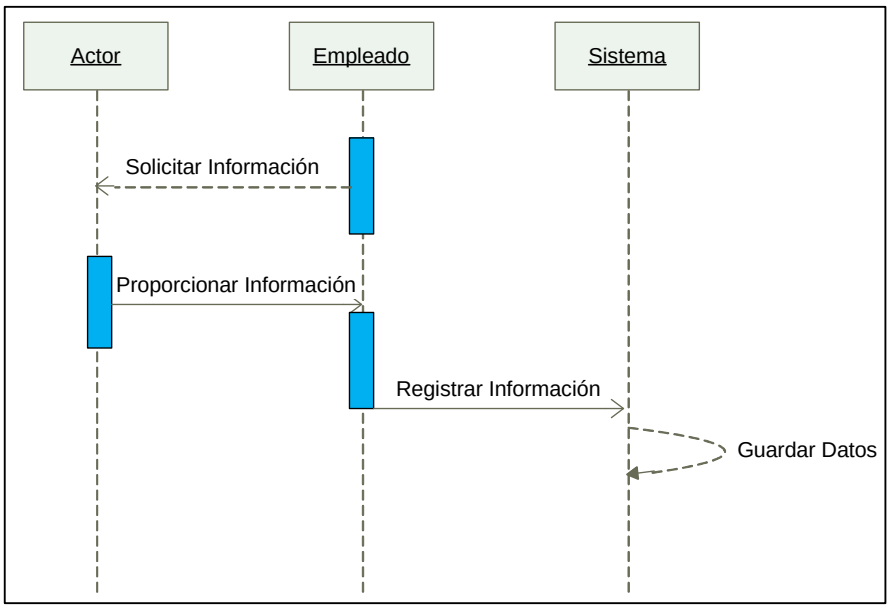


Ilustración 31. Diagrama de Secuencia.- Registro de Actores

Diagrama de Secuencia.- Subasta de Animales.

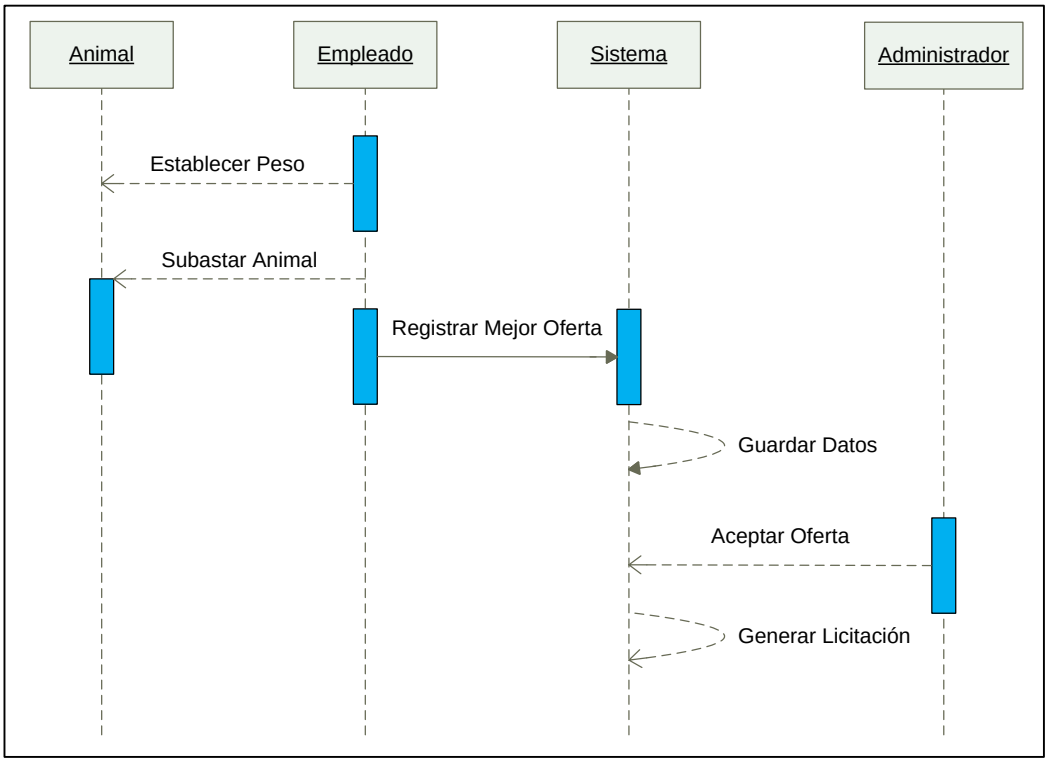


Ilustración 32. Diagrama de Secuencia.- Subasta de Animales.

***Diagrama de Secuencia*.-** Subasta Web de Animales.

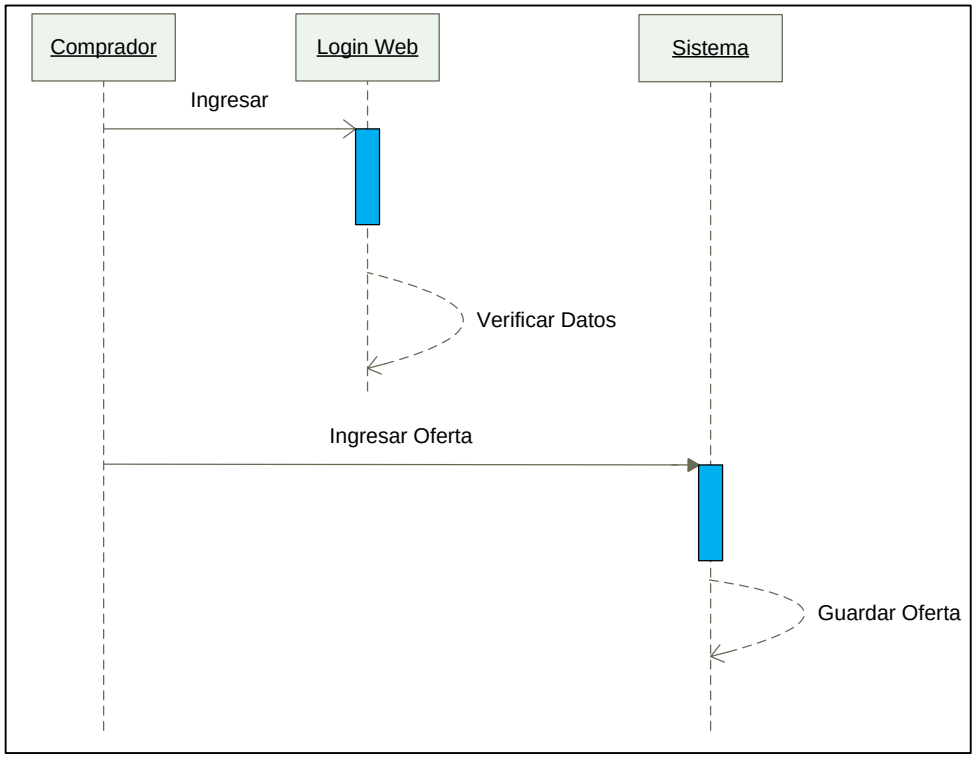


Ilustración 33. Diagrama de Secuencia.- Subasta Web de Animales

4.5.2. Diagrama de Actividades

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

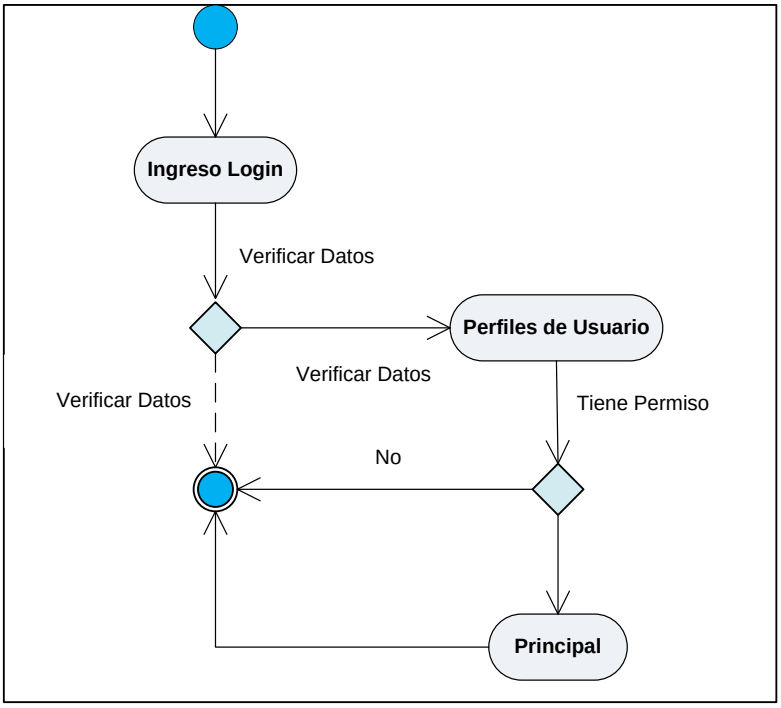


Ilustración 34. Diagrama de Actividad.- Acceso al Sistema

Diagrama de Actividad.- Subasta de Animales.

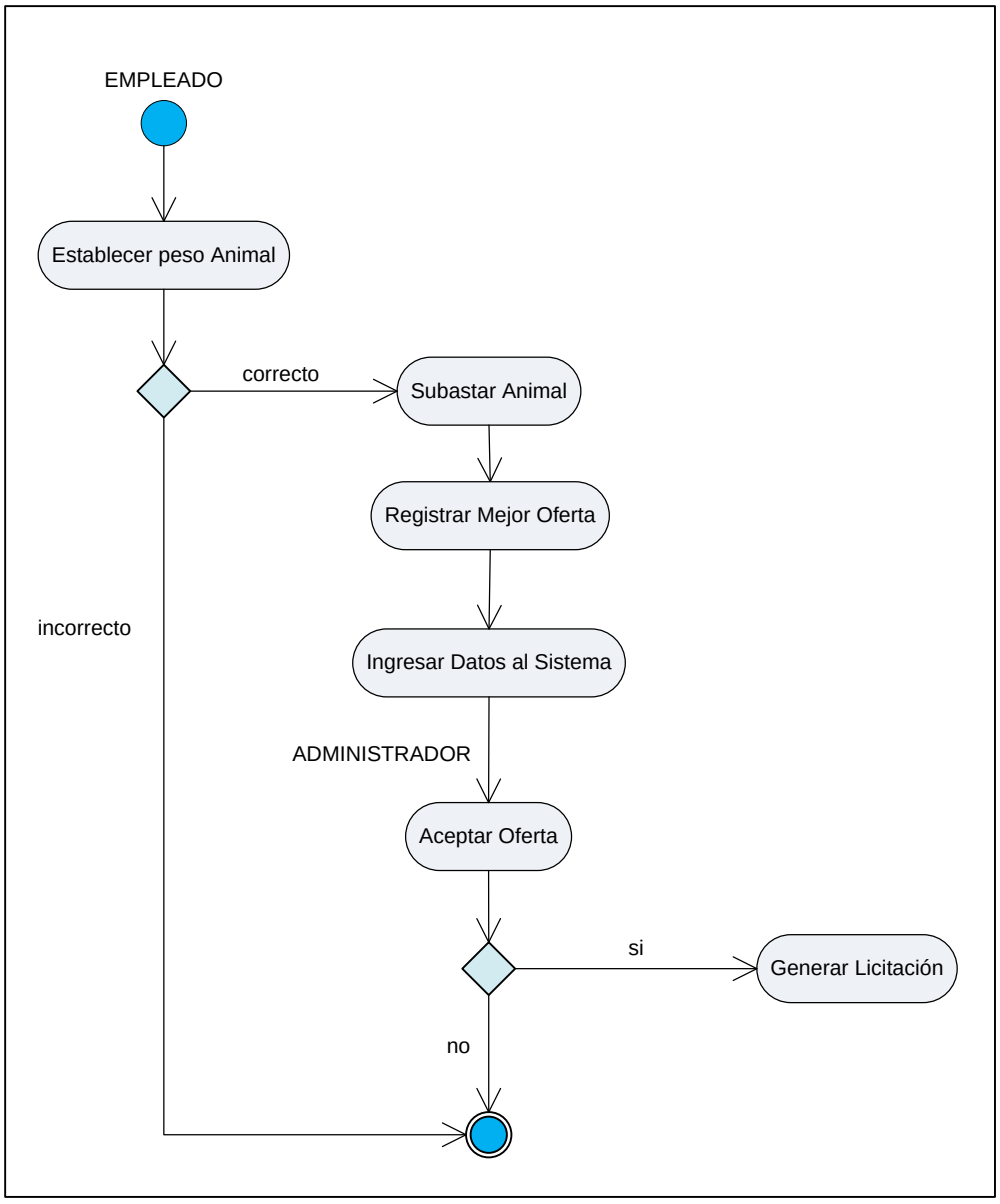


Ilustración 35. Diagrama de Actividad.- Subasta de Animales

Diagrama de Actividad.- Subasta Web de Animales.

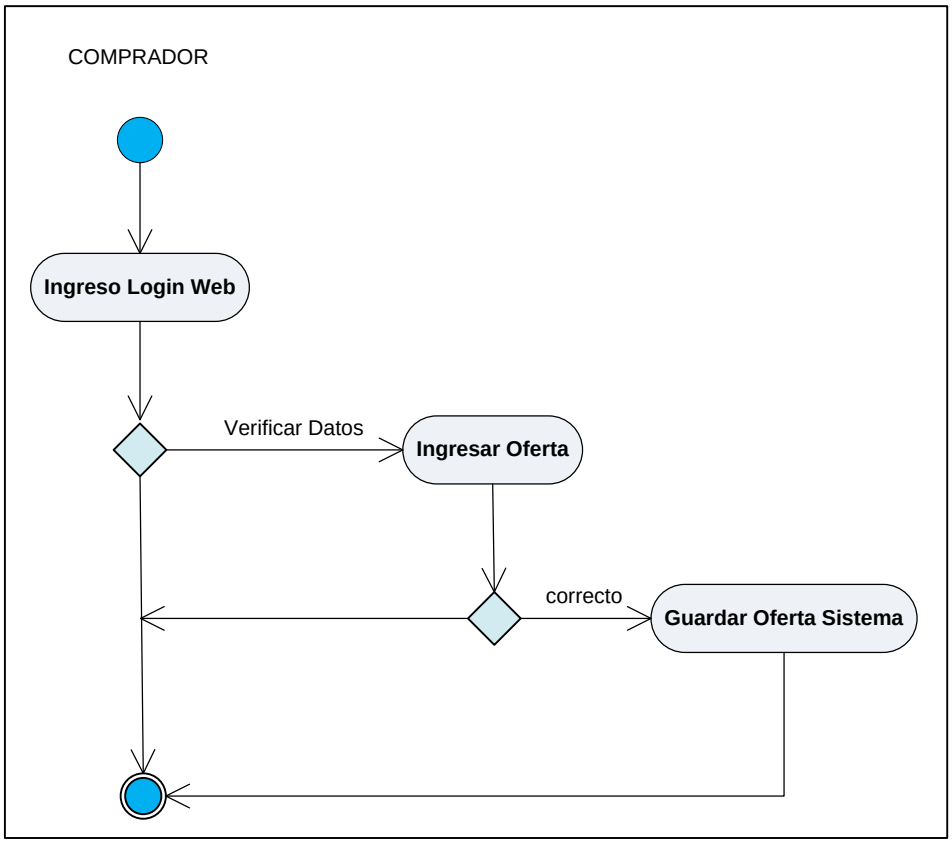


Ilustración 36. Diagrama de Actividad.- Subasta Web de Animales

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/principal.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 desde el navegador digitar 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/index.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 de Nuevo Vendedor/Comprador.

Descripción:

Para registrar los datos de los vendedores y compradores el usuario deberá solicitar la cédula de identidad. Luego deberá ingresar los datos del nuevo comprador o vendedor, se deberá verificar que los datos sean correctos, en caso de cumplir con este requisito se procederá a guardar la información.

Comprobación:

Ingresar los datos de un nuevo vendedor/comprador, por ejemplo registrar un vendedor con número de cédula XXXX, nombres XXXX, apellidos XXXX, entonces el sistema deberá verificar que no exista en la base de datos, caso contrario la aplicación generará un error.

Condiciones de ejecución:

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

Entrada:

- Datos básicos del nuevo vendedor/comprador.

Resultado esperado:

- Registro normal de los datos del vendedor ó comprador.
- 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: Subasta de Animales.

Descripción:

En este caso de pruebas el usuario administrador de Subastas, registra la subasta en el sistema, luego inicia el proceso de subasta dando a conocer las características del animal que se está subastando, posteriormente los compradores empiezan a dar sus ofertas, se ingresa la mejor oferta en el sistema. El administrador acepta la mejor oferta para cada animal y se procede a la licitación de animales.

Comprobación:

En ésta prueba se registraron 25 animales, se realizo la subasta de cada uno de ellos, se ingresaron las mejores ofertas para cada uno. El administrador acepto las ofertas, las subastan se cerraron y automáticamente el sistema realizo el proceso de licitación de animales.

Condiciones de ejecución:

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

Entrada:

- Animales.
- Vendedores.
- Compradores.
- Subasta.

4.6.4. Especificación de Caso de Prueba: Subasta Web de Animales.

Descripción:

Una vez registradas las subastas en el sistema, el comprador podrá acceder al sitio de subastas web y ofertar en las subastas que estén disponibles. Posteriormente el administrador aceptará la mejor oferta para cada animal y se procederá a la licitación de animales.

Comprobación:

En ésta prueba se registraron 25 animales, se publicaron las subastas para cada uno de ellos, el comprador ofertó en la subasta que deseaba. A continuación, el administrador acepta la mejor oferta para cada animal, las subastan se cierran y automáticamente el sistema realiza el proceso de licitación de animales.

Condiciones de ejecución:

- El usuario deberá estar logueado en el sistema y tener permiso para el módulo de Subastas y Licitación.
- El usuario comprador deberá estar registrado para tener acceso al sitio de subastas web.

Entrada:

- Animales.
- Subasta.
- Compradores.

CAPITULO V
CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones.

- Se ha diseñado un módulo que se pueda integrar con nuevos desarrollos. Que sea fiable, flexible y adaptable a las necesidades de la empresa. También incrementar la calidad y satisfacción de los usuarios.

- Con el uso de herramientas de Software Libre, una de las ventajas es el bajo o nulo coste de los productos libres que permiten proporcionar a las PYMES servicios y ampliar sus infraestructuras sin que se vean mermados sus intentos de crecimiento por no poder hacer frente al pago de grandes cantidades en licencias. Cuando se presentaron errores muy complejos en el desarrollo del sistema, se contaba con la ayuda que hay en el internet, en los foros.

- Se ha mejorado el proceso de venta de ganado mediante el sistema para que poder establecer un precio justo, transparente y competitivo. De esta manera tanto los vendedores de animales como también los compradores se beneficiarán maximizando sus ganancias.

- El diseño de la base de datos fue desarrollado en base al tamaño de la información, a la facilidad de acceso y extracción de la información requerida. Este diseño satisface las necesidades de procesamiento de los datos y de generación de informes.

- Los reportes permiten a la empresa tomar decisiones acertadas, le ayudan a apreciar e interpretar la información de una manera más rápida así como eficiente, los reportes ayudan a esquematizar la información lo que a su vez le permitirá a la empresa entenderla y organizarla.

5.2. Recomendaciones.

- Subgana es un sistema escalable, es decir, puede seguir evolucionando, debido a que ha sido implementado con una sólida y flexible arquitectura. Al sistema se le pueden agregar módulos adicionales, que pueden ser integrados sin mucha dificultad.

- Es importante mantener la aplicación y supervisar su funcionamiento, además de realizarle el mantenimiento y ajustes correspondientes.

- Sería fundamental que la empresa decida migrar sus sistemas a software libre por la infinidad de beneficios que ofrece tales como, la independencia tecnológica, el control sobre el software, la confiabilidad y estabilidad, la seguridad para con los virus, la supresión del uso de copias no licenciadas, la eliminación de monopolios, ahorro financiero y un potencial liderazgo tecnológico regional.

- Sistematizar todos los procesos internos que maneja la empresa, para optimizar tiempos y facilitar la labor al personal que maneja la Institución.

- Capacitar al personal de la empresa en las Tecnologías de Información y Comunicación (TICs), ya que existen nuevas tecnologías que los funcionarios pueden utilizar para la mejor práctica de tareas dentro de la empresa.

- A las autoridades de la Universidad mantener este tipo de convenios que ayudan a los estudiantes a desarrollar sus trabajos de grado como también ganar experiencia en el ambiente laboral.

- A nuestros docentes fomentar la investigación en los estudiantes para que cuando terminen su carrera universitaria tengan la capacidad necesaria para resolver problemas cotidianos en la vida profesional.

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. Registro.

Anteriormente no se llevaba el registro de vendedores, compradores y animales, más bien se utilizaba un comprobante de ingreso y de salida de las personas y animales, pero nunca se registraba los animales que pertenecían a cierta persona, por lo que se presentaban facilidades para cometer actos ilícitos dentro de la empresa.

Cabe señalar que la información anteriormente era insegura e inestable. Ahora se cuenta con la aplicación y así habrá una correcta sistematización de la información.

A continuación, se muestra un gráfico estimado en base a proyecciones del proceso de registro de los datos de un vendedor con los animales respectivos.

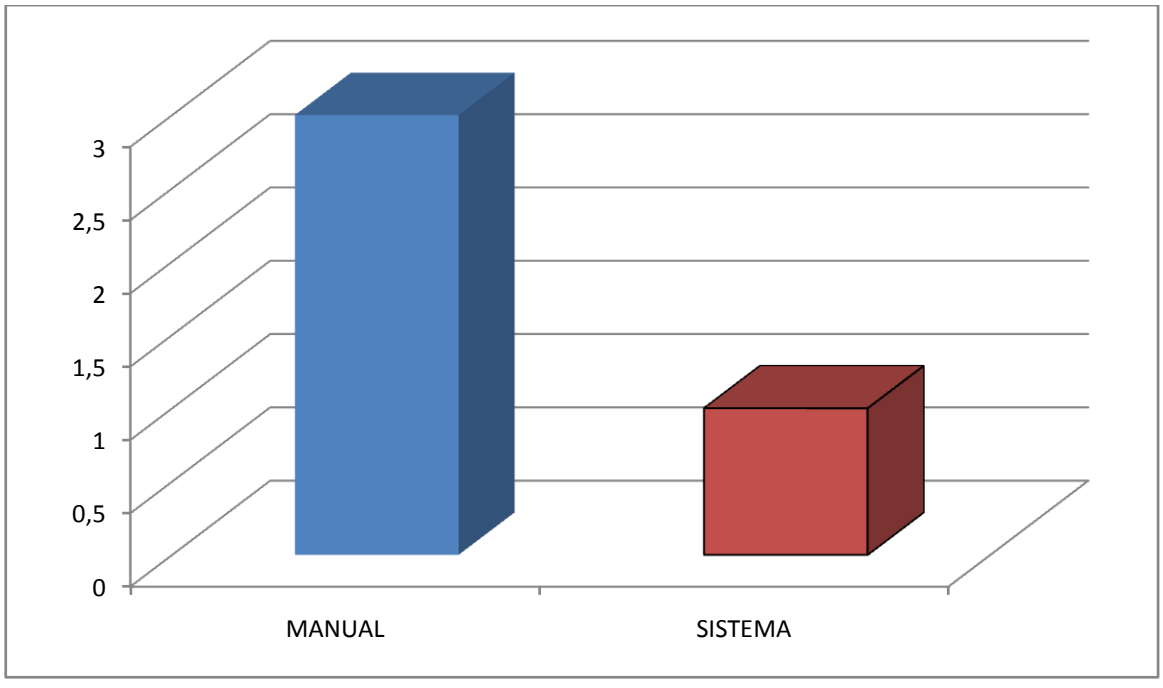


Ilustración 37. Análisis de Tiempo durante el Registro de Vendedores y Compradores

En la ilustración anterior se observa el tiempo del proceso de registro. Sin la implantación del sistema se demorará cerca de tres minutos en registrar en papel, mientras que en las pruebas efectuadas en el sistema se registra la información en aproximadamente un

minuto. Por ende se estableció conjuntamente con las personas encargadas del manejo de la aplicación que el sistema optimiza el tiempo de trabajo del empleado.

Además es importante mencionar que anteriormente en la feria ganadera nunca se registró datos de los compradores, vendedores y animales, pero con el sistema SUBGANA, se logrará llevar un control estadístico de todos los entes que actúan sobre el proceso de la venta del ganado.

5.3.2. Licitación y Subastas.

Durante la fase de pruebas se pudo comprobar que con la implementación del sistema se mejorará el proceso de subastas, ofertas y licitación, beneficiando de esta manera a todos los involucrados.

La venta de ganado que se realiza actualmente en la feria ganadera es por venta directa sin intermediarios.

Con la implementación del sistema se fortalecerá la comercialización de ganado ya que la Subasta Ganadera será una herramienta fundamental para dinamizar el mercado de compra y venta de animales en pie, bovino, porcinos y ovinos.

Se ha fijado implementar la subasta ganadera porque:

- Tanto el comprador como el vendedor se sientes motivados y no engañados.
- Los negocios se vuelven justos y transparentes.
- El comprador está seguro de lo que adquiere.
- Se motiva a mejorar la producción con calidad y el manejo técnico de los animales.
- Mejorar la situación del sector ganadero en la zona Norte del país, con la participación de los productores ganaderos en la comercialización de su producto.
- Fomentar el desarrollo socioeconómico del sector ganadero y sectores relacionados.
- Permite que los ganaderos, sobre todo los más pequeños, logren jugar un papel más protagónico dentro de la actividad, accediendo a precios más justos, así como a la posibilidad de adquirir animales de forma más segura y sencilla.
- Constituirse en la principal subasta ganadera de la región norte, de acuerdo con su aporte al desarrollo socioeconómico de la región.

A continuación, se muestra un gráfico que permita ver las oportunidades de venta de ganado con el uso del sistema y sin el uso del sistema.

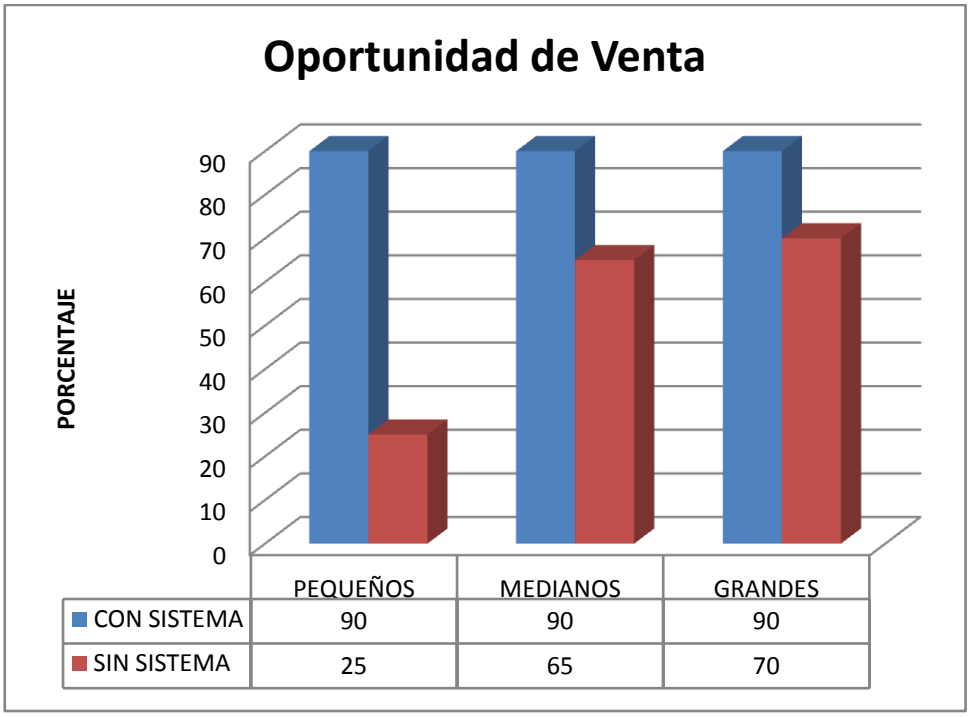


Ilustración 38. Análisis de la oportunidad de venta

En la figura anterior vemos que sin la implementación del sistema no tendrán las mismas oportunidades de venta entre los diferentes tipos de productores, estos datos se manejan en función a la actual feria ganadera. Con la implementación de la aplicación se logrará mantener igualdad de condiciones para todo tipo de productores de ganado.

5.3.3. Reportes.

Anteriormente la empresa carecía de este tipo de información ya que no se contaba con este sistema.

Con la implementación de esta aplicación se puede generar reportes estadísticos que ayuden a las personas a analizar cuáles son los animales más vendidos, o la raza más vendida de cierto tipo de ganado, con esta información se podrá ayudar a los productores de ganado a generar más ingresos y optimizar recursos económicos.

Se estima que el uso de este módulo permitirá tomar decisiones en base a resultados de sus ventas a pequeños y grandes productores.

Además es importante mencionar que este módulo permanecerá activo y actualizado todo el tiempo y de forma inmediata por lo que las personas que necesiten saber datos reales de la subasta podrán accederlo acercándose a la oficina correspondiente.

Lo que se puede mencionar como resumen general, es que antes de implementar la aplicación se demoraba mucho tiempo para sacar un dato exacto de los animales vendidos durante un cierto mes, ahora es más fácil sacar un dato exacto de cuantos animales vendimos.

A continuación, se muestra un gráfico aproximado de la generación de un reporte de animales ingresados y vendidos en función del tiempo.

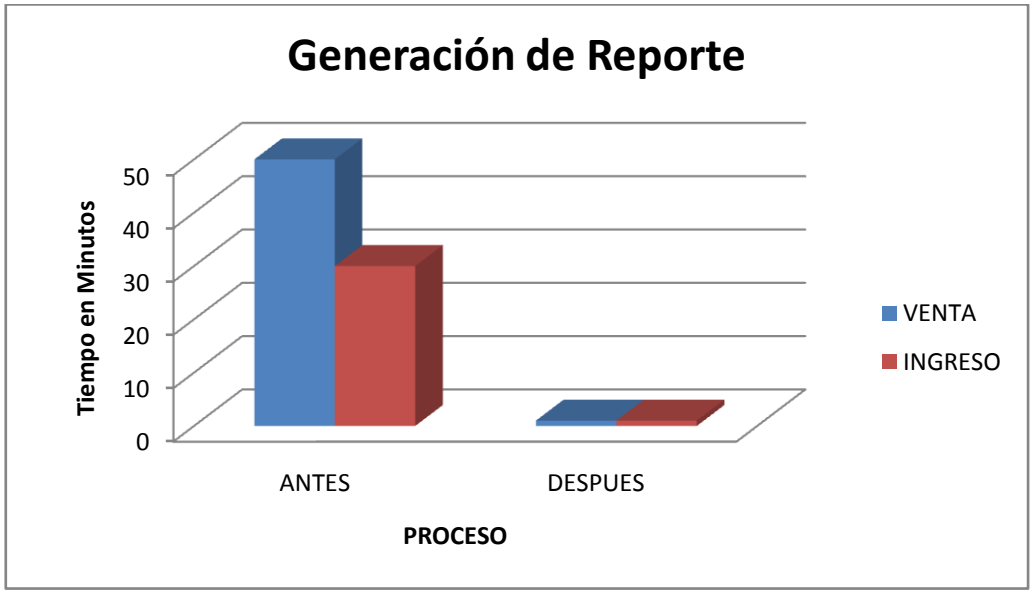


Ilustración 39. Análisis de la Generación de Reportes

En este reporte se muestran los datos aproximados que se generan al momento de generar un reporte, antes del sistema se muestra que para saber cuántos animales se vendió durante un periodo, se demora como mínimo unos 50 minutos ya que al momento no tienen una aplicación que les dé realizando el proceso de forma automática. En cambio con el sistema se genera automáticamente al seleccionar los parámetros de fechas dentro de la aplicación.

BIBLIOGRAFIA

LIBROS.

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

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

PUBLICACIONES EN LINEA.

Proyecto competitividad subasta ganadera expo Pococí. Recuperado en Mayo de 2010, de http://www.monografias.com/trabajos74/competitividad-subasta-ganadera-expo-pococi/competitividad-subasta-ganadera-expo-pococi.shtml

Wikipedia, E. (14 de Junio de 2010). *Subasta*. Recuperado el 2011, de Fundación Wikipedia Inc.: http://es.wikipedia.org/wiki/Subasta

Sliderashare.net, E. (26 de Enero de 2010). Tipos de *Subasta*. Recuperado el 2011, de http://www.slideshare.net/juan2888/tipos-de-subasta

Instituto Interamericano de Cooperación para la Agricultura. (2010). *Java Persistence Api (JPA).* Recuperado en Septiembre de 2012, de http://www.iica.int

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.