TEMA

“Sistema de Gestión de Cuidados, Crianza y Ventas de Ganado de la Agropecuaria Velbamal del Cantón Vinces”.

CAPITULO I

1. EL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

En el recorrido que se hizo en la hacienda "Agropecuaria Velbamal", observamos la desorganización en asignar los cuidados, crianzas y ventas, también se visualizó la anomalía de la aplicación de lasvacunas, baños, y la distribución de los animales en cada lote desde cuando nacen hasta que mueren.

No se encontraban los historiales de cada animal, por decir partos y abortos.

Partiendo de que en la "Agropecuaria Velbamal" el ente regulador de la administración y dominio de los animales, el mismo que no tiene un control adecuado, se ha visto la necesidad de crear e implementar un Sistema de Gestión de Cuidados, Crianza y Venta de Ganado.

Por otro lado sabemos que estos animales, los mismos que constituyen la inversión de la agropecuaria no están a un buen cuidado y conservación lo que dependerá de la eficacia del personal a cargo.

La responsabilidad compartida y la falta de un control interno que salvaguarde los recursos, impiden una buena administración, lo que no le permite a la institución mantener un adecuado manejo y control administrativo de los mismos, así como también no conocer la totalidad actual.

Se puede deducir que el ganado de la agropecuaria tiene un serio problema relacionado con la ampliación de segmento de mercado en el cual se desempeña.

Por tal motivo en la agropecuaria "Velbamal", debido al avance tecnológico y a las complicaciones generadas por el sistema actual el cual no cubre con las necesidades de los procesos que se realizan diariamente, pretende implementar un sistema que permita realizar de una forma más automatizada los procesos, dando como resultado que la información este más segura y que se lleve un mejor control de los cuidados, crianza, y ventas y administración en general, que se ha ofrecido por el sistema actual.

El manejo de las haciendas siempre se han manejado de una manera manual, pero nunca se ha llevado un control de una manera automatizada y computarizada por lo que se han presentado las siguientes complicaciones:

- Consumo innecesarios de material de papelería e insumos en general lo cual ocasiona perdida de recursos.

- Las secretarias no tienen el hardware (PC) apropiado y además la Hacienda no cuenta con el software necesario que permita llevar los registros de control de cuidados,crianza y ventas.

- Existe pérdida de tiempo al ingresar los datos debido a que no se cuenta con las herramientas necesarias que facilite dicha tarea.

- La agropecuaria no posee un archivo general de todas sus vacas, toros, etc.

- A pesar de que la utilización del sistema actual resulta sencilla su capacidad parece ser limitada.

- Se ha detectado perdidas de información referente a los datos del ganado.

En vista de las necesidades presentadas para controlar la información en general de esta entidad surgió la idea de equipar a la misma con los recursos de hardware y software necesarios que permitan llevar un mayor control de los procesos, brindando información necesaria, oportuna y útil garantizando la protección y confiabilidad de los datos, esperando satisfacer de esta forma las necesidades de los clientes con una atención rápida, eficiente, eficaz y personalizada.

1.2 FORMULACIÓN DEL PROBLEMA

Frente a la situación actual por la que atraviesa este negocio se propone como problema. **¿Cómo controlar mejor la Gestión de Cuidados, Crianza y Ventas de Ganado de la Agropecuaria Velbamal del Cantón Vinces?.**

1.3 DELIMITACION

1.3.1. Temporal

De Abril 2011 a Mayo 2012

1.3.2. Espacial

1.3.2.1. Agropecuaria Velbamal.

1.3.2.2. Ubicación:

* 1 ½ Km via Palenque

* Ciudad de Vinces.

* Cantón Vinces.

* Provincia de Los Ríos

1.3.3. Unidades de observación

* Area de administracion (Vinces)

1.4 OBJETIVOS

1.4.1 OBJETIVO GENERAL

- Desarrollar un Sistema para la eficiente Gestión "de Cuidados, Crianzas y Ventas de Ganado de la Agropecuaria Velbamal".

1.4.2 OBJETIVOS ESPECÍFICOS

- Establecer teorías con la gestión de ventas del ganado.
- Recolección y Análisis de la Información
- Implementar un Software Administrativo y validarlo con un experto.

1.5 JUSTIFICACION

En la actualidad la computadora es una de las herramientas más trascendentales para el desarrollo institucional de un ente económico, porque es necesario renovar la tecnología informática para tener una mejor perspectiva empresarial y mejorar sus recursos: financieros, profesionales y materiales.

Los procesos informáticos hoy en día constituyen uno de los aspectos más importantes de las organizaciones empresariales, para los que se dedican a la Producción o a los servicios, es necesario que la información sea procesada y almacenada de una forma más efectiva para agilizar los procesos de cuidados, crianza y ventas logrando así un control integral de las actividades. Con el desarrollo de un sistema automatizado que abarque las necesidades y una mayor relación con los requerimientos del hacendado, proporcionará una mejor efectividad en el manejo del ganado.

La propuesta de solución a este problema consiste en la elaboración de un plan de orden y administración del ganado el cual permitirá que se mejore la imagen institucional y consecuentemente se pueda ampliar el segmento de mercado en el cual se halla laborando la ganadería. Esto significará que se tendrá un mejor posicionamiento en el mercado de Los Ríos y a nivel nacional.

De aplicarse la solución propuesta muy probablemente se pueda conseguir proyectos en algunas otras empresas de ganaderías regionales o privadas del país, esto posibilitará un mayor ingreso económico.

Por todo esto se considera plenamente justificada la realización de la presente tesis como solución al problema de administración que tiene la empresa del ganado “Agropecuaria Velbamal”.

El uso de la tecnología va a servir a la agropecuaria de un medio avanzado que le permitirá manejar informaciones y datos en tiempo real, facilitando el análisis y la toma de dediciones para alcanzar niveles máximos de eficacia y eficiencia en el aspecto financiero-administrativo así como en los niveles asistenciales más próximos al usuario brindando así un servicio de alta calidad porque contara con su propio sistema PHP.

CAPITULO II

2. MARCO TEORICO

2.1 ANTECEDENTES DE LA INVESTIGACION

Al presente trabajo de investigación, no le antecede proyecto similar luego de buscar las referencias en tesis en la Universidad Técnica de Babahoyo, este trabajo surge en base al análisis realizado del comportamiento de los clientes internos y externos del comercial Agropecuaria Velbamal.

Durante el tiempo de revisión de datos se observó que existen grandes desfases e incongruencias en la gestión de cuidados, crianza y ventas, así también como la atención al cliente y la poca afluencia de la misma.

Con la inauguración el día 10 de Abril de 1993 de la Agropecuaria “VELBAMAL”, bajo la dirección de su propietario del Sr. Mario Cabezas; se rompió con los cánones tradicionales de la agropecuaria de la ciudad de Vinces, erigiéndose en un nuevo modelo que en aquella época constituyó una iniciativa singular. Representando desde su creación un concepto inédito de la administración y del servicio, claramente personalizado y en un ambiente intimista y de competitividad.

Probablemente, con la inauguración de este establecimiento, la agropecuaria de Vinces haya dado un nuevo paso en la metamorfosis que se precisaba en la ciudad. La Agropecuaria “VELBAMAL” aspira a convertirse en un símbolo emblemático de este proceso de renovación del sector. No es una agropecuaria alternativo versus tradicional de la ciudad, sino ese necesario complemento que precisaba el ramo del cuidado excelente de Ganado para representar una oferta más completa y enfocada a un mercado de alto nivel cualitativo.

La Agropecuaria “VELBAMAL”, es una agropecuaria que está situada en el mismo corazón de Vinces, en la vía 1 1/2 km Palenque.

2.2 FUNDAMENTACION CIENTIFICA

2.2.1 LA AGROPECUARIA VELBAMAL

2.2.1.1 El Sector Ganadero y Eficiencia

Sector Ganadero:[1]Pertenece al Sector de Ganado todos las Agropecuarias quedesarrollan actividades de crianzas y ventas de ganado, Es una actividad económica del sector primario encargada de la cría y domesticación de animales para el consumo humano.

Eficiencia: La Agropecuaria Velbamal como entidad privada, debe ser capaz de promover sus propios recursos eficazmente y hacer más agil y efectiva la forma de manejar sus ingresos que son realizados mediante la recaudación de ventas.

2.2.1.2 SISTEMA ACTUAL

2.2.1.2.1 CRITICAS DEL SISTEMA ACTUAL

En la actualidad la Agropecuaria "VELBAMAL", de la ciudad de Vinces no posee ningún tipo de sistema informático por lo que todos sus procesos son desarrollados de forma manual; siendo la secretaria la persona encargada de llevar el control de los mismos. Además los registros de las vacas y toros son llevados sin ningún orden específico existiendo pérdida de tiempo al momento de realizar alguna consulta.

2.2.1.2.2 FORTALEZAS Y DEBILIDADES DE LA AGROPECUARIA

Fortalezas de la Agropecuaria

- ✓ "El personal que trabaja en la agropecuaria", existe trabajo en equipo todos colaboran cuando ocurre alguna dificultad.
- ✓ Muy buena actitud de parte de los empleados hacia el cliente.
- ✓ El jefe de sector es una persona muy accesible para el personal que trabaja, no se encierra en su cargo, ayuda en lo que más puede al personal en caso de alguna dificultad y trabaja a puertas abiertas en caso de cualquier consulta.

Debilidades

- ✓ Al ser una agropecuaria, no presta mayores servicios. Su infraestructura en el caso es algo limitada.
- ✓ La mayoría del personal que trabaja en la agropecuaria no tiene estudios terminados en el Campo Agropecuario.

[1]http://www.monografias.com/Agricultura_y_Ganaderia/index.shtml

2.2.1.3SISTEMA DE GESTION

2.2.1.3.1 OBJETIVOS DEL NEGOCIO

- ✓ Tener en todo momento la mejor disposición para atender a nuestros clientes.
- ✓ Demostrar la responsabilidad, limpieza y respeto que se merecen nuestros clientes.
- ✓ Proporcionar una de las mejores instalaciones que se encuentran en la ciudad de Vinces que nos asegure su regreso.
- ✓ Ofrecer a nuestras reses con excelente calidad para un buen consumo en el ser humano.
- ✓ Reemplazar el modo actual con la adquisición de un nuevo sistema que satisfaga los requerimientos del negocio.
- ✓ Contribuir al desarrollo de la ciudad, provincia y país con el pago de los impuestos contemplados en la ley que es lo que se decreta actualmente.

2.2.1.4 GESTIÓN ADMINISTRATIVA DE UNA AGROPECUARIA

2.2.1.4.1 Funciones de la Administración de una Agropecuaria

La función primordial de la administración es producir ganancias al dueño. Para realizar esta operación, la administración debe efectuar las siguientes actividades:

- Planeación a corto y largo plazo.
- Mantener una imagen positiva y un servicio de buena calidad.
- Implantar políticas y procedimientos operativos.
- Mantener comunicación efectiva entre los clientes.
- Asegurar que en la agropecuaria tenga el personal adecuado y que estos, estén debidamente capacitados, motivados y supervisados.

2.2.1.4.2 Estructura y Personal de la Agropecuaria

Independientemente de si la agropecuaria es una simple hacienda o encierra en sus cuatro paredes todas las comunidades de una ciudad pequeña , ciertamente se trata de un negocio de gente, no sólo porque está hecho para servir al cliente, sino porque requiere de los servicios en los clientes para existir. La automatización puede ayudar, pero únicamente los seres humanos pueden proporcionar los servicios necesarios para el bienestar de las reses.

Todas las agropecuarias tienen distintas clases de animales y la mayoría ofrecen excelencia. Los que tienen éxito agregan un ingrediente adicional, el buen servicio.

Este es el único producto que no puede comprarse. La conducta humana en una sociedad libre no puede uniformarse; sólo puede ser guiada en un proceso que requiere de supervisión, atención y entrenamiento constante.

Nuestras reses varían no solo en tamaño, sino también en la clasificación como: por ejemplo cuando son terneros y cuando ya estén grandes. La administración de cada uno debe determinar los lotes y el número de empleados que serán necesarios para su propia operación. La administración de la agropecuaria es una operación de tiempo normal, 10 horas al día durante 6 días a la semana.

El jefe de sector es responsable de definir interpretar las políticas establecidas para elsector. Además, el jefe de sector, con éxito debe aplicarlas y mejorarlas, y ocasionalmente verse obligado a omitirlas por completo. El correcto desempeño de estas obligaciones requiere de un conocimiento funcional de todas las fases de la operación de la agropecuaria.

Nadie puede dar o explicar una orden correctamente sin tener alguna idea de lo que se trata.

La forma, más rápida y fácil de que el ejecutivo pierda el respeto de los empleados es dando instrucciones sin comprender sus implicaciones o la cantidad de tiempo necesario para realizarlas. De hecho, es imposible supervisar a nadie en forma correcta e inteligente sin tener por lo menos una idea general de los deberes y responsabilidad de esa persona.

2.2.1.4.3 Reses

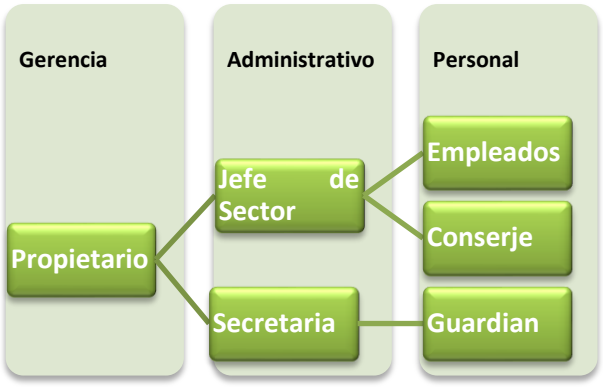
En una actividad económica como es la ganadería, con el propósito de crear un negocio productivo, que consiste en el manejo de animales domesticables, donde lo primordial de estos, son las reses.

Dependiendo de la especie ganadera, se pueden obtener diversos productos derivados, tales como la carne, la leche, los cueros, etc.

Los ganados más importantes en número a nivel mundial son los relacionados con la ganadería bovina

Para lo cual el principal responsable en el mantenimiento de la agropecuaria se delega a los empleados, estos dirigen el correcto cuidado de los vacunos.

2.2.1.4 .4 ORGANIGRAMA ESTRUCTURAL



2.2.1.4.5 Datos Básicos de las Reses de la Agropecuaria de Ganado Velbamal

- En este punto principal nos encontramos con las actividades de las reses:

  - ✓ Vaca Preñada
  - ✓ Vaca Vacia
  - ✓ Vaca Lactando
  - ✓ Vaca Seca
  - ✓ Reproductor
  - ✓ Castrados
  - ✓ Mamones

- A continuación los eventos que se encuentran en las reses de la Agropecuaria Velbamal.

  - ✓ Parto
  - ✓ Aborto
  - ✓ Enfermedad
  - ✓ Muerte
  - ✓ Normal

- A continuación la informacion obtenida de la descripcion de reses necesaria en nuestro Sistema:

  - ✓ Vaquilla
  - ✓ Vaca
  - ✓ Toro

- ✓ Ternero/a
- ✓ Novillo

- La formación de los lotes que hay en la agropecuaria se dividen en los siguientes:

  - ✓ Lote 1/destete
  - ✓ Lote 2/destete
  - ✓ Lote 3/destete
  - ✓ Lote 1/ceba
  - ✓ Lote 2/ceba
  - ✓ Lote 3/ceba

2.2.1.4 .6 VACUNAS

El elemento más importante de las reses es el tiempo de la vacunación ya que podemos evitar vender carnes contaminadas con un perfecto control para realizar las vacunas sobre la fiebre aftosa y otras enfermedades que estas reses pueden obtener.

2.2.1.4.7TIPO DE LIMPIEZA

Aquí tenemos la Informacion sobre el tipo de limpieza en la agropecuaria

- ✓ Limpieza Corral
- ✓ Limpieza de Animales

2.2.1.4.8 RAZAS

Entre las razas tenemos las siguientes:

- JERSTIN
- CHAROLAIS
- BRONSWIS
- BRAHMAN
- SAJIGUA
- SERRANA

2.2.1.4.9Condición de las Ventas

Cuando se recibe una solicitud autorización para la venta el encargado debe estar en condiciones de informar a la persona que debe conocer el estado de cada res para el

cual está aceptando pagar una cantidad de dinero y ya la tenemos calculada dentro del Sistema realizado en PHP.

2.2.1.4.9.1 Ventas de las Reses

En la agropecuaria Velbamal los precios de las reses están registrados por el tipo de Razaz en lbs y kgs.

No obstante ello, si quiere tener un panorama de los precios que pueden cobrarse por los cortes, dejamos **a continuación una tabla comparativa de los precios de carne de res:**

RAZAS	Ternero		Novillo		Vaca		Vaquilla		Toro	
	lbs	kg	lbs	kg	lbs	kg	Lbs	kg	lbs	kg
JERSTIN	1.50	50	2.00	70	1.75	4.30	1.75	4.30	2.25	4.95
CHAROLAIS	1.60	60	3.00	80	1.60	3.52	1.60	3.52	2.10	4.62
BRONSWIS	1.40	40	1.00	60	1.40	3.08	1.40	3.08	1.90	4.18
BRAHMAN	1.80	80	6.00	100	2.00	4.4	2.00	4.4	2.50	5.50
SAJIGUA	1.30	30	0.90	50	1.80	3.96	1.80	3.96	2.30	5.06
SERRANA	1.40	40	1.00	60	1.85	4.07	1.85	4.07	2.35	5.17

2.2.2 PHP

2.2.2.1 Definición de PHP

[1]El PHP, se define como un lenguaje de programación para la creación rápida de contenidos dinámicos de sitios web, como son los foros, blogs, sistemas de noticias, entre otros. También, crea aplicaciones gráficas independientes del navegador y aplicaciones para servidores. Es un lenguaje de script dentro del HTML.

Este programa computacional fue creado en Perl el año 1994 por Rasmus Lerdorf.

PHP presenta características esenciales que lo hacen ser tan eficaz. Posee un lenguaje multipataforma, es decir, un software que trabaja en diversas plataformas; conexión y aporte con un gran número de bases de dato como MySQL, Oracle, Informix, entre otros; capacidad de expandirse potencialmente por su gran cantidad de módulos; con bibliotecas externas de funciones; es libre (open source o código abierto) y por lo tanto asequible para todo el que lo desee.

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor.

¿Qué se puede hacer con PHP?

PHP puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Y esto no es todo, se puede hacer mucho más.

Existen tres campos en los que se usan scripts escritos en PHP.

- **Scripts del lado del servidor.** Este es el campo más tradicional y el principal foco de trabajo. Se necesitan tres cosas para que esto funcione. El intérprete PHP (CGI ó módulo), un servidor web y un navegador. Es necesario correr el servidor web con PHP instalado.
- **Scripts en la línea de comandos.** Puede crear un script PHP y correrlo sin ningún servidor web o navegador. Solamente necesita el intérprete

[1]**http://www.php.net/manual/es/preface.php,Autor: Bill Abt**

- **Escribir aplicaciones de interfaz gráfica.** Probablemente PHP no sea el lenguaje más apropiado para escribir aplicaciones gráficas, pero si conoce bien PHP, y quisiera utilizar algunas características avanzadas en programas clientes, puede utilizar PHP-GTK para escribir dichos programas.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, de modo que con PHP tiene la libertad de elegir el sistema operativo y el servidor de su gusto. También tiene la posibilidad de usar programación procedimental o programación orientada a objetos.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía web para una base de datos es una tarea simple con PHP. Soporta ODBC (el Estándar Abierto de Conexión con Bases de Datos), asi que puede conectarse a cualquier base de datos que soporte tal estándar.

2.2.2.2 Su primera página con PHP

Comienze por crear un archivo llamado hola.php y colocarle en el "directorio raíz" (DOCUMENT_ROOT) con el siguiente contenido:

Ejemplo 2-1. Nuestro primer script PHP: hola.php

```
<html>
<head>
 <title>Ejemplo de PHP</title>
</head>
<body>
<?php echo "<p>Hola Mundo</p>"; ?>
</body>
</html>
```

Utilice su navegador web para acceder al archivo, con la URL terminando en "/hola.php". Si está programando localmente este URL lucirá algo como http://localhost/hola.php o http://127.0.0.1/hola.php pero esto depende de la configuración de su servidor web.

El objetivo de este ejemplo es demostrar cómo puede usar las etiquetas PHP. En este

ejemplo usamos <?php para indicar el inicio de la etiqueta PHP. Después indicamos la sentencia y abandonamos el modo PHP usando ?>.

2.2.2.3INSTALACION Y CONFIGURACION

El primero es el más tradicional y el principal campo de trabajo. Se necesitan tres cosas para que funcione. El analizador PHP (CGI ó módulo), un servidor web y un navegador. Dependiendo de la versión de sistema operativo que utiliceis, probablemente tengais un servidor web (p.ej: Apache en Linux y MacOS X ó IIS en Windows).

Hay dos maneras de utilizar [2]PHP, instalando desde el servidor y PHP. Existen módulos directos (también llamados SAPI) para muchos servidores web, como Apache, Microsoft Internet Information Server, Netscape e iPlanet. Esto significa que el servidor se configura para usar el ejecutable para linea de comandos de PHP en el procesado de peticiones de ficheros PHP.

Con PHP tambien se puede escribir aplicaciones gráficas usando la extensión PHP-GTK. Esta es una forma totalmente distinta de utilizar PHP que cuando se utiliza para escribir páginas web, ya que no se genera código HTML sino que se trabaja con ventanas y objetos dentro de las mismas. Para más información sobre PHP-GTK, visitar las páginas dedicadas a esta extensión. PHP-GTK no se incluye en la distribución oficial de PHP.

2.2.2.4SINTAXIS BÁSICA

Cuando PHP interpreta un fichero, busca por las etiquetas de apertura y fin de bloque, que dicen a PHP donde empezar y finalizar la interpretación del código.

```
<p>Este texto va a ser ignorado.</p>
<?php echo 'Mientras que esto va a ser interpretado.'; ?>
<p>Esto también será ignorado.</p>
```

[2]**http://www.php.net/manual/es/preface.php, Autor: Bill Abt**

Existen cuatro tipos de pares de etiquetas de apertura y de fin de bloque que se pueden usar en PHP. Dos de estas, <?php ?> y <script language="php"></script>, siempre están disponibles.

Ejemplo sobre Etiquetas de apertura y de fin de bloque de PHP

```
1. <?php echo 'si se quiere mostrar documentos XHTML o XML, debe hacerse así'; ?>

2. <script language="php">
       echo 'algunos editores (como FrontPage) no les gusta
            las instruciones de proceso';
   </script>

3. <? echo 'esta es la forma más simple, una instrución de procesado SGML'; ?>
   <?= expression ?> Esto es una forma corta para "<? echo expression ?>"

4. <% echo 'Quizá use de forma opcional etiquetas de estilo ASP'; %>
   <%= $variable; # Esto es una forma corta para "<% echo . . ." %>
```

Las etiquetas vistas en los ejemplos uno y dos están siempre disponibles, el ejemplo uno es el más común y recomendado de los dos.

2.2.2.5TIPOS DE DATOS

PHP soporta ocho tipos primitivos.

Cuatro tipos escalares:

- boolean
- integer
- float (número de punto flotante, también conocido como double)
- string

Tipo compuesto:

- array

2.2.2.5.1 Booleanos

Este es el tipo más simple. Un boolean expresa un valor de verdad. Puede ser TRUE or FALSE.

Sintaxis

Para especificar un literal boolean, use alguna de las palabras clave TRUE o FALSE. Ambas son insensibles a mayúsculas y minúsculas.

```
<?php
$foo = True; // asigna el valor TRUE a $foo
?>
```

Usualmente, el resultado de un operador que devuelve un valor boolean es pasado a una estructura de control.

```
<?php
// == es un operador que prueba por
// igualdad y devuelve un booleano
if ($accion == "mostrar_version") {
    echo "La versión es 1.23";
}
// esto no es necesario...
if ($mostrar_separadores == TRUE) {
    echo "<hr>\n";
}
// ...porque se puede escribir simplemente:
if ($mostrar_separadores) {
    echo "<hr>\n";
}
?>
```

2.2.2.5.2Enteros

Un entero o integer es un número del conjunto $\mathbb{Z}$ = {..., -2, -1, 0, 1, 2, ...}.

Sintaxis

Los integer pueden ser especificados mediante notación decimal (base 10), hexadecimal (base 16), octal (base 8) o binaria (base 2), opcionalmente precedidos por un signo (- o +).

Para usar la notación octal, se antepone al número un 0 (cero). Para usar la notación hexadecimal, se antepone al número un 0x. Para usar la notación binaria, se antepone al número un 0b.

Ejemplo sobre Enteros literales

```
<?php
$a = 1234; // número decimal
$a = -123; // un número negativo
$a = 0123; // número octal (equivalente a 83 decimal)
$a = 0x1A; // número hexadecimal (equivalente a 26 decimal)
?>
```

2.2.2.5.3Números de punto flotante

Los números de punto flotante (también conocidos como "flotantes", "dobles" o "números reales") pueden ser especificados usando cualquiera de las siguientes sintaxis:

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

2.2.2.5.4Cadenas

Un string es una serie de caracteres donde un caracter es lo mismo que un byte. Esto significa que PHP solo soporta el conjunto de 256 caracteres y por lo tanto no tiene soporte nativo Unicode.

Sintaxis

Un string literal puede ser especificado de la siguiente manera:

- comillas simples

- comillas dobles

2.2.2.5.4.1 Comillas simples

La manera más sencilla de especificar un string es encerrarlo entre comillas simples (el caracter ').

```
<?php
echo 'Esto es una cadena sencilla';
```

2.2.2.5.4.2 Comillas dobles

Caracteres escapados

Sentencia	**Significado**
\n	avance de línea (LF o 0x0A (10) en ASCII)
\r	retorno de carro (CR o 0x0D (13) en ASCII)
\t	tabulador horizontal (HT o 0x09 (9) en ASCII)
\v	tabulador vertical (VT o 0x0B (11) en ASCII) (desde PHP 5.2.5)
\e	escape (ESC o 0x1B (27) en ASCII) (desde PHP 5.4.0)
\f	avance de página (FF o 0x0C (12) en ASCII) (desde PHP 5.2.5)
\\	barra invertida
\$	signo del dólar
\"	comillas dobles
\[0-7]{1,3}	la secuencia de caracteres que coincida con la expresión regular es un caracter en notación octal
\x[0-9A-Fa-f]{1,2}	la secuencia de caracteres que coincida con la expresión regular es un caracter en notación hexadecimal

2.2.2.5.5Arrays

Un array (matriz) en PHP es en realidad un mapa ordenado. Un mapa es un tipo de datos que asocia valores con claves.

Sintaxis

Especificación con array()

Un valor array puede ser creado por la construcción de lenguaje array(). Ésta toma un cierto número de parejas clave =>valor separadas con coma.

```
array( key =>value
    , ...
    )
// key puede ser un integer o string
// value puede ser cualquier valor
```

```
<?php
$arr = array("foo" => "bar", 12 => true);

echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

2.2.2.6 VARIABLES

Conceptos básicos

En PHP las variables se representan con un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

Los nombres de variables siguen las mismas reglas que otras etiquetas en PHP. Un nombre de variable válido tiene que empezar con una letra o un carácter de subrayado (underscore), seguido de cualquier número de letras, números y caracteres de subrayado. Como expresión regular se podría expresar como: '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'.

2.2.2.6.1 Variables Predefinidas

PHP proporciona una gran cantidad de variables predefinidas a cualquier script que se ejecute. Muchas de éstas, sin embargo, no pueden ser completamente documentadas ya que dependen del servidor que esté corriendo, la versión y configuración de dicho servidor, y otros factores. Algunas de estas variables no estarán disponibles cuando se ejecute PHP desde la línea de comandos.

La palabra clave global

En primer lugar, un ejemplo de uso de global:

Ejemplo sobre Uso de global

```
<?php
$a = 1;
$b = 2;

function Suma()
{
   global $a, $b;

   $b = $a + $b;
}

Suma();
echo $b;
?>
```

El script anterior producirá la salida "3". Al declarar $a y $b globales dentro de la función, todas las referencias a tales variables se referirán a la versión global. No hay límite al número de variables globales que se pueden manipular dentro de una función.

2.2.2.6.2 Variables variables

A veces es conveniente tener nombres de variables variables. Dicho de otro modo, son nombres de variables que se pueden definir y usar dinámicamente. Una variable normal se establece con una sentencia como:

```
<?php
$a = 'hola';
?>
```

Una variable variable toma el valor de una variable y lo trata como el nombre de una variable. En el ejemplo anterior, hola, se puede usar como el nombre de una variable utilizando dos signos de dólar. Es decir:

```
<?php
$$a = 'mundo';
?>
```

En este momento se han definido y almacenado dos variables en el árbol de símbolos de PHP: $a, que contiene "hola", y $hola, que contiene "mundo". Es más, esta sentencia:

```
<?php
echo "$a ${$a}";
?>
```

produce el mismo resultado que:

```
<?php
echo "$a $hola";
?>
```

Esto quiere decir que ambas producen el resultado: hola mundo. Para usar variables variables con matrices, hay que resolver un problema de ambigüedad. Si se escribe $$a[1] el intérprete necesita saber si nos referimos a utilizar $a[1] como una variable, o si se pretendía utilizar $$a como variable y el índice [1] como índice de dicha variable. La sintaxis para resolver esta ambigüedad es: ${$a[1]} para el primer caso y ${$a}[1] para el segundo.

Donde se hizo la llamada. Por ejemplo, en esta expresión $foo->$bar, de forma local en la clase se buscará por $bar y su valor será usado como el nombre de la propiedad de $foo. Esto también es cierto si $bar es un acceso a un array.

2.2.2.6.3Variables Desde Fuentes Externas

Formularios HTML (GET y POST)

Cuando se envía un formulario a un script PHP, las variables de dicho formulario pasan a estar automáticamente disponibles en el script gracias a PHP. Por ejemplo, consideremos el siguiente formulario:

Ejemplo sobre Variables de formulario simples

```
<form action="foo.php" method="post">
    Nombre usuario: <input type="text" name="username" /><br />
    Email:  <input type="text" name="email" /><br />
    <input type="submit" name="submit" value="¡Enviarme!" />
</form>
```

Dependiendo de su configuración y preferencias personales, existen muchas maneras de acceder a los datos de formularios HTML.

Ejemplo de Acceso a datos de un formulario simple HTML POST

```
<?php

// Disponible desde PHP 4.1.0

echo $_POST['username'];

echo $_REQUEST['username'];

   import_request_variables('p', 'p_');

echo $p_username;

// Desde PHP 5.0.0, las variables predefinidas largas se pueden

// desactivar con la directiva register_long_arrays.

echo $HTTP_POST_VARS['username'];

// Disponible si la directiva de PHP register_globals = on. A partir de

// PHP 4.2.0 el valor predeterminado de register_globals = off.

// Usar o depender de este método no es recomendable.

echo $username;?>
```

2.2.2.7Constantes

Una constante es un identificador (nombre) para expresar un valor simple. Como el nombre sugiere, este valor no puede variar durante la ejecucción del script. (A excepción de las constantes predefinidas, que en realidad no son constantes). Una

constante es sensible a mayúsculas por defecto. Por convención, los identificadores de contantes siempre suelen declararse en mayúsculas.

El nombre de una constante sigue las mismas reglas que cualquier otra etiqueta de PHP. Un nombre de constante válido empieza por una letra o subguión, seguido por cualquier número o letras, números o subguiones. Usando una expresión regular, se representaría de la siguiente manera: [a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*.

Sintaxis

Se puede definir una constante usando la función define() o también declarándola fuera de la clase con const desde PHP 5.3.0. Una vez que la constante está definida, no puede ser cambiada o redifinida en ningún momento.

Solo se puede definir como constantes valores escalares (boolean, integer, float y string). Se puede definir resource en constantes, pero debería ser evitado, porque puede causar resultados inesperados.

Para obtener el valor de una constante solo es necesario especificar su nombre. A diferencia de las variables, no se debe prefijar una constante con el signo $. También se puede usar la función constant () para leer el valor de una constante si se desea obtener el valor de una constante de forma dinámica. Use get_defined_constants() para obtener una lista de todas las constantes definidas.

Si se usa una constante que todavía no está definida, PHP asume que se está refiriendo al nombre de la constante en si, igual que si fuera una string (CONSTANT vs "CONSTANT"). Cuando esto suceda, se mostrará un error de nivel E_NOTICE. Ver también la sección en el manual de porqué $foo[bar] es

Incorrecto (a no ser que primero define()bar como constante). Si simplemente quiere comprobar si una constante está definida, use la función defined().

Estas son las diferencias entre constantes y variables:

- Las constantes no llevan el signo dólar ($), como prefijo.
- Las constantes solo pueden ser definidas usando la función define(), y no por simple asignación.
- Las constantes pueden ser definidas y accedidas desde cualquier sitio sin importar las reglas de acceso de variables.
- Las constantes no pueden ser redefinidas o eliminadas una vez se han definido. Y

- Las constantes solo deberían contener valores escalares.

2.2.2.7.1Constantes predefinidas

PHP ofrece un largo número de constantes predefinidas a cualquier script en ejecucción. Muchas de estas constantes, sin embargo, son creadas por diferentes

Extensiones, y sólo estarán presentes si dichas extensiones están disponibles, bien por carga dinámica o porque han sido compiladas.

Hay siete constantes predefinidas que cambian dependiendo de donde son usadas. Por ejemplo el valor de __LINE__ depende en la línea que se use en el script. Estas constantes especiales son sensibles a mayúsculas y son las siguientes:

Varias constantes PHP "mágicas"

Nombre	**Descripción**
__LINE__	Línea actual en el fichero.
__FILE__	Ruta completa y nombre del fichero. Si se usa dentro de un include, devolverá el nombre del fichero del include. Desde PHP 4.0.2, __FILE__ siempre contiene la ruta absoluta con symlinks resueltos, en otras versiones contenía la ruta relativa según las circunstancias.
__DIR__	Directorio del fichero. Si se utiliza dentro de un include, devolverá el directorio del fichero incluído. Esta constante es igual que dirname(__FILE__). El nombre del directorio no lleva la barra inicial a no ser que esté en el directorio root. (Fue añadida en PHP 5.3.0)
__FUNCTION__	Nombre de la función. (Añadida en PHP 4.3.0) Desde PHP 5 esta constante devuelve el nombre de la función donde fue declarada (sensible a mayúsculas). En PHP 4 su valor siempre es en minúsculas.
__CLASS__	Nombre de la clase. (Añadida en PHP 4.3.0) Desde PHP 5 esta constante devuelve el nombre de la clase donde fue declarada (sensible a mayúsculas). En PHP 4 su valor siempre es en

Varias constantes PHP "mágicas"

Nombre	Descripción
	minúsculas. El nombre de la clase incluye el namespace declarado en (p.e.j. Foo\Bar). Tenga en cuenta que a partir de PHP 5.4 __CLASS__ también funciona con traits. Cuando es usado en un método trait, __CLASS__ es el nombre de la clase del trait que está siendo utilizado.
__TRAIT__	El nombre de el trait. (Añadido en PHP 5.4.0) A partir de PHP 5.4 esta constante devuelve el trait que fué declarado (sensible a mayúsculas y minúsculas). El nombre de el trait incluye el namespace si alguno fué declarado en (p.e.j. Foo\Bar).
__METHOD__	Nombre del método de la clase. (Añadida en PHP 5.0.0.) Nombre del método devuelto donde fue declarada. (sensible a mayúsculas).
__NAMESPACE__	Nombre del espacio de nombres actual (sensible a mayúsculas). Esta constante se define en tiempo de compilación (Añadida en PHP 5.3.0) El nombre del namespace actual (sensible a mayúsculas).

Ver también get_class(), get_object_vars(), file_exists() y function_exists().

2.2.2.8 Expresiones

Las expresiones son las piedras de construcción más importantes de PHP. En PHP casi todo lo que se escribe es una expresión. La manera más simple y acertada de definir lo que es una expresión es "cualquier cosa que tiene un valor".

Las formas más básicas de expresiones son las constantes y las variables. Cuando se escribe "$a = 5", se está asignando '5' a $a. '5', obviamente, tiene el valor 5, o en otras palabras, '5' es una expresión con el valor de 5 (en este caso, '5' es una constante entera).

Después de esta asignación, se espera que el valor de $a sea 5 también, por lo que si se escribe $b = $a, se espera que esto se comporte tal como si se escribiera $b = 5.

En otras palabras, $a es también una expresión con el valor 5. Si todo funciona bien, esto es exactamente lo que sucederá.

```
<?php
function foo ()
{
   return 5;
}
?>
```

PHP soporta cuatro tipos de valores escalares: valores enteros (integer), valores de coma (punto) flotante (float), valores de cadena (string) y valores booleanos (boolean) - (valores escalares son aquellos que no se pueden descomponer en piezas más pequeñas, a diferencia de las matrices, por ejemplo). PHP también soporta dos tipos compuestos (no escalares): matrices (arrays) y objetos. Cada uno de estos tipos de valores pueden ser asignados a variables o devueltos desde funciones.

PHP es un lenguaje orientado a expresiones, en el sentido de que casi todo es una expresión. Considere el ejemplo que ya hemos tratado, '$a = 5'. Es fácil de ver que aquí hay dos valores involucrados, el valor de la constante entera '5', y el valor de $a que ha sido actualizado a 5 también. Aunque la verdad es que existe aquí un valor adicional involucrado, que es el valor de la asignación misma. La asignación evalúa al valor asignado, en este caso 5. En la práctica, esto significa que '$a = 5', sin importar lo que haga, es una expresión con el valor 5. De este modo, escribir algo como '$b = ($a = 5)' es igual que escribir '$a = 5; $b = 5;' (el punto y coma marca el final de una sentencia). Ya que las asignaciones se analizan de derecha a izquierda, también se puede escribir '$b = $a = 5'.

2.2.2.9Operadores

Un operador es algo que toma uno o más valores (o expresiones, en jerga de programación) y produce otro valor (de modo que la construcción en si misma se convierte en una expresión).

Los operadores se pueden agrupar de acuerdo con el número de valores que toman. Los operadores unarios toman sólo un valor, por ejemplo ! (el operador lógico de negación) o ++ (el operador de incremento). Los operadores binarios toman dos valores, como los familiares operadores aritméticos+ (suma) y - (resta), y la mayoría de los operadores de PHP entran en esta categoría. Finalmente, hay sólo un operador ternario? :, el cual toma tres valores; usualmente a este se le refiere simplemente

como "el operador ternario" (aunque podría tal vez llamarse más correctamente como el operador condicional).

Una lista completa de operadores de PHP sigue en la sección Precedencia de Operadores. Está también explica la precedencia y asociatividad, las cuales gobiernan exactamente como son evaluadas expresiones que contienen varios diferentes operadores.

2.2.2.9 Estructuras de Control

Introducción

Todo script PHP está construido en base a una serie de sentencias. Una sentencia puede ser una asignación, una llamada de función, un ciclo, una sentencia condicional o incluso una sentencia que no hace nada (una sentencia vacía). Las sentencias generalmente finalizan con un punto y coma. Adicionalmente, las sentencias pueden agruparse en un conjunto de sentencias, encapsulándolas entre corchetes. Un grupo de sentencias es una sentencia por sí misma también. Los diferentes tipos de sentencias son descritos en este capítulo.

- if
- else
- elseif/else if
- Sintaxis alternativa de estructuras de control
- while
- do-while
- for
- break
- continue
- switch
- declare
- return
- require
- include

2.2.2.9.1 If

El constructor if es una de las características más importantes de muchos lenguajes, incluido PHP. Permite la ejecución condicional de fragmentos de código. PHP dispone de una estructura if que es similar a la de C:

```
if (expr)

Sentencia
```

Como se describe en la sección sobre expresiones, la expresión es evaluada a su valor booleano. Si la expresión se evalúa como TRUE, PHP ejecutará la sentencia y si se evalúa como FALSE la ignorará.

El siguiente ejemplo mostraría a es mayor que b si $a es mayor que $b:

```
<?php
if ($a > $b) {
  echo "a es mayor que b";

}
?>
```

A menudo se desea tener más de una sentencia para ser ejecutada condicionalmente. Por supuesto, no hay necesidad de envolver cada sentencia con una cláusula if. En cambio, se pueden agrupar varias sentencias en un grupo de sentencias. Por ejemplo, este código mostraría a es mayor que b si $a es mayor que $b y entonces asignaría el valor de $a a $b:

```
<?php
if ($a > $b) {
  echo "a es mayor que b";
  $b = $a;
}
?>
```

Las sentencias if pueden anidarse dentro de otra sentencias if infinitamente, lo cual provee completa flexibilidad para la ejecución condicional de diferentes partes del programa.

2.2.2.9.2 else

Con frecuencia se desea ejecutar una sentencia si una determinada condición se cumple y una sentencia diferente si la condición no se cumplePor ejemplo, el siguiente código deberá mostrar a es mayor que b si $a es mayor que $b y a NO es mayor que b en el caso contrario:

```
<?php
if ($a > $b) {
  echo "a es mayor que b";
} else {
  echo "a NO es mayor que b";
}
?>
```

La sentencia else sólo es ejecutada si la expresión if es evaluada como FALSE y si hay algunas expresiones elseif - sólo se ejecuta si también todas son evaluadas como FALSEver enelseif.

2.2.2.9.3 elseif/else if

elseif, como su nombre lo sugiere, es una combinación de if y else. Del mismo modo que else, extiende una sentencia if para ejecutar una sentencia diferente en caso que la expresión if original se evalúe como FALSE. Sin embargo, a diferencia de else, esa expresión alternativa sólo se ejecutará si la expresión condicional del elseif se evalúa como TRUE. Por ejemplo, el siguiente código debe mostrar a es mayor que b, a es igual que b o a es menor que b:

```
<?php
if ($a > $b) {
    echo "a es mayor que b";
} elseif ($a == $b) {
    echo "a es igual que b";
} else {
    echo "a es menor que b";
}
?>
```

Puede haber varios elseif dentro de la misma sentencia if. La primera expresión elseif (si hay alguna) que se evalúe como TRUE sería ejecutada. En PHP también se puede

escribir 'else if' (en dos palabras) y el comportamiento sería idéntico al de 'elseif' (en una sola palabra). El significado sintáctico es ligeramente diferente pero la conclusión es que ambos resultarían tener exactamente el mismo comportamiento.

La sentencia elseif es ejecutada solamente si la expresión if precedente y cualquiera de las expresiones elseif precedentes son evaluadas como FALSE, y la expresión elseif actual se evalúa como TRUE.

2.2.2.9.4 Sintaxis alternativa de estructuras de control

PHP ofrece una sintaxis alternativa para algunas de sus estructuras de control; a saber: if, while, for, foreach, y switch. En cada caso, la forma básica de la sintaxis alternativa es cambiar el corchete de apertura por dos puntos (:) y el corchete de cierre por endif;,endwhile;, endfor;, endforeach;, o endswitch;, respectivamente.

```
<?php if ($a == 5): ?>
A es igual a 5
<?php endif; ?>
```

En el ejemplo anterior, el bloque HTML "A es igual a 5" se anida dentro de una sentencia if escrita en la sintaxis alternativa. El bloque HTML se mostraría solamente si $a es igual a 5.

La sintaxis alternativa también se aplica a else y elseif. El siguiente es una estructura if con elseif y else en el formato alternativo:

```
<?php
if ($a == 5):
    echo "a igual 5";
    echo "...";
elseif ($a == 6):
    echo "a igual 6";
    echo "!!!";
else:
    echo "a no es 5 ni 6";
endif;
?>
```

2.2.2.9.5 While

Los bucles while son el tipo más sencillo de bucle en PHP. Se comportan igual que su contrapartida en C. La forma básica de una sentencia while es:

```
while (expr)

sentencia
```

El significado de una sentencia while es simple. Le dice a PHP que ejecute las sentencias anidadas, tanto como la expresión while se evalúe como TRUE. El valor de la expresión es verificado cada vez al inicio del bucle, por lo que incluso si este valor cambia durante la ejecución de las sentencias anidadas, la ejecución no se detendrá hasta el final de la iteración (cada vez que PHP ejecuta las sentencias contenidas en el bucle es una iteración). A veces, si la expresión while se evalúa como FALSE desde el principio, las sentencias anidadas no se ejecutarán ni siquiera una vez.

Al igual que con la sentencia if, se pueden agrupar varias instrucciones dentro del mismo bucle while rodeando un grupo de sentencias con corchetes, o utilizando la sintaxis alternativa:

```
while (expr):

sentencias

   ...

endwhile;
```

Los siguientes ejemplos son idénticos y ambos presentan los números del 1 al 10:

```
<?php
/* ejemplo 1 */

$i = 1;
while ($i <= 10) {
   echo $i++;  /* el valor presentado sería
                  $i antes del incremento
                  (post-incremento) */
}

/* ejemplo 2 */
```

```
$i = 1;
while ($i <= 10):
    echo $i;
    $i++;
endwhile;
?>
```

2.2.2.9.6 Do-while

Los bucles do-while son muy similares a los bucles while, excepto que la expresión verdadera es verificada al final de cada iteración en lugar que al principio. La diferencia principal con los bucles while es que está garantizado que corra la primera iteración de un bucle do-while,la expresión verdadera sólo es verificada al final de la iteración, mientras que no necesariamente va a correr con un bucle while regular, la expresión verdadera es verificada al principio de cada iteración, si se evalúa como FALSE justo desde el comienzo, la ejecución del bucle terminaría inmediatamente.

Hay una sola sintaxis para bucles do-while:

```
<?php
$i = 0;
do {
    echo $i;
} while ($i > 0);
?>
```

El bucle de arriba se ejecutaría exactamente una sola vez, ya que después de la primera iteración, cuando la expresión verdadera es verificada, se evalúa como FALSE ($i no es mayor que 0) y termina la ejecución del bucle.

2.2.2.9.7For

Los bucles for son los ciclos más complejos en PHP. Se comportan como sus contrapartes en C. La sintaxis de un bucle for es:

```
for (expr1; expr2; expr3)

sentencia
```

La primera expresión (expr1) es evaluada (ejecutada) una vez incondicionalmente al comienzo del bucle.

En el comienzo de cada iteración, la expr2 es evaluada, Si es TRUE, el bucle continúa y las sentencias anidadas son ejecutadas. Si se evalúa como FALSE, termina la ejecución del bucle.

Al final de cada iteración, la expr3 es evaluada (ejecutada).

Cada una de las expresiones puede estar vacía o contener múltiples expresiones separadas por comas. En la expr2, todas las expresiones separadas por una coma son evaluadas pero el resultado se toma de la última parte. Que la expr2 esté vacía significa que el bucle deberá ser corrido indefinidamente (PHP implícitamente lo considera como TRUE, como en C). Esto puede no ser tan inútil como se pudiera pensar, ya que muchas veces se quiere terminar el bucle usando una sentencia condicional break en lugar de utilizar la expresión verdadera del for.

Considere los siguientes ejemplos. Todos ellos muestran los números del 1 al 10:

```
<?php
/* ejemplo 1 */
for ($i = 1; $i <= 10; $i++) {
    echo $i;
}
/* ejemplo 2 */

for ($i = 1; ; $i++) {
    if ($i > 10) {
        break;
    }
    echo $i;
}
/* ejemplo 3 */

$i = 1;
for (; ; ) {
    if ($i > 10) {
        break;
    }
```

```
    echo $i;
    $i++;

}
/* ejemplo 4 */

for ($i = 1, $j = 0; $i <= 10; $j += $i, print $i, $i++);

?>
```

PHP también soporta la sintaxis alternativa de los dos puntos para bucles for.

```
for (expr1; expr2; expr3):

sentencia

    ...

endfor;
```

2.2.2.9.8 Break

Break termina la ejecución de la estructura actual for, foreach, while, do-while o switch.

Break acepta un argumento numérico opcional el cual indica de cuantas estructuras anidadas encerradas se debe salir.

```
<?php
$arr = array('uno', 'dos', 'tres', 'cuatro', 'pare', 'cinco');
while (list(, $val) = each($arr)) {
    if ($val == 'pare') {
        break;    /* Se puede también escribir 'break 1;' aquí. */
    }
    echo "$val<br />\n";
}

/* Usando el argumento opcional. */

$i = 0;
while (++$i) {
    switch ($i) {
    case 5:
```

```
        echo "En 5<br />\n";
        break 1;  /* Sólo sale del switch. */
    case 10:
        echo "En 10; saliendo<br />\n";
        break 2;  /* Sale del switch y del while. */
    default:
        break;
    }
}
?>
```

2.2.2.10 Continue

Continue se utiliza dentro de las estructuras de bucle para saltarse el resto de la actual iteración del bucle y continuar la ejecución en la evaluación de la condición y entonces el comienzo de la siguiente iteración.

Estructura de bucle para los efectos del continue.

Continue acepta un argumento numérico opcional el cual indica hasta el final de cuantos niveles de bucles cerrados se debe saltar.

Continue 0; y continue 1; son lo mismo que ejecutar continue;.

```
<?php
while (list($key, $value) = each($arr)) {
    if (!($key % 2)) { // saltar miembros impares
        continue;
    }
    do_something_odd($value);
}

$i = 0;
while ($i++ < 5) {
    echo "Outer<br />\n";
    while (1) {
        echo "Middle<br />\n";
        while (1) {
```

```
            echo "Inner<br />\n";
            continue 3;
        }
        echo "This never gets output.<br />\n";
    }
    echo "Neither does this.<br />\n";
}
?>
```

2.2.2.9.10 Switch

La sentencia switch es similar a una serie de sentencias IF en la misma expresión. En muchas ocasiones, es posible que se quiera comparar la misma variable (o expresión) con muchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a que valor es igual. Para esto es exactamente la expresión switch.

Ejemplo: Estructura switch

```
<?php
if ($i == 0) {
    echo "i es igual a 0";
} elseif ($i == 1) {
    echo "i es igual a 1";
} elseif ($i == 2) {
    echo "i es igual a 2";
}

switch ($i) {
    case 0:
        echo "i es igual a 0";
        break;
    case 1:
        echo "i es igual a 1";
        break;
    case 2:
        echo "i es igual a 2";
        break;
}
?>
```

2.2.2.9.11 declare

El constructor declare es usado para fijar directivas de ejecución para un bloque de código. La sintaxis de declare es similar a la sintaxis de otros constructores de control de flujo:

```
declare (directive)

statement
```

La sección directive permite que el comportamiento de declare sea configurado. Actualmente, sólo dos directivas están reconocidas:

La parte statement del bloque declare será ejecutada - como se ejecuta y que efectos secundarios ocurran durante la ejecución puede depender de la directiva fijada en el bloque directive.

El constructor declare también se puede utilizar en el alcance global, afectando a todo el código que le sigue sin embargo, si el archivo con el declare fue incluido entonces no afectará al archivo padre.

```
<?php
// estos son lo mismo:

// se puede usar ésto:
declare(ticks=1) {
    // script entero aquí
}

// o se puede usar ésto:
declare(ticks=1);
// script entero aquí
?>
```

2.2.2.9.12 Return

Si se llama desde una función, la sentencia **return()** inmediatamente termina la ejecución de la función actual, y devuelve su argumento como el valor de la llamada a la función. **return()** también pondrá fin a la ejecución de una sentencia eval() o a un archivo de script.

Si se llama desde el ámbito global, entonces la ejecución del script actual se termina. Si el archivo script actual fue incluido con **include()** o **require()**, entonces el control es pasado de regreso al archivo que hizo el llamado. Además, si el archivo script actual fue incluido, entonces el valor dado a **return()** será retornado como el valor de la llamada **include()**. Si **return()** es llamado desde dentro del archivo script principal, entonces termina la ejecución del script. Si el archivo script actual fue nombrado por las opciones de configuración auto_prepend_file o auto_append_file en php.ini, entonces se termina la ejecución de ese archivo script.

2.2.2.9.13 REQUIRE

require() es idéntico a **include()** excepto que en caso de fallo producirá un error fatal de nivel E_COMPILE_ERROR. En otras palabras, éste detiene el script mientras que **include()** sólo emitirá una advertencia (E_WARNING) lo cual permite continuar el script.

2.2.2.9.14 INCLUDE

La sentencia **include()** incluye y evalúa el archivo especificado.

La siguiente documentación también se aplica a **require()**.

Los archivos son incluidos con base en la ruta de acceso dada o, si ninguna es dada, el include_path especificado. Si el archivo no se encuentra en el include_path, **include()** finalmente verificará en el propio directorio del script que hace el llamado y en el directorio de trabajo actual, antes de fallar. El constructor **include()** emitirá una advertencia si no puede encontrar un archivo, éste es un comportamiento diferente al de **require()**, el cual emitirá un error fatal..

Si una ruta es definida — ya sea absoluta (comenzando con una letra de unidad o \ en Windows o / en sistemas Unix/Linux) o relativa al directorio actual (comenzando con . o ..) — el include_path será ignorado por completo. Por ejemplo, si un nombre de archivo comienza con ../, el interprete buscará en el directorio padre para encontrar el archivo solicitado.

Para más información sobre como PHP maneja la inclusión de archivos y la ruta de accesos para incluir, ver la documentación de include_path.

Cuando se incluye un archivo, el código que contiene hereda el ámbito de las variables de la línea en la cual ocurre la inclusión. Cualquier variable disponible en esa línea del archivo que hace el llamado, estará disponible en el archivo llamado, desde ese punto

en adelante. Sin embargo, todas las funciones y clases definidas en el archivo incluido tienen el ámbito global.

Ejemplo básico de include()

```
vars.php
<?php

$color = 'verde';
$fruta = 'manzana';
?>
test.php
<?php

echo "Una $fruta $color"; // Una

include 'vars.php';

echo "Una $fruta $color"; // Una manzana verde

?>
```

Si la inclusión ocurre al interior de una función dentro del archivo que hace el llamado, entonces todo el código contenido en el archivo llamado se comportará como si hubiera sido definida dentro de esa función

require_once()

La sentencia **require_once()** es idéntica a **require()** excepto que PHP verificará si el archivo ya ha sido incluido y si es así, no se incluye (require) de nuevo.

include_once()

La sentencia **include_once()** incluye y evalúa el fichero especificado durante la ejecución del script. Es un comportamiento similar al de la sentencia **include()**,

include_once() puede ser usado en casos donde el mismo fichero podría ser incluido y evaluado más de una vez durante una ejecución particular de un script, así que en este caso, puede ayudar a evitar problemas como la redefinición de funciones, reasignación de valores de variables, etc.

2.2.2.10 Funciones

2.2.2.10.1 Funciones definidas por el usuario

Una función puede ser definida usando una sintaxis como la siguiente:

Ejemplo #1 Pseudo código para demostrar el uso de funciones

```
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
    echo "Función de ejemplo.\n";
    return $valordevuelto;
}
?>
```

Cualquier código PHP válido puede aparecer dentro de una función, incluso otras funciones y definiciones de clases.

Los nombres de las funciones siguen las mismas reglas que otras etiquetas de PHP. Un nombre de función válido comienza con una letra o guión bajo, seguido de cualquier número de letras, números, o guiones bajos. Como expresión regular se expresaría así: [a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*.

Argumentos de funciones

La información puede ser pasada a las funciones mediante la lista de argumentos, la cual es una lista de expresiones delimitadas por comas. Los argumentos son evaluados de izquierda a derecha.

PHP soporta argumentos pasados por valor (por defecto), pasados por referencia, y valores de argumentos predeterminados. Las Listas de argumentos de longitud variable también están soportadas, vea también las referencias de funciones para func_num_args(), func_get_arg(), y func_get_args().

Devolver valores

Los valores son devueltos usando la sentencia opcional return. Se puede devolver cualquier tipo, incluidos arrays y objetos. Esto causa que la función finalice su ejecución inmediatamente y pase el control de nuevo a la línea desde la que fue llamada.

2.2.2.10.2 Funciones variables

PHP soporta el concepto de funciones variables. Esto significa que si un nombre de variable tiene paréntesis anexos a él, PHP buscará una función con el mismo nombre que lo evaluado por la variable, e intentará ejecutarla. Entre otras cosas, esto se puede usar para implementar llamadas de retorno, tablas de funciones, y así sucesivamente.

Las funciones variables no funcionarán con constructores de lenguaje como echo(), print(), unset(), isset(), empty(), **include()**, **require()** y similares. Utilice funciones de envoltura para hacer uso de cualquiera de estos constructores como funciones variables.

2.2.2.10.3 Funciones internas (incluidas)

PHP se estandariza con muchas funciones y construcciones. También existen funciones que necesitan extensiones específicas de PHP compiladas, si no, aparecerán errores fatales "undefined function" ("función no definida"). Por ejemplo, para usar las funciones de image tales como imagecreatetruecolor (), PHP debe ser compilado con soporte para GD. O para usar mysql_connect(), PHP debe ser compilado con soporte para MySQL. Hay muchas funciones de núcleo que está incluidas en cada versión de PHP, tales como las funciones de string y de variable. Una llamada a phpinfo() o get_loaded_extensions() mostrará las extensiones que están cargadas en PHP. Tambiénobservamos que muchas extensiones están habilitadas por defecto y que el manual de PHP está dividido por extensiones.

Interpretar y comprender un prototipo de una función está explicado dentro de la sección del manual titulada cómo interpretar la definición de una función. Es importante comprender lo que devuelve una función o si una función funciona directamente con un valor pasado. Por ejemplo, str_replace() devolverá la cadena modificada mientras que usort() funciona con la variable actual pasada.

2.2.2.10.4 Funciones anónimas

Las funciones anónimas, también conocidas como closures (cierres), permiten la creación de funciones que no tienen un nombre especificado. Son más útiles como valor de los parámetros de llamadas de retorno, pero tienen muchos otros usos.

Ejemplo #1 Ejemplo de función anónima

```
<?php
echo preg_replace_callback('~-([a-z])~', function ($coincidencia) {
    return strtoupper($coincidencia[1]);
}, 'hola-mundo');
// imprime holaMundo
?>
```

2.2.2.11 Clases y Objetos

A partir de PHP 5, el modelo de objetos ha sido reescrito para permitir un mejor rendimiento y con más características. Este fue un cambio importante a partir de PHP 4. PHP 5 tiene un modelo de objetos completo.

Entre las características de PHP 5 están la inclusión de la visibilidad, las clases abstractas y clases y métodos finales, métodos mágicos adicionales, interfaces, clonación y tipos sugeridos.

PHP trata los objetos de la misma manera como referencias o manejadores, lo que significa que cada variable contiene una referencia de objeto en lugar de una copia de todo el objeto.

2.2.2.11.1 Class

La definición básica de clases comienza con la palabra clave class, seguido por un nombre de clase, continuado por un par de llaves que encierran las definiciones de las propiedades y métodos pertenecientes a la clase.

El nombre de clase puede ser cualquier etiqueta válida que no sea una palabra reservada de PHP. Un nombre válido de clase comienza con una letra o un guión bajo, seguido de la cantidad de letras, números o guiones bajos que sea. Como una expresión regular, se expresaría de la siguiente forma: [a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*.

Una clase puede tener sus propias constantes, variables (llamadas "propiedades"), y funciones (llamadas "métodos").

Ejemplo #1 Definición simple de una clase

```
<?php
class SimpleClass
{
    // Declaración de la propiedad
    public $var = 'a default value';

    // Declaración del método
    public function displayVar() {
        echo $this->var;
    }
}
?>
```

2.2.2.10.2Propiedades

Las variables pertenecientes a clases son llamadas "propiedades". También se les puede referir usando otros términos como "atributos" o "campos", pero para los propósitos de esta referencia vamos a utilizar "propiedades". Éstas se definen usando una de las palabras claves public, protected, o private, seguido por una declaración normal de variable. Esta declaración puede incluir una inicialización, pero esta inicialización debe ser un valor constante, es decir, debe tener la capacidad de ser evaluada en la compilación y no debe depender de información en tiempo de ejecución para ser evaluada.

La pseudo-variable $this está disponible dentro de cualquier método de clase cuando éste es invocado dentro del contexto de un objeto. $this es una referencia del objeto que invoca (usualmente el objeto al que el método pertenece, pero posiblemente sea otro objeto, si el método es llamado estáticamente desde el contexto de un objeto secundario).

Ejemplo #1 Declaración de propiedades

```
<?php
class SimpleClass
{
   // Declaraciones inválida de propiedades:
   public $var1 = 'hola ' . 'mundo';
```

```
  public $var2 = <<<EOD
hola mundo
EOD;
  public $var3 = 1+2;
  public $var4 = self::myStaticMethod();
  public $var5 = $myVar;

  // Declaraciones válida de propiedades:
  public $var6 = myConstant;
  public $var7 = array(true, false);

  // Esto se permite sólo en PHP 5.3.0 y superiores.
  public $var8 = <<<'EOD'
hola mundo
EOD;
}
?>
```

2.2.2.11.3Constantes de Clases

Es posible definir valores constantes en función de cada clase manteniéndola invariable. Las constantes se diferencian de variables comunes en que no utilizan el símbolo $ al declararlas o usarlas.

El valor debe ser una expresión constante, no por ejemplo: una variable, una propiedad, un resultado de una operación matemática, o una llamada a una función.

A partir de PHP 5.3.0, es posible hacer referencia a una clase utilizando una variable. El valor de la variable no puede ser una palabra clave (por ej., self, parent y static).

Ejemplo #1 Definición y uso de una constante

```
<?php
class MyClass
{
    const constant = 'valor constante';

    function showConstant() {
        echo  self::constant . "\n";
```

```
    }
}

echo MyClass::constant . "\n";

$classname = "MyClass";
echo $classname::constant . "\n"; // A partir de PHP 5.3.0

$class = new MyClass();
$class->showConstant();

echo $class::constant."\n"; // A partir de PHP 5.3.0
?>
```

2.2.2.11.4Auto-carga de Clases

Muchos desarrolladores que escriben aplicaciones orientadas a objetos crean un archivo fuente PHP para cada definición de clase. Una de las mayores molestias es tener que hacer una larga lista de includes al comienzo de cada script, uno por cada clase.

En PHP 5, esto ya no es más necesario. Se puede definir una función __autoload() la cual es automáticamente invocada en caso de que esté intentando utilizar una clase/interfaz que todavía no haya sido definida. Al invocar a esta función el motor de scripting está dando una última oportunidad de cargar la clase antes que PHP falle con un error.

2.2.2.11.5Constructores y destructores

Constructor

void__**construct** ([mixed$args [, $...]])

PHP 5 permite a los desarrolladores declarar métodos constructores para las clases. Aquellas que tengan un método constructor lo invocarán en cada nuevo objeto creado, lo que lo hace idóneo para cualquier inicialización que el objeto pueda necesitar antes de ser usado.

Ejemplo #1 Utilización de nuevos constructores unificados

```
<?php
class BaseClass {
   function __construct() {
       print "En el constructor de BaseClass\n";
   }
}

class SubClass extends BaseClass {
   function __construct() {
       parent::__construct();
       print "En el constructor de SubClass\n";
   }
}

$obj = new BaseClass();
$obj = new SubClass();
?>
```

2.2.2.11.6 PROPIEDADES

2.2.2.11.6.1Visibilidad

La visibilidad de una propiedad o método se puede definir anteponiendo una de las palabras claves public, protected o private en la declaración. Miembros de clases declarados como public pueden ser accedidos de cualquier lado. Miembros declarados como protected, sólo de la clase misma, por herencia y clases parent. Aquellos definidos como private, únicamentede la clase que los definió.

2.2.2.11.6.2 Visibilidad de Propiedades

Las propiedades de clases deben ser definidas como public, private, o protected. Si se declaran usando var, serán definidas como public.

Ejemplo #1 Declaración de propiedades

```
<?php
/**
 * Definición de MyClass
```

```
 */
class MyClass
{
    public $public = 'Public';
    protected $protected = 'Protected';
    private $private = 'Private';

    function printHello()
    {
        echo $this->public;
        echo $this->protected;
        echo $this->private;
    }
}

$obj = new MyClass();
echo $obj->public; // Funciona
echo $obj->protected; // Error Fatal
echo $obj->private; // Error Fatal
$obj->printHello(); // Muestra Public, Protected y Private



/**
 * Definición de MyClass2
 */
class MyClass2 extends MyClass
{
    // Se puede redeclarar los métodos public y protected, pero no el private
    protected $protected = 'Protected2';

    function printHello()
    {
        echo $this->public;
        echo $this->protected;
        echo $this->private;
    }
}
```

```
$obj2 = new MyClass2();
echo $obj2->public; // Funciona
echo $obj2->private; // Undefined
echo $obj2->protected; // Error Fatal
$obj2->printHello(); // Muestra Public, Protected2, Undefined

?>
```

2.2.2.11.6.3 Visibilidad

La visibilidad de una propiedad o método se puede definir anteponiendo una de las palabras claves public, protected o private en la declaración. Miembros de clases declarados como public pueden ser accedidos de cualquier lado. Miembros declarados como protected, sólo de la clase misma, por herencia y clases parent. Aquellos definidos como private, únicamentede la clase que los definió.

2.2.2.11.6.4 Visibilidad de Propiedades

Las propiedades de clases deben ser definidas como public, private, o protected. Si se declaran usando var, serán definidas como public.

2.2.2.11.7 Herencia de Objetos

La herencia es un principio de programación bien establecido y PHP hace uso de él en su modelado de objetos. Este principio afectará la manera en que muchas clases y objetos se relacionan unas con otras.

Por ejemplo, cuando se extiende una clase, la subclase hereda todos los métodos públicos y protegidos de la clase padre. A menos que una clase sobrescriba esos métodos, mantendrán su funcionalidad original.

2.2.2.11.8 Palabra Clave Static

Declarar propiedades o métodos de clases como estáticos los hacen accesibles sin necesidad de una instanciación de la clase. Una propiedad declarada como static no puede ser accedida con un objeto de clase instanciado, pero sí se puede con métodos estáticos.

Por motivos de compatibilidad con PHP 4, si no se utiliza ninguna declaración de visibilidad, se tratará a las propiedades o métodos como si hubiesen sido definidos como public.

Debido a que los métodos estáticos se pueden invocar sin tener creada una instancia del objeto, la pseudo-variable $this no está disponible dentro de los métodos declarados como static.

Las propiedades estáticas no pueden ser accedidas a través del objeto utilizando el operador flecha (->).

Invocar métodos no estáticos estáticamente genera un Warning a nivel de E_STRICT.

Como cualquier otra variable estática en PHP, las propiedades estáticas sólo pueden ser inicializadas utilizando una string literal o una constante; las expresiones no están permitidas. Por tanto, puede inicializar una propiedad estática con enteros o arrays por ejemplo, pero no puede hacerlo con otra variable, con el valor de devolución de una función, o con un objeto.

A partir de PHP 5.3.0, es posible hacer referencia a una clase usando una variable. El valor de la variable no puede ser una palabra clave por ej., self, parent y static.

2.2.2.11.9 Abstracción de clases

PHP 5 introduce clases y métodos abstractos. Las clases definidas como abstract seguramente no son instanciadas y cualquier clase que contiene almenos un método abstracto debe ser definida como abstract. Los métodos definidos como abstractos simplemente declaran la estructura del método, pero no pueden definir la implementación.

Cuando se hereda de una clase abstracta, todos los métodos definidos como abstract en la definición de la clase parent deben ser redefinidos en la clase child; adicionalmente, estos métodos deben ser definidos con la misma visibilidad o con una menos restrictiva. Por ejemplo, si el método abstracto está definido como protected, la implementación de la función puede ser redefinida como protected o public, pero nunca como private. Por otra parte, las estructuras de los métodos tienen que coincidir, es decir, los (type hinting) tipos sugeridos y el número de argumentos requeridos deben ser los mismos.

2.2.2.11.10 Enlace estático en tiempo de ejecución

Desde PHP 5.3.0, PHP incorpora una nueva funcionalidad llamada enlace estático en tiempo de ejecución que permite hacer referencias a la clase en uso dentro de un contexto de herencia estática.

De forma más precisa, un enlace estático en tiempo de ejecución para funcionar almacena el nombre de clase de la última llamada que no tenga "propagación". En el caso de las llamadas a métodos estáticos, se trata de la clase a la que se llamó explícitamente (normalmente, la que precede al operador ::); en los casos de llamadas a métodos que no son estáticos, se resolvería a la clase del objeto. Una "llamada con propagación" es una llamada estática que está precedida por self::,parent::, static::, o, si seguimos la jerarquía de clases, forward_static_call(). La función get_called_class()puede utilizarse para obtener un string con el nombre de la clase que realiza la llamada, y static:: revela cuál es su alcance.

2.2.2.11.11 Objetos y referencias

Uno de los puntos clave de la POO de PHP 5 que a menudo se menciona es que "por omisión los objetos se pasan por referencia". Esto no es completamente cierto. Esta sección rectifica esa creencia general, usando algunos ejemplos.

Una referencia en PHP es un alias, que permite a dos variables diferentes escribir sobre un mismo valor. Desde PHP 5, una variable de tipo objeto ya no contiene el objeto en sí como valor. Únicamente contiene un identificador del objeto que le permite localizar al objeto real. Cuando se pasa un objeto como parámetro, o se devuelve como retorno, o se asigna a otra variable, las distintas variables no son alias: guardan una copia del identificador, que apunta al mismo objeto.

2.2.2.11.11.1 Serialización de objetos

Serialización de objetos - objetos en sesiones

La función serialize () devuelve un string que contiene un flujo de bytes que representa cualquier valor que se pueda almacenar en PHP. Por otra parte, unserialize() puede restaurar los valores originales a partir de dicho string. Al utilizar serialize para guardar un objeto, almacenará todas las variables de dicho objeto. En cambio los métodos no se guardarán, sólo el nombre de la clase.

Para poder deserializar (unserialize()) un objeto, debe estar definida la clase de ese objeto. Es decir, si se tiene un objeto de la clase A, y lo serializamos, se obtendrá un string que haga referencia a la clase A y contenga todas las variables que haya en esta clase. Si se desea deseralizar en otro fichero, antes debe estar presente la definición de la clase A. Esto se puede hacer, por ejemplo, escribiendo la definición de la clase A en un fichero, para después o bien incluirlo, o bien hacer uso de la función spl_autoload_register().

```
<?php
// classa.inc:

  class A {
      public $one = 1;

      public function show_one() {
          echo $this->one;
      }
  }

// page1.php:

  include("classa.inc");

  $a = new A;
  $s = serialize($a);
  // almacenamos $s en algún lugar en el que page2.php puede encontrarlo.
  file_put_contents('store', $s);

// page2.php:

  // se necesita para que unserialize funcione correctamente.
  include("classa.inc");

  $s = file_get_contents('store');
  $a = unserialize($s);

  // now use the function show_one() of the $a object.
```

```
 $a->show_one();
?>
```

2.2.2.11.11.2 Qué son las Referencias?

Las Referencias en PHP son medios de acceder al mismo contenido de una variable mediante diferentes nombres. No son como los punteros de C; por ejemplo, no se puede realizar aritmética de punteros con ellas, realmente no son direcciones de memoria, etc. Observe que en PHP el nombre de la variable y el contenido de la variable son cosas diferentes, por lo que el mismo contenido puede tener diferentes nombres. La analogía más próxima es con los archivos y los nombres de archivos de Unix - los nombres de variables son entradas de directorio, mientras que el contenido de las variables es el archivo en sí. Las referencias se pueden vincular a enlaces duros en sistemas de archivos Unix.

2.2.2.12 Seguridad

PHP es un potente lenguaje, y su intérprete, bien como módulo del servidor web o bien como binario CGI, puede acceder a ficheros, ejecutar comandos o abrir conexiones de red desde el servidor. Estas propiedades hacen que, por omisión, sea inseguro todo lo que se ejecute en un servidor web. PHP está diseñado específicamente para ser un lenguaje más seguro para escribir aplicaciones CGI que Perl or C. Partiendo de un correcto ajuste de opciones de configuración para tiempo de ejecución y en tiempo de compilación, y el uso de prácticas de programación apropiadas, pueden proporcionarle la combinación de libertad y de seguridad que necesita.

La flexibilidad de configuración de PHP rivaliza igualmente con la flexibilidad de su código. PHP puede ser usado para construir completas aplicaciones de servidor, con toda la potencia de un usuario de consola, o se puede usar sólo desde el lado del servidor implicando un menor riesgo dentro de un entorno controlado. El cómo construir ese entorno, y cómo de seguro es, depende del desarrollador PHP.

2.2.2.12.1 Seguridad de Bases de Datos

Hoy en día, las bases de datos son componentes cardinales de cualquier aplicación basada en la web permitiendo a los sitios web que tengan una variedad de contenido dinámico. Esta información muy sensible o secreta puede ser almacenada en una base de datos, por lo que debe considerar fuertemente proteger su base de datos.

Para devolver o almacenar cualquier información usted necesita conectarse a la base de datos, enviar una consulta legítima, devolver el resultado, y cerrar la conexión. Hoy en día, el lenguaje de consultas comunmente utilizado en esta interacción es el Lenguaje Estructurado de Consultas SQL.

Como puede suponer, PHP no protege su base de datos por sí mismo. Las siguientes secciones piensan ser una introducción a lo más básico de cómo acceder y manipular base de datos dentro de scripts de PHP.

2.2.2.12.2 Diseñando la base de datos

El primer paso es siempre crear una base de datos, a menos que quiera utilizar una de terceras personas. Cuando una base de datos es creada, ésta es asignada a un propietario, el que ha ejecutado la sentencia de creación. Usualmente, sólo el propietario o un superusuariopuede hacer cualquier cosa con los objetos en esa base de datos, y para permitir a otros usuarios que puedan utilizarla, debe concederles privilegios.

Puede crear distintos usuarios de la base de datos para cada aspecto de su aplicación con permisos muy limitados a los objetos. La mayoría de privilegios que son requeridos deberían ser solamente otorgados, y así evitar que el mismo usuario pueda interactuar con la base de datos en diferentes casos y usos.

2.2.3. MYSQL

2.2.3.1 Definición de MySQL

[1]**MySQL** es un sistema de gestión de bases de datosrelacional, multihilo y multiusuario con más de seis millones de instalaciones.MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

2.2.3.2 USO DEL CLIENTE MYSQL

El objetivo es mostrar el uso del programa cliente mysql para crear y usar una sencilla base de datos. mysqlalgunas veces referido como "monitor mysql", es un programa interactivo que permite conectarnos a un servidor MySQL, ejecutar algunas consultas, y ver los resultados. mysql puede ser usado también en modo batch: es decir, se pueden colocar toda una serie de consultas en un archivo, y posteriormente decirle a mysql que ejecute dichas consultas.Una vez que mysql está instalado en alguna máquina y que disponemos de un servidor MySQL al cuál podemos conectarnos. Si este no es el caso, tenemos que contactar con nuestro administrador MySQL.Para ver la lista de opciones proporcionadas por mysql, lo invocamos con la opción --help: shell> mysql --help A continuación se describe el proceso completo de creación y uso de una base de datos en MySQL. Para conectarse al servidor, usualmente necesitamos de un nombre de usuario (login) y de una contraseña (password), y si elservidor al que nos deseamos conectar está en una máquina diferente de la nuestra, también necesitamos indicar el nombre o la dirección IP de dicho servidor. Una vez que conocemos estos tres valores, podemos conectarnos de la siguiente manera: shell> mysql -h NombreDelServidor -u NombreDeUsuario -p Cuando ejecutamos este comando, se nos pedirá que proporcionemos también la contraseña para el nombre de usuario que estamos usando.

Si la conexión al servidor MySQL se pudo establecer de manera satisfactoria, recibiremos el mensaje de bienvenida y estaremos en el prompt de mysql :

shell>mysql -h casita -u blueman -p Enter password: ****** Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 5563 to server version: 3.23.41 Type 'help;' or '\h' for help. Type '\c' to clear the buffer. mysql>

Este prompt nos indica que mysql está listo para recibir comandos. Algunas instalaciones permiten que los usuarios se conecten de manera anónima al servidor corriendo en la máquina local. Si es el caso de nuestra máquina, debemos de ser capaces de conectarnos al servidor invocando a mysql sin ninguna opción.

[1] http://dev.mysql.com/doc/internals/en

2.2.3.3 LENGUAJES DE PROGRAMACIÓN

[2]Existen varias interfaces de programación de aplicaciones que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySQL, incluyendo C, C++, C#, Pascal, Delphi (via dbExpress), Eiffel, Smalltalk, Java (con una implementación nativa del driver de Java), Lisp, Perl, PHP, Python.

Cada uno de estos utiliza una interfaz de programación de aplicaciones específica. También existe una interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL.

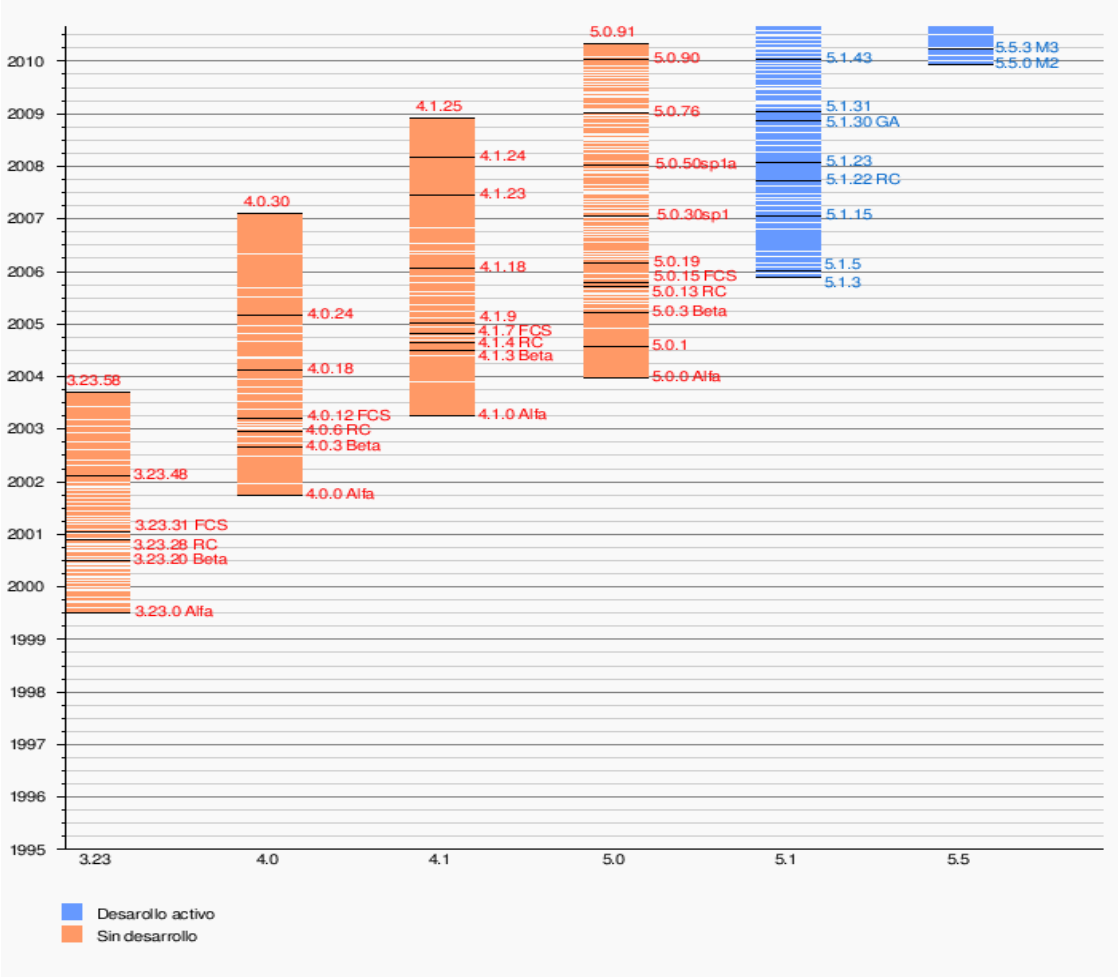
2.2.3.4 APLICACIONES

MySQL es muy utilizado en aplicaciones web, como Drupal o phpBB, en plataformas Linux/Windows-Apache-MySQL-PHP/Perl/Python, y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL.MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación.

En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones. Sea cual sea el entorno en el que va a utilizar MySQL, es importante monitorizar de antemano el rendimiento para detectar y corregir errores tanto de SQL como de programación.

[2]http://dev.mysql.com/doc/internals/en

2.2.3.5VERSIONES DE MYSQL



2.2.3.6 Plataformas

MySQL funciona sobre múltiples plataformas, incluyendo:

- AIX
- BSD
- FreeBSD
- HP-UX
- Kurisu OS
- GNU/Linux
- Mac OS X
- NetBSD
- Novell Netware
- OpenBSD
- OS/2 Warp

- QNX
- SGI IRIX
- Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Vista, Windows 7 y Windows Server (2000, 2003 y 2008).
- OpenVMS[10]

2.2.3.7 Características

[3]Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones.Poco a poco los elementos de los que carecía MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

- Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.

- Disponibilidad en gran cantidad de plataformas y sistemas.
  - Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferente velocidad de operación, soporte físico, capacidad, distribución geográfica, transacciones...
  - Transacciones y claves foráneas.
  - Conectividad segura.
  - Replicación.
  - Búsqueda e indexación de campos de texto. MySQL es un sistema de
  - administración de bases de datos. Una base de datos es una colección estructurada de tablas que contienen datos. Esta puede ser desde una simple lista de compras a una galería de pinturas o el vasto volumen de información en una red corporativa. Para agregar, acceder a y procesar datos guardados en un computador, usted necesita un administrador como MySQL Server. Dado que los computadores son muy buenos manejando grandes cantidades de información, los administradores de bases de datos juegan un papel central en computación, como aplicaciones independientes o como parte de otras aplicaciones.

[3]http//www.programacion.com/php/kelvin torres

MySQL es un sistema de administración relacional de bases de datos. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido.

MySQL es software de fuente abierta. Fuente abierta significa que es posible para cualquier persona usarlo y modificarlo. Cualquier persona puede bajar el código fuente de MySQL y usarlo sin pagar. Cualquier interesado puede estudiar el código fuente y ajustarlo a sus necesidades. MySQL usa el GPL

(GNU General Public License) para definir que puede hacer y que no puede hacer con el software en diferentes situaciones. Si usted no se ajusta al GPL o requiere introducir código MySQL en aplicaciones comerciales, usted puede comprar una versión comercial licenciada.

2.2.3.7.1Características distintivas

- Las siguientes características son implementadas únicamente por MySQL:
- Permite escoger entre múltiples motores de almacenamiento para cada tabla. En MySQL 5.0 éstos debían añadirse en tiempo de compilación, a partir de MySQL 5.1 se pueden añadir dinámicamente en tiempo de ejecución:
    - Los hay nativos como MyISAM, Falcon, Merge, InnoDB, BDB, Memory/heap, MySQL Cluster, Federated, Archive, CSV, Blackhole y Example
    - Desarrollados por partners como solidDB, NitroEDB, ScaleDB, TokuDB, Infobright (antes Brighthouse), Kickfire, XtraDB, IBM DB2). InnoDB Estuvo desarrollado así pero ahora pertenece también a Oracle
    - Desarrollados por la comunidad como memcache, httpd, PBXT y Revision
- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

2.2.3.8Tipos de compilación del servidor

Hay tres tipos de compilación del servidor MySQL:

- Estándar: Los binarios estándares de **MySQL** son los recomendados para la mayoría de los usuarios, e incluyen el motor de almacenamiento InnoDB.
- Max (No se trata de MaxDB, que es una cooperación con SAP): Los binarios incluyen características adicionales que no han sido lo bastante probadas o que normalmente no son necesarias.
- MySQL-Debug: Son binarios que han sido compilados con información de depuración extra. No debe ser usada en sistemas en producción porque el código de depuración puede reducir el rendimiento.

2.2.3.9 Especificaciones del código fuente

MySQL está escrito en una mezcla de C y C++. Hay un documento que describe algunas de sus estructuras internas en el código de esta página

2.2.3.10 Desarrollo del proyecto

El desarrollo de MySQL se fundamenta en el trabajo de los desarrolladores contratados por la empresa MySQL AB quienes se encargan de dar soporte a los socios comerciales y usuarios de la comunidad MySQL y dar solución a los problemas encontrados por los usuarios. Los usuarios o miembros de la comunidad MySQL pueden reportar bugs revisando el manual en línea que contiene las soluciones a problemas encontrados; el historial de cambios ; la base de datos bugs que contiene bugs reportados y solucionados y en las listas de correo MySQL.

2.2.3.11 Estructuras organizativas/asociativas o de decisión

La dirección y el patrocinio de los proyectos MySQL están a cargo de la empresa MySQL AB quien posee el copyrigth del código fuente MySQL, su logo y marca registrada. MySQL, Inc. y MySQL GmbH son ejemplos de empresas subsidiarias de MySQL AB. Están establecidas en los Estados Unidos y Alemania respectivamente. MySQL AB, cuenta con más de 200 empleados en más de 20 países y funcionan bajo la estrategia de teletrabajo.

2.2.3.12 Licencia

La licencia GNU GPL de MySQL obliga a que la distribución de cualquier producto derivado (aplicación) se haga bajo esa misma licencia. Si un desarrollador desea incorporar MySQL en su producto pero desea distribuirlo bajo otra licencia que no sea la GNU GPL, puede adquirir una licencia comercial de MySQL que le permite hacer justamente eso

2.3 HIPOTESIS Y VARIABLES

2.3.1 HIPÓTESIS GENERAL

La aplicación de un sistema informatico sobre cuidados, crianza y ventas mejoraría la gestión administrativa de la Agropecuaria “VELBAMAL” de la ciudad de Vinces.

2.3.1.1 HIPÓTESIS ESPECÍFICAS

- Se obtendría un registro detallado de Ganados Vacunos, de un modo más seguro ya que contaría con una base de datos que estará ligada al sistema a aplicar.

- Los informes a realizar por parte de la gerencia de la Agropecuaria estarían estrictamente controlados para evitar los errores que se presentan con la forma manual.

- La sistematización de los procesos administrativos permitiría controlar la información producida en las actividades de la Agropecuaria.

2.3.2 VARIABLES E IDENTIFICADORES

2.3.2.1 VARIABLE INDEPENDIENTE

Gestión de Ganado basado en el Sistema PHP

La informática ha puesto una revolución en la última década, su utilidad en la vida cotidiana resulta indiscutible, los campos de aplicación cada día son mayores. Por lo que es necesario un software para generar eficiencia administrativa y operativa que le permita conocer cuando y como se asignan los recursos, en una buena Crianza de Vacas y Toros, Y poder darles ayuda a sus clientes cuando ellos le realicen consulta, en una aplicación generada en PHP, con la base de datos en MYSQL.

2.3.2.2 VARIABLE DEPENDIENTE

Gestión de Cuidado, Crianza y Ventas de la Agropecuaria

La Gestión de Cuidado, Crianza y Venta esla actividad de laAgropecuaria "VELBAMAL" para definir, alcanzar y evaluar sus propósitos con el adecuado uso de los recursos disponibles para conseguir determinados objetivos.

2.3.2.2.1 Descripción de Identificadores

Meses

Tiempo de respuesta de los procesos realizados por el Sistema de Cuidados, Crianza, Ventas mediante el Sistema PHP Runner.

2.4 OPERACIONALIZACION DE LAS VARIABLES

HIPOTESIS	VARIABLES	DEFINICION CONCEPTUAL	PARAMETROS	INDICADORES	INSTRUMENTOS
La aplicación de un sistema de cuidados, crianza y ventas mejorara la gestión administrativa de la agropecuaria mediante la optimización de sus procesos. ➢ Se obtendrá un registro detallado de las Vacas y toros, de un modo seguro y confiable. ➢ Los informes a realizar por parte de la gerencia estarán estrictamente controlados. ➢ La sistematización de los procesos permitirá controlar la información producida en las actividades de la Agropecuaria. ➢ La herramienta proporcionada facilitara el funcionamiento de los procesos administrativos. ➢ Se facilitara el acceso de los Precios de las vacas y toros para realizar rapidas eficientes ventas.	Gestión de Ganado basado en el Sistema PHP Gestión de cuidado, crianza y ventas de la Agropecuaria	La informática ha puesto una revolución en la última década, su utilidad en la vida cotidiana resulta indiscutible, los campos de aplicación cada día son mayores.Generar eficiencia administrativa y operativa que le permita conocer cuando y como se asignan los recursos, en una buena crianza de vacas y toros, y poder darle ayuda a sus clientes cuando ellos le realicen consulta, en una aplicación generada en PHP, con la base de datos en MYSQL. La Gestión de Cuidado, Crianza y Venta es la actividad de la Agropecuaria "VELBAMAL" para definir, alcanzar y evaluar sus propósitos con el adecuado uso de los recursos disponibles para conseguir determinados objetivos.	✓ Base de Datos Tiempo	PROGRAMACION OPEN SOURCE Meses	✓ Programación PHP Runner ✓ MYSQL ✓ Revisión ✓ Documental ✓ Entrevista ✓ Encuestas ✓ Observación

CAPITULO III

3. MARCO METODOLOGICO

3.1 MODALIDAD DE LA INVESTIGACION

En este capítulo semuestra la metodología mediante la cual se obtuvo la información para la elaboración de la presente investigación. Teniendo como finalidad dar a conocer el método que se siguió para la recopilación de datos y posteriormente el análisis de datos. Esta se llevará a cabo en dos etapas: la primera por una investigación tipo exploratoria y la segunda por una recopilación de datos, aplicando un cuestionario de satisfacción al cliente en el la Agropecuaria que es objeto de estudio.

3.2 TIPO DE INVESTIGACION

La primera etapa está formada por una investigación de tipo exploratoria, debido a que esta sirve para familiarizarse con la agropecuaria que será objeto de estudio y poder obtener la información sobre la posibilidad de llevar a cabo una investigación completa.

Mientras que la segunda etapa estará formada por la recopilación de datos, mediante el cuestionario de satisfacción al cliente. Esta etapa sustenta la investigación bibligrafica y de campo, ya que se darán a conocer las características, ventajas y deficiencias de la calidad en el servicio que existe en la Agropecuaria "VELBAMAL".

3.2.1 INVESTIGACION BIBLIOGRAFICA

Nuestras fuentes primarias de investigación para el desarrollo de nuestra tésisde Sistema de Gestión Agropecuaria serán las distintas ordenanzas elaboradas y aprobadas por la misma agropecuria.

3.2.2 INVESTIGACION DE CAMPO

Nuestra investigación fue realizada en el departamento de adminstración, donde se concentra nuestro tema de tesis, por lo que fue allí donde realizamos las investigaciones y consulta y como se llevo a cabo la recolección de los permisos dándonos fuentes al contribuyente y también para poder mantener la información correcta.

3.2.3 POBLACION Y MUESTRA DE LA INVESTIGACION

3.2.3.1 UNIVERSO O POBLACIÓN

La Agropecuaria “VELBAMAL”, está localizado en el Cantón Vinces, de la Provincia de Los Ríos, Parroquia Santa Martha. La población objeto de la investigación estará constituida por los empleados de la agropecuaria “VELBAMAL”, del cual tomaremos un tamaño muestral para poblaciones infinitas para realizar la encuesta.

Por otra parte se realizara varias entrevistas a los Propietarios y empleados de la agropecuaria, el mismo que cuenta con un total de: 1 Jefe de Sector o Administrador, 1 Secretaria, 4 Personas de Limpieza, 4 Personas de Cuidados, 2 Personas para las Ventas, 1 Responsable de Transporte, 1 Mensajero y 2 guardianes. Resulta inconveniente entrevistar al total de los sujetos de estudio pues la investigación procura obstaculizar en lo más mínimo las operaciones diarias de la Agropecuaria, previendo que algún Cliente pudiera molestarse por alguna alteración del servicio que está acostumbrado a recibir. Por tal motivo se selecciono una muestra aleatoria confiable que sea representativa de la población total.

3.2.3.2 TAMAÑO DE LA MUESTRA

Para la recolección de la información se tomará una muestra representativa de los empleados de la agropecuaria.

A continuación las siguientes preguntas que se realizaron.

3.2.3.2.1 PREGUNTAS

1.- ¿Está usted de acuerdo en que se implemente el sistema informático para mejorar la calidad del manejo de la Agropecuaria?

Si ☐ **No** ☐

2.- ¿Cree usted que la Implementación del Sistema le ayudaría agilizar las ventas diarias del Ganado?

Si ☐ **No** ☐

3.- ¿Desearía usted que se implemente el sistema, en el que se muestra el cuidado y crianza de los vacunos, de manera controlada?

Si ☐ **No** ☐

4.- ¿El sistema a implementar mostrará los cálculos de manera personalizada y automatizada para una rapidez en cuanto se refiera a venta del animal? ¿Cree usted que le ayudaría?

Si ☐ **No** ☐

5.- ¿Para la vacunación del Ganado el sistema mostrará en pantalla la aplicación de la vacuna respectiva en determinada fecha? ¿Está de acuerdo?

Si ☐ **No** ☐

6.- ¿Esta usted preparado para poder manejar un sistema Informático?

Si ☐ **No** ☐

7.- ¿El sistema a implementar mostrará un listado detallado de las vacunas, las que se vanaplicar referente a cada enfermedad y el modo de aplicación?

Si ☐ **No** ☐

8.- ¿Considera ud que se debería utilizar un Sistema Informatico para evitar las demoras y las constantes perdidas de informacion cuando se lleva manera manual?

Si ☐ **No** ☐

9.- ¿De que manera ayudaría automatizar el Sistema de Ganado Velbamal Desearía que el sistema cuente con un informe datallado de los ingresos y egresos para obtener veracidad y seguridad de todo el Ganado?

Si ☐ **No** ☐

10.- ¿Estaria ud de acuerdo con el respaldo de una base de datos en el Sistema?

Si ☐ **No** ☐

3.2.3.2.2 INTERPRETACIÓN SOBRE LAS ENCUESTAS

Las encuestas serán aplicadas a 16 Empleados de la Agropecuaria “VELBAMAL”, obteniendo de esta forma que el tamaño total de la muestra son: **16 encuestados**.

3.2.3.3 METODOS, TECNICAS E INSTRUMENTOS DE LA INVESTIGACION

3.2.3.3.1 MÉTODO GENERAL

El método general que se establecerá para el desarrollo de este proyecto es el Científico, debido a que nos permitirá alcanzar conocimientos validos con la utilización de instrumentos confiables mediante la aplicación de una secuencia de pasos.

Con la aplicación de este método se podrá plantear una interrogante la cual será producto de la observación de un fenómeno, en nuestro caso dicho fenómeno seria los procesos administrativos que se realizan en la agropecuaria. Para luego someterlo a un proceso de experimentación con la finalidad de hallar posibles soluciones a los problemas que hayan sido detectados en el mismo.

3.2.3.3.2 MÉTODOS ESPECÍFICOS

3.2.3.3.2.1 DESCRIPTIVO

Para el desarrollo de este estudio se tomo como referencia el método descriptivo ya que este se utilizara para especificar las propiedades importantes de las personas en nuestro caso los clientes para que puedan ser sometidos a un análisis. Dicho en otras palabras al seleccionar una serie de preguntas, se describirá cada una de ellas de manera independiente, después de haberlas medido.

Es importante seleccionar un grupo cuyos miembros estén disponibles al investigador. En ciertas ocasiones los datos de algunas variables pueden ser recolectadas sin tener un acceso directo a los sujetos. Variables como estancias anteriores o rango de edad pueden ser consultadas en el historial o registro de los Datos que posee la Agropecuaria.

3.2.3.3.2.2 EXPERIMENTAL

Este método ha sido seleccionado debido a que se presenta mediante la manipulación de una variable experimental no comprobada, en nuestro caso sería el Sistema PHP que se desea implementar, el cual trabajara en condiciones rigurosamente

controladas, con la finalidad de disminuir las causas o efectos de las conductas que han sido observadas lo cual seria los procesos que se manejan actualmente en la Agropecuaria.

3.2.3.4 TÉCNICAS

Para la realización del sistema de deberá aplicar varias técnicas de investigación con la finalidad de recolectar la información necesaria que después se someterá a una intensa tarea de tabulación de datos. Entre las principales técnicas aplicadas tenemos las siguientes:

3.2.3.4.1 ENTREVISTA

A los propietarios y empleados de la Agropecuaria “VELBAMAL”, a efectos de conocer el funcionamiento manual de la información y poder llevarla de la realidad abstracta al sistema informático.

3.2.3.4.2 OBSERVACIÓN

De todos los procesos que se realizan en la Agropecuaria para tener una visión objetiva de las necesidades de esta con la finalidad de crear estrategias de solución.

3.2.3.4.3 ENCUESTA

A los clientes de la agropecuaria, ya que son los usuarios que intervienen indirectamente con el sistema y nos pueden brindar una opinión externa de este; según sus experiencias vividas y que mejoras desearían en el mismo.

3.2.3.5 INSTRUMENTO

Esta encuesta está compuesta de preguntas cerradas en donde generalmente se toman 6 respuestas cerradas, donde se incluyen las variables de calidad. Para concluir con una pregunta que le permita establecer si el cliente regresaría a la agropecuaria en su próxima visita, y poder determinar de esta forma su lealtad a la empresa.

El cuestionario comienza con preguntas demográficas para obtener información general de los clientes cuya información se utilizo esencialmente para construir el perfil de los clientes encuestados.

3.2.3.6 TABULACION DE RESULTADOS

3.2.3.6.1 TABULACIÓN Y ANALISIS DE DATOS

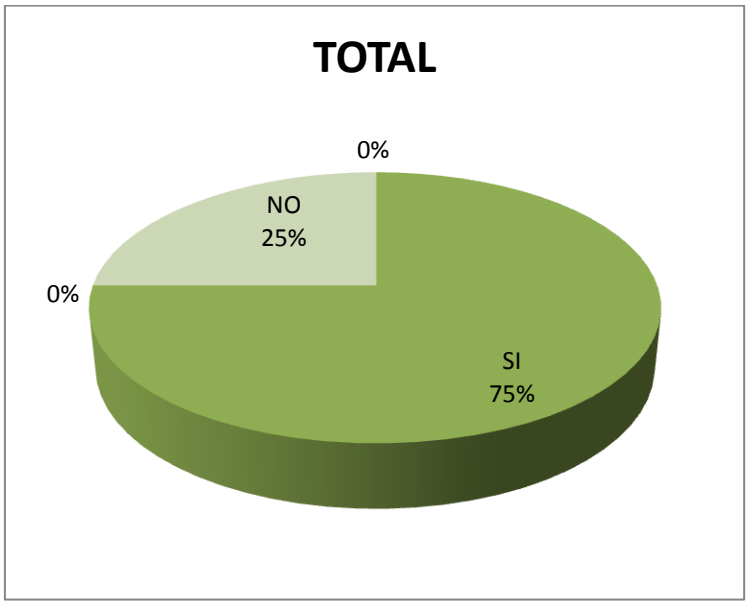
En esta sección se muestran los resultados obtenidos a lo largo de esta investigación. Tomando en cuenta los datos obtenidos en la agropecuaria “VELBAMAL”, donde se realizo la investigación exploratoria mediante la aplicación de una encuesta a los clientes del mismo.

La información obtenida se registró en cuadros y gráficos, donde se colocaron todos los datos necesarios correspondientes a los clientes, a los cuales se les aplico la encuesta para el análisis de los requerimientos y por ende los resultados del sistema.

CUADRO # 1

Total y porcentaje de los resultados de la pregunta: "¿**Está usted de acuerdo en que se implemente el sistema informático para mejorar la calidad del manejo de la Agropecuaria?".**

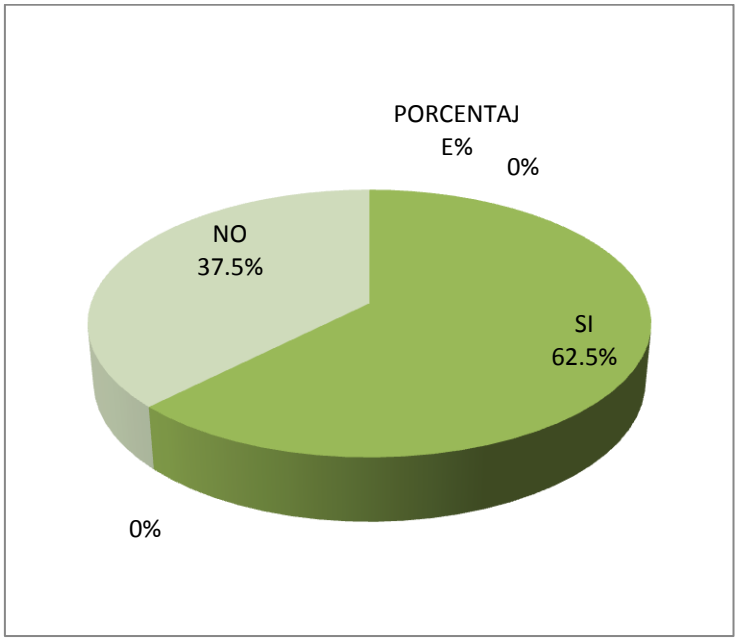
RESPUESTAS	TOTAL	PORCENTAJE
SI	**12**	**75%**
NO	**4**	**25%**
TOTAL	16	100%



CUADRO # 2

Total y porcentaje de los resultados de la pregunta: "¿**Cree usted que la Implementación del Sistema le ayudaría agilizar las ventas diarias del Ganado?".**

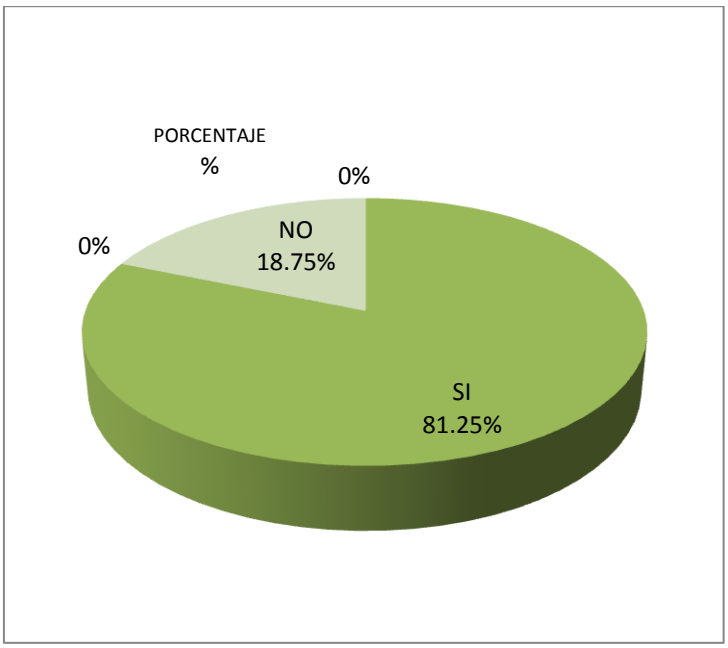
RESPUESTAS	TOTAL	PORCENTAJE
SI	**10**	**62.5%**
NO	**6**	**37.5%**
TOTAL	16	100%



CUADRO # 3

Total y porcentaje de los resultados de la pregunta: **"¿Desearía usted que se implemente el sistema, en el que se muestra el cuidado y crianza de los vacunos, de manera controlada?".**

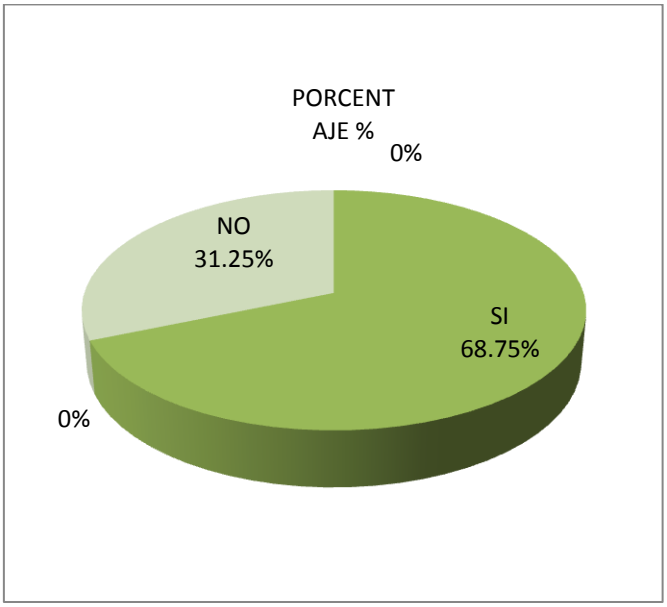
RESPUESTAS	TOTAL	PORCENTAJE
SI	**13**	**81.25%**
NO	**3**	**18.75%**
TOTAL	16	100%



CUADRO # 4

Total y porcentaje de los resultados de la pregunta: **"El sistema a implementar mostrará los cálculos de manera personalizada y automatizada para una rapidez en cuanto se refiera a venta del animal. ¿Cree usted que le ayudaría?".**

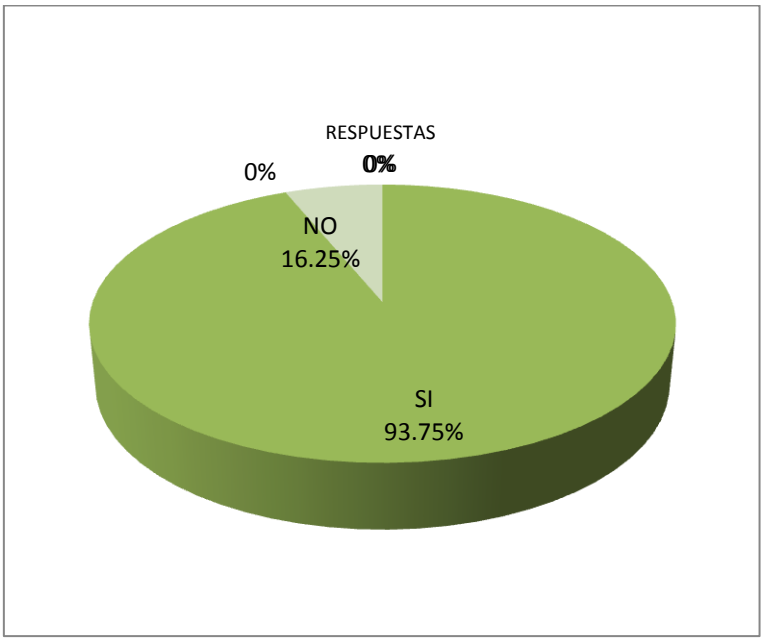
RESPUESTAS	TOTAL	PORCENTAJE
SI	**11**	**68.75%**
NO	**5**	**31.25%**
TOTAL	16	100%



CUADRO # 5

Total y porcentaje de los resultados de la pregunta: **"PARA LA VACUNACIÓN DEL GANADO EL SISTEMA MOSTRARA EN PANTALLA LA APLICACIÓN DE LA VACUNA RESPECTIVA EN DETERMINADA FECHA..¿ESTÁ DE ACUERDO? ".**

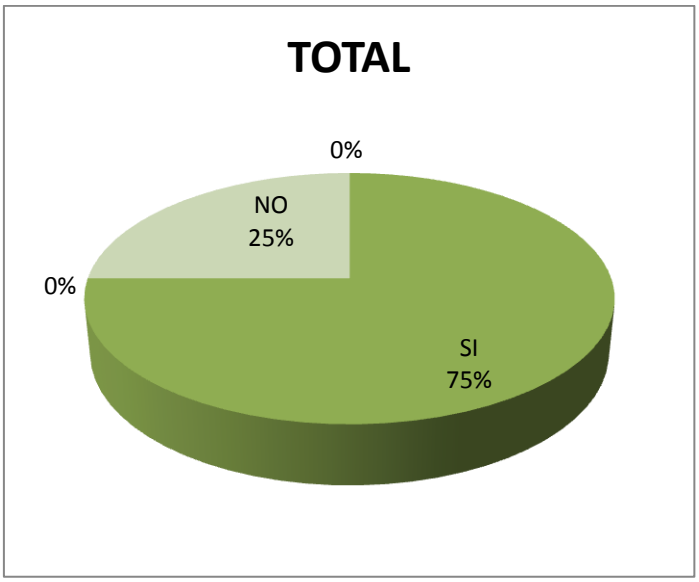
RESPUESTAS	TOTAL	PORCENTAJE
SI	**15**	**93.75%**
NO	**1**	**6.25%**
TOTAL	16	100%



CUADRO # 6

Total y porcentaje de los resultados de la pregunta: **"¿ESTA USTED PREPARADO PARA PODER MANEJAR UN SISTEMA INFORMÁTICO?".**

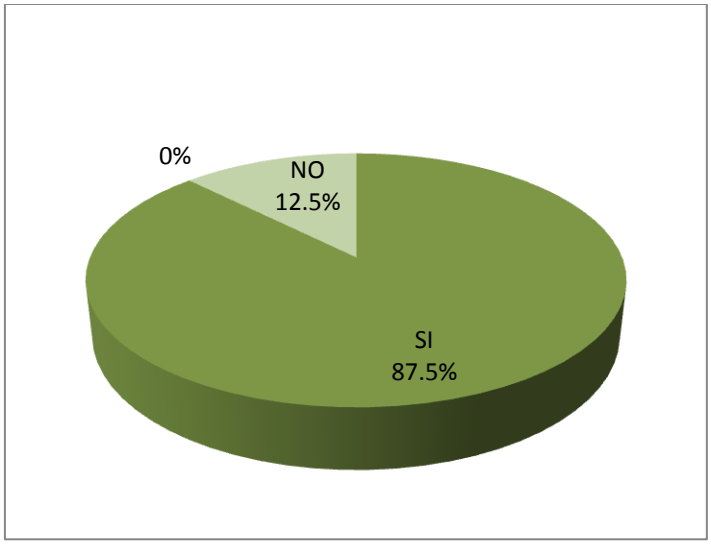
RESPUESTAS	TOTAL	PORCENTAJE
SI	**12**	**75%**
NO	**4**	**25%**
TOTAL	16	100%



CUADRO # 7

Total y porcentaje de los resultados de la pregunta: **"¿LE GUSTARÍA OBSERVAR UN LISTADO DETALLADO DE LAS VACUNAS PARA EL GANADO APLICAR REFERENTE A LA ENFERMEDAD, FECHA Y MODO DE APLICACIÓN? ".**

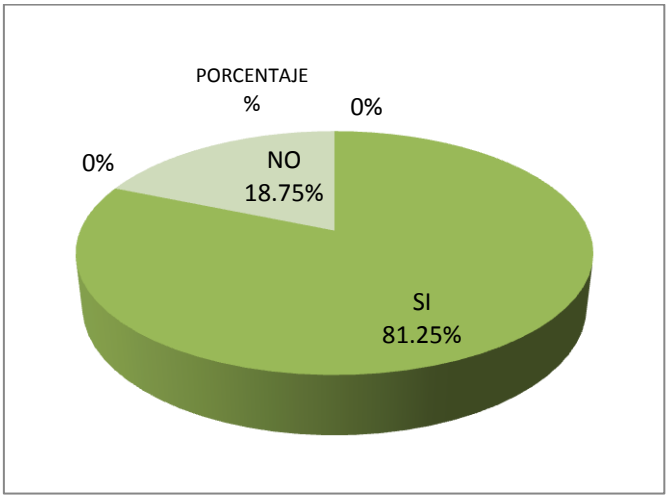
RESPUESTAS	TOTAL	PORCENTAJE
SI	**14**	**87.5%**
NO	**2**	**12.5%**
TOTAL	16	100%



CUADRO # 8

Total y porcentaje de los resultados de la pregunta: **"CONSIDERA UD QUE SE DEBERÍA UTILIZAR UN SISTEMA INFORMATICO PARA EVITAR LAS DEMORAS Y LAS CONSTANTES FALLAS.? ".**

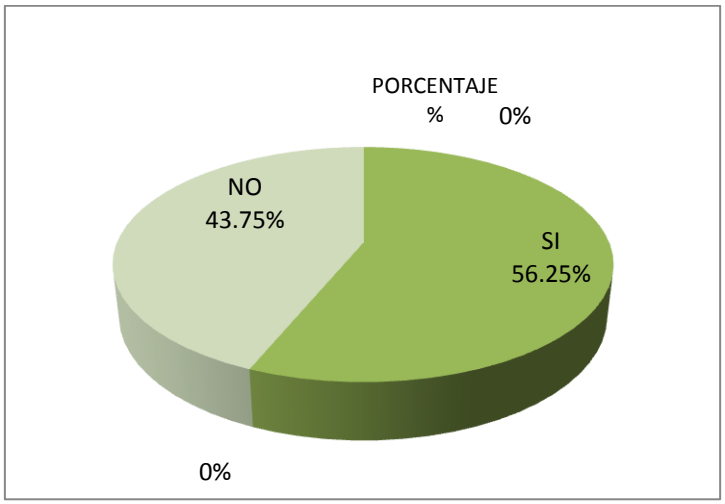
RESPUESTAS	TOTAL	PORCENTAJE
SI	**13**	**81.25%**
NO	**3**	**18.75%**
TOTAL	16	100%



CUADRO # 9

Total y porcentaje de los resultados de la pregunta: **"¿DESEARIA QUE EL SISTEMA CUENTE CON UN INFORME DATALLADO DE LOS INGRESOS Y EGRESOS PARA OBTENER VERACIDAD Y SEGURIDAD DE TODO EL GANADO.? ".**

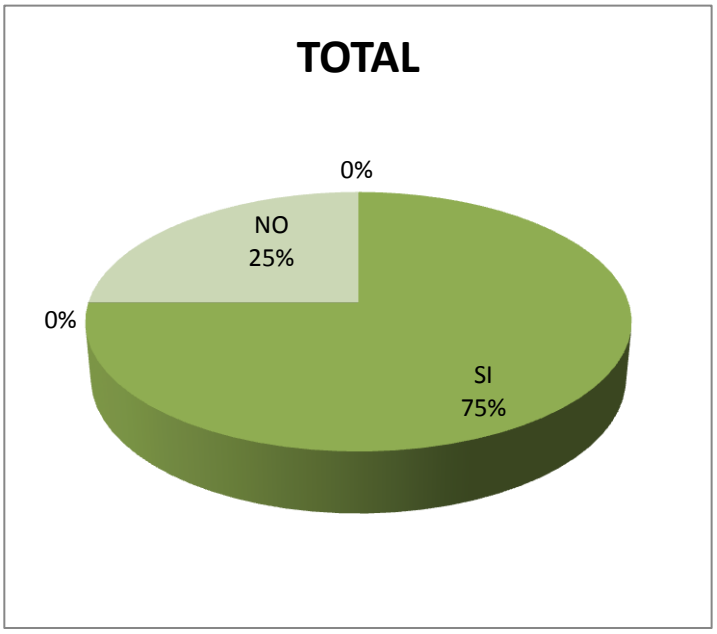
RESPUESTAS	TOTAL	PORCENTAJE
SI	**9**	**56.25%**
NO	**7**	**43.75%**
TOTAL	16	100%



CUADRO # 10

Total y porcentaje de los resultados de la pregunta: **"ESTARIA UD DE ACUERDO CON EL RESPALDO DE UNA BASE DE DATOS EN EL SISTEMA?".**

RESPUESTAS	TOTAL	PORCENTAJE
SI	**12**	**75%**
NO	**4**	**25%**
TOTAL	16	100%



3.2.3.7 VERIFICACION DE LA HIPOTESIS

En esta sección se muestran los resultados obtenidos a lo largo de esta investigación. Si bien, el diseño del punto ideal para calcular la calidad en el servicio se obtuvo de la media de las respuestas con respecto al rango de calificación, tomando en cuenta el puntaje mas alto de todas las preguntas de los cuestionarios.

Los resultados de las entrevistas realizadas al personal inmerso en la agropecuaria determinan que existe una insatisfacción en el sistema actual ya que el mismo no cumple las expectativas del propietario así como la de los empleados, al no existir un control de centralizado de toda la información que se maneja.

3.3 CONCLUSIONES

El sistema manual usado en la actualidad en la Agropecuaria “VELBAMAL”, de la ciudad de Vinces cubre el 40% de los requerimientos, de los propietarios del mismo.
De acuerdo con la información recolectada, en la Agropecuaria “VELBAMAL”, cuenta con un personal convencido de las ventajas que ofrece la automatización de las tareas y la necesidad de manejar información precisa y veraz.

3.4 RECOMENDACIONES

Hay que considerar, el uso adecuado de cada una de las herramientas de hardware y software del sistema, para que la información que se transmita sea veraz y oportuna, lográndose así una comunicación efectiva que influya directamente en la toma de decisiones, por ser la información uno de los recursos esenciales en la solución de problemas en una organización.

CAPITULO IV

4.1 INTRODUCCIÓN

Las Empresas Ganaderas, como la mayoría de las organizaciones se enfrentan a un entorno continuamente cambiante y altamente competitivo, caracterizado por una serie de particularidades como una fuerte estacionalidad, una importante rigidez derivada de una inversión en un Sistema Completo para tener una buena atención a la exigencia del cliente.

Es necesario comprender a la explotación agropecuaria que modernamente podríamos denominar también como "empresa agropecuaria",un conjunto armónico que funciona (no importando como desde este punto de vista), con la aplicación de la fuerza de trabajodel hombre, sobre el capital y los recursos agropecuarios, mediante una adecuada administracion, y una creación de un sistema persiguiendo objetivos precisos.

Debemos observar a la hacienda agropecuaria como un todo, como un conjunto, como una unidad, que se encuentra y funciona dentro un contexto determinado: el medio productivo regional y por lo tanto también inserto en el medio productivo nacional. Hacienda, cuyo tamaño, forma, rubros productivos, intensidad de aplicación de los recursos etc., responden a leyes naturales, económicas, políticas, culturales, familiares y hasta racionales del productor, que regulan sus actividades, sus flujos internos, sus influencias y sus relaciones con otras unidades productivas de la región.

Todo ello contribuye a resaltar la importancia de la realización de una tesis de estas características como una fuente adicional para la administración de recursos en el Sector Agropecuario, la gestión en particular para la dirección de este tipo de empresas.

Por otra parte, en los últimos años se ha detectado un interés creciente por los sistemas informáticos por cuanto aporta numerosas ventajas a las empresas que lo utilizan. En este sentido, debe destacarse el escaso desarrollo de este tipo de sistemas en el entorno Agropecuario.

4.2. OBJETIVOS DE LA PROPUESTA

En el presente informe se encuentra detallado todo el trabajo investigativo que ha sido desarrollado en el Sistema Gestión De Cuidados, Crianzas y Ventas de Ganado de la Agropecuaria “Velbamal” en la cuidad de Vinces, el cual ofrece a los clientes un servicio eficiente y de alta calidad.

Por tanto el objetivo principal de esta documentación es mostrar la información más importante del sistema a emplear en la actualidad en la Agropecuaria, junto con las características de este y describir la naturaleza del estudio realizado para de esta manera poder compararlo con el sistema propuesto.

4.2.1 OBJETIVO GENERAL

- Desarrollar un Sistema para la eficiente Gestión “De Cuidados, Crianzas y Ventas de Ganado de la Agropecuaria Velbamal”.

4.2.2 OBJETIVOS ESPECIFICOS

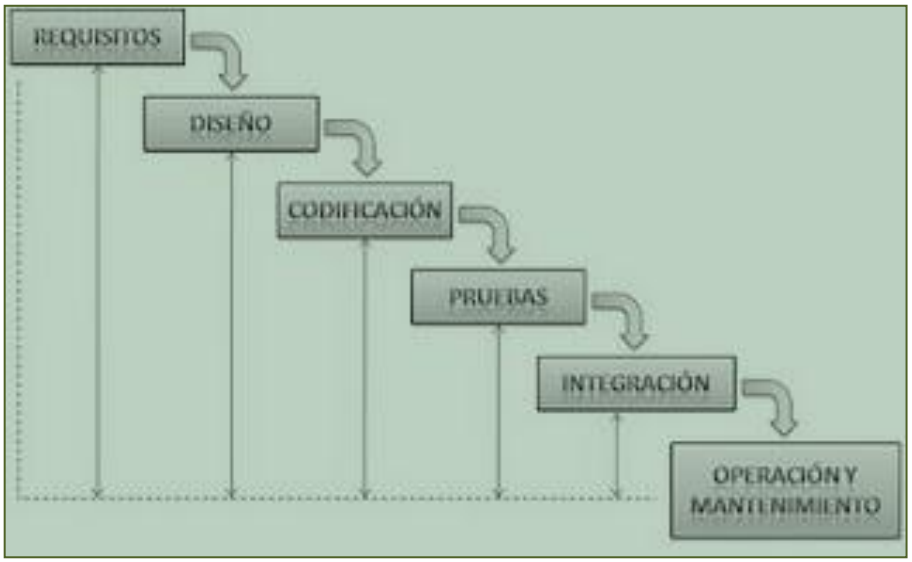
- Obtener un registro detallado del Ganado, de una manera mucho más segura por medio de una base de datos que estará ligada al sistema a aplicar.
- Establecer un orden y control sobre los informes a realizar por parte de la gerencia de la Agropecuaria evitando los errores que se presentan con la forma manual.
- Sistematizar los procesos de gestión para controlar la información producida en las actividades de la Agropecuaria.
- Recolección y Análisis de la Información
- Implementar un Software de Gestión y validarlo con un experto.
- Promover, Fortalecer Y Ayudar El Sector Ganadero trabajando con todos los propietarios de fincas, pequeños, medianos y grandes productores, también a las diferentes entidades del sector.
- Obtener un registro detallado de las Vacas, Toros, de una manera mucho más segura por medio de una base de datos que estará ligada al sistema a aplicar.
- Establecer un orden y control sobre los informes a realizar por parte de la gerencia de la Agropecuaria evitando los errores que se presentan con la forma manual.
- Sistematizar los procesos administrativos para controlar la información producida en las actividades de la Agropecuaria.

4.3. METODOLOGIA DE DESARROLLO UTILIZADA

El modelo a desarrollar el siguiente proyecto investigativo es mediante modelo cascada.

- MODELO CASCADA.-Este modelo admite la posibilidad de hacer iteraciones, es decir, durante las modificaciones quese hacen en el mantenimiento se puede ver por ejemplo la necesidad de cambiar algo en el diseño, lo cualsignifica que se harán los cambios necesarios en la codificación y se tendrán que realizar de nuevo laspruebas, es decir, si se tiene que volver a una de las etapas anteriores al mantenimiento hay que recorrerde nuevo el resto de las etapas.

  Despuésde cadaetapa se realiza una revisión para comprobarsi se puede pasara la siguiente.



Desde el mismo instante en el que se consive la idea del desarrollo de un software hasta que este ya ha quedado obsoleto, estamos hablando de un ciclo de vida de nuestro sistema.

Elparadigma del ciclo de vida abarca las siguientes actividades:

- **Requisitos del software:** El proceso de recopilación de los requisitos se centra eintensifica especialmente en el software. El ingeniero de software debe comprender el ámbito de lainformacióndel software,así como la función,el rendimiento y las interfaces requeridas.
- **Diseño:** El diseño del software se enfoca en cuatro atributos distintos del programa: la estructura de losdatos, la arquitecturadel software,el detalle procedimentaly la caracterizaciónde la interfaz.

- **Codificación:** El diseño debe traducirse en una forma legible para la maquina. El paso de codificación realiza esta tarea.
- **Prueba:** La prueba se centra en la lógica interna del software, y en las funciones externas, realizandopruebasque asegurenque la entrada definida producelos resultados que realmentese requieren.
- **Integración**: Una vez que tenemos los módulos codificados, hay que ensamblarlos. Desgraciadamente el proceso no consiste simplemente en unir piezas. Suelen aparecer problemas con las interfaces entre los módulos, con la comunicación de datos compartidos, con el encadenamiento de flujos de ejecución, etc.
- **Mantenimiento:** El software sufrirá cambios después de que se entrega al cliente. Los cambios ocurrirándebido a que hayan encontrado errores, a que el software deba adaptarse a cambios del entorno externo (sistema operativo o dispositivos periféricos), o debido a que el cliente requiera ampliaciones funcionaleso del rendimiento.

CARACTERISTICAS

- Es el más utilizado.
- Es una visión del proceso de desarrollo de software como una sucesión de etapas que producen productosintermedios.
- Para que el proyecto tenga éxito deben desarrollarse todas las fases.
- Las fases continúan hasta que los objetivos se han cumplido.

VENTAJAS

- ✓ Minimiza las tareas de desarrollo repetidas y por tanto el esfuerzo de desarrollo invertido en total.
- ✓ Minimiza la carga de planificación de los ciclos iterativos de otros ciclos de vida.
- ✓ Permite afrontar la complejidad de proyectos grandes de una manera muy ordenada y aumenta así las posibilidades de éxito.
- ✓ Ayuda a trabajar mejor con equipos de desarrollo de relativamente baja calificación por el alto control de cada actividad y sus resultados.

INCONVENIENTES

- Retroceder en las fases para corregir errores que se han cometido en fases previas o adaptar el proyecto a cambios resulta muy difícil y costoso en esfuerzo.

- Aunque la documentación elaborada permite un seguimiento bueno del proyecto para una persona calificada, los resultados tangibles para el cliente aparecen prácticamente al final del proyecto, algo que muchas veces no aceptan los clientes.

4.4 ANALISIS PREVIO

Es una disposición de procedimientos o programas relacionados de manera que juntos forman una sola unidad. Un conjunto de hechos, procedimientos o programas y reglas clasificadas y dispuestas de manera ordenada mostrando un plan lógico en la unión de las partes. Un método plan o procedimiento clasificación para hacer algo.

- Esto se lleva a cabo teniendo en cuenta ciertos principios para el Análisis:
- Debe presentarse y entenderse el dominio de la información de un problema.
- Definir las funciones que debe realizar el software.
- Representar el comportamiento del software a consecuencias de acontecimientos externos.

El proceso debe partir desde la información esencial hasta el detalle de la Implementación.

- La función del Análisis puede ser dar soporte a las actividades de un negocio, o desarrollar un producto que pueda venderse para generar beneficios. Para conseguir este objetivo, un Sistema basado en computadoras hace uso de seis elementos fundamentales:
- Software, que son Programas de computadora, con estructuras de datos y su documentación que hacen efectiva la logísticametodología o controles de requerimientos del Programa.
- Hardware, dispositivos electrónicos y electromecánicos, que proporcionan capacidad de cálculos y funciones rápidas, exactas y efectivas (Computadoras, Censores, maquinarias, bombas, lectores, etc.), que proporcionan una función externa dentro de los Sistemas.
- Personal, son los operadores o usuarios directos de las herramientas del Sistema.
- Base de Datos, una gran colección de informaciones organizadas y enlazadas al Sistema a las que se accede por medio del Software.
- Documentación, Manuales, formularios, y otra información descriptiva que detalla o da instrucciones sobre el empleo y operación del Programa.
- Procedimientos, o pasos que definen el uso específico de cada uno de los elementos o componentes del Sistema y las reglas de su manejo y mantenimiento.

Un Análisis de Sistema se lleva a cabo teniendo en cuenta los siguientes objetivos en mente:

- Identificar las necesidades del Dueño de la Agropecuaria.
- Evaluar que conceptos tiene el cliente del sistema para establecer su viabilidad.
- Realizar un Análisis Técnico y económico.
- Asignar funciones al Hardware, Software, Personal, Base de datos, y otros elementos del Sistema.
- Crear una definición del sistema que forme el fundamento de todo el trabajo de Ingeniería.

Para lograr estos objetivos se requiere tener un gran conocimiento y dominio del Hardware, Software, y Administración de Base de datos.

4.5 DISEÑO

Las tablas con las que consta nuestro sistema son las siguientes:

- Reses
- Vacunación
- Limpieza
- Comprador
- Empleado
- Lotes
- Ventas
- Descripciones
- Eventos
- Usuario
- Egresos
- Actividades
- Historial_Res
- Raza

- Seguimiento_Res
- Tipo_Limpieza
- Vacuna

4.5.1 LISTADO DE REQUERIMIENTOS Y FUNCIONES QUE QUE TENDRA EL SISTEMA

REQUERIMIENTOS DE HARDWARE

- Disco Duro 80 GB
- Memoria 512 MB
- Procesador 2.8 GHz
- CD-ROM
- Cd Rw
- Monitor
- Teclado
- Mouse
- Impresora

REQUERIMIENTOS DE SOFTWARE

- Microsoft Windows Xp / 7
- Sql Yog Enterprise
- Wamp server 1.6.1

Las funciones con las que contará el Sistema son las siguientes:

- Interfaz de Autenticación de Usuario:
- Menú principal

Gestión de Reses.

- Ingreso de Datos de Nuevas Reses.
- Actualización de Datos de las Reses.
- Eliminación de Reses.
- Consultas de Reses.

Gestión de Vacunación.

- Ingreso de Datos de Vacunación
- Actualización de Datos de Vacunación.
- Eliminación de Vacunación.
- Consultas de Vacunación.

Gestión de Limpieza.

- Ingreso de Datos de Limpieza.
- Actualización de Datos de Limpieza.
- Eliminación Tipo de Limpieza.
- Consultas de Limpieza.

Gestión de Comprador.

- Ingreso de Datos de Nuevo Comprador.
- Actualización de Datos de Comprador.
- Eliminación de Comprador
- Consultas de Comprador.

Gestión de Empleado.

- Ingreso de Datos de Nuevo Empleado.
- Actualización de Datos de Empleado.
- Eliminación de Empleado.
- Consultas de Empleado.

Gestión de Lotes.

- Ingreso de Datos de Nuevo Lote.
- Actualización de Datos Lote.
- Eliminación de Lote.
- Consultas de Lote.

Gestión de Ventas.

- Ingreso de Datos de Venta.
- Consultas de Ventas.

Gestión de Descripciones.

- Ingreso de Datos de Descripciones.
- Actualización de Datos de Descripciones..
- Eliminación de Descripciones.
- Consultas de Descripciones.

Gestión de Eventos.

- Ingreso de Datos de Evento.
- Actualización de Datos del Evento.

- Eliminación de Evento.
- Consultas de Evento.

Gestión de Usuarios Registrados

- Ingreso de Datos de Usuarios.
- Actualización de Datos de Usuario.
- Eliminación de Usuario.
- Consultas de Usuario Registrado.

Gestión de Egresos.

- Ingreso de Datos de Egreso.
- Actualización de Datos de Egreso.
- Eliminación de Egreso.
- Consultas de Datos de Egreso.

Gestión de Actividades

- Ingreso de Actividad.
- Modificación de Actividad.
- Eliminación de Actividad.
- Consulta de Actividad.

Gestión de Historial_Res

- Ingreso de Datos del Historial.
- Actualización de Historial.
- Eliminación de Historial.

- Consultas de Historial.

Gestión de Raza

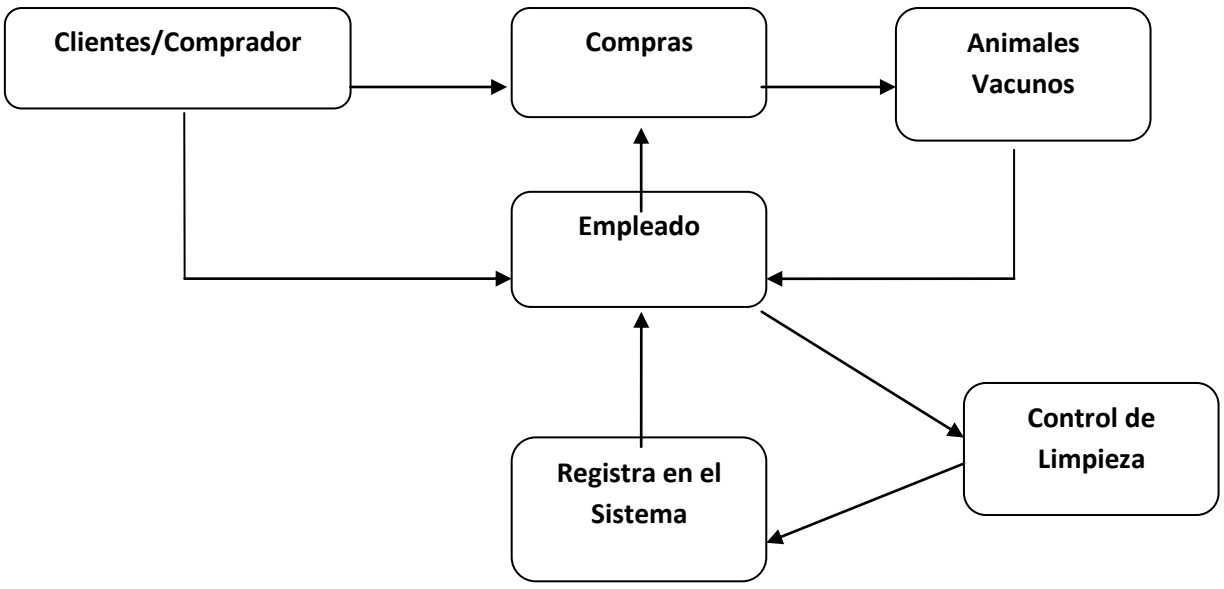
- Ingreso de Datos de Raza.
- Actualización de Datos de Raza.
- Eliminación de Raza.
- Consultas de Raza.

OPCIONES QUE TENDRA EL SISTEMA

- Listado de reses por evento
- Listado de reses vendidas
- Listado de reses por actividad
- Listado de reses por raza

4.5.2 BASE DE DATOS (MODELO CONCEPTUAL Y MODELO FISICO)

4.5.2.1 MODELO CONCEPTUAL



4.5.2.2 MODELO FISICO

La base de datos constara con las siguientes tablas

Nombre de la Tabla	Nombre del atributo	Tipo de dato	Longitud	Clave
Usuarios	Cod_Usuario	int	10	Pk
	Usuario	varchar	200	
	Tipo	varchar	200	
	Estado	varchar	200	
Comprador	Cedula_RUC	varchar	10	Pk, Fk
	Nombres	varchar	200	
	Sexo	varchar	200	
	Dirección	varchar	200	
	Telefono	varchar	50	
	Ocupacion	varchar	200	
	Email	varchar	200	
	Foto	varchar	200	
	Comentario	varchar	200	
Descripcion_reses	Cod_Descrpicion	int	200	Pk
	Nombre	varchar	200	
	Observacion	varchar	200	
Empleados	Cedula_Empleado	varchar	10	Pk
	Nombres	varchar	200	

	Sexo	varchar	200	
	Dirección	varhar	200	
	Teléfono	varchar	50	
	Tipo_Empleado	varchar	200	
	Foto	varchar	200	
Evento	Cod_Evento	int	11	Pk
	Nombre	varchar	200	
	Observación	varchar	200	
Historial_reses	Cod_Historial	int	11	Pk
	Cod_Res	int	11	Fk
	Estado	varchar	200	
Limpieza	Cod_Limpieza	int	11	Pk
	Cod_Res	int	11	Fk
	Cod_Tipo_Limpieza	Int	11	Fk
	Fecha	datetime		
	Cedula_Empleado	varchar	10	
	Comentario	varchar	200	
Lotes	Cod_Lotes	int	11	Pk
	Descripción	varchar	200	
	Capacidad	int	11	

Raza	cod_raza	int	4	Pk
	Raza	varchar	200	
	Valor_Libra	decimal	10,2	
	Valor_Mes_Preñada	decimail	10,2	
	Observación	varchar	200	
Seguimiento_res	Cod_Seguimiento	int	11	Pk
	Cod_Res	int	11	Fk
	Fecha_Seguimiento	date		
	Peso	int	11	
	Cod_Lotes	int	11	Fk
	Cod_Descripcion	int	11	Fk
	Cod_Actividad	int	11	Fk
	Cod_Evento	int	11	Fk
	Comentario	varchar	200	
Tipo_Limpieza	Codigo_Tipo_Limpieza	int	11	Pk
	Nombre	varchar	200	
	Observación	varchar	200	
Vacuna	Cod_Vacuna	int	11	Pk
	Nombre	Varchar	200	
	Uso	Varchar	500	

	Comentario	varchar	200	
Actividades	Cod_Actividad	Int	11	Pk
	Nombre	varchar	200	
	Observacion	varchar	200	
Egreso	Cod_Egreso	int	11	Pk
	Fecha_Egreso	Date		
	Detalle	varchar	400	
	Valor	decimal	19.4	
	Comentario	varchar	200	
Reses	Cod_Res	int	11	Pk
	Nombre	varchar	200	
	Fecha_Nacimiento	Date		
	Peso	int	11	
Vacunacion	Cod_Vacunacion	int	11	Pk
	Cod_Res	int	11	Fk
	Cod_Vacuna	int	11	Fk
	Fecha_Aplicacion	Datetime		
	Comentario	varchar	200	
Ventas	Cod_Venta	int	11	Pk
	Cod_Res	int	11	Fk

	Ced_RUC	Varchar	10	
	Cedula_Empleado	varchar	10	
	Fecha_Transaccion	Date		
	Valor_Libra	decimal	19.4	
	Peso_Animal	int	4	
	Adicional_Preñada	decimal	10	
	Subtotal	decimal	10	
	Descuento	int	4	
	Valor_Pagar	decimal	10	
	Observacion	varchar	200	

4.5.3 DICCIONARIO DE DATOS

TABLA ACTIVIDADES

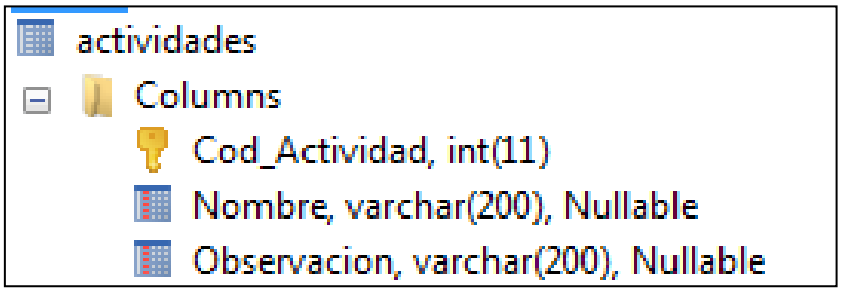


TABLA COMPRADOR

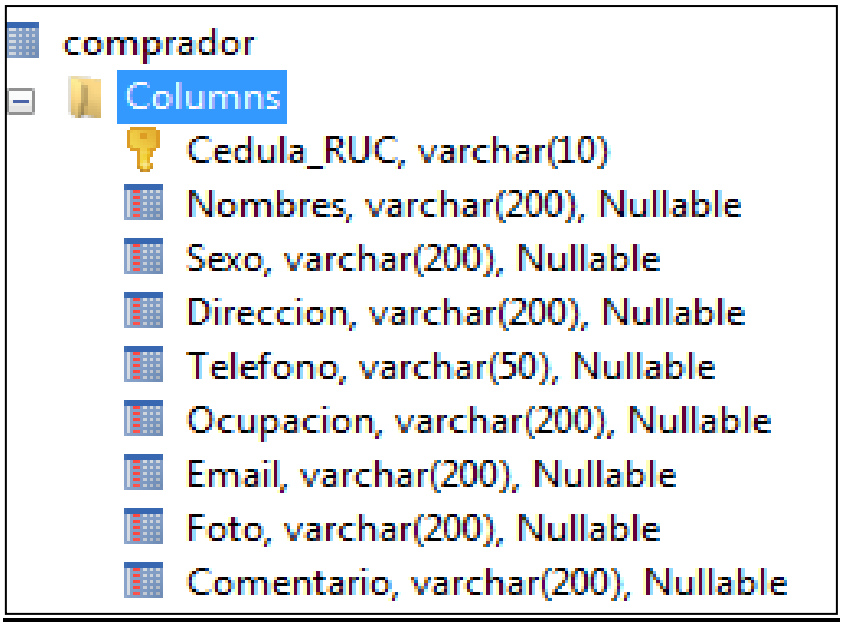


TABLA DESCRIPCION_RESES

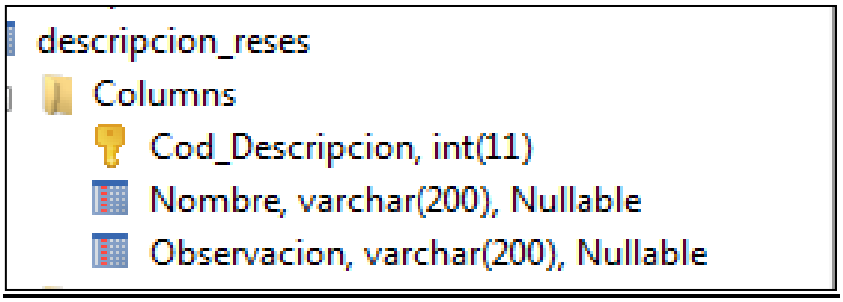


TABLA EGRESOS

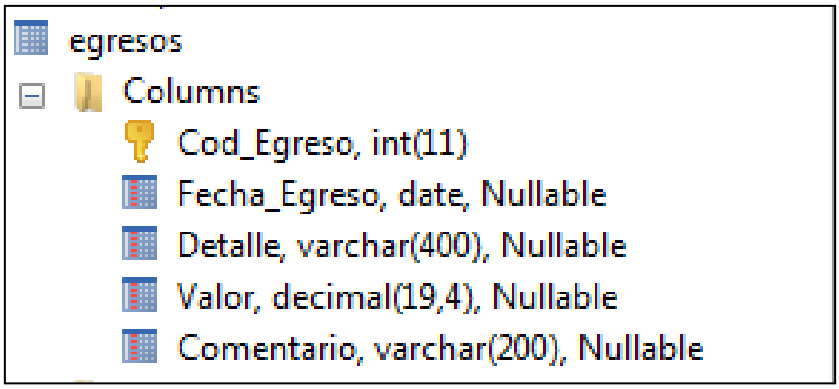


TABLA EMPLEADOS

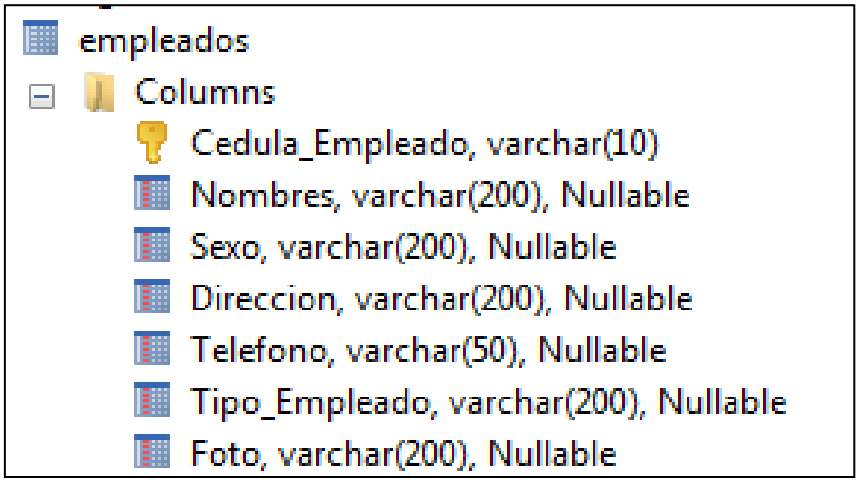


TABLA EVENTO

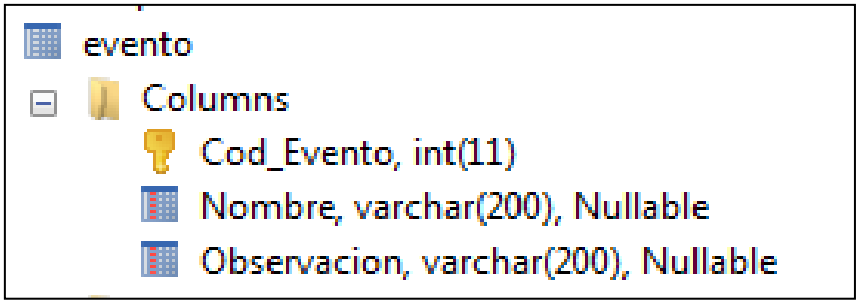


TABLA HISTORIAL RES

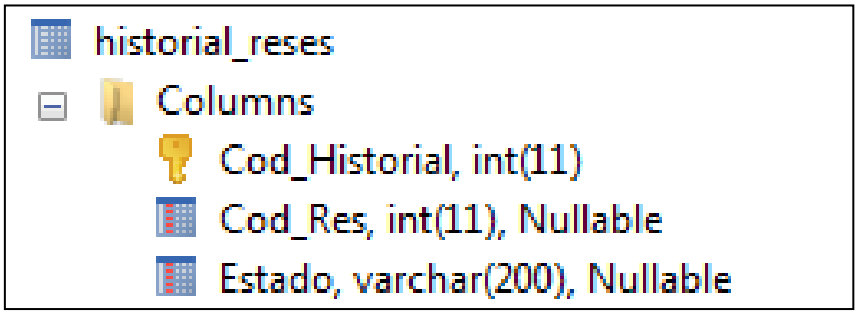


TABLA LIMPIEZA

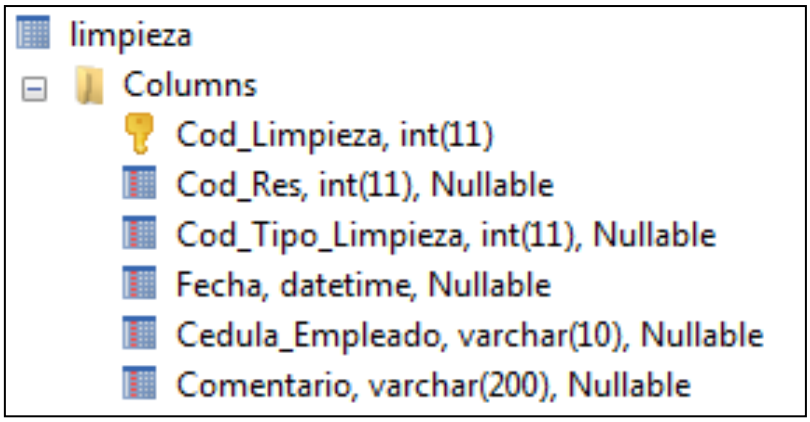


TABLA LOTES

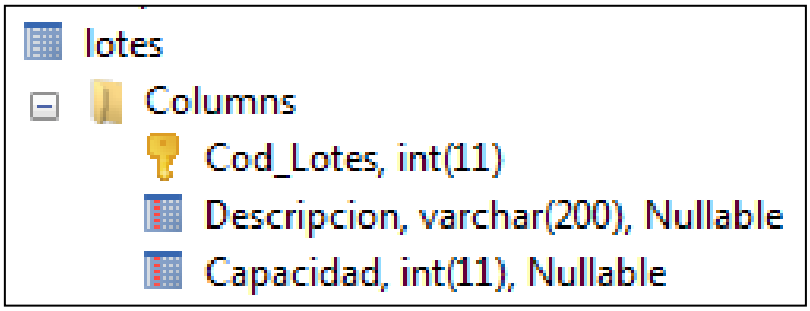


TABLA RAZA

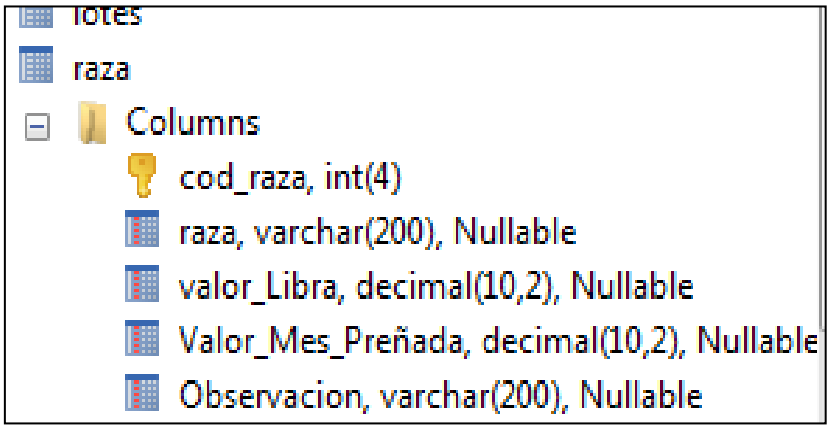


TABLA RESES

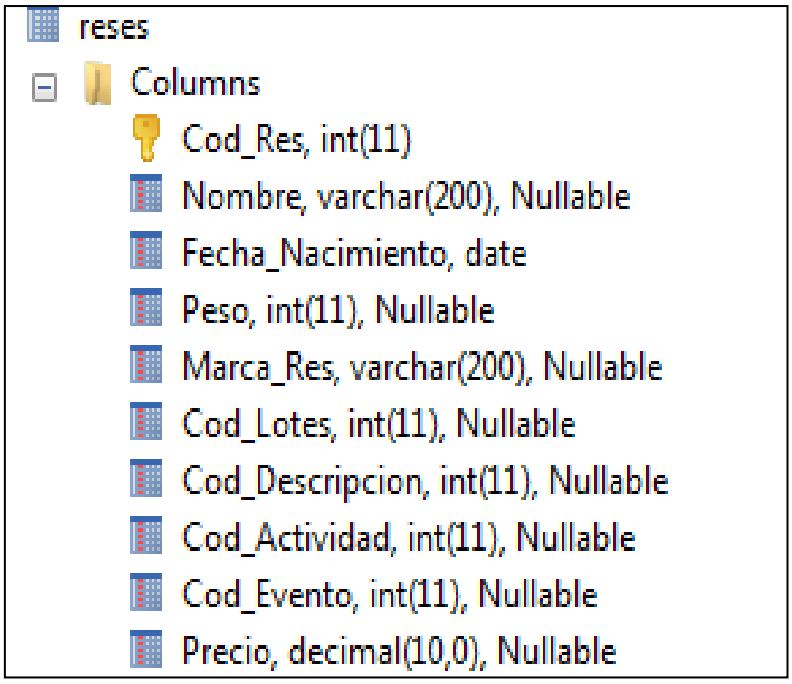


TABLA SEGUIMIENTO RES

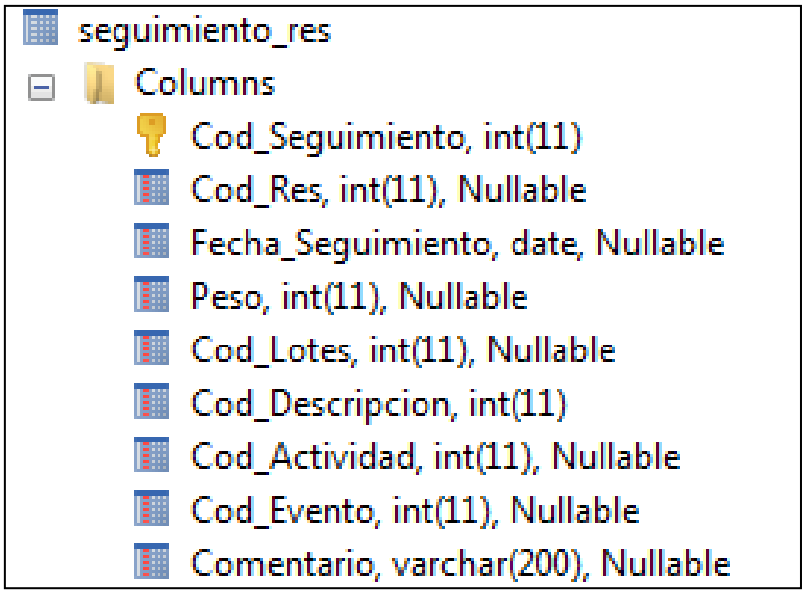


TABLA TIPO LIMPIEZA

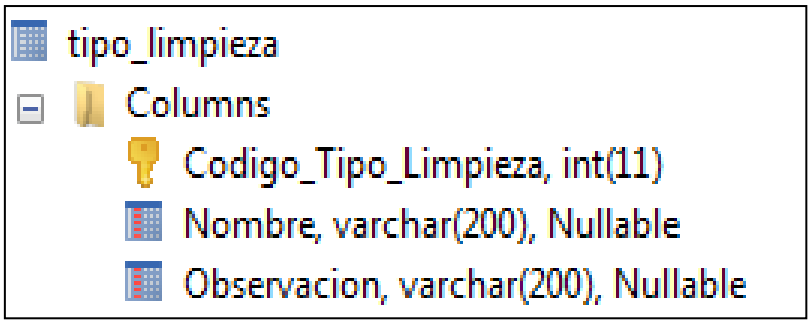


TABLA USUARIOS

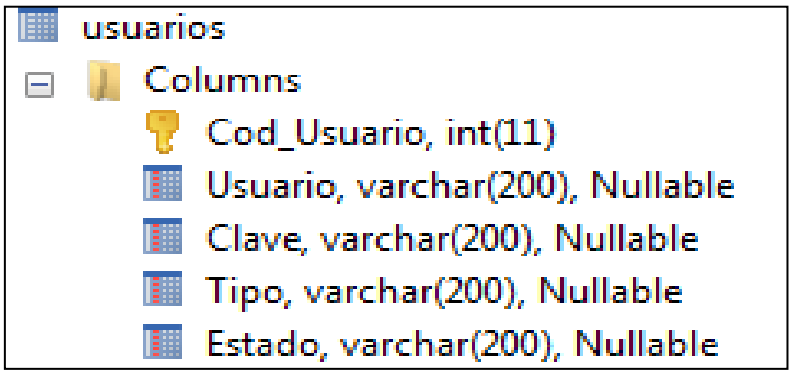


TABLA VACUNACION

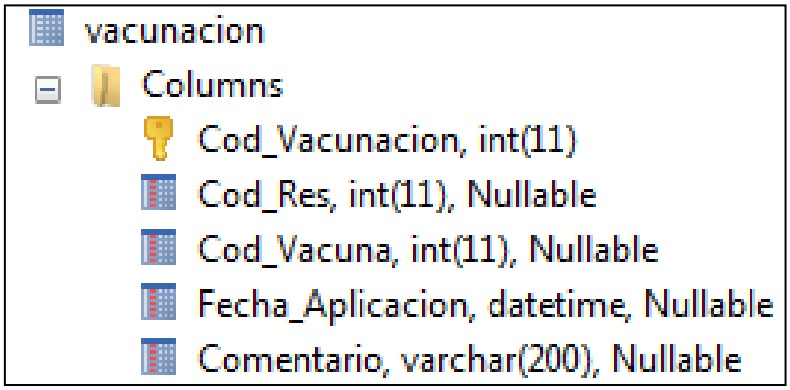
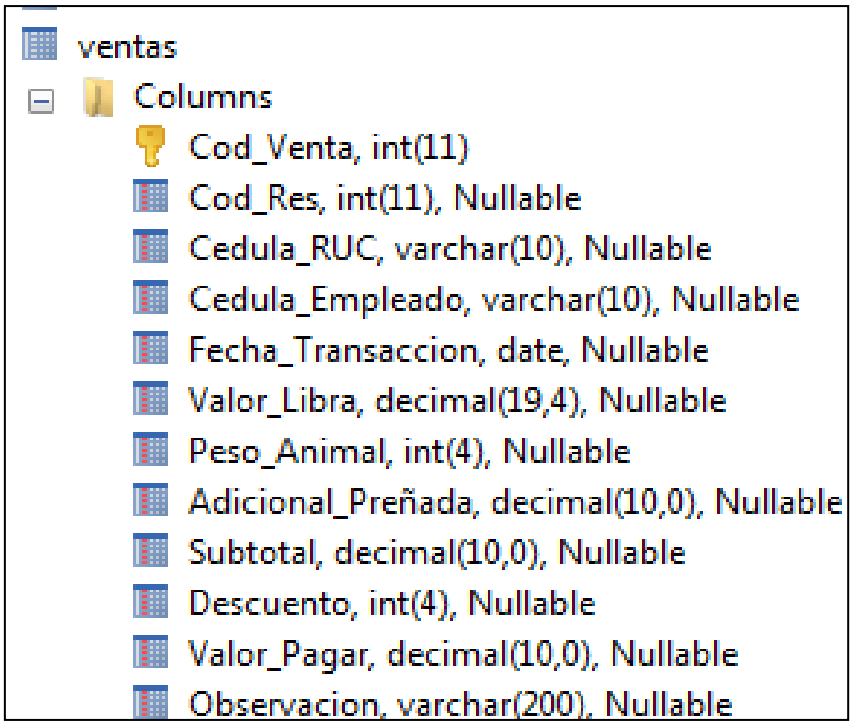


TABLA VENTAS



4.5.4 SCRIPT DE BASE DE DATOS

```
/*
sqlyog - free mysql gui v5.11
host - 5.0.18-nt : database - velbamal
*********************************************************************
server version : 5.0.18-nt
*/
set names utf8;
set sql_mode='';
create database if not exists `velbamal`;

use `velbamal`;
/*table structure for table `actividades` */
drop table if exists `actividades`;

create table `actividades` (
  `cod_actividad` int(11) not null auto_increment,
  `nombre` varchar(200) character set utf8 default null,
  `observacion` varchar(200) character set utf8 default null,
primary key  (`cod_actividad`)
```

```
) engine=innodb default charset=latin1;
/*data for the table `actividades` */
/*table structure for table `comprador` */
drop table if exists `comprador`;

create table `comprador` (
  `cedula_ruc` varchar(10) character set utf8 not null,
  `nombres` varchar(200) character set utf8 default null,
  `sexo` varchar(200) character set utf8 default null,
  `direccion` varchar(200) character set utf8 default null,
  `telefono` varchar(50) character set utf8 default null,
  `ocupacion` varchar(200) character set utf8 default null,
  `email` varchar(200) character set utf8 default null,
  `foto` varchar(200) character set utf8 default null,
  `comentario` varchar(200) character set utf8 default null,
primary key  (`cedula_ruc`)
) engine=innodb default charset=latin1;
/*data for the table `comprador` */
/*table structure for table `descripcion_reses` */
drop table if exists `descripcion_reses`;

create table `descripcion_reses` (
  `cod_descripcion` int(11) not null auto_increment,
  `nombre` varchar(200) character set utf8 default null,
  `observacion` varchar(200) character set utf8 default null,
primary key  (`cod_descripcion`)
) engine=innodb default charset=latin1;
/*data for the table `descripcion_reses` */
/*table structure for table `egresos` */
drop table if exists `egresos`;

create table `egresos` (
  `cod_egreso` int(11) not null auto_increment,
  `fecha_egreso` date default null,
  `detalle` varchar(400) character set utf8 default null,
  `valor` decimal(19,4) default null,
  `comentario` varchar(200) character set utf8 default null,
primary key  (`cod_egreso`)
) engine=innodb default charset=latin1;
/*data for the table `egresos` */
```

```
/*table structure for table `empleados` */
drop table if exists `empleados`;

create table `empleados` (
  `cedula_empleado` varchar(10) character set utf8 not null,
  `nombres` varchar(200) character set utf8 default null,
  `sexo` varchar(200) character set utf8 default null,
  `direccion` varchar(200) character set utf8 default null,
  `telefono` varchar(50) character set utf8 default null,
  `tipo_empleado` varchar(200) character set utf8 default null,
  `foto` varchar(200) character set utf8 default null,
primary key  (`cedula_empleado`)
) engine=innodb default charset=latin1;
/*data for the table `empleados` */
/*table structure for table `evento` */
drop table if exists `evento`;

create table `evento` (
  `cod_evento` int(11) not null auto_increment,
  `nombre` varchar(200) character set utf8 default null,
  `observacion` varchar(200) character set utf8 default null,
primary key  (`cod_evento`)
) engine=innodb default charset=latin1;
/*data for the table `evento` */
/*table structure for table `historial_reses` */
drop table if exists `historial_reses`;

create table `historial_reses` (
  `cod_historial` int(11) not null auto_increment,
  `cod_res` int(11) default null,
  `estado` varchar(200) character set utf8 default null,
primary key  (`cod_historial`),
key `fk_historial_reses` (`cod_res`)
) engine=innodb default charset=latin1;
/*data for the table `historial_reses` */
/*table structure for table `limpieza` */
drop table if exists `limpieza`;

create table `limpieza` (
```

```
  `cod_limpieza` int(11) not null auto_increment,
  `cod_res` int(11) default null,
`cod_tipo_limpieza` int(11) default null,
`fecha` datetime default null,
  `cedula_empleado` varchar(10) character set utf8 default null,
  `comentario` varchar(200) character set utf8 default null,
primary key  (`cod_limpieza`)
) engine=innodb default charset=latin1;
/*data for the table `limpieza` */
/*table structure for table `lotes` */
drop table if exists `lotes`;

create table `lotes` (
  `cod_lotes` int(11) not null auto_increment,
  `descripcion` varchar(200) character set utf8 default null,
  `capacidad` int(11) default null,
primary key  (`cod_lotes`)
) engine=innodb default charset=latin1;
/*data for the table `lotes` */
/*table structure for table `raza` */
drop table if exists `raza`;

create table `raza` (
  `cod_raza` int(4) not null auto_increment,
`raza` varchar(200) default null,
  `valor_libra` decimal(10,2) default null,
  `valor_mes_preñada` decimal(10,2) default null,
`observacion` varchar(200) default null,
primary key  (`cod_raza`)
) engine=innodb default charset=latin1;
/*data for the table `raza` */
/*table structure for table `reses` */
drop table if exists `reses`;

create table `reses` (
  `cod_res` int(11) not null auto_increment,
  `nombre` varchar(200) character set utf8 default null,
`fecha_nacimiento` date not null,
`peso` int(11) default null,
  `marca_res` varchar(200) character set utf8 default null,
```

```
  `cod_lotes` int(11) default null,
  `cod_descripcion` int(11) default null,
  `cod_actividad` int(11) default null,
  `cod_evento` int(11) default null,
  `precio` decimal(10,0) default null,
primary key  (`cod_res`)
) engine=innodb default charset=latin1;
/*data for the table `reses` */
/*table structure for table `seguimiento_res` */
drop table if exists `seguimiento_res`;

create table `seguimiento_res` (
  `cod_seguimiento` int(11) not null auto_increment,
  `cod_res` int(11) default null,
`fecha_seguimiento` date default null,
`peso` int(11) default null,
  `cod_lotes` int(11) default null,
  `cod_descripcion` int(11) not null,
  `cod_actividad` int(11) default null,
  `cod_evento` int(11) default null,
  `comentario` varchar(200) character set utf8 default null,
primary key  (`cod_seguimiento`)
) engine=innodb default charset=latin1;
/*data for the table `seguimiento_res` */
/*table structure for table `tipo_limpieza` */
drop table if exists `tipo_limpieza`;

create table `tipo_limpieza` (
  `codigo_tipo_limpieza` int(11) not null auto_increment,
`nombre` varchar(200) character set utf8 default null,
  `observacion` varchar(200) character set utf8 default null,
primary key  (`codigo_tipo_limpieza`)
) engine=innodb default charset=latin1;
/*data for the table `tipo_limpieza` */

/*table structure for table `usuarios` */
drop table if exists `usuarios`;

create table `usuarios` (
  `cod_usuario` int(11) not null auto_increment,
```

```
  `usuario` varchar(200) character set utf8 default null,
  `clave` varchar(200) character set utf8 default null,
  `tipo` varchar(200) character set utf8 default null,
  `estado` varchar(200) default null,
primary key  (`cod_usuario`)
) engine=innodb default charset=latin1;
/*data for the table `usuarios` */
/*table structure for table `vacuna` */
drop table if exists `vacuna`;

create table `vacuna` (
  `cod_vacuna` int(11) not null auto_increment,
  `nombre` varchar(200) character set utf8 default null,
  `uso` varchar(500) character set utf8 default null,
  `comentario` varchar(200) character set utf8 default null,
primary key  (`cod_vacuna`)
) engine=innodb default charset=latin1;
/*data for the table `vacuna` */
/*table structure for table `vacunacion` */
drop table if exists `vacunacion`;

create table `vacunacion` (
  `cod_vacunacion` int(11) not null auto_increment,
  `cod_res` int(11) default null,
  `cod_vacuna` int(11) default null,
`fecha_aplicacion` datetime default null,
`comentario` varchar(200) character set utf8 default null,
primary key  (`cod_vacunacion`)
) engine=innodb default charset=latin1;
/*data for the table `vacunacion` */
/*table structure for table `ventas` */
drop table if exists `ventas`;

create table `ventas` (
  `cod_venta` int(11) not null auto_increment,
  `cod_res` int(11) default null,
  `cedula_ruc` varchar(10) character set utf8 default null,
  `cedula_empleado` varchar(10) character set utf8 default null,
`fecha_transaccion` date default null,
  `valor_libra` decimal(19,4) default null,
```

```
  `peso_animal` int(4) default null,
  `adicional_preñada` decimal(10,0) default null,
  `subtotal` decimal(10,0) default null,
  `descuento` int(4) default null,
  `valor_pagar` decimal(10,0) default null,
`observacion` varchar(200) default null,
primary key  (`cod_venta`)
) engine=innodb default charset=latin1;
/*data for the table `ventas` */
```

4.5.5 DIAGRAMAS DE CASO DE USO

En los diagramas de caso de uso de la agropecuaria, es donde se representan gráficamente los procesos que transcurrren en la misma, asi como la interación que existe entre los casos de uso y los administradores. Con el objetivo de comporender mejor el funcionamiento de los procesos que ocurren, que estamos estudiando sie implementan en el siguiente diagrama de casos de uso de la Agropecuaria.

4.5.5.1 CASO DE USO # 1

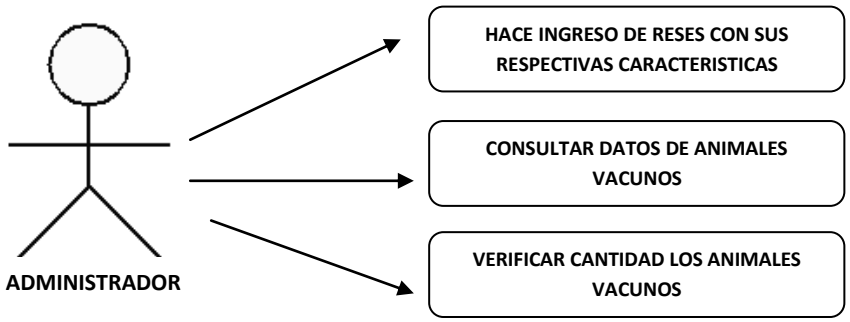
CASO DE USO: Requerimientos para realizar control de ingresos de datos de la agropecuaria.

ACTORES: Administrador

OBJETIVO: Ingresar y verificar cantidad de animales vacunos.

RESUMEN: El administrador consulta el número de animales vacunos disponibles y verifica datos.

DIAGRAMA DE CASO DE USO # 1



4.5.5.2 CASO DE USO # 2

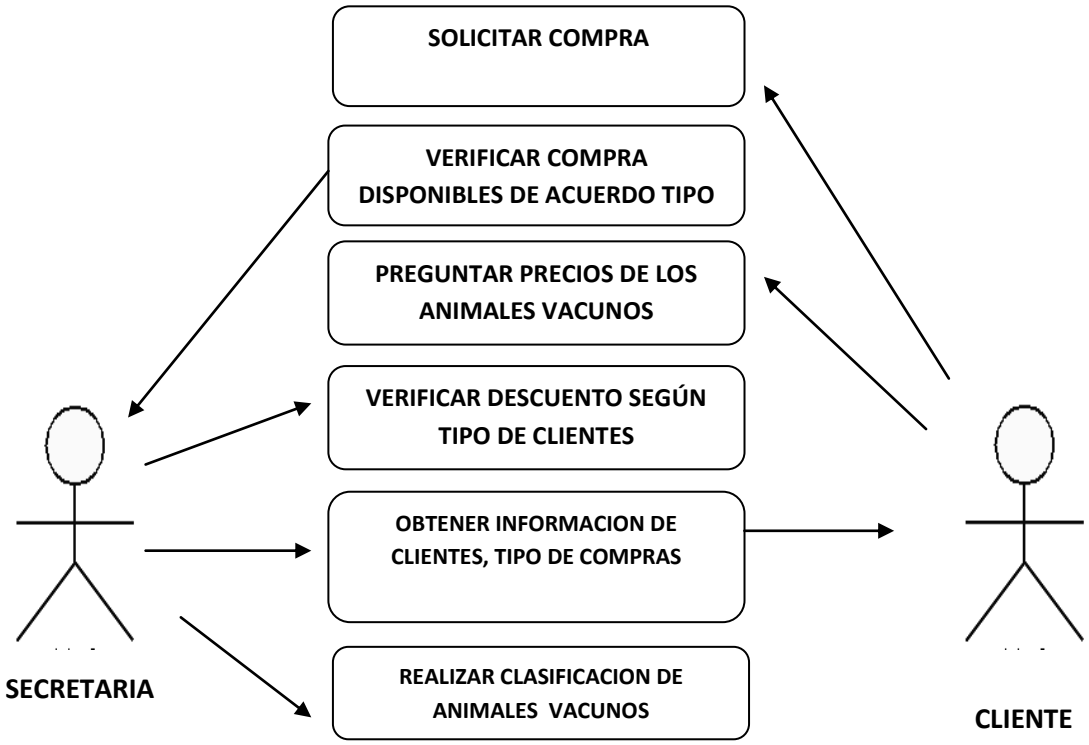
CASO DE USO: Requerimientos para realizar una venta.

ACTORES: Secretaria

OBJETIVO: Registrar una venta.

RESUMEN: Llega el cliente a la agropecuaria y solicita una compra, se verifica la disponibilidad de la misma **y** si el cliente ya ha sido nuestro consumidor antes se le asigna un tipo de descuento. Se obtiene los datos del cliente, tipo cedula, teléfono y se registra la misma en el sistema.

DIAGRAMA DE CASO DE USO # 2



4.5.5.3 CASO DE USO # 3

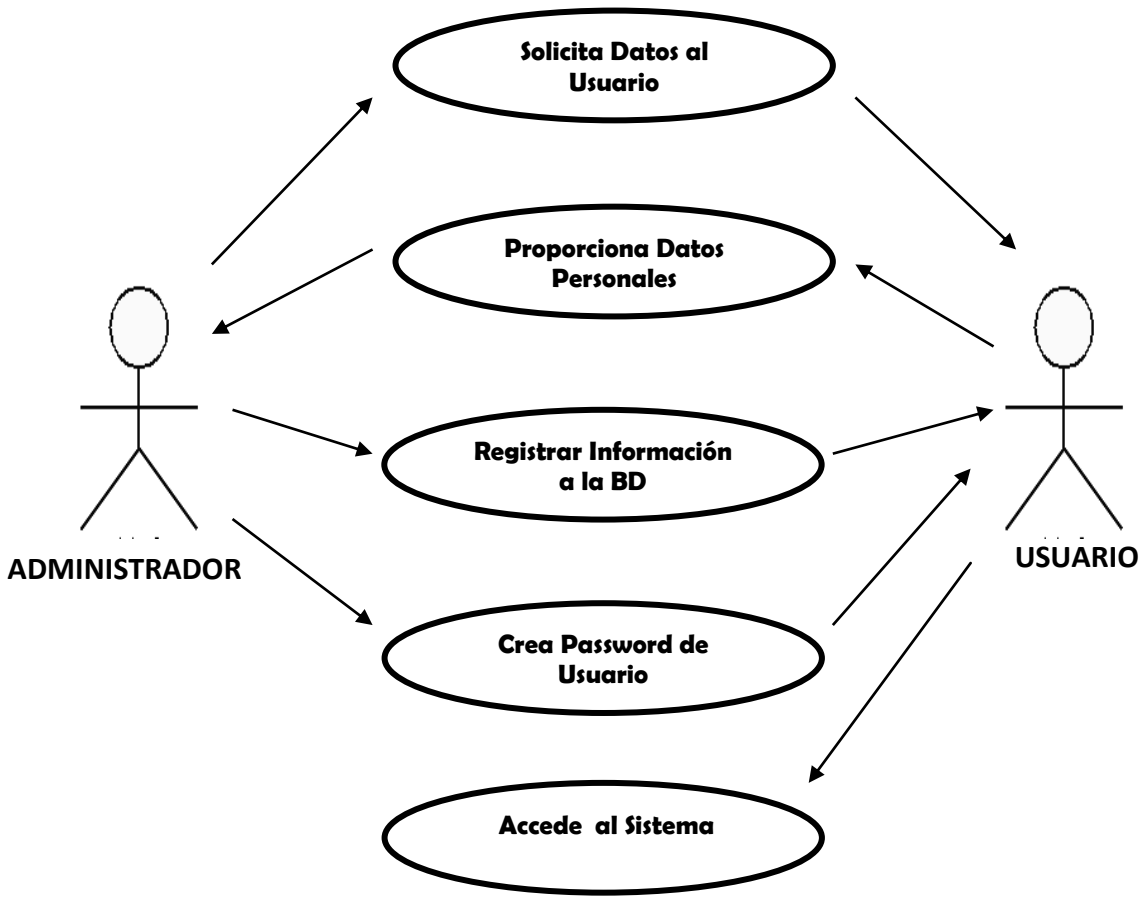
Caso de Uso: Ingreso de Usuarios

Actores: Administrador, Usuario (operador)

Objetivo: Registrar y autorizar usuario al sistema informático

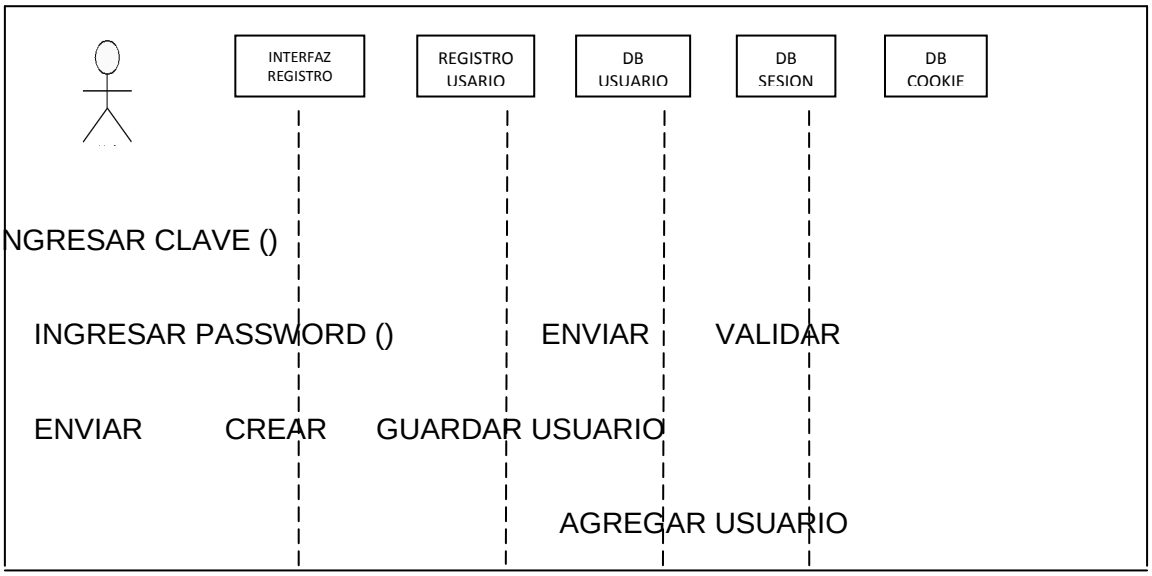
Resumen: El administrador es la persona encargada de nombrar y autorizar usuarios a efectos de realizar la función.

DIAGRAMA DE CASO DE USO # 3

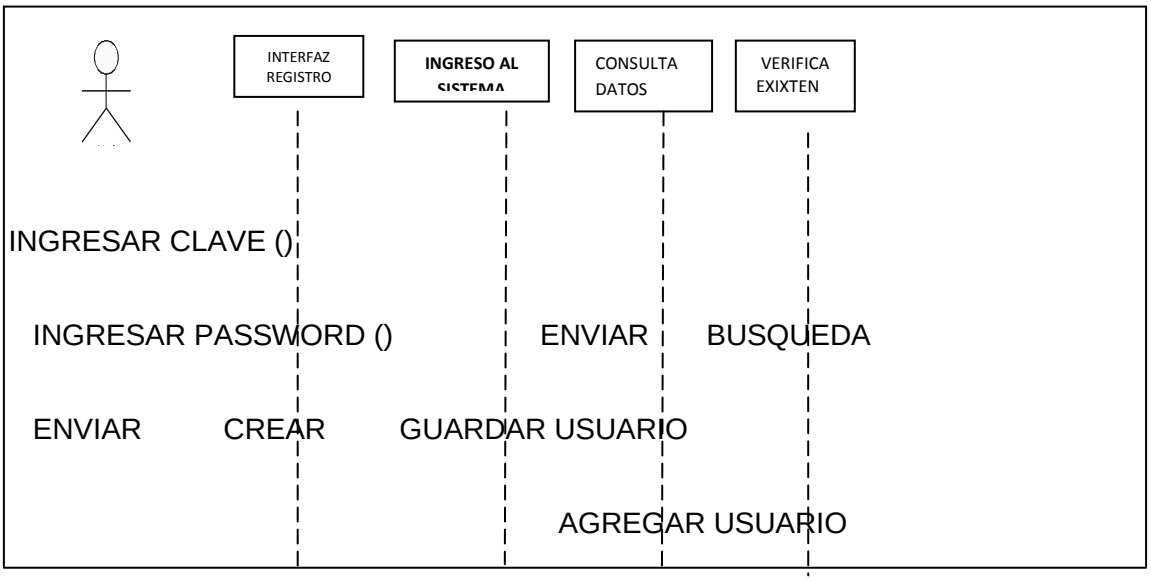


4.5.6 DIAGRAMAS DE SECUENCIA

4.5.6.1 DIAGRAMAS DE SECUENCIA (administrador)



4.5.6.1 DIAGRAMAS DE SECUENCIA (administrador)

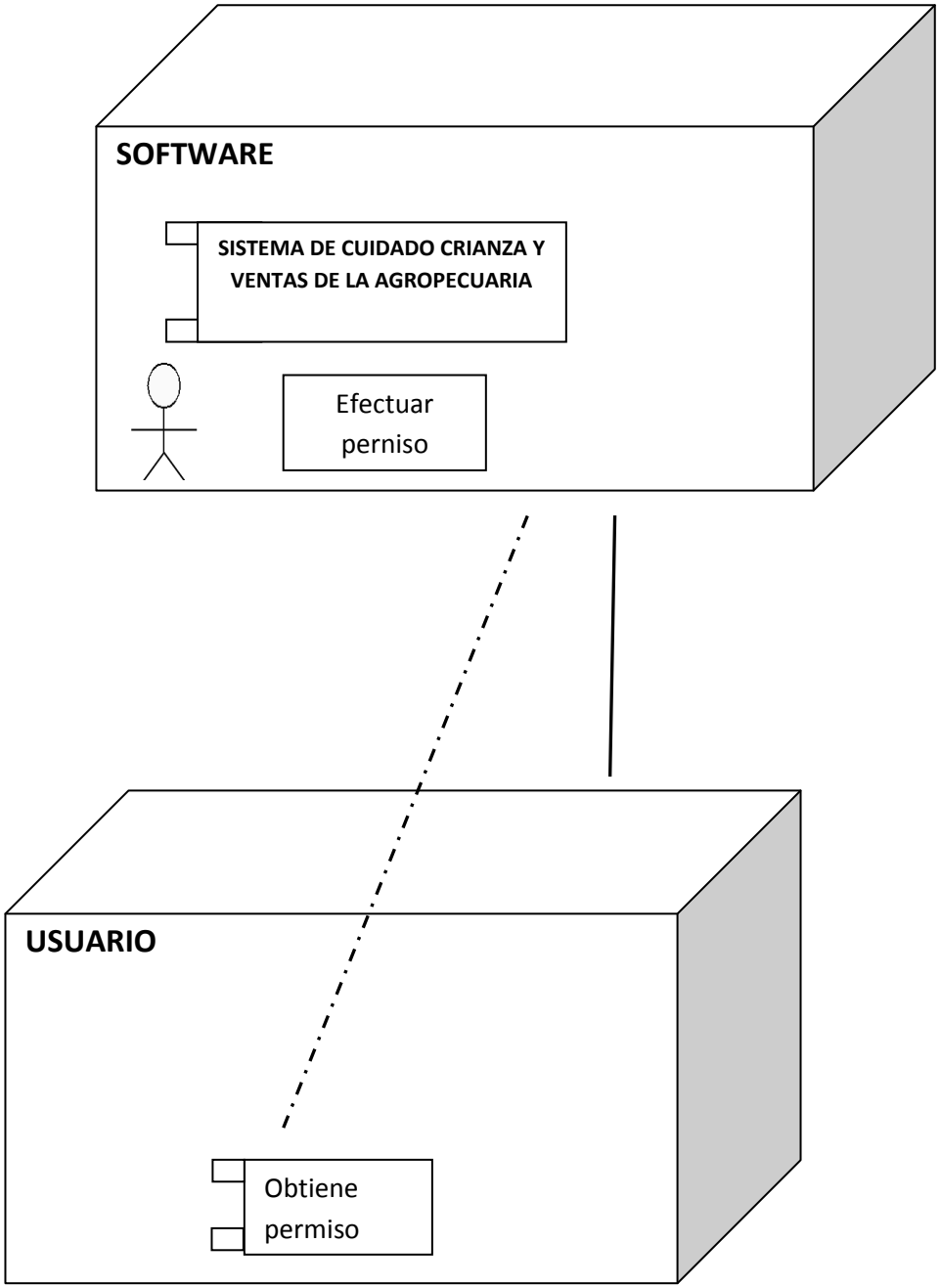


4.5.7 DIAGRAMAS DE ACTIVIDAD

El diagrama de actividad, es donde se ilustra de una forma explicita las labores realizadas en la agropecuaria Velbamal.

4.5.8 DIAGRAMAS DE DESPLIEGUE

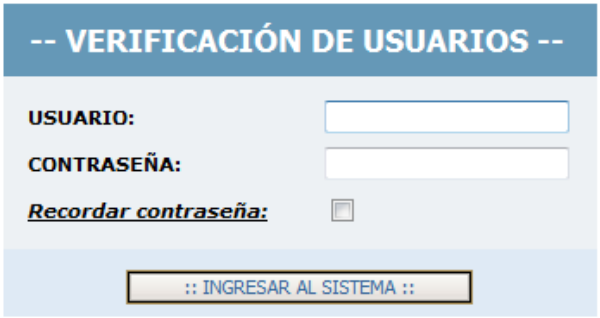
En el Diagrama de Despliegue utilizaremos para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes.



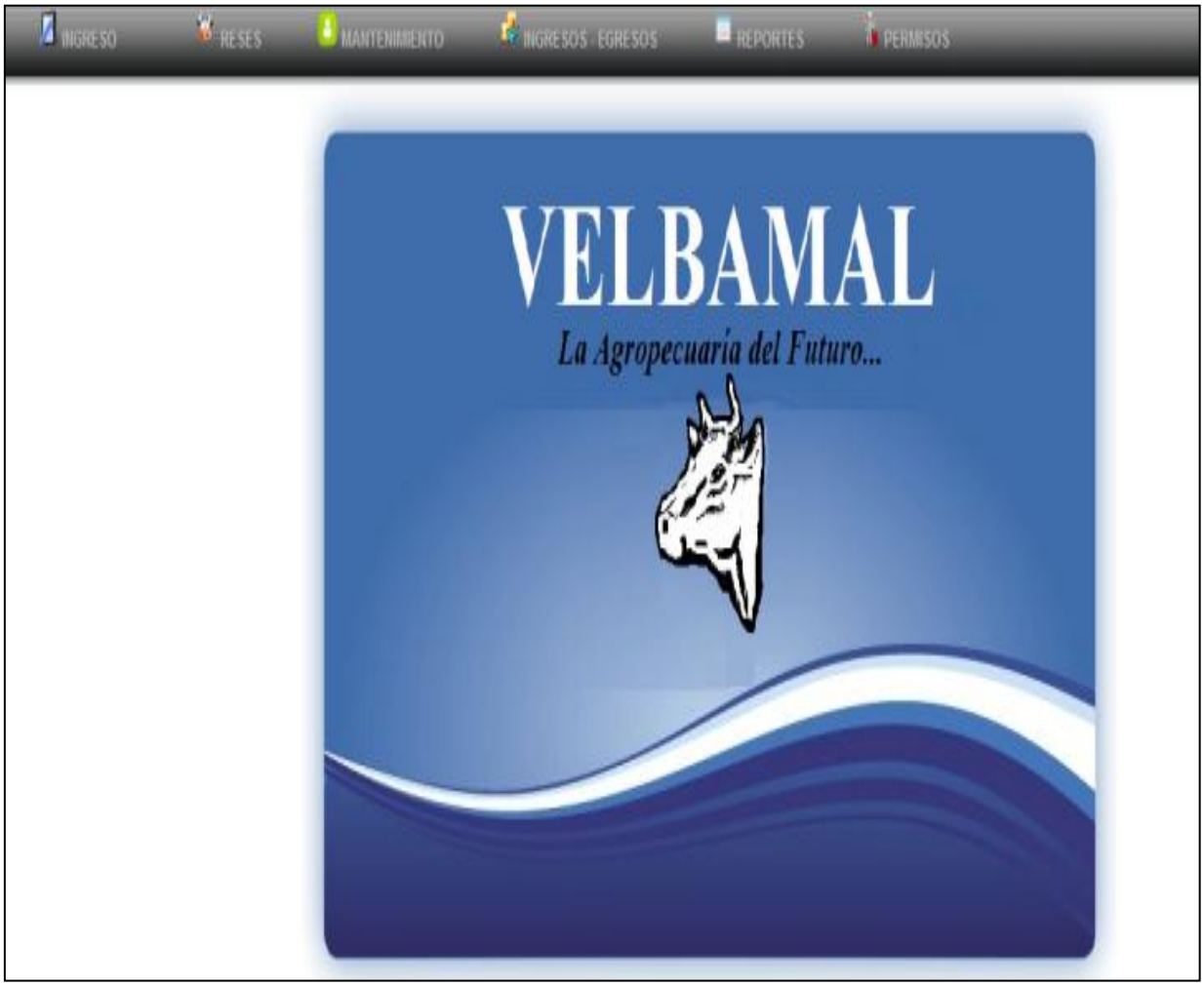
4.5.9 DISEÑO DE INTERFASES

La que se muestra a continuación es la patalla de inicio del sistema.

La siguiente pantalla es donde se valida el usuario.



A continuación se detalla las opciones del sistema, en la que encontramos a modo de menú y submenú respectivamente que tenemos en el sistema.



4.5.10 DISEÑO DE SALIDAS

Permite visualizar la información de l sistema.

Aquí obtenemos los resultados e informaciones generadas por ingresos y egresos que se realiozan en la agropecuaria, por el cual se ve el estdo de la misma.

4.6 DESARROLLO

El sistema se genero mediante php y mysql.

CONEXIÓN

```
<html>
<head>
<title>:::: Conexion con la Base de Datos :::: </title>
</head>
<body>


<?php
function Conectarse()
{
if (!($link=mysql_connect("localhost","velbamal","123velbamal789")))
```

```
{
echo "Error conectando a la base de datos.";
exit();
}
if (!mysql_select_db("velbamal",$link))
{
echo "Error seleccionando la base de datos.";
exit();
}
return $link;
}
?>


</body>
</html>
```

RESES

```
<?php
@ini_set("display_errors","1");
@ini_set("display_startup_errors","1");

include("include/dbcommon.php");

header("Expires: Thu, 01 Jan 1970 00:00:01 GMT");
header("Pragma: no-cache");
header("Cache-Control: no-cache");

include('include/xtempl.php');
include("include/reses_variables.php");
include("include/historial_reses_settings.php");
include("include/limpieza_settings.php");
include("include/seguimiento_res_settings.php");
include("include/vacunacion_settings.php");
include("include/ventas_settings.php");
include('classes/runnerpage.php');
include('classes/listpage.php');
include("classes/searchpanel.php");
include("classes/searchcontrol.php");
include("classes/searchclause.php");
include("classes/panelsearchcontrol.php");

if(!@$_SESSION["UserID"])
```

```
{
$_SESSION["MyURL"]=$_SERVER["SCRIPT_NAME"]."?".$_SERVER["QUERY_STRING"];
header("Location: login.php?message=expired");
return;
}
if(!CheckSecurity(@$_SESSION["_".$strTableName."_OwnerID"],"Search") &&
!CheckSecurity(@$_SESSION["_".$strTableName."_OwnerID"],"Add"))
{
if(IsAdmin())
echo "<p>"."No tiene permiso para acceder a esta tabla"."<br><a
href=\"admin_rights_list.php\">"."Proceder a Administrar"."</a> "."para configurar los premisos de
usuario"."</p>";
else
echo "<p>"."No tiene permiso para acceder a esta tabla"." <a href=\"login.php\">"."Regresar a la
página de conexión"."</a></p>";
exit();
}

//        Include necessary files in accordance with mode displaying page
if (postvalue("mode")=="")
{
$mode = LIST_SIMPLE;
include('classes/listpage_simple.php');
include("classes/searchpanelsimple.php");
}
elseif(postvalue("mode") == "ajax")
{
$mode = LIST_AJAX;
include('classes/listpage_simple.php');
include('classes/listpage_ajax.php');
include("classes/searchpanelsimple.php");
}
elseif(postvalue("mode") == "lookup")
{
include('classes/listpage_embed.php');
include('classes/listpage_lookup.php');
include("classes/searchpanellookup.php");
$mode=LIST_LOOKUP;
//determine which field should be used to select values
$params["lookupSelectField"] = "Cod_Res";
}
elseif(postvalue("mode")=="listdetails")
{
include('classes/listpage_embed.php');
```

```
include('classes/listpage_dpinline.php');
$mode=LIST_DETAILS;
}
$xt = new Xtempl();

// Modify query: remove blob fields from fieldlist.
// Blob fields on a list page are shown using imager.php (for example).
// They don't need to be selected from DB in list.php itself.
$gQuery->ReplaceFieldsWithDummies(GetBinaryFieldsIndices());

$options = array();
//array of params for classes
$options["pageType"] = PAGE_LIST;
$options["id"] = postvalue("id") ? postvalue("id") : 1;
$options["mode"] = $mode;
$options['xt'] = &$xt;
$options['masterPageType'] = postvalue("masterpagetype");
$options["masterTable"] = postvalue("mastertable");
$options["masterId"] = postvalue("masterid");
$options["firstTime"] = postvalue("firsttime");

$i = 1;
while(isset($_REQUEST["masterkey".$i]))
{
$options["masterKeysReq"][$i] = $_REQUEST["masterkey".$i];
$i++;
}
$pageObject = ListPage::createListPage($strTableName, $options);


// prepare code for build page
$pageObject->prepareForBuildPage();

$includesArr = array();
$masterTablesInfoArr = GetMasterTablesArr($strTableName);
for($i=0;$i<count($masterTablesInfoArr);$i++)
{
if($masterTablesInfoArr[$i]['dispInfo'])
$includesArr[] = getabspath("include/".$masterTablesInfoArr[$i]['mShortTable']."_masterlist.php");
}

//include files if need
for($i=0;$i<count($includesArr);$i++)
include($includesArr[$i]);
```

```
// show page depends of mode
$pageObject->showPage();

?>
```

SEGUIMIENTO DE RES

```
<?php
@ini_set("display_errors","1");
@ini_set("display_startup_errors","1");

include("include/dbcommon.php");

header("Expires: Thu, 01 Jan 1970 00:00:01 GMT");
header("Pragma: no-cache");
header("Cache-Control: no-cache");

include('include/xtempl.php');
include("include/seguimiento_res_variables.php");
include('classes/runnerpage.php');
include('classes/listpage.php');
include("classes/searchpanel.php");
include("classes/searchcontrol.php");
include("classes/searchclause.php");
include("classes/panelsearchcontrol.php");

if(!@$_SESSION["UserID"])
{
	$_SESSION["MyURL"]=$_SERVER["SCRIPT_NAME"]."?".$_SERVER["QUERY_STRING"];
	header("Location: login.php?message=expired");
	return;
}
if(!CheckSecurity(@$_SESSION["_".$strTableName."_OwnerID"],"Search") &&
!CheckSecurity(@$_SESSION["_".$strTableName."_OwnerID"],"Add"))
{
	if(IsAdmin())
		echo "<p>"."No tiene permiso para acceder a esta tabla"."<br><a
href=\"admin_rights_list.php\">"."Proceder a Administrar"."</a> "."para configurar los premisos de
usuario"."</p>";
	else
		echo "<p>"."No tiene permiso para acceder a esta tabla"." <a
href=\"login.php\">"."Regresar a la página de conexión"."</a></p>";
	exit();
}
```

```
//	Include necessary files in accordance with mode displaying page
if (postvalue("mode")=="")
{
	$mode = LIST_SIMPLE;
	include('classes/listpage_simple.php');
	include("classes/searchpanelsimple.php");
}
elseif(postvalue("mode") == "ajax")
{
	$mode = LIST_AJAX;
	include('classes/listpage_simple.php');
	include('classes/listpage_ajax.php');
	include("classes/searchpanelsimple.php");
}
elseif(postvalue("mode") == "lookup")
{
	include('classes/listpage_embed.php');
	include('classes/listpage_lookup.php');
	include("classes/searchpanellookup.php");
	$mode=LIST_LOOKUP;
	//determine which field should be used to select values
			$params["lookupSelectField"] = "Cod_Seguimiento";
}
elseif(postvalue("mode")=="listdetails")
{
	include('classes/listpage_embed.php');
	include('classes/listpage_dpinline.php');
	$mode=LIST_DETAILS;
}
$xt = new Xtempl();

// Modify query: remove blob fields from fieldlist.
// Blob fields on a list page are shown using imager.php (for example).
// They don't need to be selected from DB in list.php itself.
$gQuery->ReplaceFieldsWithDummies(GetBinaryFieldsIndices());

$options = array();
//array of params for classes
$options["pageType"] = PAGE_LIST;
$options["id"] = postvalue("id") ? postvalue("id") : 1;
$options["mode"] = $mode;
$options['xt'] = &$xt;
$options['masterPageType'] = postvalue("masterpagetype");
```

```
$options["masterTable"] = postvalue("mastertable");
$options["masterId"] = postvalue("masterid");
$options["firstTime"] = postvalue("firsttime");

$i = 1;
while(isset($_REQUEST["masterkey".$i]))
{
	$options["masterKeysReq"][$i] = $_REQUEST["masterkey".$i];
	$i++;
}
$pageObject = ListPage::createListPage($strTableName, $options);


// prepare code for build page
$pageObject->prepareForBuildPage();

$includesArr = array();
$masterTablesInfoArr = GetMasterTablesArr($strTableName);
for($i=0;$i<count($masterTablesInfoArr);$i++)
{
	if($masterTablesInfoArr[$i]['dispInfo'])
		$includesArr[] =
getabspath("include/".$masterTablesInfoArr[$i]['mShortTable']."_masterlist.php");
}

//include files if need
for($i=0;$i<count($includesArr);$i++)
	include($includesArr[$i]);

// show page depends of mode
$pageObject->showPage();

?>
```

4.7 PRUEBAS

4.7.1 IMPLEMENTACION DEL SISTEMA

4.7.1.1 REQUERIMIENTOS DE HARDWARE

Para la realización de este proyecto se necesitan los siguientes requerimientos de Hardware.

- ✓ 1 PC
- ✓ 2 GB de memoria RAM DDR-2
- ✓ Procesador Pentium IV de 3.2 GHz
- ✓ Disco duro de 120 GB
- ✓ Motherboard Asrock
- ✓ 1 Tarjeta de Red inalámbricas con una Velocidad de 100/10
- ✓ 1 Monitores LCD
- ✓ 1 Mouse
- ✓ 1 Teclado
- ✓ 1 Impresora HP 4200 Multifunción
- ✓

4.7.1.2 REQUERIMIENTOS DE SOFTWARE

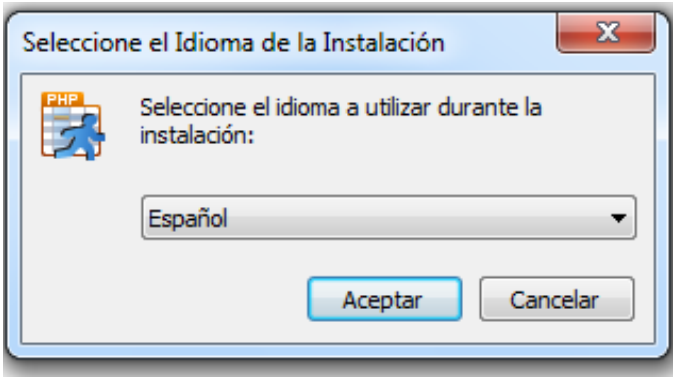
El Software utilizado para nuestro sistema es:

- ✓ Windows XP Profesional SP2
- ✓ Software de PHPRUNNER
- ✓ Software de Base de datos MYSQL.
- ✓ Software de Office 2007 Word.

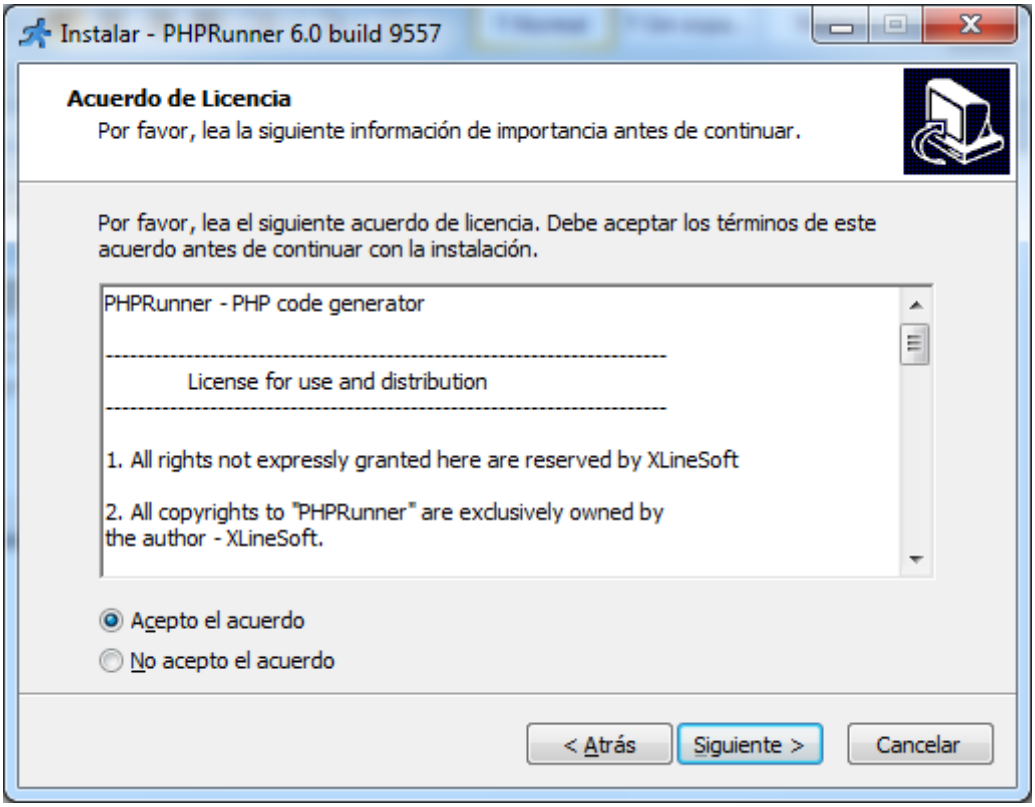
4.7.1.3 PROCESO DE INSTALACION

4.7.1.3.1 INSTALACION DE PHPRUNNER

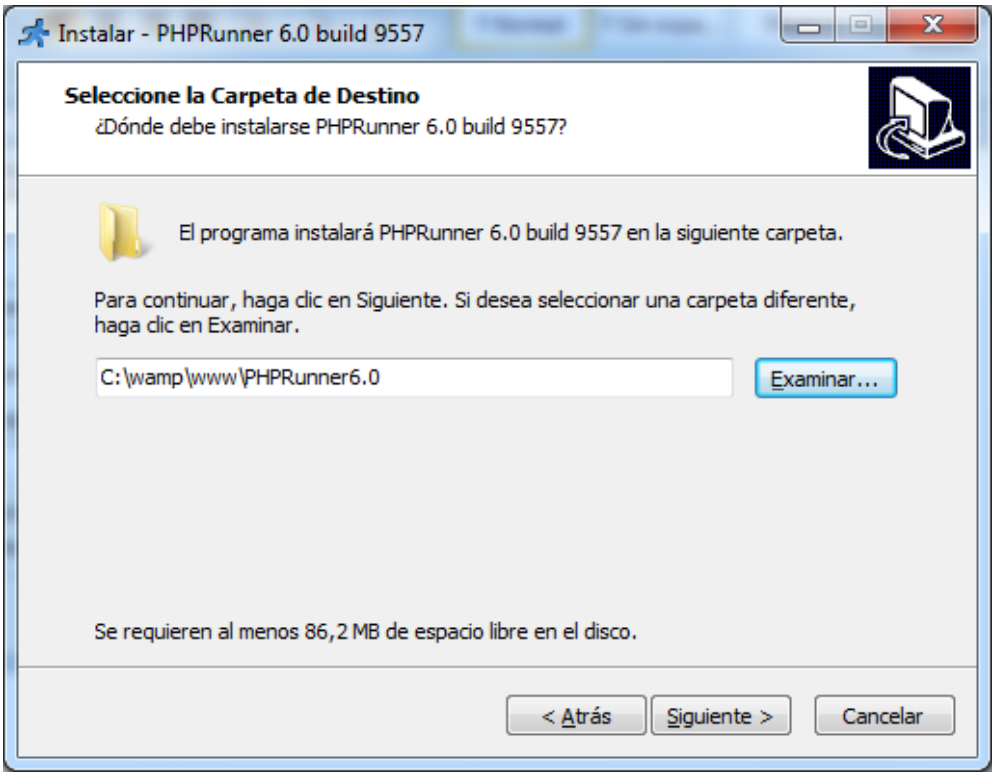
1.- Como primer paso seleccionamos el idioma…



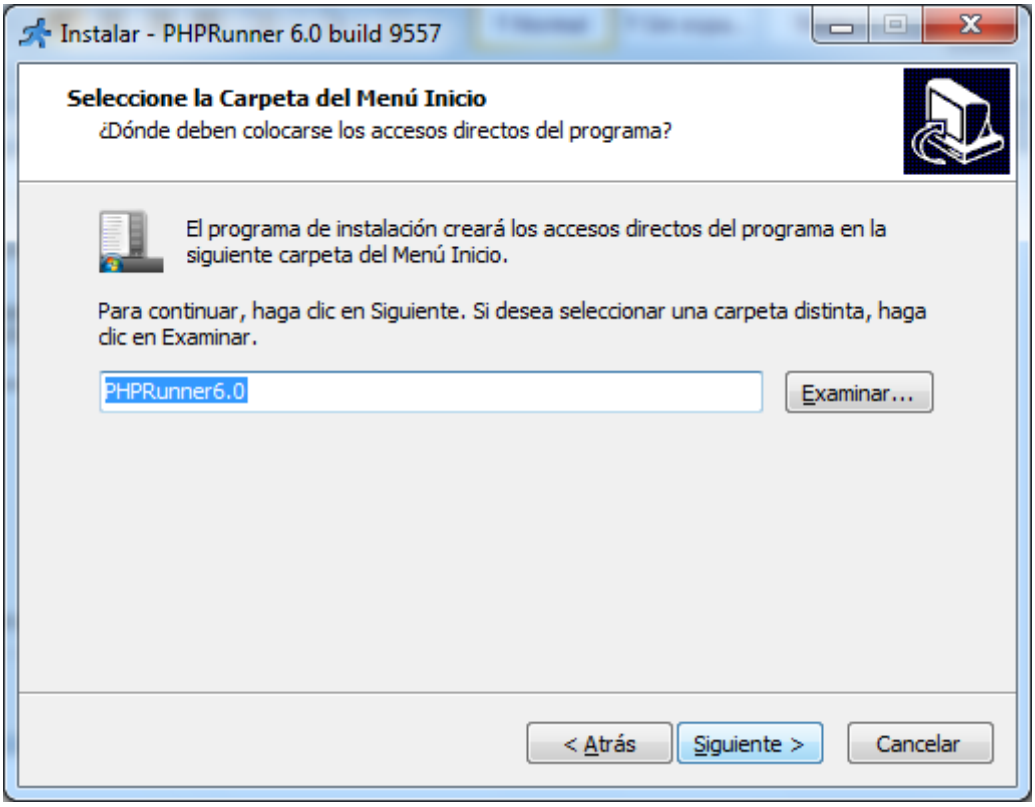
2.- Aceptamos la licencia.



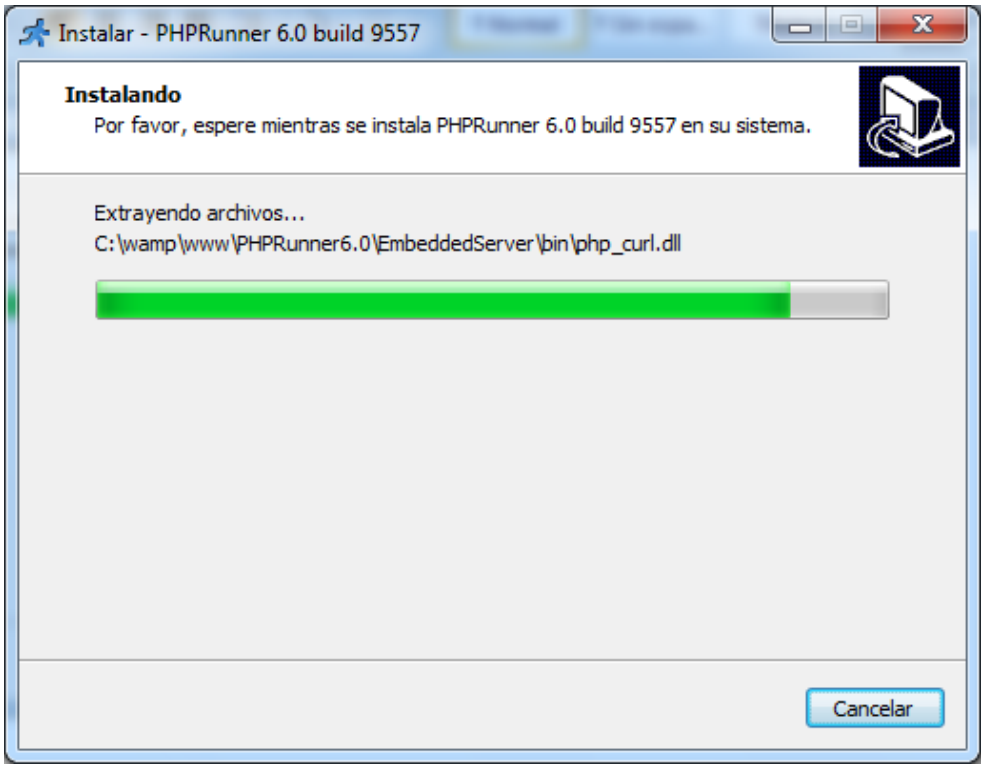
3.- Seleccionamos la ruta que va a tener nuestro archivo.



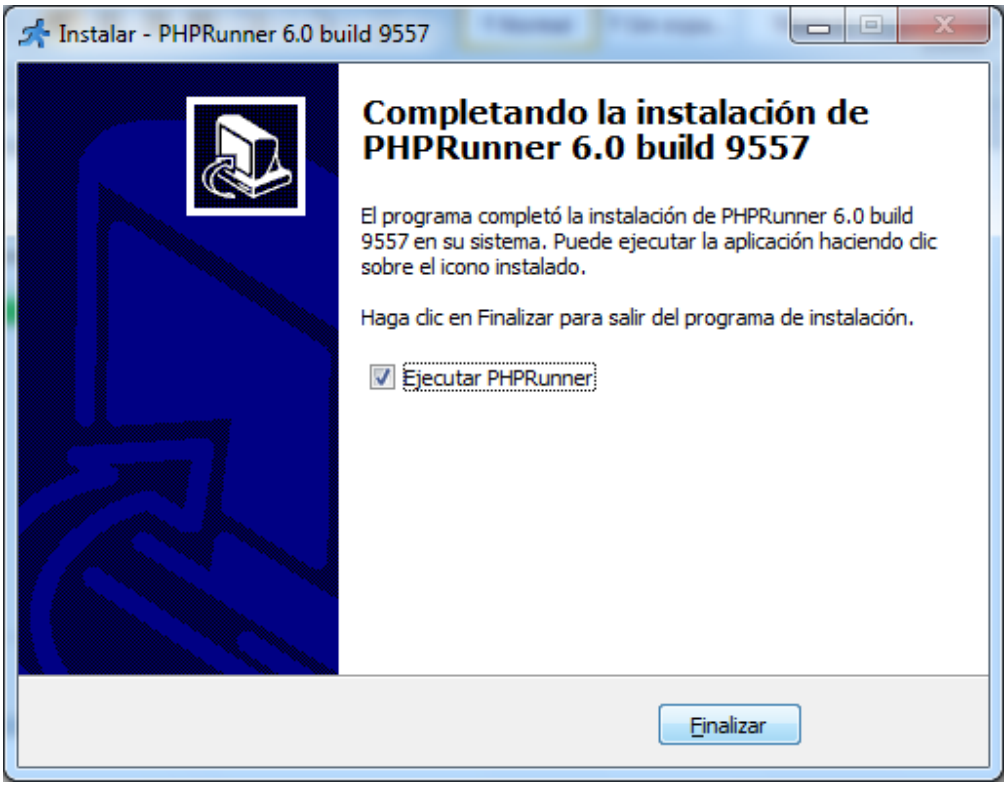
4.- El nombre de la carpeta con que vamos aencontrar phprunner



5.- Inicia la instalación

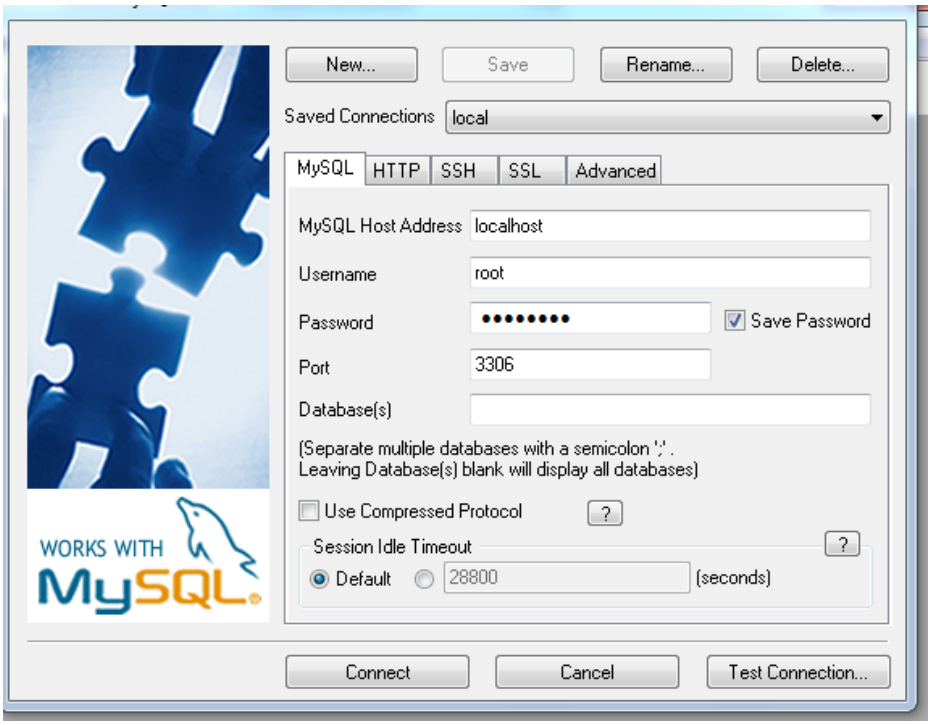


6.- Finalizando la instalación.

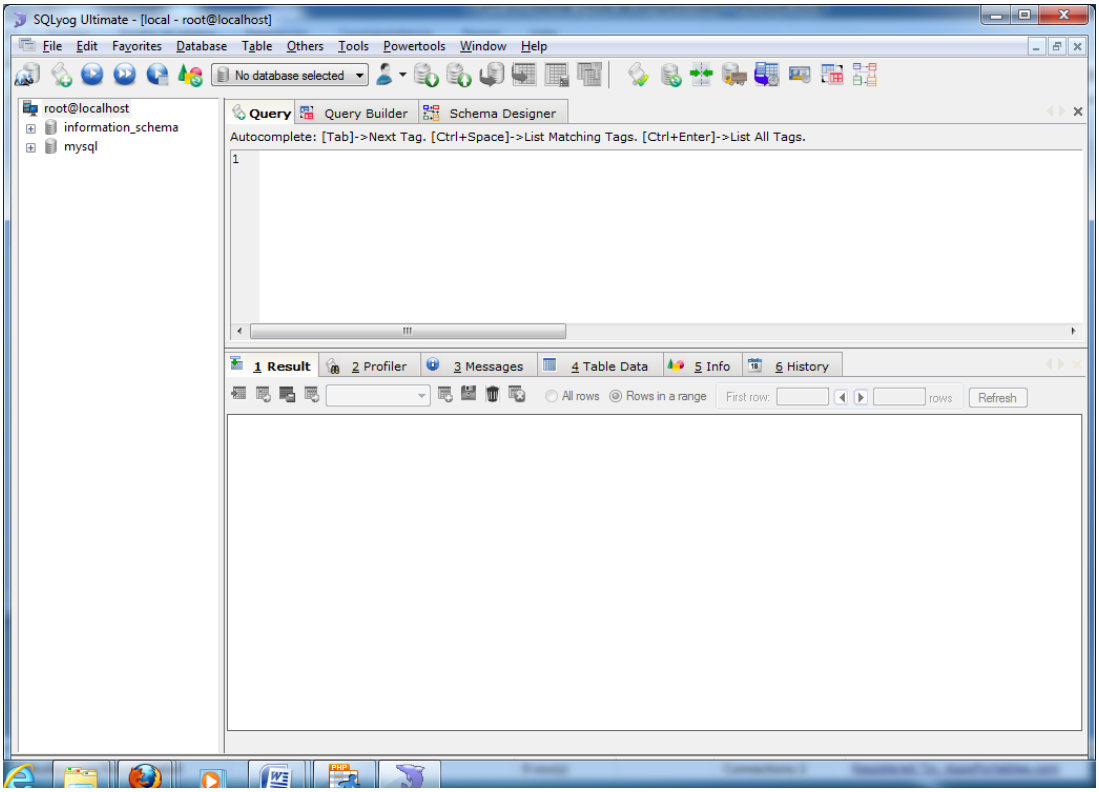


4.7.1.3.2 INSTALACION DE MYSQL

1.-La instalación de mysql, comienza con la coneccion del usuario y el password.



2.- la pantalla siguiete es donde nos permite crear la base de datos.



4.8 CONCLUSIONES Y RECOMENDACIONES PARA UNA EFICIENTE MPLEMENTACION DEL SOFTWARE

CONCLUSION

Como conclusión podemos describir en este sistema a parte de su evidente utilidad es muy ameno y fácil de manipular ya que su interfaz es muy didáctica, además de contar con una base de datos capaz de almacenar grandes volúmenes de información lo cual evita la tediosa tarea de sacar respaldos en periodos muy cortos de tiempo.

La atención personalizada y buen proceso de ventas en la Agropecuaria son la clave para llegar al éxito en el mundo de las Agropecuarias además de un buen servicio en lo que atención al cliente se refiere.

RECOMENDACIONES

- Reemplazar los equipos de oficinas obsoletos (maquinas de escribir, fax, otros), por nuevos equipos con tecnología de punta (computadores); además de acondicionar el ambiente laboral para proteger los equipos informáticos.
- Se debe considerar, el uso adecuado de cada una de las herramientas de hardware y software del sistema, para que la información que se transmita sea veraz y oportuna, lográndose así una comunicación efectiva que influya directamente en la toma de decisiones, por ser la información uno de los recursos esenciales en la solución de problemas en una organización.

- Mientras se desarrolla el Sistema informático propuesto; se recomienda revisarlo periódicamente, con la finalidad de mejorarlo o por el contrario obviar pasos innecesarios.

- Para evitar que el Sistema se vuelva lento con el pasar del tiempo o por el acumulamiento de información es necesario sacar respaldos en un periodo de cada seis meses además de efectuar un mantenimiento preventivo en el mismo periodo de tiempo, a efectos de mantener los equipos hardware en buen estado.
- Mantener la privacidad de las claves de cuentas de usuarios para disminuir el riesgo de ataques externos al sistema y cerrar la sesión de trabajo del mismo, en caso de ausencia temporal del usuario para prevenir el mal manejo de los procesos.

BIBLIOGRAFIA

- Lee, Craig (June 14, 2010), ADO.NET Entity Framework Unleashed (1st ed.), Sams, pp. 600, ISBN 0-672-33074-1,
- http://www.informit.com/store/product.aspx?isbn=0672330741[dead link]
- Lerman, Julia (August, 2010), Programming Entity Framework (2nd ed.), O'Reilly Media, pp. 912, ISBN 978-0-596-80726-9,
- http://oreilly.com/catalog/9780596807252/
- Jennings, Roger (February 3, 2009), Professional ADO.NET 3.5 with LINQ and the Entity Framework (1st ed.), Wrox, pp. 672, ISBN 0-470-18261-X, http://www.wiley.com/WileyCDA/WileyTitle/productCd-047018261X.html
- FREEDMAN, Alan. (1991), Diccionario de Computación, McGrawHill, 5ta ED, España.
- Kendall & Kendall; Análisis y Diseño de Sistemas; 3ª Edición; Pearson Educación.
- Roger S. Pressman; Ingeniería del Software;4ª Edición; Mc Graw Hill.

- Kendall & Kendall; Análisis y Diseño de Sistemas; 3ª Edición; Pearson Educación.
- Roger S. Pressman; Ingeniería del Software;4ª Edición; Mc Graw Hill.

LINKOGRAFIAS

http://www.php.net/manual/es/preface.php, Autor: Bill Abt

http//www.programacion.com/php/kelvin torres

http://www.codeplex.com/datadictionary

http://go.microsoft.com/fwlink/?LinkId=82745

http://es.wikipedia.org/w/index.php?title=Microsoft_SQL_Server&oldid=50249512

http://es.wikipedia.org/w/index.php?title=Visual_Basic_.NET&oldid=50336212

MANUAL DE USUARIO

Objetivo

El objetivo de este documento es guiar al usuario para que lleve a cabo de manera exitosa el Sistema de Gestión de Cuidado Crianza y ventas de Ganado en la Agropecuaria Velbamal, teniendo para ello una explicación breve de los pasos a seguir y de la utilización de todas las opciones incluidas en este proceso.

PAGINA: Index.php

Abra el Internet Explorer y diríjase a la barra de direcciones y digite la dirección que le permitirá ingresar al Sistema, la dirección que deberá digitar tiene el siguiente formato: http://www.localhost.VELBAMAL, donde **VELBAMAL**es el nombre del archivo en donde se encuentra guardadonuestro Sistema con su respectiva base de datos.

Una vez digitada la dirección presione la tecla ENTER y le aparecerá una pantalla como la mostrada a Continuación.

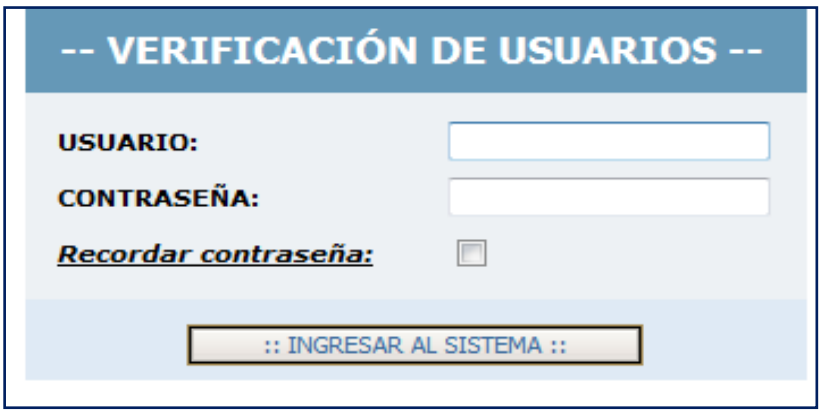
Clic derecho para avanzar a la siguiente pantalla donde encontramos login.php.

PAGINA: login.php

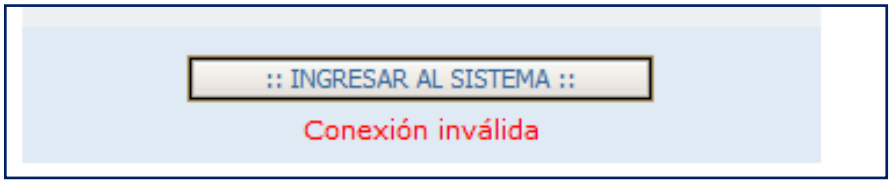
Acceso al sistema

Para acceder al sistema deberá digitar su usuario y su clave, que le serán suministrados por el administrador del sistema de su franquicia, una vez ingresados estos valores, puede presionar la tecla ***"ENTER"*** o hacer click en el botón ***"INGRESAR AL SISTEMA"***.

Usuario: Aquí ingresamos el nombre del usuario registrado en la base de datos.



De lo contrario si no lo ingresamos correctamente nos saldrá el mensaje de Conexión inválida



Estructura General de Pantallas

El Sistema mostrará una pantalla como se muestra en la Portada , está constituida por cuatro partes que son:

- DATOS GENERALES DE CONEXIÓN.
- MENÚ DE OPCIONES.
  Ingreso.
  Reses.
  Mantenimiento
  Ingresos-Egresos
  Reportes
  Permisos

- HISTORIAL DE RESES
- ÁREA DE TRABAJO.

PAGINA: Portada

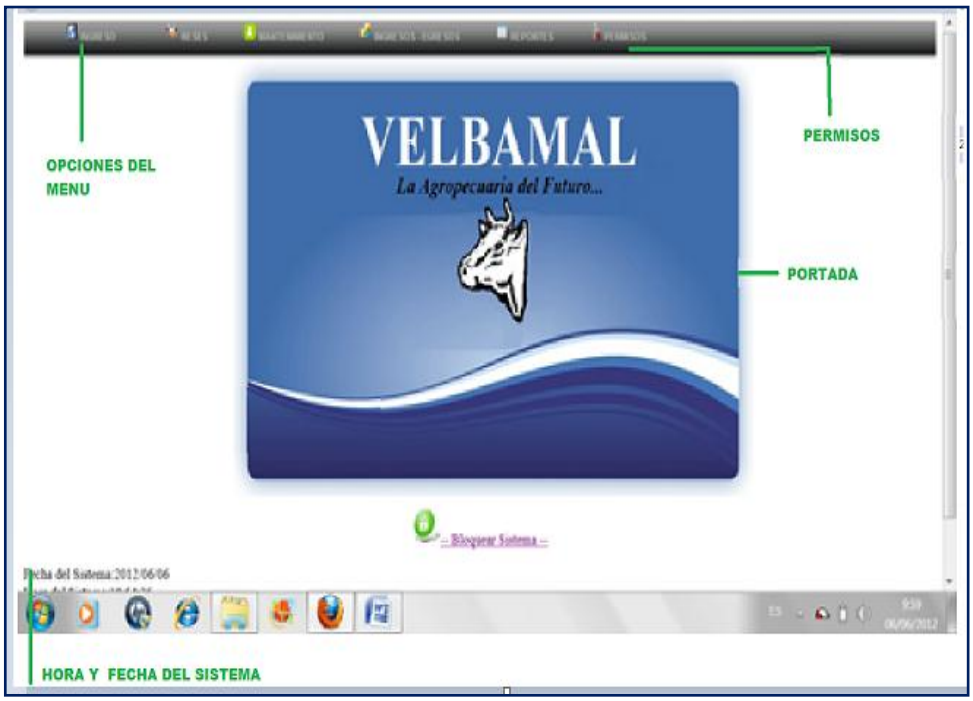
- **Datos Generales de Conexión**

Información de conexión

En la sección de **Datos Generales de Conexión** se puede observar la siguiente información:

- Usuario conectado.
- Nombre Corto de la Empresa sobre os que se realizan las consultas y transacciones.
- Punto de Venta sobre los que se realizan las consultas y transacciones.
- Estado de la Agropecuaria asignada al usuario y en la que se registrarán los cobros realizados y la pérdida donde observamos el Saldo en la Agropecuaria.
- Versión del sistema con el que se encuentra trabajando.

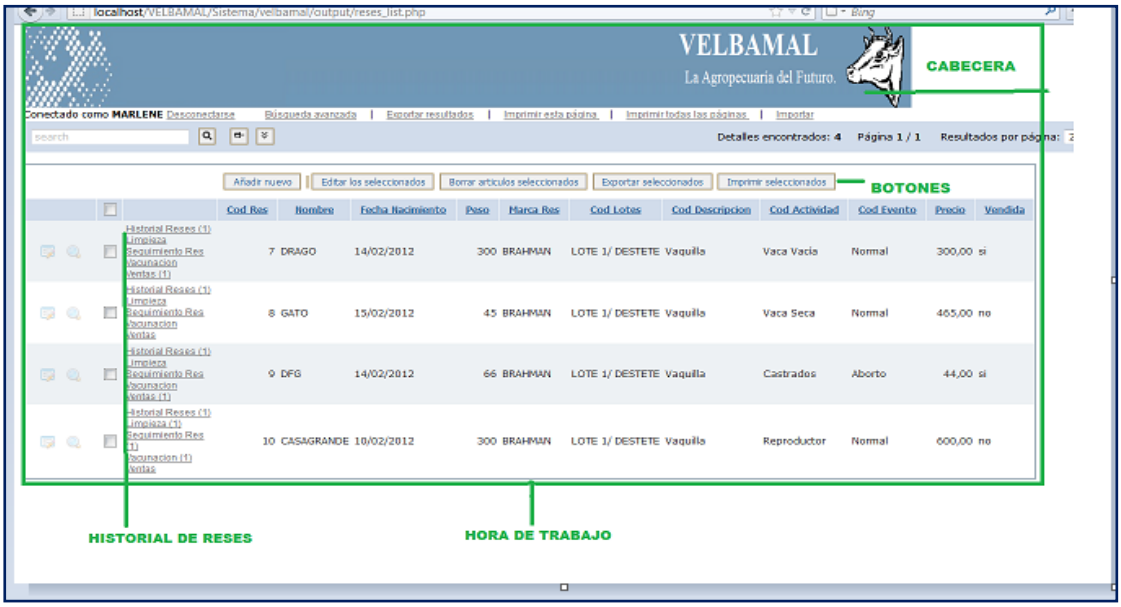
A continuación tenemos la portada del sistema en donde podemos visualizar las opciones del Menú, Permisos, Hora y Fecha de la entrada en el Sistema.



El Programa constara con la Fecha que entra el Usuario al Sistema y la Hora Programada en la maquina.

Fecha del Sistema:2012/06/06
Hora del Sistema:05:56:55

Dentro de cada Submenú tendremos datos como los que tenemos en esta página en el cual podemos ingresar, actualizar, eliminar, exportar, imprimir los datos sea en Word o Excel.



También contamos con el Año de Vigencia, lugar y Provincia donde pertenece este Sistema, con sus Botones de Regreso a la Pantalla Principal.

HISTORIAL DE RESES

En esta sección cuenta con un **"Historial de Reses"**es en donde se muestra una lista de los ingresos de las Reses, los seguimientos en la Reses, iniciando con la más reciente.

ÁREA DE TRABAJO

En esta sección se mostrarán todas las pantallas que le permitirán observar en el Sistema PHP, ya sea en pantallas de cuidado, crianza y ventas realizadas en la Agropecuaria.

ESTRUCTURA DE PANTALLAS

A continuación los Menú en donde obtendremos los respectivos Submenú desde la pagina Principal en nuestro Sistema.

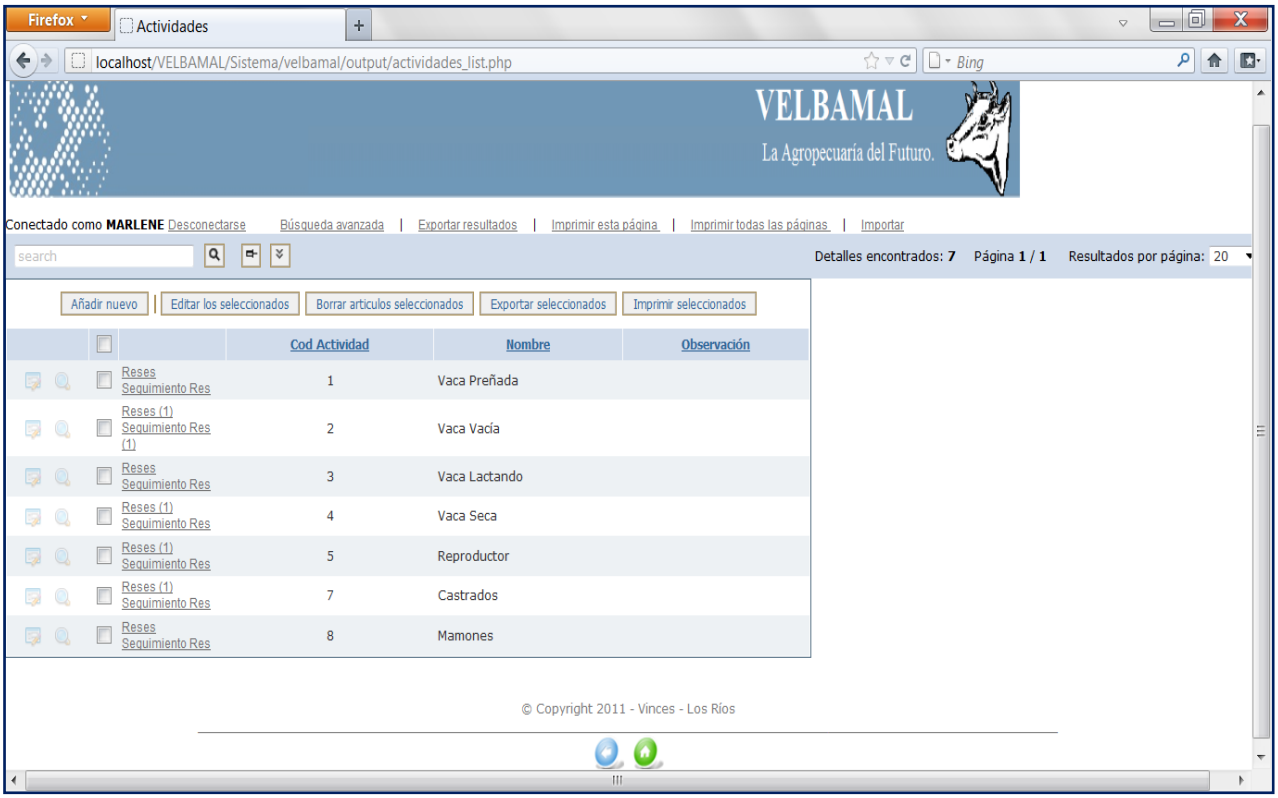
MENU: Ingreso

En este Menú tenemos ingreso de: Actividades, Evento, Descripción de Res, Lotes, Vacunas, Tipo de Limpieza, Raza de Res, en donde encontraremos cada cualidad de estos animales de una forma constante y verídica.

A continuación la información de cada uno de ellos:

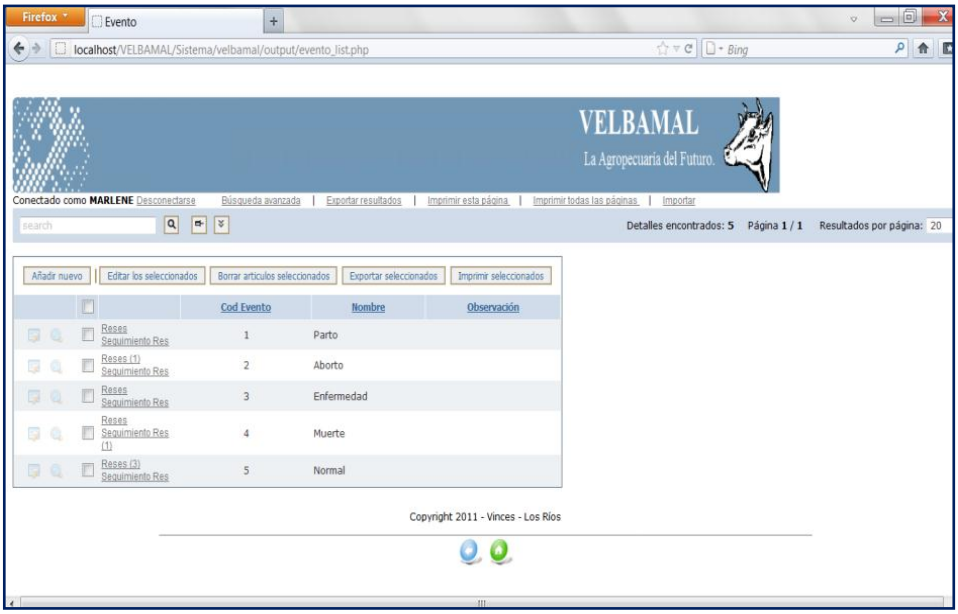
Actividades

En esta pantalla encontramos las Actividades del Animal es decir si es una Vaca verificaremos si está preñada, Vacía, Lactando, Seca, Mamones O si es un Toro si esta en Reproducción o esta Castrado.



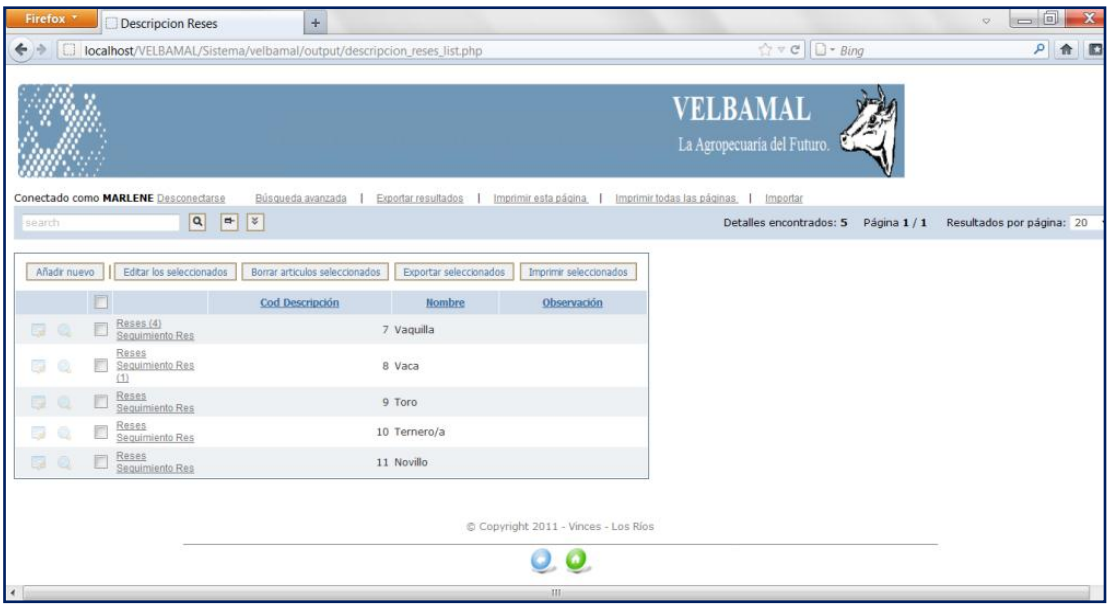
EVENTOS

Observamos en esta pantalla que tenemos los eventos del animal: Parto, Aborto, Enfermedad, Muerte Normal.



DESCRICION DE RES

En la pantalla de la Descripción de Res tenemos si es una vaquilla, vaca, toro ternero, novillo.



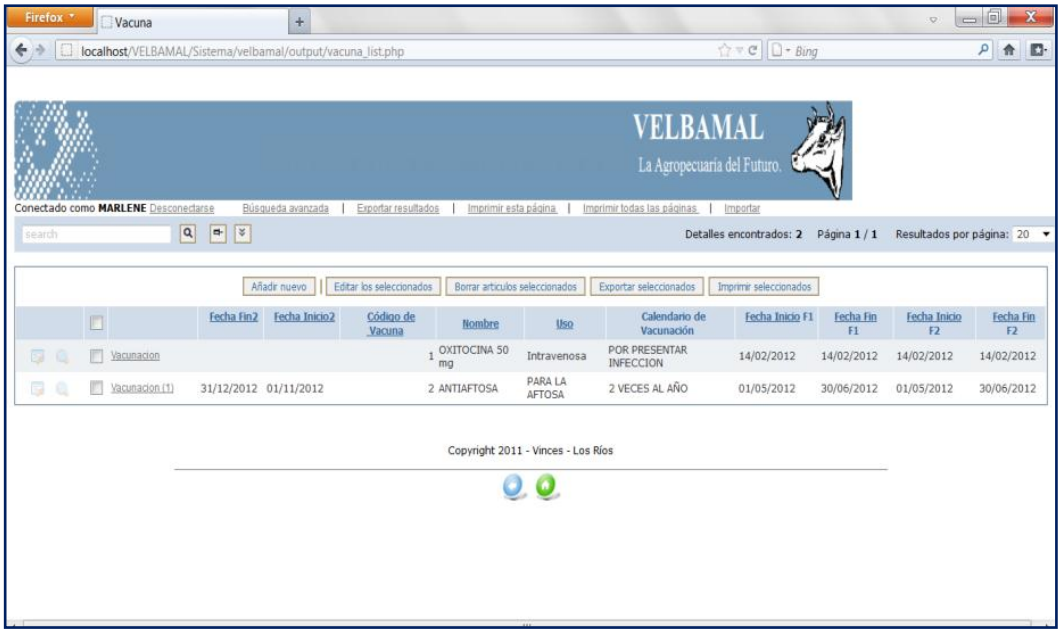
LOTES

En esta Agropecuaria encontramos 3 Lotes, destete, engorde, ceba.

VACUNAS

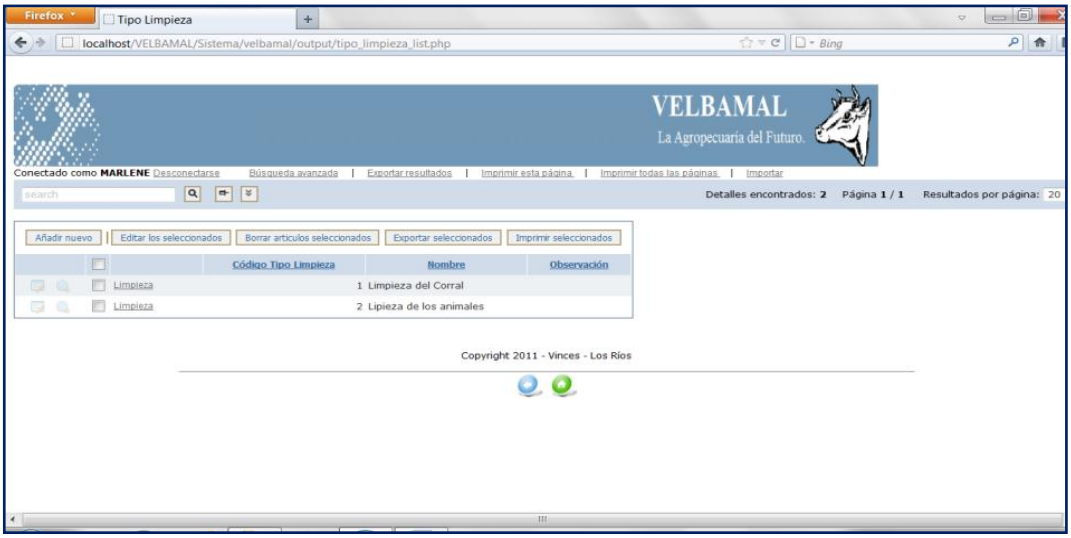
En la Pantalla de Vacuna tenemos Codigo de la Vacuna, Nombre de la Vacuna, Uso de la Vacuna, Calendario de Vacunación, Fecha de Inicio y Fecha de Fin

Y de aquí saldrá el Calendario Alarma que tendrá cada Animal Vacuno para el tipo de Vacunación que se ingrese en el Sistema



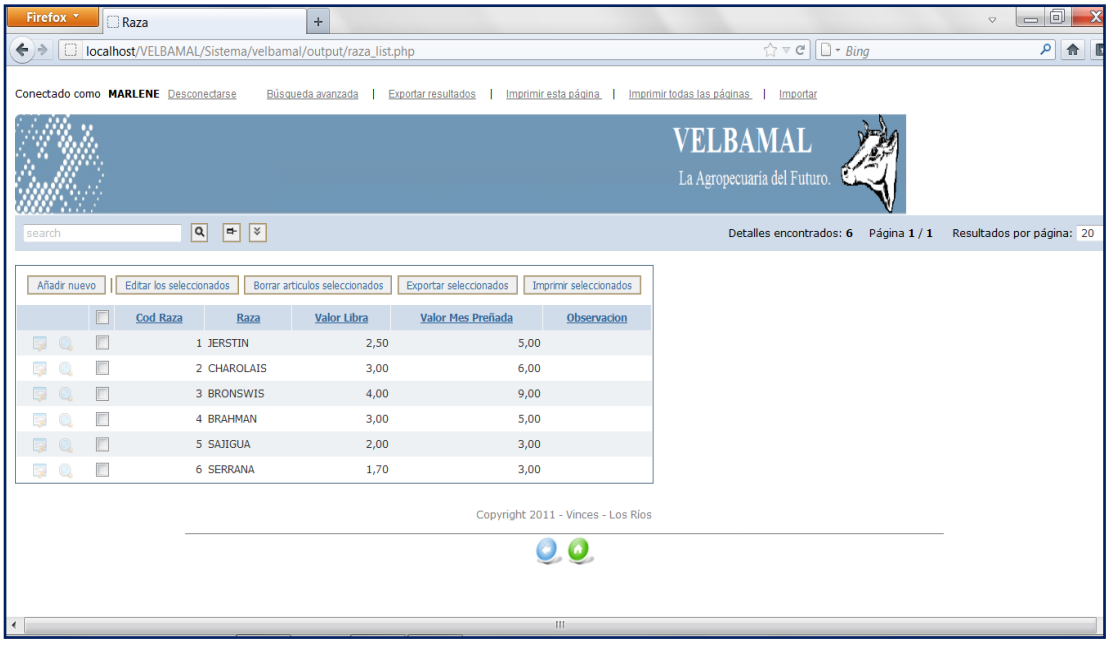
TIPO DE LIMPIEZA

En este cuadro tenemos 2 clases de Limpieza para estos Animales en la Agropecuaria son: Limpieza de Corral y la Limpieza de los Animales.



RAZA DE RES

La raza de Res observamos el Código de Res, Raza del Animal, Valor libra, Valor Preñada, Observación de cada animal.

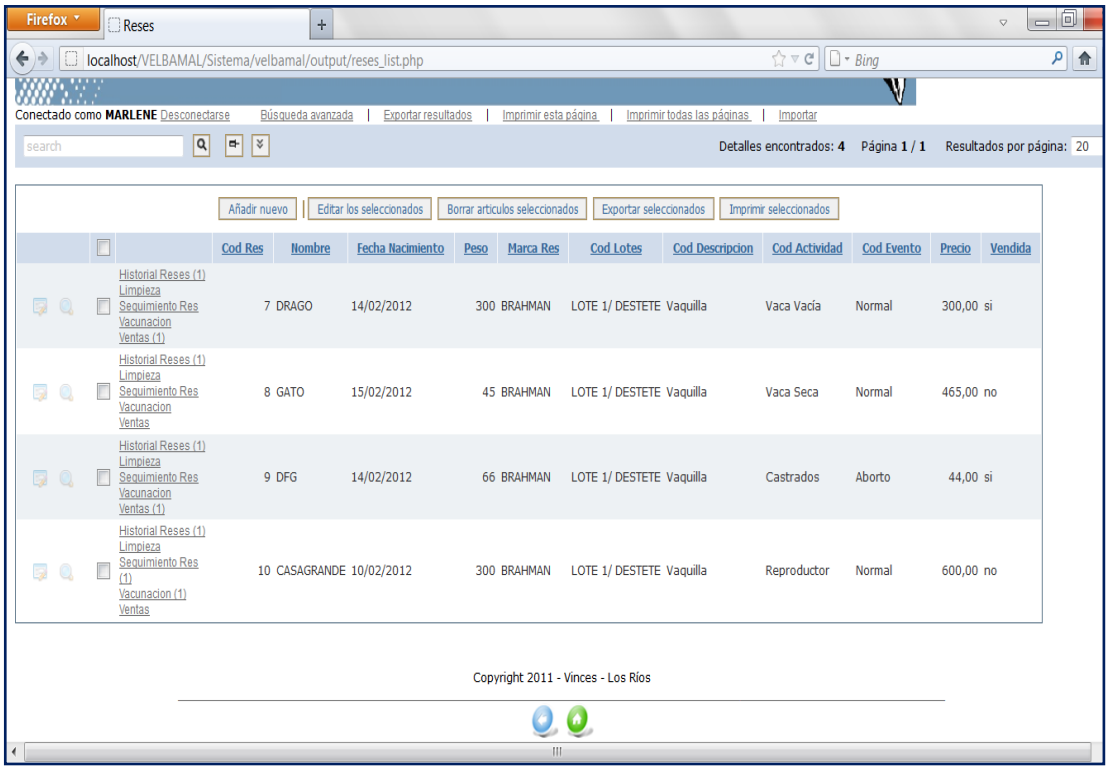


PANTALLA: Reses

Este Menú siguiente es el de Reses donde encontramos Res, Seguimiento de Res, Limpieza, Vacunación

RESES

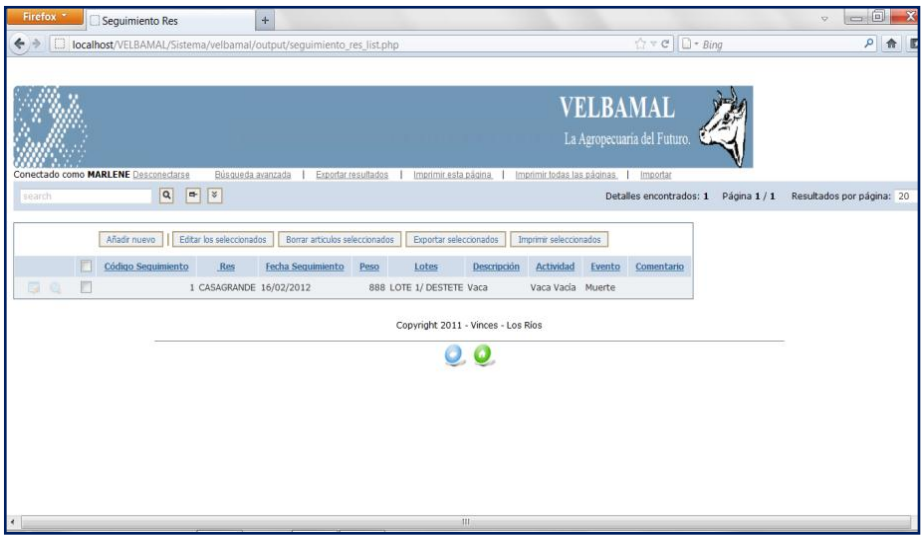
En esta pantalla ingresamos los datos de la Res que tenemos en la Agropecuaria Velbamal como: el Código Res, Nombre Res, Fecha Nacimiento, Peso, Marca Res, Cód. Lotes, Cód. Descripciones, Cód. Actividad, Cód. Evento, Precio, Vendida. En donde podemos ingresar, Actualizar, elimina o hacer Consultas de las Reses de la Agropecuaria.



SEGUIMIENTO DE RES

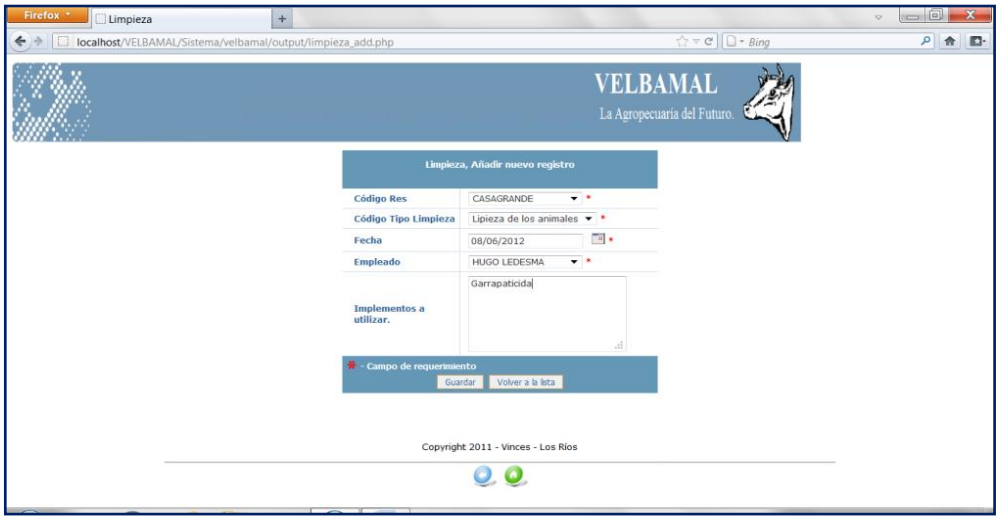
En esta pantalla visualizamos cada Seguimiento del Animal Vacuna de la Agropecuaria y encontramos lo siguiente a ingresar: Codigo de Seguimiento, Res, Fecha de Seguimiento, Peso, Descripción, Actividad, Eventos, Comentario.

En donde podemos Ingresar, Actualizar, Eliminar, Consultar.



LIMPIEZA

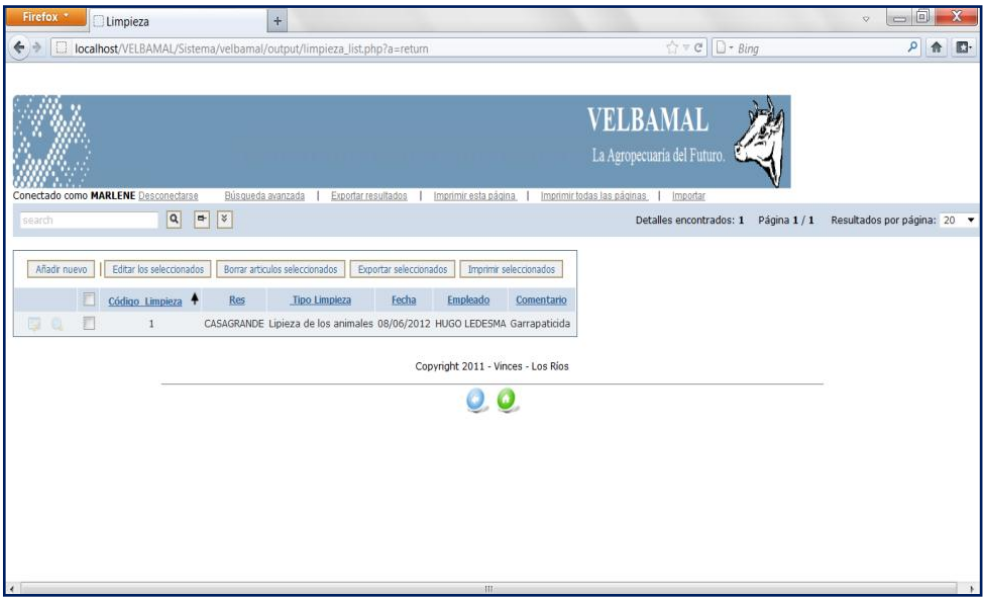
En esta pantalla Hemos ingresado los datos para el tipo de Limpieza del Animal y el empleado que lo realizo.



En esta pantalla Hemos ingresado los datos para el tipo de Limpieza del Animal y el empleado que lo realizo.

En este cuadro ya tenemos ingresado los datos los campos que utilizamos son los siguientes: Código Limpieza, Res, Tipo de Limpieza, Fecha, Empleado y el Comentario.

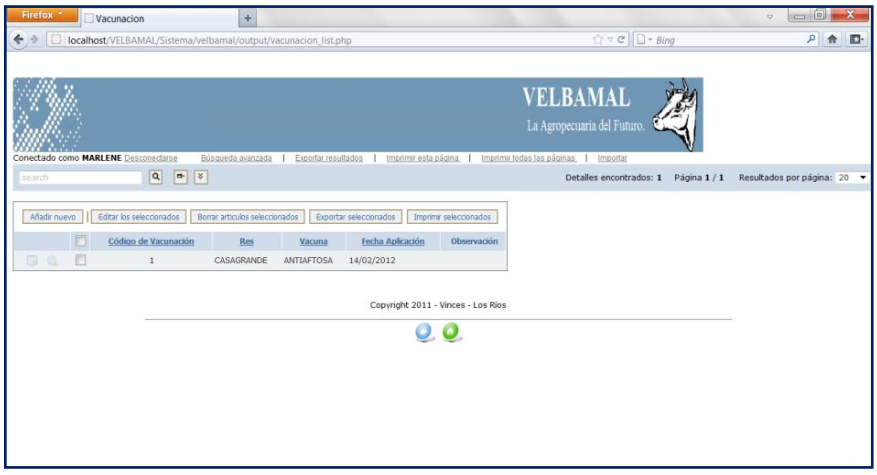
En donde podemos Ingresar, Actualizar, Eliminar, Consultar, los datos.



VACUNACION

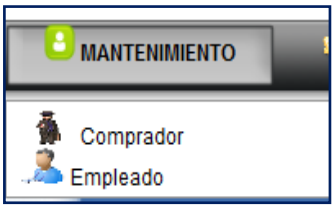
Aquí tenemos que ingresar la Vacunación de cada Res con su Respectivo Nombre de Vacuna en los siguientes campos: Cod. de Vacunación, Res, Vacuna, Fecha de Aplicación, Observaciones.

En donde podemos Ingresar, Actualizar, Eliminar, Consultar , los datos.



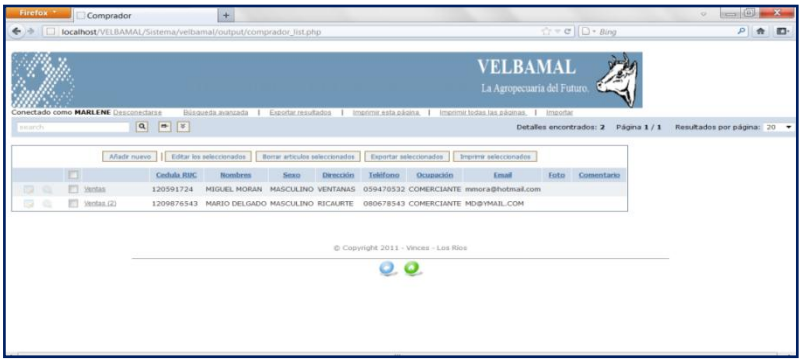
Pantalla: Mantenimiento

En el Menú Mantenimiento tenemos Comprador y Empleado



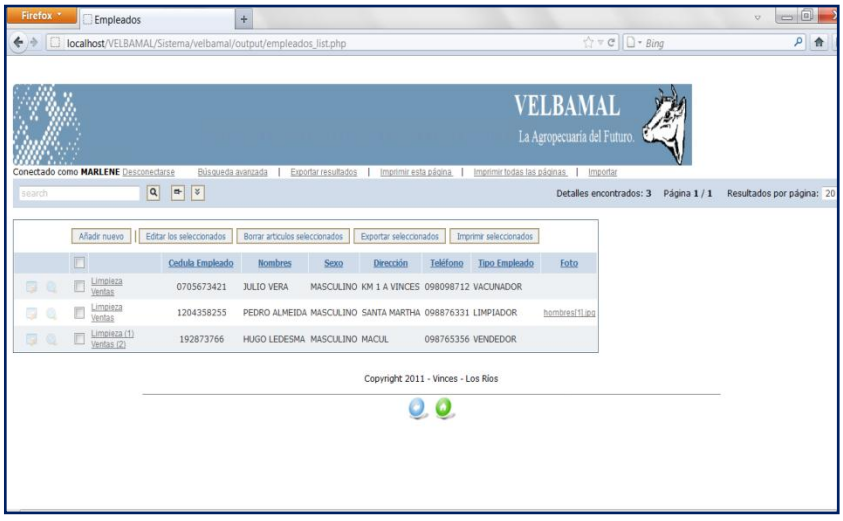
En esta siguiente Pantalla observamos en el Campo Comprador el ingreso de los datos del Comprador como: Cedula Ruc, Nombres, Sexo, Dirección, Teléfono, Ocupación, Email, Foto, Comentario.

En donde podemos Ingresar, Actualizar, Eliminar, Consultar, los datos.

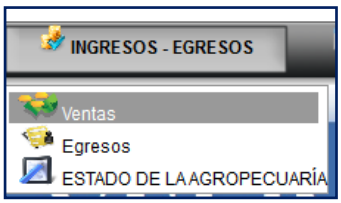


EMPLEADOS

En esta pantalla vamos a ingresar a los empleados con sus respectivos datos: cedula empleado, Nombres, Sexo, Dirección, Teléfono, Tipo de Empleado, Foto, donde podemos Ingresar, Actualizar, Eliminar, Consultar, los datos.

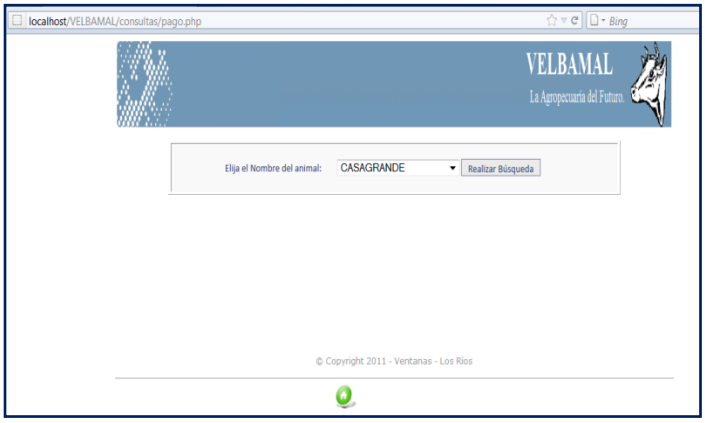


Pantalla: INGRESOS - EGRESOS



VENTAS

En esta pantalla escogeremos el nombre del Animal Vacuno de la Agropecuaria que se va a vender damos clic en el botón de Realizar Búsqueda.

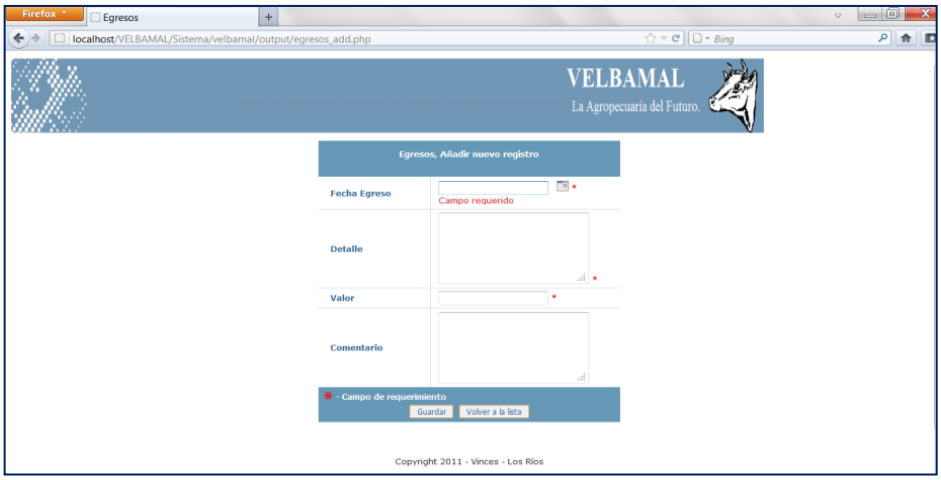


Datos de la Venta:

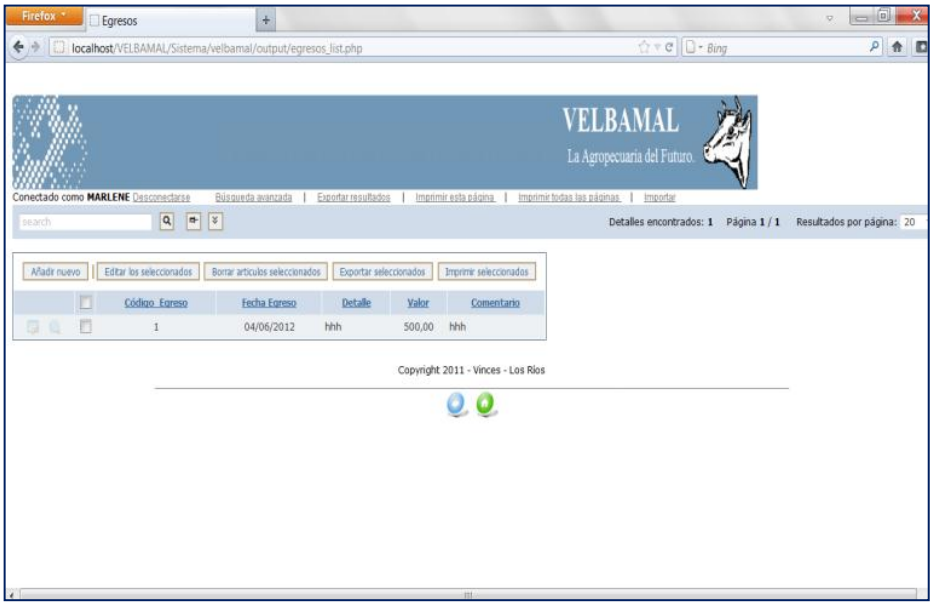
Aquí encontramos los Detalle Especifico del Animal los siguientes campos: Comprador, Empleado, Fecha de transacción, Valor por libra, Peso del animal, Adicional (preñada), Subtotal, Descuento, Iva, Valor a Pagar, Observación. .

EGRESOS

Aquí tenemos la pantalla Egreso en el cual ingresamos los datos correspondientes



Una vez que ingresamos los datos correspondientes llenamos los siguientes campos: Codigo Egreso, Fecha Egreso, Detalle, Valor, Comentario.



ESTADO DE LA AGROPECUARIA

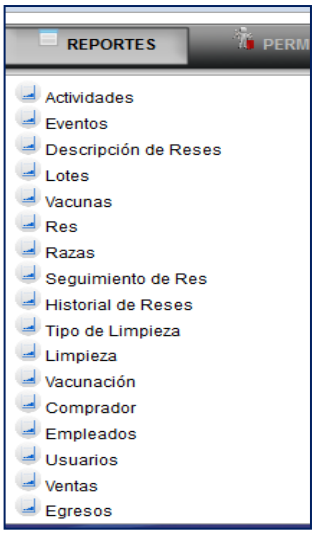
En esta pantalla se Presenta el Reporte de Estado de la Ganadería. En el reporte de los Egresos tenemos: fecha Egreso, Detalle, Valor y su total de Egresos.

En el Reporte de los Ingresos encontramos: Fecha transacción, Res Vendida, Valor y el total de Ingresos.

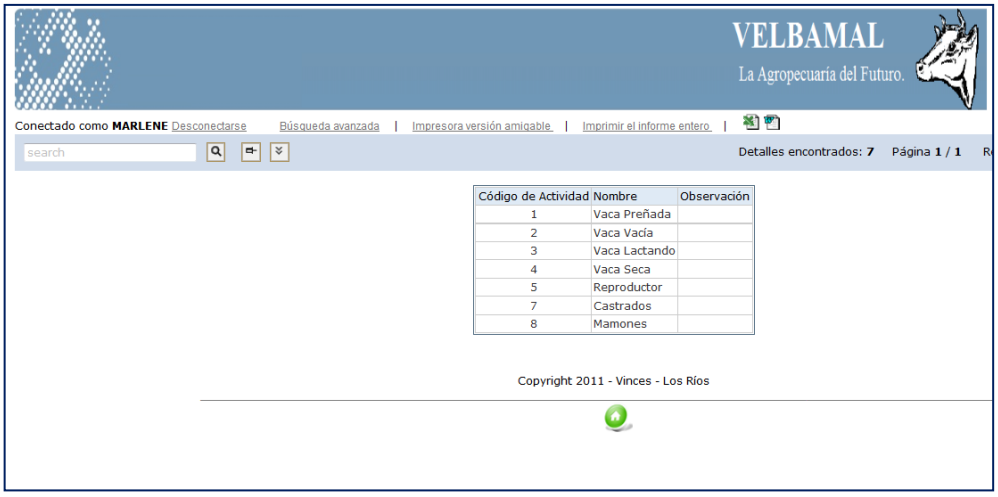
Como punto final aquí observamos el Saldo Final en la Agropecuaria.

Pantalla: Reportes

En esta pantalla se despliega las Actividades, Eventos, Descripción de Reses, Lotes, Vacunas, Res, Razas, Seguimiento de Res, Historial de Reses, Tipo de Limpieza, Limpieza, Vacunación, Comprador, Empleados, Usuarios, Ventas, Egresos.

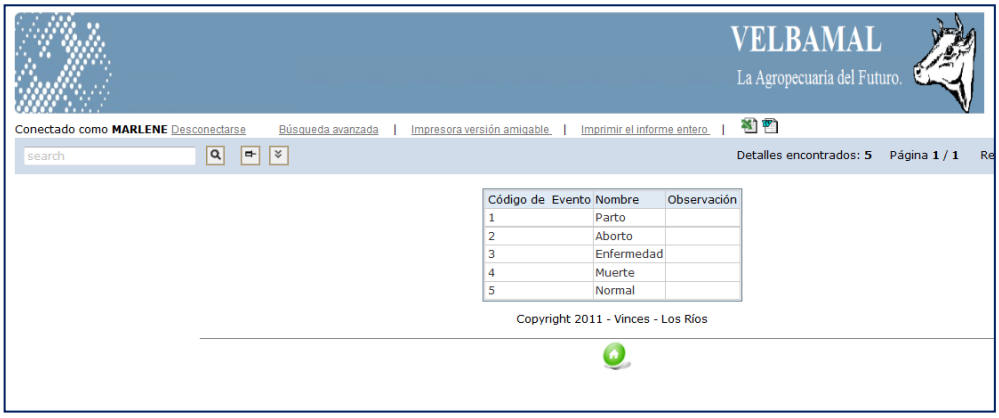


ACTIVIDADES



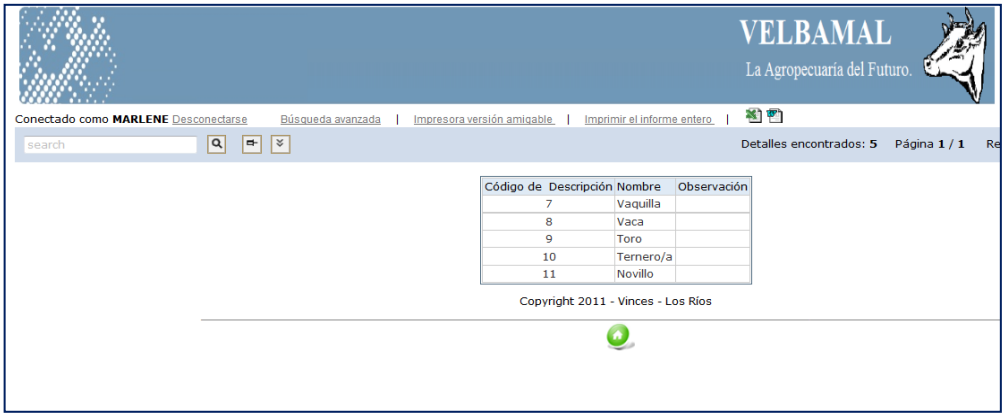
Código de Actividad	Nombre	Observación
1	Vaca Preñada	
2	Vaca Vacía	
3	Vaca Lactando	
4	Vaca Seca	
5	Reproductor	
7	Castrados	
8	Mamones	

EVENTOS



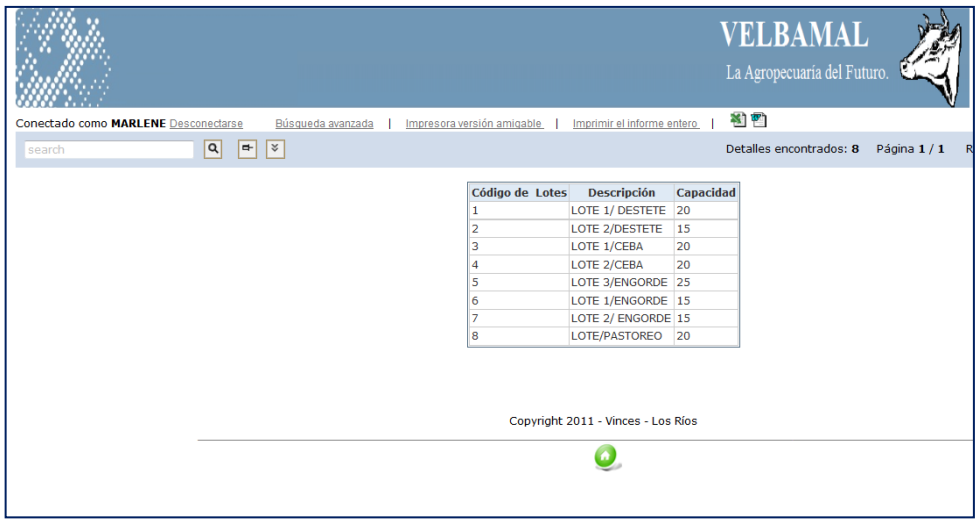
Código de Evento	Nombre	Observación
1	Parto	
2	Aborto	
3	Enfermedad	
4	Muerte	
5	Normal	

DESCRIPCION DE RESES



Código de Descripción	Nombre	Observación
7	Vaquilla	
8	Vaca	
9	Toro	
10	Ternero/a	
11	Novillo	

LOTES

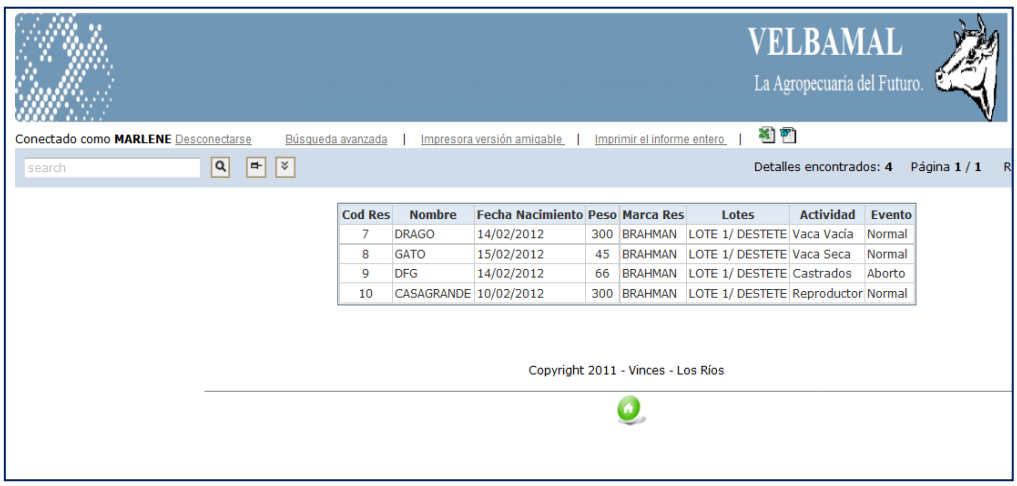


Código de Lotes	Descripción	Capacidad
1	LOTE 1/ DESTETE	20
2	LOTE 2/DESTETE	15
3	LOTE 1/CEBA	20
4	LOTE 2/CEBA	20
5	LOTE 3/ENGORDE	25
6	LOTE 1/ENGORDE	15
7	LOTE 2/ ENGORDE	15
8	LOTE/PASTOREO	20

VACUNAS

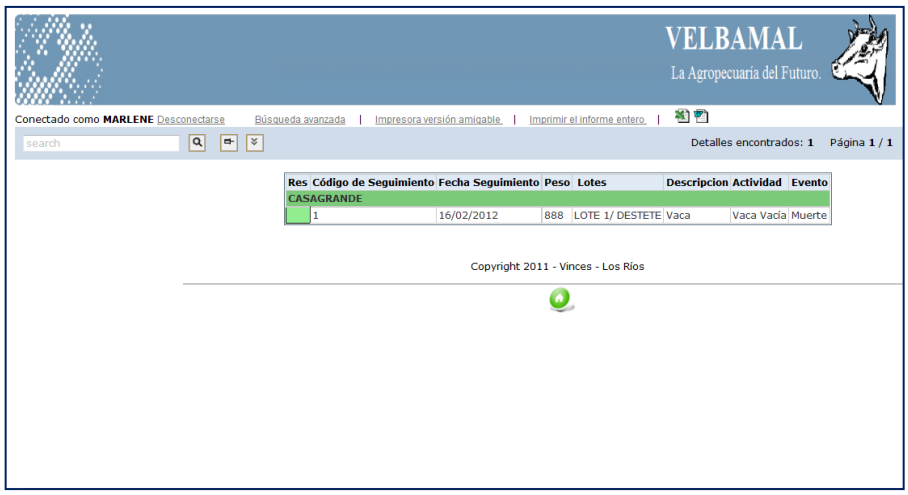
Código de Vacuna	Nombre	Uso	Comentario
1	OXITOCINA 50 mg	Intravenosa	POR PRESENTAR INFECCION
2	ANTIAFTOSA	PARA LA AFTOSA	2 VECES AL AÑO

RESES

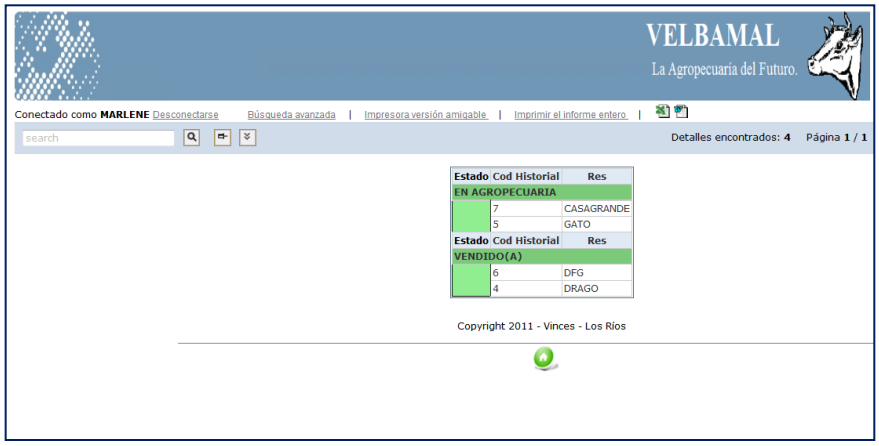


Cod Res	Nombre	Fecha Nacimiento	Peso	Marca Res	Lotes	Actividad	Evento
7	DRAGO	14/02/2012	300	BRAHMAN	LOTE 1/ DESTETE	Vaca Vacía	Normal
8	GATO	15/02/2012	45	BRAHMAN	LOTE 1/ DESTETE	Vaca Seca	Normal
9	DFG	14/02/2012	66	BRAHMAN	LOTE 1/ DESTETE	Castrados	Aborto
10	CASAGRANDE	10/02/2012	300	BRAHMAN	LOTE 1/ DESTETE	Reproductor	Normal

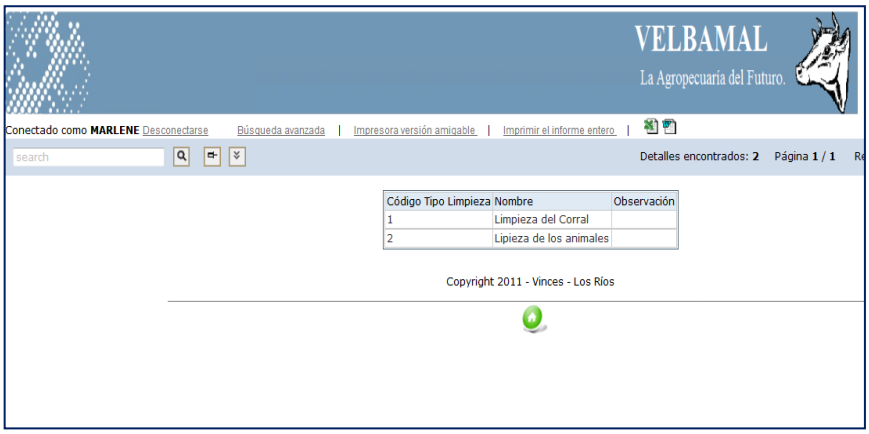
SEGUIMIENTO DE RES



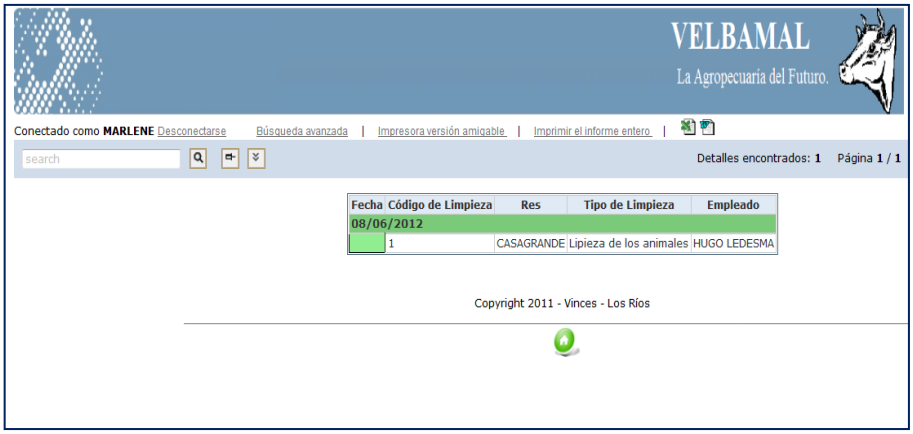
HISTORIAL DE RES



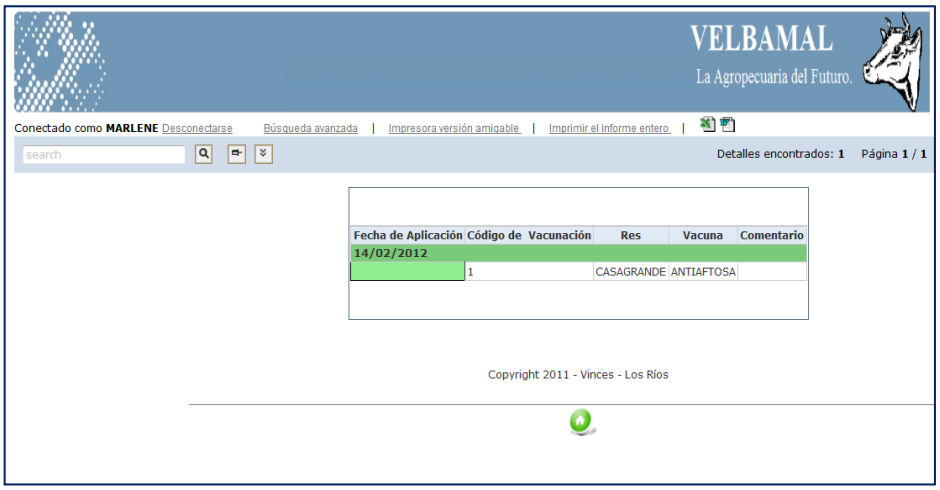
TIPO DE LIMPIEZA



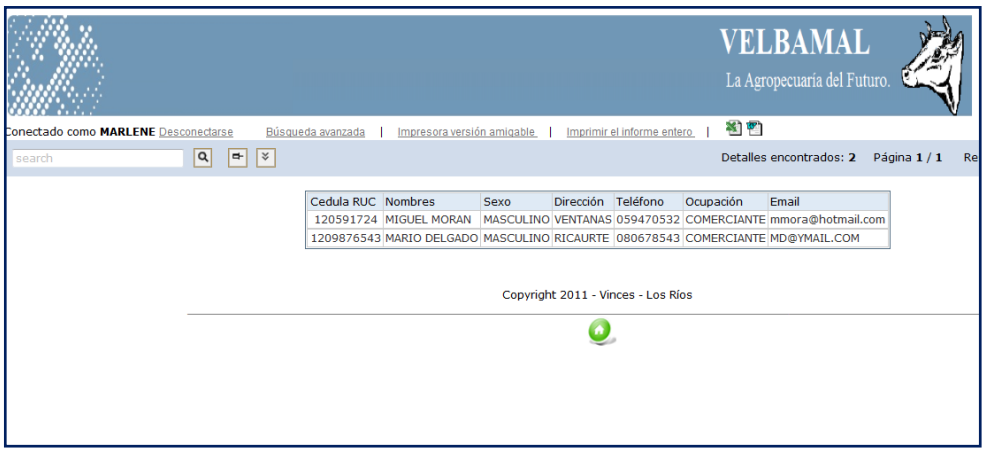
LIMPIEZA



VACUNACION



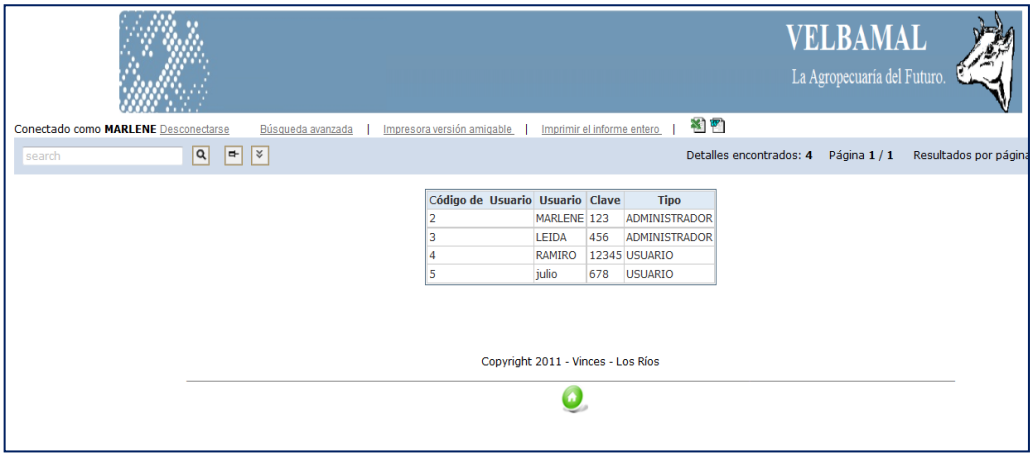
COMPRADOR



EMPLEADOS

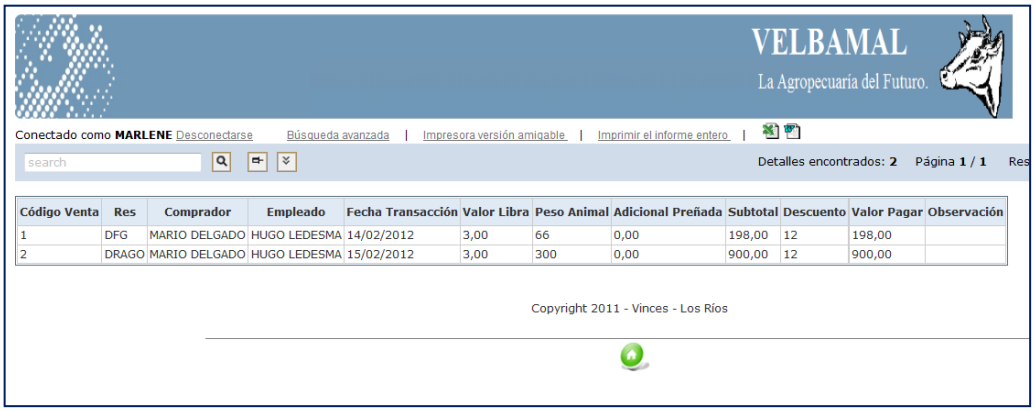
Cédula Empleado	Nombres	Sexo	Dirección	Teléfono	Tipo Empleado
0705673421	JULIO VERA	MASCULINO	KM 1 A VINCES	098098712	VACUNADOR
1204358255	PEDRO ALMEIDA	MASCULINO	SANTA MARTHA	098876331	LIMPIADOR
192873766	HUGO LEDESMA	MASCULINO	MACUL	098765356	VENDEDOR

USUARIO



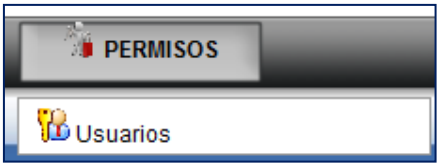
Código de Usuario	Usuario	Clave	Tipo
2	MARLENE	123	ADMINISTRADOR
3	LEIDA	456	ADMINISTRADOR
4	RAMIRO	12345	USUARIO
5	julio	678	USUARIO

VENTAS



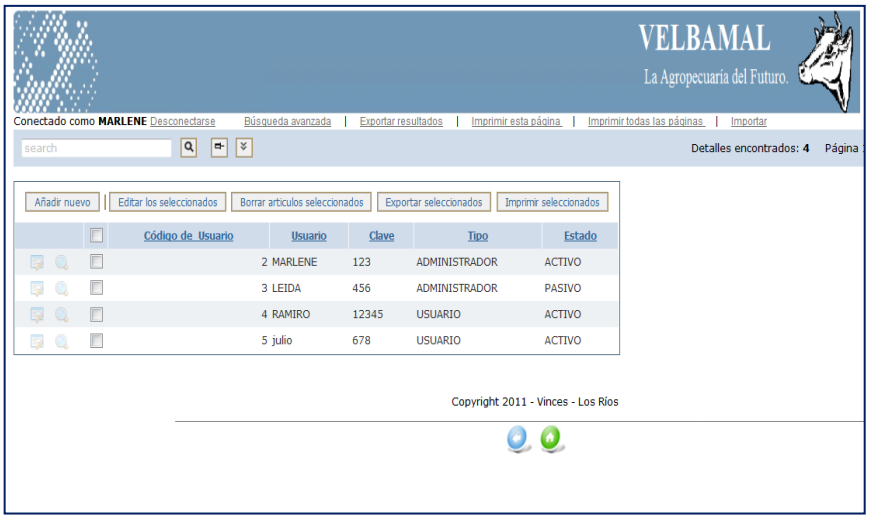
Código Venta	Res	Comprador	Empleado	Fecha Transacción	Valor Libra	Peso Animal	Adicional Preñada	Subtotal	Descuento	Valor Pagar	Observación
1	DFG	MARIO DELGADO	HUGO LEDESMA	14/02/2012	3,00	66	0,00	198,00	12	198,00	
2	DRAGO	MARIO DELGADO	HUGO LEDESMA	15/02/2012	3,00	300	0,00	900,00	12	900,00	

Pantalla: Permisos



USUARIOS

En este cuadro tenemos la pantalla Usuarios en el cual tenemos el nombre de Usuario la clave el tipo dependiendo si es Administrador o Usuario y el Estado en el que se encuentra si es Activo o Pasivo.



ANEXOS