UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ESCUELA DE CIENCIAS DE LA COMPUTACIÓN

TEMA:

Desarrollo e implementación de un aplicativo Web para la gestión de concursos de la Asociación de Caballos de Paso utilizando patrones de diseño Modelo-Vista-Controlador

TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN INFORMÁTICA

AUTOR:

Luis Armando Lima Taramuel

DIRECTOR:

Ing. Daniel Guamán.

Loja – Ecuador
2011

CERTIFICACIÓN

Ing. Daniel Guamán
DIRECTOR DE TESIS

CERTIFICA:

Que el presente trabajo de investigación, previo a la obtención del título de Ingeniero en Informática, realizado por el estudiante Luis Armando Lima Taramuel, ha sido cuidadosamente revisado por el suscrito y he podido constatar que cumple con todos los requisitos de fondo y de forma establecidos por la Universidad Técnica Particular de Loja por lo que autorizo su presentación.

Loja, abril 29 de 2011

…….
Ing. Daniel Guamán

CESION DE DERECHOS DE AUTOR

Yo Luis Armando Lima Taramuel, declaro ser autor del presente trabajo y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales.

Adicionalmente declaro conocer y aceptar la disposición del Art. 67 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: “Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través o con el apoyo financiero, académico o institucional (operativo) de la Universidad”.

……
Luis Armando Lima Taramuel

DECLARACION DE AUTORIA

Las ideas emitidas en el contenido del informe final de la presente investigación son de exclusiva responsabilidad del autor.

…….
Luis Armando Lima Taramuel

AGRADECIMIENTO

A Dios, mi esposa y mis hijos
por todo el amor, apoyo y colaboración
que me han brindado
para conseguir este objetivo.

A la Universidad Técnica Particular de Loja
por la excelente labor educativa que realiza
y por haberme dado la oportunidad de
realizar el presente trabajo de investigación.

DEDICATORIA

El presente trabajo de investigación
está dedicado a mis padres,
porque ellos me han enseñado
con su ejemplo a luchar
para conseguir mis más
anhelados sueños.

ESQUEMA DE CONTENIDOS

Dada la importancia de los patrones de diseño en el desarrollo de sistemas, así como la utilización de frameworks y metodologías agiles, en el presente proyecto de investigación se realiza un estudio breve de estos elementos así como la aplicación práctica de los mismos mediante el desarrollo de una aplicación informática real.

Lo indicado se expone a través del siguiente esquema de contenidos:

- **Estado del arte.-** Marco teórico de: patrones de diseño en general, frameworks de desarrollo, programación extrema, arquitecturas de software, patrón de diseño Modelo-Vista-Controlado y CodeIgniter.

- **Propuesta de la solución.-** En base a la situación actual de la organización y realización de concursos de caballos de paso, se realiza la propuesta de la solución a fin de cubrir las necesidades de automatización planteadas por el usuario. En este capítulo se inicia con la aplicación práctica de lo expuesto en el capitulo anterior.

- **Diseño de la solución.-** En este capítulo se realiza y documenta las actividades para el diseño de la solución, entre las más importantes se tiene: tarjetas CRC, arquitectura de la aplicación, modelo de datos, diseño y modelado de interfaces y el modelo lógico de la aplicación.

- **Implantación de la aplicación.-** Se describe las actividades relacionadas con la implantación de la aplicación entre las cuales tenemos: estándar de codificación utilizado por el framework CodeIgniter, pruebas de aceptación, migración de datos existentes para el nuevo sistema, procedimiento de instalación y tutorías luego de la instalación.

- **Conclusiones y recomendaciones.-** Se detallan las conclusiones y recomendaciones útiles para proyectos similares obtenidas de la experiencia en el desarrollo del presente proyecto.

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

CAPITULO I

1. ESTADO DEL ARTE

En el presente capítulo se describirá en forma breve acerca de los Patrones de Diseño, Arquitecturas de Software, Frameworks y Proceso de Desarrollo, necesarios para el presente proyecto en el cual se hace uso del patrón de diseño Modelo-Vista-Controlador, Programación Extrema y framework CodeIgniter.

1.1 PATRONES DE DISEÑO

En el desarrollo de sistemas como en otros ámbitos de la vida en los que se tenga por objetivo la construcción de algo nuevo, como puede ser: un edificio, un auto o un sistema informático, se debe recordar que los problemas que se presenten en el proceso alguien anteriormente ya los resolvió, es ahí donde surgen los patrones de diseño los cuales ayudan a utilizar las soluciones ya probadas y documentadas por otras personas a problemas que son comunes en una situación en particular como es el caso del diseño orientado a objetos.

Para los desarrolladores de sistemas, lo ideal sería contar con una colección de estos patrones con los problemas que resuelven, pero se debe considerar que los mismos tienen su grado de complejidad, partiendo de que son creados por otras personas y aplicados por quienes tienen la necesidad de dar una solución a un problema, sin embargo, una vez que se comprende su funcionamiento y aplicación los diseños serán mucho más flexibles, modulares y especialmente reutilizables.

Los patrones de diseño no son líneas de código que puedan reutilizarse después, tampoco se tratan de complicados diseños específicos de un dominio para una aplicación completa. Los patrones de diseño son “descripciones de clases y objetos relacionados que están particularizados para resolver un problema de diseño general en un determinado contexto” (Gamma Erich, 2003, p.4).

Los patrones de diseño están clasificados en base a dos criterios: propósito y ámbito (Tabla 1.1).

Propósito				
		De Creación	**Estructurales**	**De comportamiento**
Ámbito	**Clase**	Factory Method	Adapter (de clases)	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Adapter (de objetos) Bridge Composite Decorator Facade Flywieght Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Tabla 1.1: Patrones de diseño
Realizado por: Luis Lima
Fuente: Gamma Erich, 2003, p.9

En base al propósito, es decir, en función de lo que cada patrón realiza, tenemos la siguiente clasificación: creacionales, estructurales y de comportamiento.

- **Patrones Creacionales:** Están relacionados con los procesos de creación de objetos, solucionan problemas de creación de instancias. Ayudan a encapsular y abstraer dicha creación.

- **Patrones Estructurales:** Se relacionan con la composición de clases u objetos, separan la interfaz de la implementación, se ocupan de cómo las clases y objetos se agrupan para formar estructuras más grandes.

- **Patrones de Comportamiento:** Describen el modo en que clases y objetos interactúan y distribuyen sus responsabilidades, así como los algoritmos que encapsulan; es decir, muestran la comunicación entre los objetos.

En base a su ámbito, tenemos patrones que se aplican a clases o a objetos. Los patrones de clases se ocupan de las relaciones entre las clases y sus subclases que se establecen a través de la herencia establecida en tiempo de compilación. Los patrones de objetos a su vez tratan de la relación entre objetos, las cuales se pueden cambiar en tiempo de ejecución.

Hay varias formas de organizar los patrones. En muchos casos ciertos patrones se usan juntos; otros son alternativas y en otros casos dan resultados parecidos, a pesar de que tengan propósitos diferentes. “Tener muchas formas de pensar en los patrones le hará comprender mejor qué es lo que hacen, como compararlos y cuándo aplicarlos.” (Gamma Erich, 2003, p.10).

A continuación una lista resumen con los patrones de diseño orientado a objetos más utilizados según Gamma (2003:71):

1.1.1 Patrones de creación

- **Abstract Factory**.- Proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas.

- **Builder**.- Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.

- **Factory Method**.- Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan qué clase instanciar.

  Este patrón se utiliza cuando se tiene múltiples variantes de una sola entidad. Por ejemplo, se tiene una clase button la cual tiene diferentes variaciones como: ImageButton, InputButton y FlashButton; dependiendo del lugar, se necesita crear diferentes botones, es aquí donde se puede utilizar Factory Method para crear los botones.

- **Prototype**.- Especifica los tipos de objetos a crear por medio de una instancia prototípica, y crear nuevos objetos copiando este prototipo.

- **Singleton**.- Garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a ella.

  Si se requiere pasar una instancia específica de una clase a otra, se puede usar Singleton para evitar tener que pasar la instancia vía el constructor o un argumento, con lo cual se logra acceder desde distintos lugares de la programación e incluso de diferentes clases.

1.1.2 Patrones estructurales

- **Adapter**.- Convierte la interfaz de una clase en otra distinta que es la que esperan los clientes con la finalidad de que pueda ser utilizada por una clase que de otra forma no lo haría.

  Un buen ejemplo, es cuando se crea una clase dominio para las clases de tabla y llamar sus funciones una a una, se puede encapsular todos estos métodos en un método que utilice una clase adaptador. Esto no sólo permite reutilizar cualquier acción que se desee, sino también evitará que se tenga que reescribir el código si se requiere usar la misma acción en un sitio distinto.

- **Bridge**.- Desvincula una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.

- **Composite**.- Combina objetos en estructuras de árbol para representar jerarquías de parte-todo, esto permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.

- **Decorator**.- Añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.

  El momento más adecuado para usar Decorator es cuando se tiene una entidad que necesita tener un nuevo comportamiento sólo si la situación así lo requiere, por ejemplo, se tiene un enlace HTML que realiza cosas diferentes dependiendo de la página en la que se encuentre, como puede ser mostrar el link subrayado si se encuentra en una página y resaltado si está en otra.

- **Facade**.- Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema

- **Flyweight**.- Usa el compartimiento para permitir un gran número de objetos de grano fino de forma eficiente.

- **Proxy**. Proporciona un sustituto o representante de otro objeto para controlar el acceso a éste.

1.1.3 Patrones de comportamiento

- **Chain of Responsibility**.- Evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición, para esto crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.

- **Command**.- Encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer la operaciones.

- **Interpreter**.- Dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.

- **Iterator**.- Proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.

- **Mediator**.- Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.

- **Memento**.- Representa y externaliza el estado interno de un objeto sin violar la encapsulación, de forma que éste puede volver a dicho estado más tarde.

- **Observer**.- Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.

- **State**.- Permite que un objeto modifique su comportamiento cada vez que cambia su estado interno. Parecerá que cambia la clase del objeto.

- **Strategy**.- Define una familia de algoritmos, encapsula uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.

- **Template Method**.- Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite

que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.

- **Visitor**.- Representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.

El uso de patrones de diseño tiene sus ventajas y desventajas las cuales se resumen en la Tabla 1.2.

Ventajas	Desventajas
Son soluciones concretas: Los patrones son un conjunto de recetas de diseño, se pueden clasificar, pero cada patrón es independiente de los demás.	El uso de patrones no se refleja directamente en el código: Es fácil determinar que patrón de diseño se ha utilizado pero no es posible hacer ingeniería inversa.
Son soluciones técnicas: Dada una determinada situación, los patrones indican cómo resolverla mediante un diseñó orientado a objetos. Existen patrones orientados a un leguaje específico y otros de carácter general.	Es difícil reutilizar la implementación de un patrón: Las clases del patrón son roles genéricos, pero en la implementación son clases concretas.
Se aplican en soluciones muy comunes: Proceden de la experiencia y tiene utilidad demostrada en problemas del diseño orientado a objetos.	El uso de patrones supone cierta carga en la implementación: Se usan más clases de las estrictamente necesarias. A menudo un mensaje se resuelve mediante delegación de varios mensajes a otros objetos.
Son soluciones simples: Resuelven un problema particular utilizando un pequeño número de clases relacionadas de forma determinada, no indican como diseñar un sistema completo, se centra en aspectos puntuales del mismo.	
Facilitan la reutilización de las clases y del propio diseño: Favorecen la reutilización de clases ya existentes y la programación de clases reutilizables. La propia estructura del patrón es reutilizada cada vez que se aplica.	
Brindan una estructura conocida por todos los programadores, de manera que la forma de trabajar no resulte distinta entre los mismos.	
Permiten tener una estructura de código común a todos los proyectos que implemente una funcionalidad genérica.	
El software construido es más fácil de comprender, mantener y extender.	

Tabla 1.2: Patrones de diseño - Ventajas y Desventajas

Realizado por: Luis Lima

De lo expuesto en la Tabla 1.2 se puede determinar que la ventaja más importante del uso de patrones de diseño es que evitan que se reinvente la rueda en la construcción de las aplicaciones debido a que existe mucho trabajo ya realizado, probado y aceptado que en un entorno similar al problema objeto de estudio ya aporta una solución satisfactoria que ahorrará

entre otras cosas mucho tiempo evitando la búsqueda de soluciones a problemas ya resueltos anteriormente.

En diseño de aplicaciones es fundamental anticiparse a nuevos requisitos y a cambios en los ya existentes de manera que el diseño soporte estas variaciones y evolucionen sin mayores complicaciones, sin correr el riesgo de rediseñar por completo. Los cambios no previstos pueden implicar el rediseño de la aplicación lo que involucra: redefiniciones y nuevas implementaciones de clases, modificar los clientes y repetir el ciclo de pruebas, lo cual siempre resulta ser muy costoso. Los patrones de diseño ayudan a evitar este tipo de problemas al dejar que algún aspecto de la estructura del sistema varíe de manera independiente, lo que da como resultado sistemas más robustos frente a cambios concretos.

Como parte central de la investigación realizada nos centraremos en el patrón de diseño Modelo-Vista-Controlador el cual se expone a continuación.

1.2 PATRON DE DISEÑO "MODELO-VISTA-CONTROLADOR"

Para el diseño de aplicaciones Web con interfaces enriquecidas y que ofrezcan una fuerte interactividad con el usuario se utiliza el patrón de diseño Modelo-Vista-Controlador. Se parte del hecho que la lógica de un interfaz de usuario cambia con más frecuencia que los almacenes de datos y la lógica de negocio. Si se realiza un diseño que mezcle los componentes de interfaz y de negocio, cuando se requiera cambiar la interfaz de usuario, tendremos que modificar necesariamente los componentes de negocio, esto trae consigo mayor trabajo y una alta probabilidad de error.

Con este patrón se trata de realizar un diseño que separe la vista del modelo, con la finalidad de mejorar la reusabilidad, de esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

En el patrón de diseño de software Modelo-Vista-Controlador (MVC) todo el proceso está dividido en 3 capas:

- **Modelo**: En esta capa se hace el levantamiento de todos los objetos que debe utilizar el sistema, es decir, es el proveedor de recursos. Es en donde se encapsulan los datos y la lógica del negocio, esto hace que sea independiente de la base de datos que se utilice así como de los medios de representación de los datos.

- **Vista**: Es la capa de presentación la cual muestra la información del modelo al usuario, se agrupan todas las clases y archivos que tengan relación de la interfaz de usuario. Cada vista tiene asociado un componente controlador que se indica a continuación.

- **Controlador**: El controlador viene a ser el orquestador de los diferentes eventos generados en la interfaz de usuario, se encarga de llamar en el modelo al experto del negocio que sabe que es lo que hay que hacer con la petición del usuario. Una vez que el modelo ha realizado su tarea se lo comunica al controlador. El controlador invoca a la vista o interfaz para que se actualice con los cambios hechos en el modelo. En resumen, el Controlador es el que escucha los cambios en la vista y se los envía al modelo, el cual le regresa los datos a la vista, es un ciclo donde cada acción del usuario causa que se inicie un nuevo ciclo.

En la Figura 1.1 se muestran las responsabilidades de cada capa del patrón de diseño MVC y cómo interactúan entre sí.

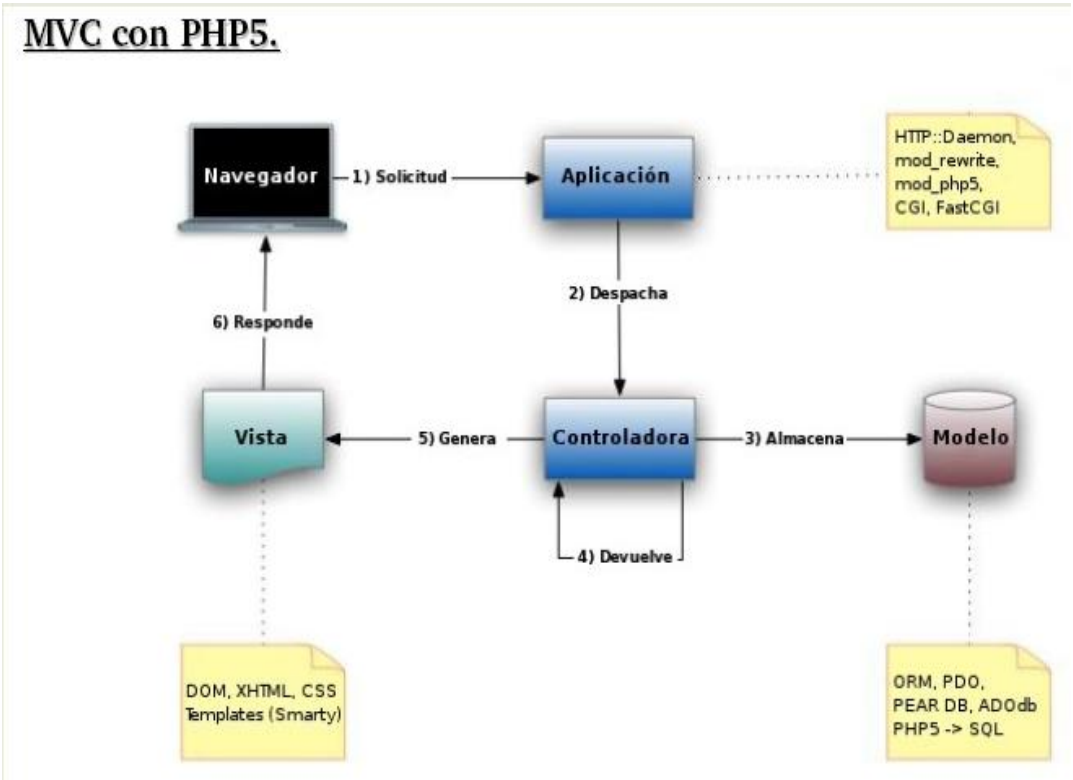


Figura 1.1: MVC con PHP

Fuente:http://s3.amazonaws.com/ppt-download/patrones-de-diseo-y-orienacin-a-objetos-en-php52422.pdf

La interacción que se muestra en la Figura 1.1 es la siguiente:

1. Solicitud: El usuario ejecuta un navegador, pide un URL solicitando a la Aplicación.
2. Despacha: La Aplicación delega a la capa **Controladora** la petición del usuario.
3. Almacena: Consulta en la capa **Modelo** por medio de la capa de datos del manejador de base de datos existente.
4. Devuelve: La capa de Datos devuelve los resultados en datos puros a la capa **Controladora**.
5. Genera: La capa **Controladora** genera la capa de **Vista** en base a los tipos de datos generados.
6. Responde: La **Vista** generada se envía como respuesta a la solicitud del navegador.

Con lo expuesto se resume las responsabilidades básicas de cada componente del patrón MVC de la siguiente manera:

- El **modelo** es responsable de:
    - Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
    - Define las reglas de negocio.
- El **controlador** es responsable de:
    - Recibe los eventos de entrada.
    - En base a reglas de gestión de eventos se realizan peticiones al modelo o a las vistas.
- Las **vistas** son responsables de:
    - Recibir datos del modelo y los muestran al usuario.
    - Tienen un registro de su controlador asociado (normalmente porque además lo instancia).

MVC implementa varios patrones a la vez, así tenemos que la relación entre la Vista y el Controlador es una implementación del patrón Strategy. MVC utiliza otros patrones de diseño, tales como el Factory Method y el Decorator. Factory Method se utiliza para especificar la clase controlador predeterminada de una vista y Decorator para añadir la capacidad de

desplazamiento de una vista, sin embargo, las principales relaciones se dan entre los patrones Observer, Composite y Strategy.

“Si una aplicación no está diseñada siguiendo el patrón de diseño MVC, inevitablemente se tendrá problemas de escalabilidad, mantenimiento y de extensión” (Caballero Leonardo,2006). Se puede afirmar que en aplicaciones Web la programación utilizando el patrón de diseño MVC no es una alternativa sino una necesidad que evitará en el futuro muchos dolores de cabeza gracias a sus características que en resumen son las siguientes:

- Patrón de diseño orientado a objetos.
- Adecuado para aplicaciones Web con alta interacción humana.
- Separa clara y consistentemente las preocupaciones en las capas indicadas anteriormente.
- Permite múltiples representaciones (vistas) de la misma información (modelo).
- Facilidad para agregar, eliminar o modificar interfaces de usuario.
- Facilita los desarrollos simultáneos con actualizaciones de interfaces, lógica del negocio o incluso introducir una nueva aplicación sin la necesidad de afectar a otro código fuente.
- Garantiza la reducción del código fundamentalmente porque los modelos pueden ser utilizados en varias vistas.
- Los desarrolladores centran su atención en un solo aspecto de la aplicación al mismo tiempo.

Estas características conducen a que un proyecto de software Web se implemente de una forma ágil con un producto sostenible y mantenible en el tiempo y que en el futuro puede migrar en cualquier dirección. Estas son las razones fundamentales por las que se utilizará el patrón MVC en el presente proyecto.

1.3 ARQUITECTURA DE SOFTWARE

Existen muchas definiciones de Arquitectura de Software, sin embargo, una buena definición sería que la Arquitectura de Software es la organización al más alto nivel de un sistema o aplicación informática que representa su estructura incluyendo los componentes de software con sus propiedades de visibilidad externa y relación entre ellos para alcanzar la misión del sistema.

La Arquitectura de Software es muy importante en el desarrollo de aplicaciones por tres razones fundamentales:

- **Decisiones tempranas de diseño**.- Define las limitaciones en la implementación, dicta la estructura organizacional, oculta o muestra los atributos del sistema y hace más fácil controlar los cambios.

- **Comunicación entre las personas involucradas.-** La abstracción que representa la arquitectura es la base para el entendimiento del problema con lo cual se puede hacer una buena negociación tanto en tiempo como costos.

- **Abstracción trasferible de un sistema**.- La arquitectura de un sistema se puede utilizar en otro que tenga las mismas características.

La Arquitectura de Software en la actualidad es un requisito clave en el desarrollo de sistemas debido la complejidad de éstos y la necesidad implícita de seguridad, escalabilidad y distribución. Es así que la arquitectura ha tenido una evolución bastante rápida en los últimos años, lo cual se resume en la tabla Evolución de Arquitecturas (Tabla 1.3) en la que se detallan los tipos más comunes de arquitecturas existentes y sus características.

Arquitectura	Características
Aplicaciones Monolíticas	Interfaces gráficas de usuario (GUI). Servicios de presentación, negocios y persistencia en la misma máquina. No hay concurrencia de usuarios. Alto acoplamiento entre capas.
Cliente - Servidor	Clientes pesados, no estándar. Conexiones dedicadas a BD Protocolos pesados. Ejecución remota de SQLs . Alta administración. Bajo rendimiento. Alto tráfico de red. Baja accesibilidad.
Cliente - Servidor Mejorada	Lógica de negocios en BD. Clientes pesados, no estándar. Conexiones dedicadas a la BD. Mejora en rendimiento. Alta administración. Baja escalabilidad y flexibilidad. Baja portabilidad.
Arquitectura de tres niveles	Reutilización de lógica de negocio para diferentes clientes o sistemas. Mejora la escalabilidad. Mejora la flexibilidad. Independencia de la base de datos.
Arquitectura de N-niveles	Bajo costo de administración de clientes. Alta accesibilidad. Alta flexibilidad. Alta disponibilidad y tolerancia a fallos. Alta escalabilidad. Independencia de DB.
Orientada a servicios (SOA)	Heterogeneidad. Escalabilidad. Disponibilidad. Distribución. Manejabilidad de Procesos. Administración y monitoreo de procesos, servicios e infraestructura.

Tabla 1.3: Evolución de Arquitecturas de Software

Realizado por: Luis Lima

En este punto es necesario diferenciar lo que es Arquitectura y lo que es Diseño, ya que desde un punto de vista ligero podríamos llegar a la conclusión de que son lo mismo, sin embargo, estos difieren radicalmente en tres áreas: Nivel de abstracción, entregables y áreas de enfoque. Estas diferencias se detallan en la Tabla 1.4.

Área	Arquitectura	Diseño
Nivel de abstracción	Se lo realiza a un nivel muy alto	El nivel de abstracción es a bajo nivel con un enfoque en los detalles.
Entregables	Los entregables constituyen la planeación de subsistemas, interfaces con sistemas externos, frameworks, componentes reutilizables, prototipo arquitectónico.	Los entregables constituyen el diseño detallado de los componentes y las especificaciones de codificación.
Áreas de enfoque	Se enfoca fundamentalmente en la selección de tecnologías, requerimientos no funcionales como la calidad del servicio (QoS), manejo de riesgos.	Se centra en los requerimientos funcionales.

Tabla 1.4: Diferencias entre Arquitectura y Diseño
Realizado por: Luis Lima

Con lo indicado se puede determinar fácilmente que las decisiones tomadas en Arquitectura causan un alto impacto en los proyectos de desarrollo de sistemas debido a que la arquitectura conlleva un conjunto de decisiones estratégicas de diseño, lineamientos, reglas y patrones que limitan el diseño e implementación de un sistema.

Como parte de esta investigación nos centraremos en la **arquitectura en tres capas** utilizada en el desarrollo de la solución web incluida en el presente proyecto de investigación.

1.3.1 ARQUITECTURA EN TRES CAPAS

Es una de las arquitecturas más utilizadas en la actualidad, cuyo objetivo primordial es el de separar la aplicación en tres capas: capa de presentación, capa de negocio y capa de datos.

- **Capa de presentación.-** En esta capa se ubican todas las interfaces de usuario (formularios Windows, html) con sus controles visuales y

de eventos que permiten presentar y capturar información del usuario. Esta capa únicamente tiene relación con la capa de negocio.

- **Capa de negocio**.- En esta capa se ubica el código que define las reglas del negocio (aplicación). Es resultado del análisis de todos los procesos del negocio incluidos en la automatización. Esta capa se comunica con la capa de presentación y la de datos.

- **Capa de datos**.- En esta capa residen los datos y el código para acceder a los mismos con la finalidad de realizar las cuatro operaciones básicas sobre ellos: insertar, actualizar, consultar y borrar. Estas operaciones son solicitadas por la capa de negocio.

Estas capas se pueden ubicar en un mismo servidor o en diferentes dependiendo de la complejidad y características de la aplicación a construir, esto debido a que esta división es lógica. La división física, es decir, el número de servidores sobre los que se distribuyen las tres capas lógicas se conocen como niveles.

La ejecución desde el punto de vista de los niveles se distribuye de la siguiente manera:

- **Front-End**.- Es donde se ejecutan las interfaz de usuario del cliente, en definitiva son los Browser.

- **MiddleWare**.- Recibe solicitudes a través de la red. Estos son mensajes (XML, SOAP, JSON) que se envían mediante protocolos de transporte (HTTP, TCP, UDP), en términos generales es El Servidor Web.

- **BackEnd**.- Base de datos o algún proceso externo a nuestro software como puede ser la relación con otro sistema.

En la Figura 1.2 se muestra ejemplo de esta arquitectura con una distribución en tres niveles.

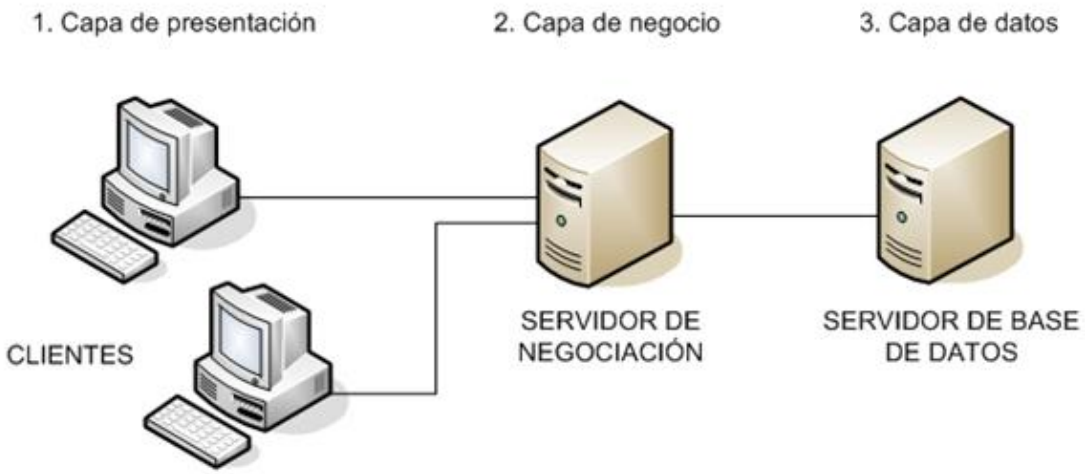


Figura 1.2: Arquitectura en tres capas
Fuente:http://upload.wikimedia.org/wikipedia/commons/e/ea/Tres_capas.PNG

El patrón de diseño MVC revisado anteriormente tiene una arquitectura asociada a la de tres capas, aunque su objetivo es mucho más fino ya que se centra en la secuencia de ejecución, desde que se produce un evento en la capa de presentación hasta que el mismo es atendido en forma completa.

El hecho de que esta arquitectura separe el problema en tres capas permite también dividir el trabajo con responsables diferentes, ya sea a nivel de colaboradores como equipos de hardware, lo que es una ventaja muy importante para afrontar fundamentalmente los cambios o el incremento de necesidades. Esta es la razón por la que se ha seleccionado esta arquitectura para el desarrollo de la solución web que se revisará más adelante.

1.4 FRAMEWORKS DE DESARROLLO

Los Frameworks cuya traducción sería “marcos de trabajo”, son una estructura básica o genérica de soporte sobre la cual otro proyecto puede ser organizado y desarrollado en su totalidad. En otras palabras un framework de desarrollo se puede considerar como una aplicación genérica incompleta y configurable a la que se le pueden añadir las piezas faltantes para construir una aplicación completa.

En general los frameworks son un conjunto de buenas prácticas de desarrollo representados en una serie de librerías (toolkits) bajo un único esquema de colaboración que resuelven un problema en particular el cual se puede tomar como base para resolver nuevos problemas con las mismas características y así crear aplicaciones de manera rápida.

El momento en que se decide utilizar un framework para construir una aplicación, estamos adoptando la arquitectura del framework, es decir, heredará la estructura general, su partición en clases y objetos, las responsabilidades clave, cómo colaboran las clases y objetos y el hilo de control, esto permite al diseñador o programador de la aplicación concentrarse en las funcionalidades propias del sistema que se está desarrollando.

Los frameworks hacen hincapié en la reutilización del diseño frente a la reutilización del código, sin embargo, un framework incluye normalmente subclases concretas listas para trabajar, lo cual permite construir aplicaciones rápidamente y que éstas tengan estructuras parecidas por lo que son más fáciles de mantener y resultan más consistentes para los usuarios.

Encontramos frameworks que parten del paradigma MVC que como se indicó anteriormente "separa en la aplicación la gestión de los datos, las operaciones, y la presentación" y frameworks que pueden llegar a tal detalle de definir los nombres de los archivos su estructura y los estándares de programación.

Los frameworks maduros incorporan varios patrones de diseño. Los patrones ayudan a hacer que la arquitectura del framework sirva para muchas aplicaciones diferentes sin necesidad de rediseño. Esto implica que entre mayor conocimiento se tenga de los diferentes tipos de patrones de diseño se comprenderá mejor un framework. La mayoría de los frameworks para web utilizan el patrón de diseño MVC.

Un framework no está necesariamente ligado a un lenguaje de programación, sin embargo, entre más experto sea el framework que se utilice se relacionará más a un lenguaje en particular, por ejemplo, uno de los frameworks más populares es **Rails**[1] que trabaja con el lenguaje de programación Ruby[2]. **Ruby on Rails** conocido como RoR es un framework de aplicaciones web que se basa en el patrón Modelo-Vista-Controlador.

Los principales objetivos que persigue un framework son los siguientes:

- Acelerar el proceso de desarrollo.
- Reutilizar código ya existente.

[1] Rails: Framework para el desarrollo de aplicaciones Web de uso libre basado en el patrón de diseño MVC.

[2] Ruby: Es un lenguaje de programación basado en scripts, multiplataforma, orientado a objetos y de uso libre.

- Promover buenas prácticas de desarrollo como es el uso de patrones.
- Desarrollo estructurado.
- Disminuir el esfuerzo en el desarrollo.
- Aprovechar funcionalidades ya implementadas.
- Centrarse directamente en la solución del problema.
- Trabajar directamente con metodologías de desarrollo Ágiles.

Los frameworks existentes en su mayoría tienen las siguientes características:

- **Abstracción de URLs y sesiones**.- No es necesario manipular directamente las URLs ni las sesiones, el framework ya se encarga de hacerlo.
- **Acceso a datos**.- Incluyen las herramientas e interfaces necesarias para integrarse con herramientas de acceso a datos.
- **Controladores**.- La mayoría de frameworks implementa una serie de controladores para gestionar eventos, como una introducción de datos mediante un formulario o el acceso a una página. Estos controladores se adaptan fácilmente a las necesidades de un proyecto en particular.
- **Autenticación y control de acceso**.- Incluyen mecanismos para la identificación de usuarios mediante login y password y permiten restringir el acceso a determinas páginas a determinados usuarios.
- Internacionalización.
- Separación entre diseño y contenido.

En la actualidad encontramos una infinidad de frameworks, en la Tabla 1.5 se detallan las más comunes que se utilizan con lenguajes de programación PHP y Java.

PHP	JAVA
Yii	Struts
CodeIgniter	Hibernate
CakePHP	JavaServerFaces
Zend	Spring
Symfony	Apache Wicket
PHPDevShell	Google Web Toolkit

Tabla 1.5: Frameworks de desarrollo

Realizado por: Luis Lima

Como caso de estudio en este proyecto de investigación se trabajará con el framework PHP **CodeIgniter**, con el cual se aplicará el patrón de diseño MVC para la construcción de una aplicación Web.

Existen muchos frameworks como los indicados en la Tabla 1.5 que tienen características similares a CodeIgniter que se podían haber utilizado, sin embargo, se ha seleccionado CodeIgniter por cuatro motivos fundamentales:

- Trabaja con PHP como lenguaje de programación.
- Utiliza el patrón de diseño MVC.
- La aplicación será completamente Web.
- Se encuentra bajo una licencia Open Source.

1.4.1 FRAMEWORK CODEIGNITER

Codeigniter es un framework orientado a personas que construyen sus aplicaciones web utilizando PHP. Permite desarrollar proyectos mucho más rápido de lo que se podría hacer si se lo escribiese desde cero, para esto provee muchas librerías para tareas repetitivas y necesarias así como una interface simple y estructura lógica para acceder a esas librerías. CodeIgniter permite enfocarse principalmente en el proyecto minimizando la cantidad de código necesaria para una tarea dada.

Codeigniter se encuentra bajo una licencia open source Apache/BSD-style, es compatible con PHP4 y PHP5 y está basado en el patrón de diseño **Modelo-Vista-Controlador** que permite separar el diseño con el código, agilizar las funciones a la base de datos y crear librerías independientes para ser reutilizadas en cualquier instancia de la aplicación. Al utilizarse librerías en una aplicación se logra agilizar y optimizar la programación a nivel Web.

Codeigniter viene con un conjunto de librerías que permiten realizar las tareas de desarrollo Web típicamente necesarias, como: acceder a una base de datos, enviar un email, validar datos de un formulario, mantener sesiones, manipular imágenes, trabajar con datos XML-RPC, etc.

El sistema puede ser fácilmente extendido a través de la inclusión de plugins y librerías, o a través de extensión de clases o ganchos del sistema.

Las “características principales de Codeigniter” (Martinez Pablo,2009, p.22) son las siguientes:

- Sistema Basado en el patrón de diseño Modelo-Vista-Controlador.
- Compatible con PHP 4.

- Extremadamente Liviano.
- Clases de base de datos llenas de características con soporte para varias plataformas.
- Soporte de Active Record para Base de Datos.
- Formulario y Validación de Datos.
- Seguridad y Filtro XSS.
- Manejo de Sesión.
- Clase de Envío de Email. Soporta Archivos Adjuntos, email de texto/HTML, múltiples protocolos (sendmail, SMTP, and Mail) y más.
- Librería de Manipulación de Imagen (cortar, redimensionar, rotar, etc.). Soporta GD, ImageMagick, y NetPBM .
- Clase de Carga (upload) de Archivo.
- Clase de FTP.
- Localización.
- Encriptación de Datos.
- Puntos de referencia.
- Historial de Errores.
- Perfilando la Aplicación.
- Scaffolding.
- Clase de Calendario.
- Clase de Agente del Usuario.
- Clase de Codificación Zip.
- Clase de Motor de Plantillas.
- Clase de Trackback.
- Librería XML-RPC.
- Clase de Prueba de Unidad.
- URLs amigables a motores de búsqueda.
- Ruteo de URI Flexible.
- Soporte para Ganchos, Extensiones de Clase y Plugins.
- Larga librería de funciones "asistentes".

En el desarrollo del presente proyecto se explota la mayoría de estas características principalmente las siguientes:

- Uso del patrón de diseño Modelo-Vista-Controlador.
- Active Record para Base de Datos.
- Formulario y Validación de Datos.
- Manejo de Sesión.
- Encriptación de Datos.
- Clase de Codificación Zip.
- URLs amigables a motores de búsqueda.
- Seguridad y Filtro XSS.

En la Tabla 1.6 se hace una comparación de las características más importantes de CodeIgniter con respecto a otros frameworks PHP.

Framework	PHP 4	PHP 5	MVC	Modules	ORM	EDP	Auth	Cache	Validator	Ajax	License	DL	RV
Zend Framework	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	New BSD		
CakePHP	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	MIT		
symfony	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	MIT		
CodeIgniter	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	Apache/BSD		
PHP for Applications	✗	✓	✓	✓	✓	✗	✗	✗	✓	✓	LGPL		
Yii	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	BSD		
DooPHP	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	New BSD		
Fat-Free	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	GNU GPL		
PHP on TRAX	✗	✓	✓	✓	✓	✗	✗	✗	✗	✓	GPL		

Tabla 1.6: Comparación de Frameworks PHP
Fuente: http://www.bestwebframeworks.com/php/
Realizado por: Luis Lima

Se puede evidenciar que no existe mayor diferencia con otros frameworks, sin embargo, por las características de licencia y soporte de PHP 4 y PHP5 CodeIgniter es el más adecuado para el presente proyecto.

En la Figura 1.3 se muestra como fluyen los datos a través de Codeigniter o una aplicación construida con este framework:

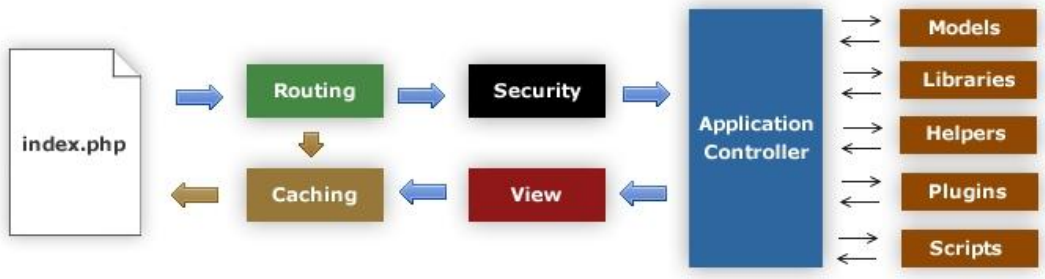


Figura 1.3: Diagrama de flujo de la aplicación
Fuente: http://lax.franhp.net/CodeIgniter_Spanish_UserGuide.pdf
Realizado por: Luis Lima

El flujo de datos es el siguiente:

1. El index.php sirve como controlador frontal, inicializando los recursos básicos necesarios para correr CodeIgniter.

2. El Router examina la petición HTTP para determinar que debe ser hecho con él.
3. Si un archivo de caché existe, es enviado directamente al explorador, sobrepasando el sistema de ejecución normal.
4. Seguridad. Antes que el controlador sea cargado, la petición HTTP y cualquier dato suministrado por el usuario es filtrado por seguridad.
5. El controlador carga los modelos, librerías, plugins, asistentes y cualquier otro recurso necesario para procesar la petición específica.
6. La Vista finalizada es enviada al explorador web para ser vista. Si el cacheo está habilitado, la vista es cacheada primero para que las peticiones subsecuentes puedan ser utilizadas.

Desde el punto de vista técnico y arquitectónico CodeIgniter cumple con los siguientes objetivos:

- **Instanciación Dinámica**.- Los componentes son cargados y las rutinas ejecutadas únicamente cuando se requieren.

- **Poco Acoplamiento**.- "Acoplamiento es el grado en que los componentes de un sistema dependen entre ellos" (EllisLab Inc, 2010, p.24). CodeIgniter crea aplicaciones con poco acoplamiento.

- **Singularidad del Componente**.- Cada clase y sus funciones son altamente autónomas.

Las aplicaciones creadas con CodeIgniter cumplen estos objetivos por lo que resultan ser sistemas: instanciados dinámicamente, poco acoplados y con gran singularidad de componentes.

1.5 PROCESO DE DESARROLLO

Un proceso de desarrollo de software tiene como propósito fundamental la producción de software de manera que éste cumpla los requisitos del cliente tanto en tiempo, costo, alcance y calidad. Este proceso es enteramente intelectual afectado por las habilidades e ingenio de cada persona o grupo de trabajo. Lo indicado en términos globales se muestra en la Figura 1.4.

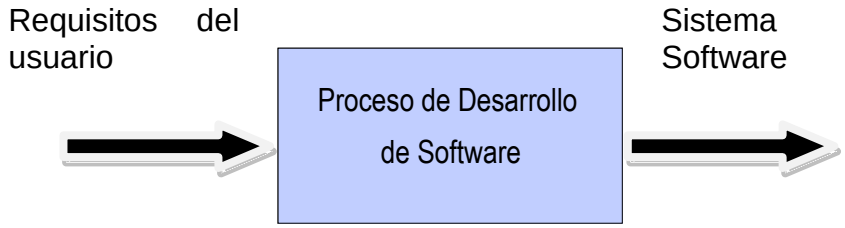


Figura 1.4: Proceso de desarrollo de software
Fuente:Jacobson Ivan, 2000, p.4.
Realizado por: Luis Lima

El proceso de desarrollo de software no es único. No existe un proceso de software universal que sea efectivo para todos los contextos de proyectos de desarrollo. Debido a esta diversidad, es difícil automatizar todo un proceso de desarrollo de software, sin embargo, existen muchos modelos genéricos de proceso de software ampliamente utilizados y que en el presente documento se revisarán brevemente algunos de ellos.

1.5.1 CODIFICAR Y CORREGIR

Se utilizó en los inicios del desarrollo de software, el cual consiste en dos pasos básicos: Escribir código y corregir los problemas en él. Es decir, en primera instancia se implementa algo de código y luego se piensa en aspectos de: requisitos, diseño, validación y mantenimiento, esto lógicamente causa problemas en la estructura del código debido a las múltiples correcciones lo que hace que el mantenimiento sea muy costoso e incluso que el software sea rechazado por el cliente.

1.5.2 MODELO EN CASCADA

Conocido como el “ciclo de vida clásico” (Pressman Roger, 2006, p.50), el cual sugiere un enfoque sistemático, secuencial hacia el desarrollo del software, inicia con la especificación del requerimiento, continúa con la planeación, el modelado, la construcción y el despliegue, finalmente el soporte del software terminado.

1.5.3 DESARROLLO EVOLUTIVO

Los requisitos del cliente con frecuencia cambian incluso durante el desarrollo del sistema, para afrontar esta situación surge el modelo de desarrollo evolutivo, el cual realiza una implantación inicial del sistema para luego exponerla a los comentarios del usuario, refinarla en N versiones hasta que se desarrolle el sistema adecuado.

Este modelo es iterativo lo que le da la ventaja de obtener una rápida realimentación del usuario, ya que las actividades de especificación, desarrollo y pruebas se ejecutan en cada iteración.

1.5.4 DESARROLLO INCREMENTAL

El enfoque incremental de desarrollo surgió como una forma de reducir la repetición del trabajo en el proceso de desarrollo y dar oportunidad de retraso en la toma de decisiones de los requisitos hasta adquirir experiencia con el sistema. Es una combinación del Modelo de Cascada y Modelo Evolutivo, en el cual los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario los elementos necesarios para evaluarlo.

1.5.5 DESARROLLO EN ESPIRAL

El modelo de desarrollo en espiral es actualmente uno de los más conocidos, es un modelo evolutivo combinado con los aspectos sistemáticos del modelo en cascada. El ciclo de desarrollo se representa como una espiral, en lugar de una serie de actividades sucesivas con retrospectiva de una actividad a otra.

Cada proyecto de software requiere de una forma particular para abordar el problema. Las propuestas comerciales y académicas actuales promueven procesos iterativos, donde en cada iteración puede utilizarse uno u otro modelo de proceso, considerando un conjunto de criterios (Por ejemplo: grado de definición de requisitos, tamaño del proyecto, riesgos identificados, entre otros).

1.6 METODOLOGÍAS PARA DESARROLLO DE SOFTWARE

Una metodología de desarrollo no es más que un proceso de software detallado y completo. Las metodologías son una combinación de los modelos de procesos genéricos definidos anteriormente, a demás, define con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, etc.

En la actualidad existe una gran cantidad de metodologías las que las podríamos clasificar en 4 grandes grupos dependiendo de sus

características: Metodologías Estructuradas, Metodologías Orientadas a Objetos, Metodologías Tradicionales y Metodologías Agiles.

1.6.1 METODOLOGÍAS ESTRUCTURADAS

Las metodologías estructuradas se utilizaron en conjunto con la Programación Estructurada. Estas metodologías se basan en la estructuración y descomposición funcional de los problemas en unidades más pequeñas relacionadas entre sí, los sistemas básicamente se ven como entradas, proceso y salidas. Los proyectos desarrollados con estas metodologías trabajan muy bien con lenguajes estructurados como el COBOL.

Como ejemplos de estas metodologías tenemos: MERISE, MÉTRICA, SSADM, Gane y Sarson, Ward y Mellor, Yourdon y DeMarco.

1.6.2 METODOLOGÍAS ORIENTADAS A OBJETOS

Estas metodologías surgen en paralelo con la evolución de los lenguajes de programación orientada a objetos, permiten que el software se construya a partir de objetos de comportamiento específico lo que hace que el producto resultante sea extensible y reutilizable. Como parte de la evolución de estas metodologías surge Unified Modeling Language (UML) que es la notación orientada a objetos más utilizada en la actualidad.

Algunas metodologías orientadas a objetos que utilizan la notación UML son: Rational Unified Process (RUP), OPEN, MÉTRICA.

1.6.3 METODOLOGÍAS TRADICIONALES

Las metodologías tradicionales o no ágiles son aquellas en las que se realiza una fuerte planificación durante todo el proceso de desarrollo y una intensa etapa de análisis y diseño antes de la construcción del sistema.

Todas las propuestas metodológicas antes indicadas pueden considerarse como metodologías tradicionales, a excepción de RUP.

1.6.4 METODOLOGÍAS ÁGILES

Un proceso es ágil cuando el desarrollo de software es incremental, cooperativo, sencillo y adaptable, esto hace que el proceso de desarrollo se pueda manipular de forma impredecible ante un posible cambio de los

requisitos del usuario con adaptaciones incrementales en base a la retroalimentación periódica que da el usuario.

Entre las metodologías ágiles tenemos:

- Extreme Programming.
- Scrum.
- Familia de Metodologías Crystal.
- Feature Driven Development.
- Rational Unified Process (RUP).
- Dynamic Systems Development Method.
- Adaptive Software Development.
- Open Source Software Development.

Por la naturaleza de la presente investigación y las características del sistema a desarrollar se utilizará la metodología ágil Extreme Programming la cual se expone a continuación.

1.7 EXTREME PROGRAMMING (XP)

La Programación Extrema es una metodología ágil que utiliza “un enfoque orientado a objetos como su paradigma de desarrollo preferido” (Pressman Roger, 2006, p.84).

XP se centra en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. En esta metodología es fundamental la realimentación continua entre el cliente y el equipo de desarrollo, la misma que debe ser fluida entre todos los participantes con la finalidad de obtener soluciones simples y tener la capacidad de reacción frente a los cambios que se presenten considerando que los requerimientos de los clientes siempre sufrirán cambios ya sea al inicio, durante y después del proyecto.

Cuatro prácticas esenciales caracterizan a XP que la diferencian de otras metodologías: liberación limitada, semana de trabajo de 40 horas, cliente en el sitio y programación en parejas. En la Figura 1.5 se ilustran estas cuatro prácticas.

Figura 1.5: Prácticas esenciales de XP
Fuente: E. Kendall Kenneth, 2005, p.73
Realizado por: Luis Lima

- La liberación limitada significa que en lugar de liberar una versión completamente desarrollada que puede tardar mucho tiempo, se hacen liberaciones parciales en corto tiempo y se continúa con mejoras continuas al producto inicial.

- La semana de trabajo de 40 horas significa que el equipo de desarrollo se compromete a trabajar intensamente durante una semana típica de trabajo y así tener las horas de descanso necesarias.

- Cliente en el sitio consiste en contar con un usuario experto en el negocio como parte del proyecto de desarrollo. Esta persona es quién escribe las historias de usuario.

- La programación en parejas consiste en que un programador trabaja junto a otro de su elección. “La programación en parejas ahorra tiempo, reduce las distracciones, activa la creatividad y es una forma divertida de programar” (E. Kendall Kenneth, 2005, p.170)

El marco de trabajo de la Programación Extrema se centra en cuatro actividades fundamentales: planeación, diseño, codificación y pruebas que se ilustran en la Figura 1.6.

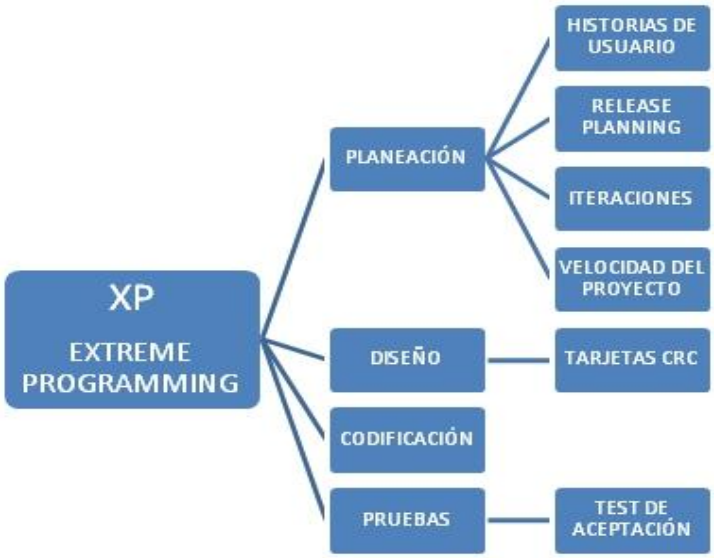


Figura 1.6: Actividades fundamentales de XP
Realizado por: Luis Lima

- **Planeación**.- En esta actividad se crean las ***historias del usuario***, que básicamente describen en lenguaje natural las necesidades requeridas por cada usuario para el sistema que se construirá lo que equivaldría a los casos de uso en el proceso unificado.

  Después de tener ya definidas las historias de usuario es necesario crear un plan de publicaciones denominado en inglés ***Release Planning***, donde se indican las historias de usuario que se crearán en cada versión del programa y las fechas en que se publicarán estas versiones. Las historias incluidas en cada versión se seleccionan en función de la prioridad dada por el usuario.

  Todo proyecto en XP se debe dividir en ***iteraciones*** de aproximadamente tres semanas de duración, al inicio de cada iteración el usuario debe definir las historias incluidas en el Release Planning que se deben implementar.

  En base al número de historias que se pueden implementar en cada iteración se determina la ***velocidad del proyecto*** que representa la rapidez con la que se desarrolla el proyecto.

- **Diseño**.- El diseño en XP debe ser simple y apegado estrictamente a lo indicado en las historias del usuario. Si se encuentra un diseño complicado XP recomienda la creación de prototipos con la finalidad de reducir el riesgo cuando comience la verdadera implementación y así validar las fechas de entrega de las historias con problemas de

diseño. De igual forma XP apoya la re-fabricación que no es más que realizar el diseño de manera continua a medida que se construye el sistema.

El único producto entregable en el diseño son las ***tarjetas CRC*** (clase-responsabilidad-colaborador).

Según Pressman (2006:225) "El modelado de Clase-Responsabilidad-Colaborador proporciona un medio simple de identificar y organizar las clases relevantes para los requisitos del sistema o producto".

El objetivo de las tarjetas CRC es obtener una representación organizada de las clases. Las responsabilidades son los atributos y operaciones para la clase. Los colaboradores son aquellas clases que se necesitan para que otra clase reciba la información que requiere para completar una responsabilidad.

En la Figura 1.7 se muestra un ejemplo de las tarjetas CRC que se utilizarán en el presente proyecto; en el mismo se representa a la clase **PlanodePlanta** con sus responsabilidades y colaboradores. Esta clase no es parte del proyecto de desarrollo, se la utiliza como ejemplo para ilustrar el uso de las tarjetas CRC.

Clase: **PlanodePlanta**	
Responsabilidad	**Colaborador**
Define el nombre/tipo del plano de planta	
Maneja la posición del plano de planta	
Escala el plano de planta para su despliegue	
Incorpora paredes, puertas y ventanas	**Pared**
Muestra la posición de las cámaras de video	**Cámara**

Figura 1.7: Tarjeta CRC
Fuente: Pressman Roger, 2006, p.226
Realizado por: Luis Lima

Algo muy importante que XP recomienda en esta fase es nunca añadir funcionalidad extra al programa aunque se piense que en el futuro será utilizada.

- **Codificación**.- XP recomienda que una vez realizado el diseño no se realice inmediatamente la codificación, antes se deben desarrollar una serie de pruebas de unidad que ejerciten las historias que se incluirán en el incremento, es decir, se debe preparar un plan de pruebas lo

que hace que el desarrollador se centre en lo que debe codificar, de esta forma una vez terminado la codificación se pueden realizar pruebas inmediatamente.

Algo importante de la programación extrema es que en esta actividad de codificación se recomienda el trabajo en pareja con lo que se pretende la resolución de problemas de manera rápida así como el aseguramiento de la calidad.

- **Pruebas**.- Como ya se indicó anteriormente, las pruebas de unidad ya se las tiene preparadas incluso antes de la codificación, estas deben estar realizadas de manera que se puedan repetir fácilmente cuando exista modificación del código. Esto ayuda a probar los desarrollos incluso de manera diaria lo que proporciona al equipo una indicación continua del progreso o si es el caso de que las cosas van mal con lo que se puede tomar acciones correctivas a tiempo.

  Las pruebas de aceptación son las que realiza el usuario o cliente y se enfocan en las características generales y la funcionalidad del sistema definidas en las historias.

En la Figura 1.8 se ilustra el proceso de la programación extrema para producir un incremento del sistema que se está desarrollando.

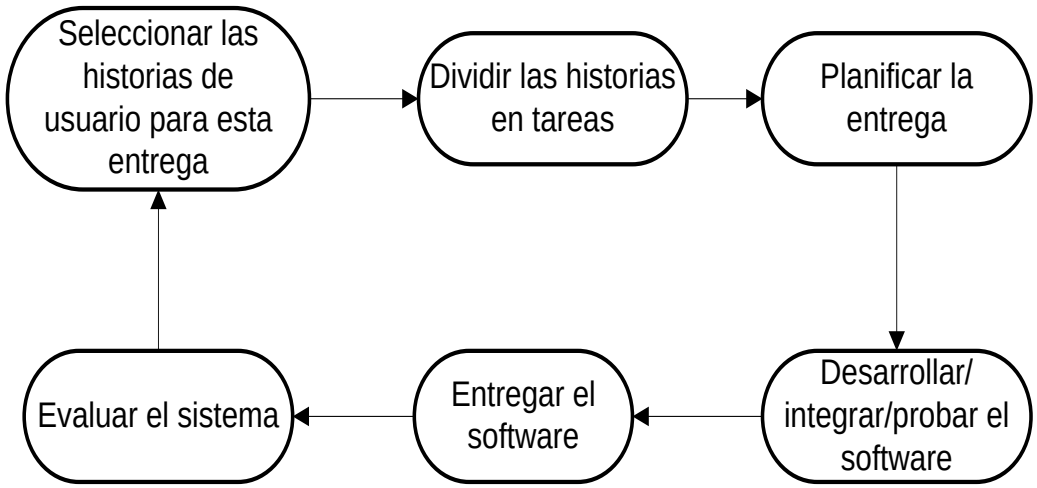


Figura 1. 8: Ciclo de entrega en la programación extrema
Fuente: Sommerville Ian, 2005, p.364
Realizado por: Luis Lima

De lo expuesto anteriormente se puede concluir que la programación extrema es una metodología de desarrollo de sistemas basado en los valores de la simplicidad, la comunicación, la retroalimentación y coraje para afrontar los cambios todo esto complementado con un equipo constantemente informado para que conozca en donde está y que es lo que debe ajustar en su accionar.

1.8 RESUMEN

Los patrones de diseño es un tema fundamental en el desarrollo de software actual ya que permite principalmente obtener un software de calidad con características de reutilización y extensibilidad.

Los patrones de diseño son complejos tanto en su implementación como en su comprensión, sin embargo, con la utilización de los frameworks de desarrollo esta complejidad se reduce principalmente en los patrones más utilizados.

En los desarrollos Web el patrón de diseño más idóneo es el Modelo-Vista-Controlador (MVC), el cual divide el problema en tres capas: Model, View y Controller.

Los frameworks de desarrollo son aplicaciones incompletas que al agregarle los elementos que faltan tendremos una solución acorde a lo requerido sin tener que empezar desde cero.

En la presente investigación se utilizará el **framework PHP CodeIgniter**, el cual implementa el patrón de diseño MVC con una arquitectura en tres capas. Este framework utiliza como lenguaje de programación PHP y está relacionado directamente con metodologías ágiles de desarrollo como Extreme Programming (XP).

CAPITULO II

2. PROPUESTA DE LA SOLUCIÓN

En el presente capítulo se inicia con la aplicación práctica de lo expuesto en el anterior mediante el desarrollo de una aplicación Web para una necesidad real, se pone especial énfasis en la utilización de: patrones de diseño, frameworks de desarrollo y programación extrema.

En base a la metodología XP, este capítulo corresponde a la fase de **Planeación**.

2.1 PROCESO ACTUAL

La Asociación de Caballos de Paso (Asopaso) organiza varios concursos al año en los que participan caballos en representación de los distintos socios y criadores, estos concursos se realizan en varias ciudades del Ecuador. Todos los procesos que tienen relación con los concursos son realizados actualmente de forma **manual** con formularios preimpresos y hojas electrónicas de Excel lo que dificulta el control y la administración de los concursos.

2.1.1 DIFICULTADES

Las principales dificultades que presenta el proceso manual que se lleva actualmente son las siguientes:

- No se tiene un registro real de los caballos de paso ni su historial, se tiene un histórico en Excel el mismo que no otorga ninguna garantía de estar actualizado.

- El proceso de inscripción a los diferentes concursos se realiza mediante formularios impresos que son enviados a cada socio quienes confirman por esta vía los inscritos. Con estos formularios se registran en documentos Excel para los respectivos controles.

  En el Anexo A se puede apreciar un ejemplo del formulario utilizado actualmente en los procesos manuales.

- El registro de los propietarios y criadores de caballos se realiza en base a formularios y se registran en documentos Excel.

- La impresión de los Catálogos se realiza desde los documentos Excel. El Catálogo es un documento impreso en el que constan todos

los caballos inscritos clasificados por categorías, tiene los espacios necesarios para registrar manualmente las ubicaciones por cada caballo y es entregado a todos los participantes para su respectivo control.

En el Anexo B se muestra un ejemplo del Catálogo utilizado.

- En la realización del evento el registro de las ubicaciones en las distintas categorías se lo realiza manualmente en los Catálogos de cada concurso.

- El cómputo final se lo hace de forma manual en documentos Excel, dando paso a errores involuntarios.

  En el Anexo C se muestra un ejemplo del documento Excel utilizado en un concurso.

- Los informes presentados el día final del evento únicamente considera los mejores caballos en base a los puntajes totales, para informes detallados se tiene que esperar aproximadamente una semana hasta elaborar los cuadros en Excel con todos los resultados.

2.1.2 DESVENTAJAS

Las principales desventajas de este proceso son las siguientes:

- Los tiempos de respuesta para la inscripción como para la presentación de los resultados es bastante alto y existe malestar en los organizadores y participantes que requieren conocer inmediatamente los mismos.

- Al no tener un registro correcto de los caballos, criadores y propietarios frecuentemente se presentan errores en la inscripción al registrar caballos en categorías fuera de edad lo que al final genera contratiempos en la ejecución del concurso.

- Para caballos inscritos el día del evento el registro es manual lo que complica el cómputo ya que se tiene que considerar estas inscripciones sin mayor respaldo.

- Al manejarse todo el proceso en documentos impresos y hojas Excel no se tiene un histórico real de los concursos realizados

- Los documentos Excel generados durante los concursos son responsabilidad de la persona encargada de la organización, los cuales son archivados en un computador personal sin ninguna política o procedimiento específico.

2.1.3 RIESGOS

Este proceso tiene muchos riesgos evidentes, entre los que mencionaremos los tres más importantes:

- No registrar todos los caballos inscritos.

- Inscribir caballos en categorías que no corresponden debido a la edad del mismo.

- El cómputo de resultados finales tiene un alto grado de error y puede ser manipulado por personas ajenas al proceso.

Con la frecuencia que se vienen realizando estos concursos y el alto número de participantes, el personero de Asopaso encargado de administrar estos eventos se vio en la necesidad de automatizar todos los procesos con la utilización de un sistema informático, esto posibilita el trabajo conjunto de los temas explicados en el capítulo I mediante el desarrollo de un aplicativo que resuelva la necesidad del cliente y a su vez permite aplicar de manera práctica el patrón de diseño MVC y el framework de desarrollo CodeIgniter que es uno de los objetivos del presente proyecto.

Las necesidades que se requieren cubrir con el sistema informático están descritas brevemente en las historias de usuario redactadas por el cliente las mismas que se revisan a continuación.

2.2 HISTORIAS DE USUARIO

En las historias de usuario se describen brevemente las características que el sistema debe tener desde el punto de vista del cliente. Para las historias de usuario del presente proyecto se utilizará el formato descrito en el Anexo D.

Las historias de usuario disponibles sobre las que se trabajará se describen a continuación:

HISTORIA DE USUARIO	
No. 1	**Usuario responsable:** Administrador
Nombre historia: Funcionalidad general	
Prioridad: Alta	
Iteración asignada: 1-4	
Programador responsable: Luis Lima	
Descripción: ➢ Como requisitos generales del sistema se requiere principalmente: auditoría, seguridad y la accesibilidad desde cualquier punto geográfico debido fundamentalmente a que los concursos se realizan en varias ciudades del país. ➢ Los cambios realizados al sistema deben ser auditados, se requiere conocer quién y cuándo realizó alguna acción de actualización, borrado o inserción de datos, las auditorías registradas serán revisadas en función de las necesidades por el administrador de la aplicación. ➢ Los accesos a las distintas opciones del sistema se controlarán mediante roles los cuales en principio serían los siguientes: administrador, consultas, registro y juez. El rol administrador tendrá acceso a todas las funcionalidades del sistema y es quien gestiona los usuarios con sus respectivos roles. Para la aplicación existirá un solo administrador.	
Observaciones:	

Figura 2.1: Historia de usuario – Funcionalidad general

Realizado por: Luis Lima

HISTORIA DE USUARIO	
No. 2	**Usuario responsable:** Administrador
Nombre historia: Parámetros generales	
Prioridad: Alta	
Iteración asignada: 1-4	
Programador responsable: Luis Lima	

Descripción: Se requiere poder ingresar en el sistema los siguientes parámetros: tipos de caballos, categorías y concursos.

- Tipos de caballos.- En principio se trabajará con los siguientes tipos: yeguas, potros y capones, sin embargo, se podría incrementar en función de las necesidades.
- Categorías.- Una categoría en términos generales define qué caballos pueden participar en ella en función de la edad y sexo de los mismos, define los puntos que otorga a los ganadores, etc. Las categorías tiene la siguiente información general:
  - Código asignado a la categoría por la Asociación.
  - Nombre de la categoría.
  - Edad inicial y final de los caballos para asignar la categoría.
  - Si es puntuada o no, es decir, si la categoría otorga puntos a los caballos que se ubiquen en los primeros puestos.
  - Puntos por ubicación, en cada categoría existe un puntaje diferente por ubicación.
- Concursos.- Para cada evento que se realice, previamente se debe parametrizar en el sistema el concurso con la información general del mismo. Con la parametrización del concurso se inicia formalmente el evento en el sistema debido a que todos los procesos siguientes están atados a un concurso. En la definición de un concurso se registra la siguiente información:
  - Nombre del concurso
  - Fecha sin formato específico. Ej: 11 Y 12 DE JULIO DEL 2.009
  - Descripción del lugar donde se realiza el concurso.
  - Nombre del Juez único.
  - Nombre del Juez Adjunto.
  - Nombre del Comisario General.
  - Nombre del Juez de Cómputo.
  - Categorías participantes.
  - Fecha de cálculo especificada en dd/mm/yyyy, la cual se utiliza como referencia para el cálculo de la edad de los caballos inscritos.

Observaciones:

Figura 2.2: Historia de usuario – Parámetros generales

Realizado por: Luis Lima

HISTORIA DE USUARIO	
No. 3	**Usuario responsable:** Administrador
Nombre historia: Registro de personas y caballos	
Prioridad: Alta	
Iteración asignada: 2-4	
Programador responsable: Luis Lima	

Descripción: Se requiere registrar tanto las personas y caballos que participan en un concurso.

- Personas.- Las personas son todas aquellas entidades que tienen alguna relación con el concurso (propietarios o criadores de caballos), de ellos se tiene la siguiente información:
  - Nombres completos separados en: primer nombre, segundo nombre, primer apellido y segundo apellido.
  - Prefijo (iniciales del nombre).
- Caballos.- Sobre los caballos se debe registrar la siguiente información:
  - Prefijo (otorgado por el prefijo del criador).
  - Nombre del caballo.
  - Registro otorgado por la Asociación. Ej: 00-1-0014.
  - Fecha de nacimiento.
  - Padre.
  - Madre.
  - Criador.
  - Propietario.
  - Tipo de caballo.

  Se debe contar con opciones de consulta tanto para las personas y caballos registrados en las cuales no se pueda modificar la información.

Observaciones:

- El formulario utilizado en el proceso manual se encuentra en el Anexo A.

Figura 2.3: Historia de usuario – Registro de personas y caballos

Realizado por: Luis Lima

HISTORIA DE USUARIO	
No. 4	**Usuario responsable:** Administrador
Nombre historia: Gestión de concursos	
Prioridad: Alta	
Iteración asignada: 3-4	
Programador responsable: Luis Lima	
Descripción: Para la realización del concurso se debe realizar las siguientes acciones en el sistema: ➢ Inscripciones.- Es el registro de los caballos que participarán en el concurso. La inscripción se realiza en función del código de registro de cada caballo, el sistema debe determinar de manera automática la categoría en la que participa el caballo en función de la fecha de nacimiento y la fecha de cálculo establecida. Las inscripciones deben tener un cierre con lo cual se permita generar el catálogo del concurso. Se debe contar con los siguientes reportes: Inscritos por categoría y total de caballos inscritos con los cuales se elabora el catálogo del concurso. ➢ Calificación.- Proceso que se realiza durante la ejecución del concurso, la calificación es por concurso y categoría en la cual se registra el puesto de cada caballo y si tiene mención honrosa o no. ➢ Cierre del concurso.- Proceso mediante el cual se da por finalizado el concurso y se realiza el cómputo final de los resultados, luego de este proceso no se puede modificar la información del concurso. Para el cómputo de los resultados se debe utilizar lo especificado en el Reglamento para Concursos, el cual fue entregado en formato digital.	
Observaciones:	

Figura 2.4: Historia de usuario – Gestión de concursos y resultados

Realizado por: Luis Lima

HISTORIA DE USUARIO	
No. 5	**Usuario responsable:** Administrador
Nombre historia: Resultados	
Prioridad: Alta	
Iteración asignada: 4	
Programador responsable: Luis Lima	
Descripción: Una vez realizado el cierre del concurso, se debe poder obtener los siguientes resultados tanto en pantalla como impresos: ➢ Resultados.- Informes en pantalla e impresos de los ganadores del concurso. Se deben tener resultados por: o Categoría. o Generales clasificados por: mejor criador, mejor expositor y mejor reproductor. ➢ Estos informes deben incluir el detalle de los puntos obtenidos en cada categoría en la que haya participado el ganador.	
Observaciones: ➢ Ver Anexo C.	

Figura 2.5: Historia de usuario – Resultados
Realizado por: Luis Lima

2.3 PROCESO PROPUESTO

El proceso que se propone para lograr el objetivo se centra en desarrollar e implementar un aplicativo Web para la gestión de los concursos que realiza la Asociación de Caballos de Paso con el que se **automaticen todos los procesos** relacionados con los concursos. El sistema agilizará y garantizará principalmente los siguientes aspectos:

- Registro e inscripción correcta de: caballos, criadores y propietarios con toda la información requerida.
- Cómputo inmediato de los datos para entregar resultados oportunos en los concursos.
- Presentación de informes de forma visual e impresa a los socios de manera ágil, a demás, tendrá información histórica de todos los concursos, participantes y resultados.

- Acceso al sistema desde cualquier lugar a través del Internet.

En la Tabla 2.1 se detallan los aspectos más importantes considerados para la definición del tipo de aplicación a desarrollar entre Web o escritorio.

CARACTERÍSTICAS DE LA APLICACIÓN	TIPO DE APLICACIÓN	
	WEB	ESCRITORIO
Se requiere poder acceder al sistema desde varios puntos separados geográficamente.	X	X
No se contará con enlace dedicado desde los lugares en los que se realicen los concursos.	X	
No se contará con infraestructura propia, se utilizará la ofrecida por un Hosting.	X	
Los socios requieren mantener actualizada su información desde cualquier lugar u oficina a través del Internet.	X	
Se debe utilizar herramientas de uso libre tanto para la base de datos como para la programación por definición del cliente.	X	X
Facilidad de actualizaciones	X	
Como segunda fase se piensa incorporar la aplicación a dispositivos móviles	X	

Tabla 2.1: Análisis para selección del tipo de aplicación

Realizado por: Luis Lima

De lo expuesto en la Tabla 2.1 se determina fácilmente que la mejor solución se la obtiene con un aplicativo Web siendo este el tipo seleccionado.

2.2.1 INFRAESTRUCTURA

Por requerimientos del cliente la aplicación a desarrollar debe ser Web, en tal virtud no se cuenta con una infraestructura propia. La infraestructura a utilizar es la otorgada por el proveedor de alojamiento web[3] (web hosting) que en este caso es Hostmonster. Hostmonster es una empresa que ha proporcionado soluciones de alojamiento desde 1996 tanto a empresas como Web personales.

La infraestructura mínima que debe proporcionar el alojamiento web se detalla en la Tabla 2.2.

[3] Alojamiento web: Servicio que provee a los usuarios de internet un sistema para poder almacenar cualquier contenido accesible vía Web.

EQUIPO	SERVICIO	SO
1 Servidor con procesador Pentium IV , 1 GHz de velocidad, 1 GB de memoria RAM, 40 GB en disco.	Servidor Web / Servidor de BDD	GNU/Linux
Acceso a Internet con 128 Kbps de velocidad	Internet	
1 PC con procesador Pentium IV, 1 Ghz de velocidad, 2 56 MB de memoria RAM, 20GB en disco.	Cliente	Windows, GNU/Linux/Mac
Acceso a Internet para el cliente con 128 Kbps de velocidad.	Internet	

Tabla 2.2: Infraestructura

Realizado por: Luis Lima

Como se puede apreciar en la Tabla 2.2, los requisitos en infraestructura son mínimos y pueden ser cubiertos sin problemas por varios proveedores de Web Hosting.

2.2.2 HERRAMIENTAS

En las Tablas 2.3 y 2.4 se describen las herramientas utilizadas tanto para el desarrollo de la aplicación como para la ejecución de la misma.

CARACTERÍSTICAS	HERRAMIENTAS	VERSION
PHP + MySQL	Xampp	1.7.3
Framework	CodeIgniter	1.7.2
Base de datos	MySQL	5.1.41
Interfaz de administración de BDD	PhpMyAdmin	3.2.4
Depurador del navegador Firefox	Firefox Firebug	1.6
IDE de desarrollo	Eclipse para PHP	Galileo
Librerías de Javascript	JQuery	1.3.2
Diagramas UML	StarUML	5.0

Tabla 2.3: Herramientas de desarrollo

Realizado por: Luis Lima

CARACTERÍSTICAS	HERRAMIENTAS	VERSION
Servidor Web + PHP 5	Apache	2.2
Base de datos	MySQL	5.1.41
Browser	Internet Explorer	8.0 o superior
	Mozilla Firefox	3.6.12 o superior

Tabla 2.4: Herramientas de ejecución

Realizado por: Luis Lima

Todo el software utilizado es gratuito y de código abierto, razón fundamental para su elección, además, están orientados a construir aplicaciones Web de manera rápida y tienen una rica historia en el campo del desarrollo de software.

En el capítulo I se realizó un análisis para la selección del Framework CodeIgniter, en la Tabla 2.5 se realiza una comparativa para la selección de la base de datos en la cual se determina que la mejor alternativa para este proyecto es MySql.

CARACTERÍSTICAS	DBMS	
Nombres:	**MySql**	**PostgreSQL**
Soporte de grandes volúmenes de datos	Si	Si
Licencia	GPL	BSD
Documentación con PHP	Alta	Medio
Utilización de desarrolladores PHP	Alta	Medio
Integración con CodeIgniter	Si	Si
Conocimiento por parte del desarrollador	Alta	Medio

Tabla 2.5: Infraestructura mínima

Realizado por: Luis Lima

Una vez realizado el análisis de la situación actual, revisado las historias de usuario y planteado la solución se procede a planificar las iteraciones lo cual se revisa a continuación.

2.3 ITERACIONES

Se han definido 4 iteraciones para cubrir las necesidades planteadas, en cada una de ellas se han incluido las historias de usuario en función de su prioridad de construcción, esto se detalla en la Tabla 2.6 :

ITERACION	HISTORIA DE USUARIO
1	- 01 Funcionalidad general - 02 Parámetros generales
2	- 03 Registro de personas y caballos - Otros cambios
3	- 04 Gestión de concurso - Otros cambios
4	- 05 Resultados - Otros cambios

Tabla 2.6: Iteraciones
Realizado por: Luis Lima

Cada iteración tiene un tiempo estimado de 3 semanas, con lo cual se tiene un tiempo total de 12 semanas para concluir el sistema.

Una vez definidas las historias de usuario en cada iteración se puede sugerir un calendario de eventos para el proyecto que concluye con la entrega de las funcionalidades requeridas por el cliente. En la Tabla 2.7 se resume el calendario de eventos con los que se trabajaría en adelante.

EVENTO	FECHA
Inicio del proyecto	03/Nov/2011
Primera entrega	22/Nov/2011
Segunda entrega	06/Dic/2011
Tercera entrega	03/Ene/2011
Cuarta entrega	31/Ene/2011

Tabla 2.7: Calendario de eventos
Realizado por: Luis Lima

2.4 DEFINICIÓN DEL ALCANCE – RELEASE PLANNING

En la primera versión del sistema se incluirán las cinco historias de usuario en su totalidad, para lo cual se cubrirá el siguiente alcance dividido en módulos como se detalla a continuación:

2.4.1 MÓDULO DE PARÁMETROS

Este módulo contempla las funcionalidades con las que el usuario realiza la parametrización general del sistema como son: Tipos de Caballos,

Mantenimiento de las distintas Categorías en las que pueden participar los caballos y la Creación de Concursos con sus respectivos atributos.

- **Tipos de Caballo.-** Contempla las opciones básicas de: inserción, modificación, consulta y borrado de las diferentes tipos de caballos existentes.

- **Mantenimiento de Categorías**.- Contempla las opciones para la creación, modificación, consulta y eliminación de categorías. Estas funcionalidades permitirían definir los atributos propios de la categoría y como se debe realizar la calificación en cada una y las requisitos para participar.

- **Creación del Concurso**.- Contempla las funcionalidades para crear y modificar un concurso con información referente a fechas de realización, jueces, lugar, categorías participantes, etc.

2.4.2 MÓDULO DE REGISTRO

Este módulo contempla las funcionalidades para el registro y mantenimiento tanto de caballos como de sus criadores y propietarios (Personas). El registro de caballos es la base maestra de los animales que pueden participar en un concurso.

- **Registro de Personas**.- Insertar, modificar, consultar y borrar personas o instituciones que de una u otra forma tienen caballos que participan en los concursos.

- **Registro de Caballos**.- Insertar, modificar, consultar y borrar la información relacionada con los caballos participantes en el concurso.

2.4.3 MÓDULO DE CONCURSOS

En este módulo se realiza los procesos propios del concurso como son:

- **Inscripciones**.- Funcionalidad para realizar la inscripción de caballos en las distintas categorías para un determinado concurso. La inscripción debe ser únicamente de los caballos registrados en el sistema.

- **Cierre de Inscripciones.-** Hasta una determinada fecha el sistema debería permitir inscripciones, luego se realiza el cierre en el que se

asigna los números a los participantes, luego de este proceso no se pueden realizar más inscripciones.

- **Calificación**.- Funcionalidad que permite registrar la ubicación y calificación de los caballos durante la realización del concurso en función de su participación y la calificación dada por el juez.

- **Cierre del Concurso.-** Comprende el cierre como tal del concurso y el cómputo de las ubicaciones y puntajes de cada participante para seleccionar los mejores criadores, reproductores y propietarios. Luego de realizado este proceso toda la información del concurso no puede ser modificada.

2.4.4 MÓDULO DE RESULTADOS

Son informes tanto en pantalla como impreso de los resultados del concurso. Estos resultados pueden ser de una Categoría específica o resultados Generales tanto para criadores, propietarios y reproductores.

- **Resultados por Categoría**.- Reporte donde constan los resultados de todos los caballos inscritos en cada categoría de un concurso en particular.

- **Generales**.- Informe en pantalla con opción de impresión de las mejores ubicaciones por tipo, es decir: mejor criador, mejor reproductor y mejor expositor.

2.4.5 MÓDULO DE CONSULTAS

Consultas básicas en pantalla de personas y caballos registrados en el sistema.

- **Personas Registradas.-** Consulta de todas las personas registradas la cual tiene facilidades de búsquedas por cualquier campo desplegado en pantalla.

- **Caballos Registrados.-** Consulta de todos los caballos registrados la cual debe dar la facilidad de búsquedas por cualquier campo desplegado en pantalla.

2.4.6 MÓDULO DE REPORTES

Reportes utilizados para la elaboración de Catálogos en cada concurso, estos son:

- **Inscritos por categoría**.- Listado en el que se detallan todos los caballos inscritos por categoría para un concurso en particular. Este reporte se utiliza de matriz para la elaboración de los catálogos que se entregan a los participantes.

- **Total caballos inscritos**.- Listado en el que se detallan todos los caballos inscritos ordenados de manera ascendente en función del número asignado en la inscripción.

2.4.7 MÓDULO DE MANTENIMIENTO

Este módulo gestionará la Seguridad y Respaldos del sistema.

- **Seguridad**.- Se administra los usuarios, esto es creación, modificación e inactivación de usuarios, además, permitiría asignar los roles requeridos para cada usuario. Esta funcionalidad estará activa para el usuario administrador.

- **Respaldos**.- Funcionalidad que permitiría al usuario administrador realizar respaldos periódicos de la base de datos.

Las funcionalidades indicadas se describen en la Figura 2.6. de manera general.

Figura 2.6: Funcionalidades del sistema Asopaso
Realizado por: Luis Lima

2.5 OPORTUNIDADES DE MEJORA

En lo funcional la aplicación en su primera versión contemplará todas las necesidades del cliente para la gestión de los concursos, sin embargo, se deja planteado como oportunidades de mejora para una siguiente versión los siguientes aspectos:

- Inclusión de JQuery Mobile[4] con la finalidad de que esta aplicación Web opere en dispositivos móviles, principalmente con las opciones de consulta.

- Definición de un estándar de categorías para todos los concursos, al momento estas cambian de un concurso a otro lo que el sistema en su primera versión considera, sin embargo, una vez el cliente llegue a un acuerdo se debe adecuar el sistema para simplificar la definición de nuevos concursos y las categorías participantes.

[4] JQuery Mobile: Framework que proporciona herramientas para desarrollar interfaces dinámicas especialmente para pantallas táctiles, tanto de tablets como de smartphones.

- Generación de archivos Excel en los que se incluya: Inscritos por categoría, total de caballos inscritos, personas, caballos, etc. Estos reportes en su primera versión se generan en PDF.

Estas funcionalidades no son prioritarias y en acuerdo con el usuario no se implementarán hasta la siguiente versión.

2.6 RIESGOS DEL PROCESO PROPUESTO

En el proceso propuesto no se encuentran mayores riesgos por cuanto existe la participación directa del cliente y se ha entregado la información necesaria para el entendimiento del problema, sin embargo, como todo proyecto de sistemas tiene sus problemas que se exponen en la Tabla 2.8:

No.	Riesgo	Probabilidad	Impacto	Plan de acción
1	Cambio de los requisitos por parte del cliente, el alcance se extiende.	80%	2	Ajustes en el cronograma
2	La estimación del tiempo para el proceso propuesto puede ser bajo	60%	2	Ajustes en el cronograma
3	Usuarios finales se resistan al uso del sistema	40%	3	Plan de capacitación a todos los usuarios
4	Cambios en el reglamento del concurso	30%	3	Ajustes en el cronograma
5	Cliente cancele el proyecto por temas de organización interna	10%	1	Notificación formal.
6	Poca experiencia en el lenguaje PHP, puede causar un retraso en las entregas.	10%	2	Soporte de un experto
7	La tecnología seleccionada no cumpla las expectativas.	10%	1	Evaluación en las entregas parciales

Tabla 2.8: Riesgos del proceso propuesto

Realizado por: Luis Lima

Valores de impacto:

1: catastrófico
2: critico
3: marginal
4: despreciable

En función de los riesgos identificados se han tomado las acciones de mitigación correspondientes detalladas en la tabla 2.8 en la columna Plan de acción, sin embargo, ellos estarán presentes a lo largo del proyecto.

Los riesgos de alto impacto al proyecto realmente no tienen probabilidad real de que ocurran, sin embargo, están considerados con la finalidad de estar informados todos los involucrados en el proyecto incluyendo el cliente, esto asegura que el proceso propuesto tendrá el éxito esperado dentro de los parámetros de tiempo, costo, alcance y calidad.

2.7 RESUMEN

En la metodología XP, este capítulo tiene relación con la fase de planeación por lo que se hace un análisis de la situación actual del proceso vigente para la realización de concursos de caballos.

El proceso actual en la realización de concursos es manual en base a formularios impresos y hojas de cálculo lo que causa muchos inconvenientes principalmente de control y entrega de resultados. Los riesgos presentes en este proceso son muy probables de ocurrir como es el de presentar resultados erróneos a los participantes.

El personero encargado de este proceso, vio la necesidad de automatizar la gestión de los concursos para lo cual redactó las historias de usuario con una breve descripción de lo que se necesitaría en un sistema informático.

En base a las historias de usuario descritas se planteó la solución con la implementación de un sistema informático Web. No se contará con una infraestructura propia ya que se utilizará un servicio de alojamiento Web en Internet. Las herramientas a utilizar son libres y de código abierto al igual que la base de datos.

Las iteraciones definidas para la culminación del sistema a construir son cuatro, en cada una de ellas se cubrirán las historias de usuario en función de su prioridad definida con el cliente considerando su importancia y necesidad secuencial en el proyecto.

Finalmente se realiza la definición del alcance del sistema a construir, éste está dividido en módulos con la finalidad de tener una mejor organización y visión. Los módulos que el sistema contemplará son los siguientes:

- Módulo de parámetros
- Módulo de registro

- Módulo de concurso
- Módulo de resultados
- Módulo de consultas
- Módulo de reportes
- Módulo de mantenimiento

Los principales riesgos asociados con la solución planteada básicamente se relacionan con cambios en las definiciones iniciales o en el reglamento que rige el concurso lo que lógicamente causaría cambios en el cronograma.

CAPITULO III

3. DISEÑO DE LA SOLUCIÓN

En la metodología XP este capítulo tiene relación con la fase de Diseño. Se documenta todo el diseño de la solución iniciando con la identificación de las tarjetas CRC en base a las historias de usuario hasta el diseño y modelado de las interfaces de usuario.

Los artefactos utilizados corresponden a los definidos por la metodología de desarrollo seleccionada complementados con los diagramas de clases correspondientes al patrón MVC y los modelos Entidad – Relación.

3.1 TARJETAS CRC

Del análisis de las historias de usuario expuestas en el capítulo anterior se tienen identificadas inicialmente las siguientes tarjetas CRC que definirán las clases principales de la aplicación:

Clase: Concurso	
Responsabilidad	**Colaborador**
Define información del concurso	
Define categorías participantes	Categoría

Figura 3.1: Tarjeta CRC - Tipo caballo
Realizado por: Luis Lima

Clase: Categoría	
Responsabilidad	**Colaborador**
Almacena información de la categoría	Puntos categoría
Define el puntaje	
Define las edades de los caballos a participar en cada una	

Figura 3.2: Tarjeta CRC - Categoría
Realizado por: Luis Lima

Clase: Puntos_categoría	
Responsabilidad	**Colaborador**
Define los puntos a ganar por ubicación	Categoría

Figura 3.3: Tarjeta CRC - Puntos categoría
Realizado por: Luis Lima

Clase: Persona	
Responsabilidad	**Colaborador**
Define información de criadores	
Define información de propietarios	

Figura 3.4: Tarjeta CRC - Persona
Realizado por: Luis Lima

Clase: Caballo	
Responsabilidad	**Colaborador**
Define información de caballos	
Registra información del criador	Persona
Registra información del propietario	Persona

Figura 3.5: Tarjeta CRC - Caballo
Realizado por: Luis Lima

Clase: Inscripción	
Responsabilidad	**Colaborador**
Entrega información del caballo inscrito	Caballo
Determina la categoría en la que se inscribe el caballo	Categoría Caballo
Asigna número de inscripción	

Figura 3.6: Tarjeta CRC – Inscripción
Realizado por: Luis Lima

Clase: Calificación	
Responsabilidad	**Colaborador**
Determina categoría a calificar por concurso	Concurso Categoría Caballo
Entrega información de caballos inscritos	Inscripción
Registra ubicación de cada caballo	

Figura 3.7: Tarjeta CRC - Calificación

Realizado por: Luis Lima

Clase: Resultado	
Responsabilidad	**Colaborador**
Consulta información de la calificación en cada categoría	Calificación
Registra puntajes totales	
Devuelve información del concurso	Concurso
Cierra el concurso	

Figura 3.8: Tarjeta CRC - Resultado

Realizado por: Luis Lima

Las tarjetas CRC indicadas se determinaron partiendo del escenario macro de la realización de un concurso, las clases resultantes se complementarán con las definidas en cada capa en base a la arquitectura seleccionada.

3.2 ARQUITECTURA

Existen muchas soluciones agradables para el desarrollo Web, sin embargo, la solución más común en la actualidad para este tipo de aplicaciones es la arquitectura en tres capas expuesta en el capítulo I que es la que se adoptará para este sistema. La arquitectura en tres capas que se utilizará está representada en la Figura 3.9.

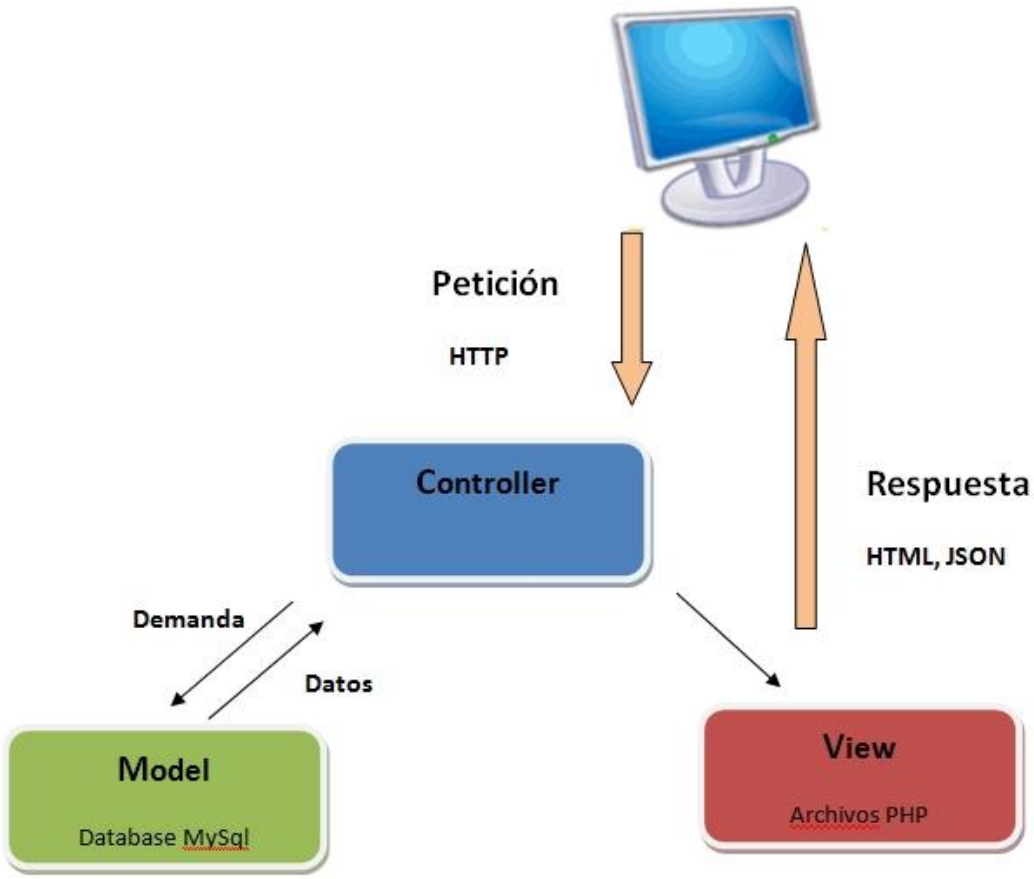


Figura 3. 9: Arquitectura tres capas
Realizado por: Luis Lima

Como se mencionó en el capítulo I, para la implementación de esta arquitectura se utilizara el framework PHP CodeIgniter, el cual organiza las capas establecidas por el patrón MVC de la siguiente manera:

3.2.1 CAPA 1 (MODELS)

Esta capa representa la estructura de datos. Sus clases de modelo contienen funciones que ayudan a recuperar, insertar y actualizar información en la base de datos. CodeIgniter para su distribución física en el directorio models incluye todos los scripts requeridos por el proyecto para trabajar sobre la base de datos.

Los métodos identificados en las clases de esta capa se describen a continuación:

Método	Parámetros	Retorna	Descripción
getSelect		Arreglo de tipos	Listado de todos los tipos existentes
insertar	Instancia de Mtb_tipos_caballo	Void	Recibe una instancia de tipos caballo y almacena en la base de datos
actualizar	Clave primaria y arreglo	Void	Actualiza la base de datos en función de la clave primaria y el arreglo
borrar	Clave primaria	Void	Borra en la base datos en función de la calve primaria
contar		Integer	Realiza el conteo de registros de tipos caballo
obtenerDatos	vsidx::string vsord::string vstart::integer vlimit::integer	Arreglo de tipos caballo	Obtiene todas los tipos_caballos existentes filtrando según parámetros

Tabla 3.1: Métodos del modelo Mtb_tipos_caballo

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
getSelect		Arreglo de categorías	Listado de todas las categorías existentes
getSelectArr		Arreglo de código y nombre	Listado de todas las categorías existentes
insertar	Instancia de Mtb_categorias	Void	Recibe una instancia de categoría y almacena en la base de datos
actualizar	Clave primaria y arreglo	Void	Actualiza la base de datos en función de la clave primaria y el arreglo
borrar	Clave primaria	Void	Borra en la base datos en función de la calve primaria
contar		Integer	Realiza el conteo de registros de categorías
obtenerDatos	vsidx::string vsord::string vstart::integer vlimit::integer vbuscar::string	Arreglo de categorías	Obtiene todas las categorías existentes filtrando según parámetros

Tabla 3.2: Métodos del modelo Mtb_categorias

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
insertar	Instancia de Mtb_puntos_categoria	Void	Recibe una instancia de puntos categoría y almacena en la base de datos
actualizar	Clave primaria y arreglo	Void	Actualiza la base de datos en función de la clave primaria y el arreglo
borrar	Clave primaria	Void	Borra en la base datos en función de la calve primaria
contar	vCategoria::integer	Integer	Realiza el conteo de registros en función de la categoría enviada
obtenerDatos	vCategoria::integer vsidx::string vsord::string vstart::integer vlimit::integer	Arreglo de puntos por categoría	Obtiene todas las categorías existentes filtrando según parámetros

Tabla 3.3: Métodos del modelo Mtb_puntos_categoria

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
getSelect		Arreglo de objetos	Listado de todas los concursos existentes
insertar	Instancia de Mtb_concursos	Void	Recibe una instancia de concursos y almacena en la base de datos
actualizar	Instancia de Mtb_concursos	Void	Recibe una instancia de concursos y almacena en la base de datos
Borrar	Clave primaria	Void	Borra en la base datos en función de la clave primaria
contar		Integer	Realiza el conteo de concursos existentes
obtenerDatos	vsidx::string vsord::string vstart::integer vlimit::integer	Arreglo de concursos	Obtiene todas los concursos existentes filtrando según parámetros
obtenerDatoscc	vConcurso::integer vsidx::string vsord::string vstart::integer vlimit::integer	Arreglo de categorías	Obtiene las categorías habilitadas para un determinado concurso
cerrarConcurso	vConcurso::integer	boolean	Realiza el cierre del concurso con lo cual no se puede realizar ninguna acción posterior

Tabla 3.4: Métodos del modelo Mtb_concursos

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
getSelect		Arreglo de objetos	Listado de todas las categorías existentes
insertar	Instancia de Mtb_categorias_concurso	Void	Recibe una instancia de categorías concurso y almacena en la base de datos
Borrar	Clave primaria	Void	Borra en la base datos en función de la clave primaria

Tabla 3.5: Métodos del modelo Mtb_categorias_concurso

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
getSelect		Arreglo de objetos	Listado de todas las personas existentes
getSelecArr		Arreglo de código y nombre	Lista todas las personas existentes
insertar	Instancia de Mtb_personas	Void	Recibe una instancia de personas y almacena en la base de datos
actualizar	Instancia de Mtb_personas	Void	Recibe una instancia de personas y actualiza en la base de datos
Borrar	Clave primaria	Void	Borra en la base datos en función de la clave primaria
contar		Integer	Realiza el conteo de concursos existentes
obtenerDatos	vsidx::string vsord::string vstart::integer vlimit::integer vBuscar::string	Arreglo de personas	Obtiene todas las personas existentes filtrando según parámetro vBuscar
getSelectRegsitro	registro::integer	Instancia de personas	Obtiene una instancia de personas en función del parámetro registro

Tabla 3.6: Métodos del modelo Mtb_personas

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
getSelect	elSex:char	Arreglo de objetos	Listado de todos los caballos existentes en función del sexo
getSelecArr		Arreglo de código y nombre	Lista todas los caballos existentes
insertar	Instancia de Mtb_personas	Void	Recibe una instancia de Mtb_caballos y almacena en la base de datos
actualizar	Instancia de Mtb_personas	Void	Recibe una instancia de Mtb_caballos y actualiza en la base de datos
Borrar	Clave primaria	Void	Borra en la base datos en función de la clave primaria
contar		Integer	Realiza el conteo de caballos existentes
obtenerDatos	vsidx::string vsord::string vstart::integer vlimit::integer vBuscar::string	Arreglo de Mtb_caballos	Obtiene todas los caballos existentes filtrando según parámetro vBuscar
getSelectRegistro	registro::integer	Instancia de Mtb_caballos	Obtiene una instancia de Mtb_caballos en función del parámetro registro

Tabla 3.7: Métodos del modelo Mtb_caballos

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
getSelect		Arreglo de objetos	Listado de todas los tipos de calificación
insertar	Instancia de Mtb_tipos_calificacion	Void	Recibe una instancia de Mtb_tipos_calificacion y almacena en la base de datos
actualizar	Instancia de Mtb_tipos_calificacion	Void	Recibe una instancia de Mtb_tipos_calificacion y actualiza en la base de datos
Borrar	Clave primaria	Void	Borra en la base datos en función de la clave primaria
contar		Integer	Realiza el conteo de tipos de calificación existentes
registros	vsidx::string vsord::string vstart::integer vlimit::integer	Arreglo de tipos de calificación	Obtiene todas los tipos de calificación existentes

Tabla 3.8: Métodos del modelo Mtb_tipos_calificacion

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
getSelect	vConcurso::integer	Arreglo de objetos	Listado de todas las inscripciones por concurso
insertar	Instancia de Mtb_inscripciones	Void	Recibe una instancia de Mtb_inscripciones y almacena en la base de datos
actualizar	Instancia de Mtb_ inscripciones	Void	Recibe una instancia de Mtb_inscripciones y actualiza en la base de datos
Borrar	Clave primaria	Void	Borra en la base datos en función de la clave primaria
contar	vConcurso::integer	Integer	Realiza el conteo de las inscripciones por concurso
obtenerDatos	vConcurso::integer vsidx::string vsord::string vstart::integer vlimit::integer vBuscar::string	Arreglo de inscripciones	Obtiene todas las inscripciones existentes filtrando según parámetro de concurso y filtros de búsqueda
getSelectEncadenado	registro::integer vConcurso::integer	Instancia de categorías	Obtiene una instancia de mtb_categorias en función del registro y concurso
inscripcionTardia	vConcurso::integer vCategoria::integer	Integer	Realiza la inscripción de participantes en función de los puestos alcanzados en una categoría y concurso, retorna el número de registros realizados.
cerrarInscripciones	vConcurso::intener	boolean	Marcar en la instancia de concurso correspondiente como cerrada la inscripción, retorna mensaje true o fase
calificar	codigoConcurco::intener codigoCategoria	void	Registra la calificación de las categorías en un concurso

Tabla 3.9: Métodos del modelo Mtb_inscripciones

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
porCategoria	vConcurso::integer vCategoria::integer	Arreglo de objetos	Listado de todas las instancias de Mtb_inscripciones por categoria
generales	vConcurso::integer tipoResultadp::integer	Arreglo de objetos	Listado de todas las instancias de Mtb_inscripciones por categoría
getTipor		Arreglo de tipos de resultados	Retorna los tipos de resultados a utilizar en informes
obtenerDetalle	concurso::integer tipoResultado::integer vcodPersona::integer	Arreglo	Listado de las instancias seleccionadas de Mtb_inscripciones

Tabla 3.10: Métodos del modelo Mtb_resultados

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
validarUsuario	usuario::string clave::string	Boolean	Autentica el usuario y la clave ingresadas

Tabla 3.11: Métodos del modelo Mlogin

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
getSelect		Arreglo de objetos	Listado de todas los usuarios existentes
insertar	Instancia de Mtb_usuarios	Void	Recibe una instancia de Mtb_usuarios y almacena en la base de datos
actualizar	Instancia de Mtb_usuarios	Void	Recibe una instancia de Mtb_usuarios y actualiza en la base de datos
Borrar	Clave primaria	Void	Borra en la base datos en función de la clave primaria
contar		Integer	Realiza el conteo de los usuarios existente
obtenerDatos	vsidx::string vsord::string vstart::integer vlimit::integer vBuscar::string	Arreglo de usuarios	Obtiene todas los usuarios existentes en base de las condiciones enviadas.
validarClave	clave1::string clave2::string vUsuario::string	boolean	Valia clave1 y clave2,si son iguales registra nueva clave en las base de datos

Tabla 3.12: Métodos del modelo Mtb_usuarios

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
getSelect		Arreglo de objetos	Listado de todas los roles por usuarios existentes
insertar	Instancia de Mtb_rol_usuarios	Void	Recibe una instancia de Mtb_rol_usuarios y almacena en la base de datos
actualizar	Instancia de Mtb_rol_usuarios	Void	Recibe una instancia de Mtb_rol_usuarios y actualiza en la base de datos
Borrar	Clave primaria	Void	Borra en la base datos en función de la clave primaria
contar	vCedula::string	Integer	Realiza el conteo de roles por usuario identificado por cedula
obtenerDatos	vCedula::string vsidx::string vsord::string vstart::integer vlimit::integer vBuscar::string	Arreglo de usuarios	Obtiene todos los roles por usuario existentes en base de las condiciones enviadas.

Tabla 3.13: Métodos del modelo Mtb_rol_usuarios

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
getSelect		Arreglo de objetos	Listado de todas los accesos registrados
insertar	Instancia de Mtb_log_acceso	Void	Recibe una instancia de Mtb_log_accesos y almacena en la base de datos

Tabla 3.14: Métodos del modelo Mtb_log_accesos

Realizado por: Luis Lima

3.2.2 CAPA 2 (VIEWS)

Esta capa contiene los scripts de los formularios que se presentan a los usuarios, normalmente será una página web o un fragmento de página. En CodeIgniter, en esta capa no se crean clases de ningún tipo, se crean archivos HTML dinámicos generados con PHP que interactúan con el usuario.

Las vistas principales en esta capa se describen en la Tabla 3.15.

Vista	Descripción	Controlador y Método relacionado
caballos_v	Pantalla para la administración de caballos	caballos_c/obtenerDatos caballos_c/getSelect personas_c/getSelect tipos_caballo_c/getSelect caballos_c/guardarDatos
categoria_v	Pantalla para la administración de las categorías	categoria_c/obtenerDatos tipos_caballo_c/getSelect tipos_calificacion/getSelect categoria_c/guardarDatos puntos_c/obtenerDatos puntos_c/guardarDatos
		puntos_c/guardarDatos
concursos_v	Pantalla para la administración de los concursos	concursos_c/obtenerDatos concursos_c/guardarDatos concursos_c/obtenerDatoscc concursos_c/guardarDatoscc
personas_v	Vista para la administración de la base de personas	personas_c/obtenerDatos personas_c/guardarDatos
tipos_caballo_v	Pantalla para reistrar los tipos de caballos existentes	tipos_caballo_c/obtenerDatos tipos_caballo_c/guardarDatos
usuarios_v	Pantalla para la administración de usuarios	usuarios_c/obtenerDatos usuarios_c/guardarDatos rol_usuario_c/obtenerDatos rol_usuario_c/guardarDatos roles_c/getSelect
ccaballo_v	Pantalla para la consulta de caballos	caballos_c/obtenerDatos caballos_c/getSelect personas_c/getSelect tipos_caballo_c/getSelect
cpersonas_v	Pantalla para la consulta de personas	personas_c/obtenerDatos
inscripciones_v	Formulario para la inscripción de caballos en un determinado concurso	concursos_c/obtenerDatos inscripciones_c/obtenerDatos inscripciones_c/guardarDatos inscripciones_c/getSelectEncadenado caballos_c/getSelectRegistro
cerrari_v	Pantalla para realizar el cierre de inscripciones	cerrari_c/procesarDatos
calificacion_v	Formulario que permite registrar las ubicaciones de los caballos en cada categoría	calificacion_c/obtenerDatos calificacion_c/guardarDatos inscripciones_c/inscripcionTardia
calculo_v	Formulario que permite realizar el cierre del concurso y el computo de los puntos y ubicaciones	calculo_c/procesarDatos
resultados_xc_v	Formulario que permite generarl el reporte de resultados por categoría	resultados_xc_c/obtenerDatos
consultas_res_v	Formulario que permite generar el reporte degenerar de resultados por tipos de calificación	consultas_res_c/obtenerDatos consultas_res_c/obtenerDetalle consultas_res_c/reporteMejorCriador

		consultas_res_c/reporteMejorExpositor consultas_res_c/reporteMejorReproductor consultas_res_c/reporteDetalle
reporte_ic_v	Formulario que permite generar el reporte de inscritos por categoría	reporte_ic_c/obtenerDatos
reporte_ti_v	Formulario que permite generar el reporte total de caballos inscritos	reporte_ti_c/obtenerDatos
perfil_v	Formulario para el cambio de clave de un usuario ingresado	perfil_c/index

Tabla 3.15: Vistas de Capa Views
Realizado por: Luis Lima

3.2.3 CAPA 3 (CONTROLLERS)

El controlador actúa como un intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para procesar la petición HTTP y generar una página Web. Contendrá los scripts mediante los cuales se gestionará las vistas y los modelos requeridos.

Los métodos identificados en las clases de esta capa se describen a continuación:

Método	Parámetros	Retorna	Descripción
index			Carga la vista categoria_v
obtenerDatos	page::integer limit::integer sidx::string sord::string	Arreglo de objetos JQGRID::json	Consulta el modelo Mtb_categorias y retorna en formato json para jqgrid
guardarDatos	oper::string		Carga el modelo Mtb_categorias y ejecuta los métodos:insertar, actualizar y borrar dependiendo del parámetro oper.
getSelect		String en formato HTML	Carga el modelo Mtb_categorias y ejecuta el método getSelect. El resultado se utiliza en select HTML.

Tabla 3.16: Métodos del controlador Categoria_c
Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
index		void	Carga la vista concursos_v
obtenerDatos	page::integer limit::integer sidx::string sord::string	Arreglo de objetos JQGRID::json	Consulta el modelo Mtb_concursos y retorna en formato json para jqgrid
guardarDatos	oper::string	void	Carga el modelo Mtb_ concursos y ejecuta los métodos:insertar, actualizar y borrar dependiendo del parámetro oper.
obtenerDatoscc	vConcurso::integer page::integer limit::integer sidx::string sord::string	Arreglo de objetos JQGRID::json	Consulta el modelo Mtb_concursos y su método obtenerDatoss y retorna en formato json para jqgrid
guardarDatoscc	vConcurso::integer oper::string	void	Carga el modelo mtb_categorias_concurso y ejecuta los métodos insertar o borrar dependiendo del parámetro oper.

Tabla 3.17: Métodos del controlador Concursos_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
index		void	Carga la vista tipos_caballo_v
obtenerDatos	page::integer limit::integer sidx::string sord::string	Arreglo de objetos JQGRID::json	Consulta el modelo Mtb_tipos_caballo y retorna en formato json para jqgrid
guardarDatos	oper::string	void	Carga el modelo Mtb_ tipos_caballo y ejecuta los métodos:insertar, actualizar y borrar dependiendo del parámetro oper.
getSelect		String en formato HTML	Carga el modelo Mtb_tipos_caballo y ejecuta el método getSelect. El resultado se utiliza en select HTML.

Tabla 3.18: Métodos del controlador Tipos_caballo_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
index	vTipo::integer	void	Carga la vista personas_v o cpersonas_v dependiente de vTipo
obtenerDatos	page::integer limit::integer sidx::string sord::string	Arreglo de objetos JQGRID::json	Consulta el modelo Mtb_personas y retorna en formato json para jqgrid
guardarDatos	oper::string	void	Carga el modelo Mtb_ personas y ejecuta los métodos:insertar, actualizar y borrar dependiendo del parámetro oper.
getSelect		String en formato HTML	Carga el modelo Mtb_ personas y ejecuta el método getSelect. El resultado se utiliza en select HTML.
getSelectRegistro	Criador::integer	String en formato HTML	Carga el modelo Mtb_ personas y ejecuta el método getSelectRegistro. El resultado se utiliza en option value HTML.

Tabla 3.19: Métodos del controlador Personas_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
index	vTipo::integer	void	Carga la vista caballos_v o ccaballos_v dependiente de vTipo
obtenerDatos	page::integer limit::integer sidx::string sord::string	Arreglo de objetos JQGRID::json	Consulta el modelo Mtb_caballos y retorna en formato json para jqgrid
guardarDatos	oper::string	void	Carga el modelo Mtb_ caballos y ejecuta los métodos:insertar, actualizar y borrar dependiendo del parámetro oper.
getSelect	elSex::char	String en formato HTML	Carga el modelo Mtb_ caballos y ejecuta el método getSelect. El resultado se utiliza en select HTML.
getSelectRegistro	registro::string	String en formato HTML	Carga el modelo Mtb_ caballos y ejecuta el método getSelectRegistro. El resultado se utiliza en option value HTML.

Tabla 3.20: Métodos del controlador Caballos_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
index		void	Carga la vista inscripciones_v
obtenerDatos	page::integer limit::integer sidx::string sord::string	Arreglo de objetos JQGRID::json	Consulta el modelo Mtb_inscripciones y retorna en formato json para jqgrid
guardarDatos	oper::string	void	Carga el modelo Mtb_ inscripciones y ejecuta los métodos:insertar, actualizar y borrar dependiendo del parámetro oper.
getSelectEncadenado	registro::string elConcurso::integer	Arreglo de objetos JQGRID::json	Carga el modelo Mtb_ inscripciones y Mtb_concursos y ejecuta el método getSelectEncadenado. El resultado es en formato json.
inscripcionTardia	vConcurso::integer vCategoria::integer	Arreglo de objetos JQGRID::json	Carga los modelos Mtb_concursos, Mtb_categorias , Mtb_inscripciones con su método insripcionTardia, retorna mensaje en formato json.

Tabla 3.21: Métodos del controlador Inscripciones_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
index	concurso::integer		Consulta el modelo Mtb_concursos y carga la vista cerrari_v
procesarDatos	concurso::integer	Arreglo de objetos JQGRID::json	Consulta el modelo Mtb_inscripciones y ejecuta el método cerrarInscricpciones , retorna mensaje en formato json.

Tabla 3.22: Métodos del controlador Cerrari_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
index		void	Consulta los modelos Mtb_concursos y Mtb_categorias y carga la vista calificacion_v
obtenerDatos	codigoConcruso::integer codigoCategoria::integer page::integer limit::integer sidx::string sord::string	Arreglo de objetos JQGRID::json	Consulta el modelo Mtb_inscripciones y retorna en formato json para jqgrid
guardarDatos	oper::string codigoCategoria::integer puesto::integer mencion::char	void	Carga el modelo Mtb_ inscripciones y ejecuta el método calificar

Tabla 3.23: Métodos del controlador Calificacion_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
index		void	Consulta los modelos Mtb_concursos carga la vista calculo_v
procesarDatos	vConcurso::integer	Arreglo JQGRID::json	Carga el modelo Mtb_ concursos y ejecuta el método cerrarConcurso, retorna mensaje en formato json

Tabla 3.24: Métodos del controlador Calculo_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
index		void	Consulta los modelos Mtb_concursos y Mtb_resulatdos y carga la vista Consultas_res_c
obtenerDatos	concurso::integer tipoResultado::integer page::integer limit::integer sidx::string sord::string	Arreglo de objetos JQGRID::json	Carga el modelo Mtb_ resultados y ejecuta el método generales, retorna en formato json para jqgrid
reporteMejorCriador	vConcurso::integer	void	Carga el modelo Mtb_ resultados y ejecuta el método generales, genera un archivo pdf.
reporteMejorExpositor	vConcurso::integer	void	Carga el modelo Mtb_ resultados y ejecuta el método generales, genera un archivo pdf.
reporteMejorReprodutor	vConcurso::integer	void	Carga el modelo Mtb_ resultados y ejecuta el método generales, genera un archivo pdf.
obtenerDetalle	concurso::integer tipoResultado::integer vCodigo::integer page::integer limit::integer sidx::string sord::string	Arreglo de objetos JQGRID::json	Carga el modelo Mtb_ resultados y ejecuta el método obtenerDetalle, retorna en formato json para jqgrid
reporteDetalle	concurso::integer tipoResultado::integer vcodPersona::integer	void	Carga el modelo Mtb_ resultados y ejecuta el método obtenerDetalle, genera un archivo pdf.

Tabla 3.25: Métodos del controlador Consultas_res_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
index		void	Consulta los modelos Mtb_concursos carga la vista resultados_cc_v
obtenerDatos	concurso::integer	void	Carga el modelo Mtb_ inscripciones y ejecuta el método getSelect. Genera archivo en formato pdf.

Tabla 3.26: Métodos del controlador Reporte_ic_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
index		void	Consulta los modelos Mtb_concursos carga la vista resultados_cc_v
obtenerDatos	concurso::integer	void	Carga el modelo Mtb_ inscripciones y ejecuta el método getSelect. Genera archivo en formato pdf.

Tabla 3.27: Métodos del controlador Reporte_ti_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
index		void	Consulta los modelos Mtb_concursos carga la vista usuarios_v
obtenerDatos	page::integer limit::integer sidx::string sord::string	Arreglo de objetos JQGRID::json	Consulta el modelo Mtb_usuarios y retorna en formato json para jqgrid
guardarDatos	oper::string	void	Carga el modelo Mtb_ usuarios y ejecuta los métodos:insertar, actualizar y borrar dependiendo del parámetro oper.

Tabla 3.28: Métodos del controlador Usuarios_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
obtenerDatos	page::integer limit::integer sidx::string sord::string vCedula:string	Arreglo de objetos JQGRID::json	Consulta el modelo Mtb_rol_usuarios y retorna en formato json para jqgrid
guardarDatos	oper::string	void	Carga el modelo Mtb_ rol_usuairos y ejecuta los métodos:insertar, actualizar y borrar dependiendo del parámetro oper.
getSelect		String en formato HTML	Carga el modelo Mtb_ rol_usuarios y ejecuta el método getSelect. El resultado se utiliza en select HTML.

Tabla 3.29: Métodos del controlador Rol_usuario_c

Realizado por: Luis Lima

Método	Parámetros	Retorna	Descripción
Index		Void	Carga la librería form_validation y el formulario perfil_v
validarClave	nuevaClave::string nuevaClave2::string cuentaUsuario::strting	String	Carga el modelo Mtb_usuarios con el que verifica el cambio de clave y retorna mensaje de cambio exitoso o no.

Tabla 3.30: Métodos del controlador Perfil_c

Realizado por: Luis Lima

Las clases descritas en la capa 1 y 3 se representan de manera gráfica en las figuras 3.10 y 3.11 respectivamente.

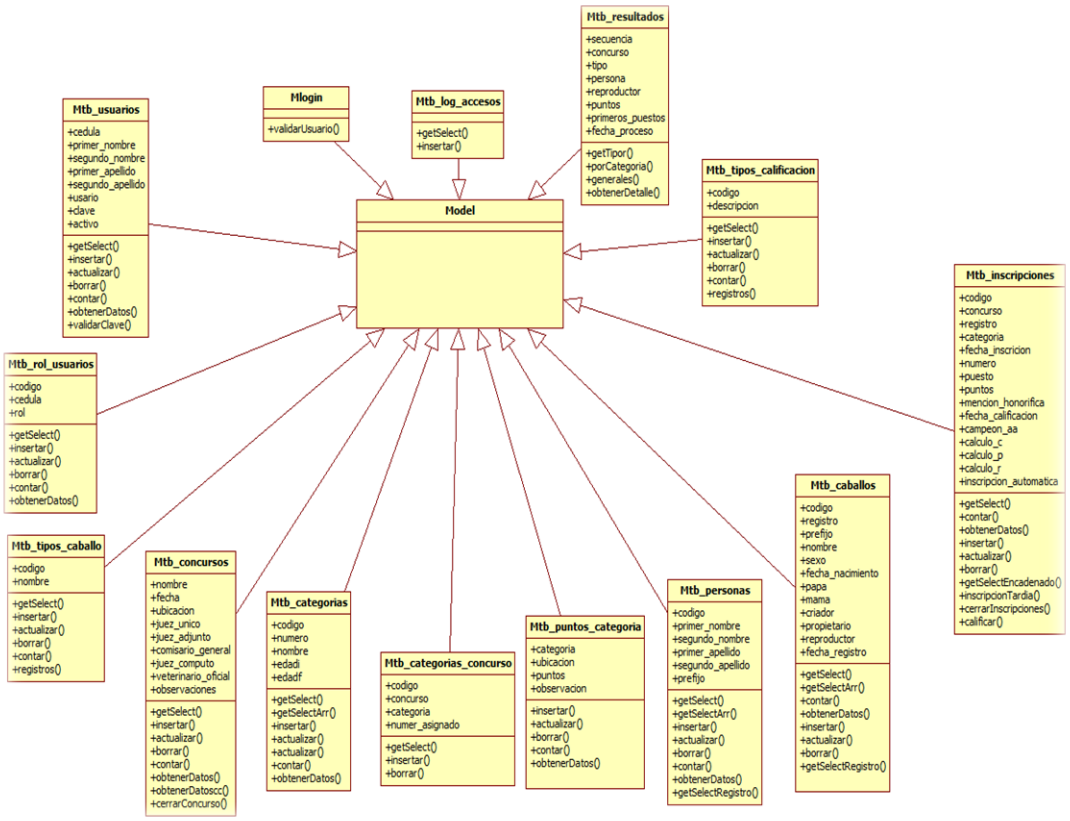


Figura 3.10: Diagrama de Clases de la capa Models

Realizado por: Luis Lima

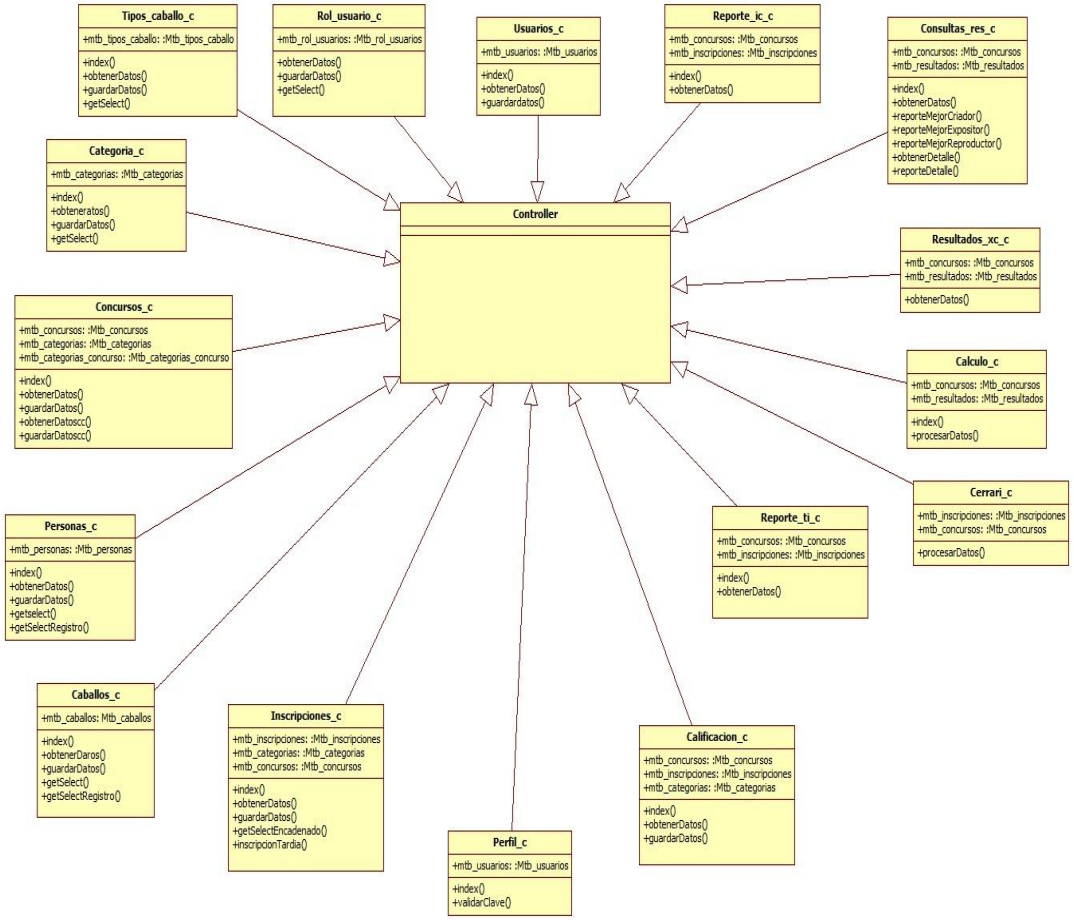


Figura 3.11: Diagrama de Clases de la capa Controller
Realizado por: Luis Lima

Como se puede apreciar todas las clases representadas son **extends**[5] (herencia) de la clase Model o Controller según correspondan, esto es una característica del patrón de diseño MVC.

3.3 MODELO DE DATOS

En base al diagrama de clases de la capa Models (Figura 3.10) descrito anteriormente se podría tener la base suficiente para el modelado de datos, sin embargo, con la finalidad de complementar la documentación se detallará el diagrama Entidad – Relación (ER). La utilización del diagrama ER no está restringido por la metodología XP.

[5] Palabra clave para aplicar la herencia de clases

3.3.1 MODELO ENTIDAD – RELACION

Los modelos Entidad – Relación (ER) son una técnica muy importante para el diseño de bases de datos, en ellos se identifican los datos más importantes llamados entidades y las relaciones entre ellas.

Los diagramas Entidad – Relación en el presente proyecto se los ha dividido en dos: el primer diagrama se enfoca en el modulo de seguridad del sistema y el segundo en los módulos que gestionan los concursos tal cual se indica en las siguientes figuras:

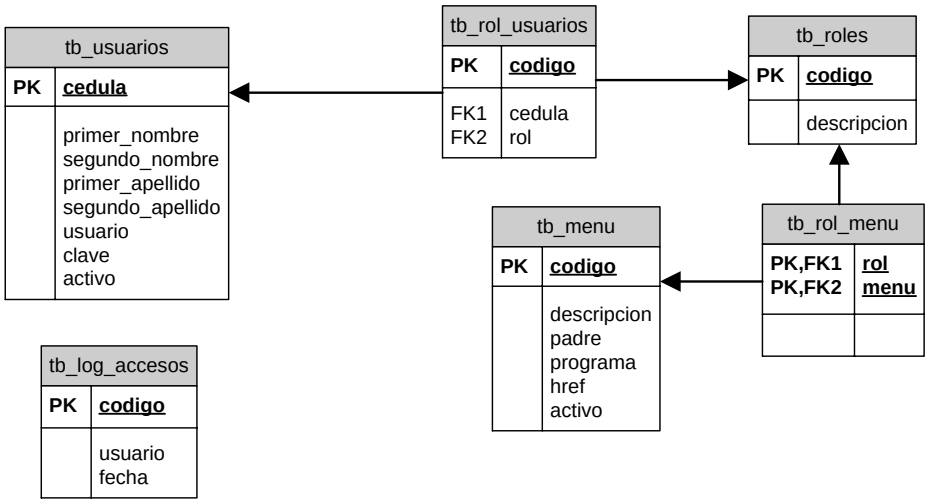


Figura 3.12: Diagrama ER – Módulo de Seguridad

Realizado por: Luis Lima

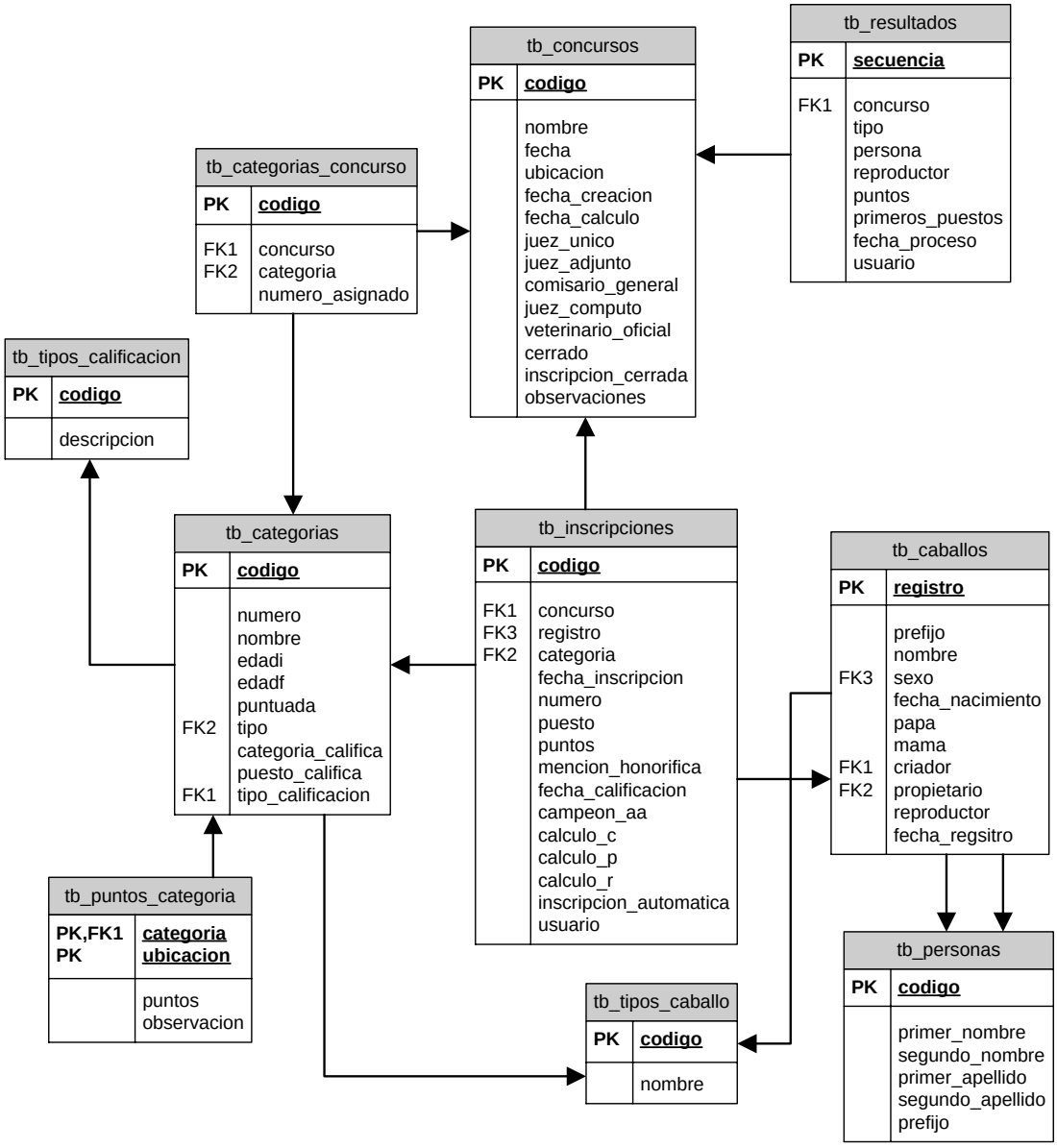


Figura 3.13: Diagrama ER – Gestión de Concursos
Realizado por: Luis Lima

3.3.2 DICCIONARIO DE DATOS

A continuación se realiza una descripción detallada de los datos que se utilizarán en el sistema para lo cual se toma de base los diagramas ER definidos anteriormente.

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CEDULA	Cédula del usuario	VARCHAR(10)	PK NOT NULL	Ingresado por el usuario
PRIMER_NOMBRE	Primer nombre del usuario	VARCHAR(45)	NULL	Ingresado por el usuario
SEGUNDO_NOMBRE	Segundo nombre del usuario	VARCHAR(45)	NULL	Ingresado por el usuario
PRIMER_APELLIDO	Primer apellido del usuario	VARCHAR(45)	NULL	Ingresado por el usuario
SEGUNDO_APELLIDO	Segundo apellido del usuario	VARCHAR(45)	NULL	Ingresado por el usuario
USUARIO	Usuario asignado	VARCHAR(45)	NULL	Ingresado por el usuario
CLAVE	Clave asignada	VARCHAR(45)	NULL	Ingresado por el usuario
ACTIVO	Estado del usuario	CHAR(1)	NOT NULL	Ingresado por el usuario

Figura 3.14: Estructura tabla tb_usuarios

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CODIGO	Código secuencial	NUMBER(9)	PK NOT NULL	Generado por el sistema
DESCRIPCION	Nombre del rol	VARCHAR(30)	NOT NULL	Ingresado por el usuario

Figura 3.15: Estructura tabla tb_roles

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CODIGO	Código secuencial	NUMBER(9)	PK NOT NULL	Generado por el sistema
CEDULA	Cédula del usuario	VARCHAR(10)	FK NOT NULL	Ingresado por el usuario
ROL	Rol asignado	NUMBER(9)	FK NOT NULL	Ingresado por el usuario

Figura 3.16: Estructura tabla tb_rol_usuarios

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CODIGO	Código secuencial	NUMBER(10)	PK NOT NULL	Generado por el sistema
DESCRIPCION	Nombre de la opción	VARCHAR(60)	NOT NULL	Ingresado por el usuario
PADRE	Menú padre de cada opción	VARCHAR(50)	NULL	Ingresado por el usuario
PROGRAMA	Programa relacionado con la opción	VARCHAR(120)	NULL	Ingresado por el usuario
HREF	Nivel para generar el menú en html	VARCHAR(10)	NULL	Ingresado por el usuario
ACTIVO	Opción de menú activo o no	CHAR(1)	DEFAULT 'S'	Ingresado por el usuario

Figura 3.17: Estructura tabla tb_menu

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
ROL	Código del rol	NUMBER(9)	PK FK NOT NULL	Generado por el sistema
MENU	Código del menú	NUMBER(9)	Pk FK NOT NULL	Ingresado por el usuario

Figura 3.18: Estructura tabla tb_rol_menu

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CODIGO	Código del tipo de caballo	CHAR(1)	PK NOT NULL	Ingresado por el usuario
NOMBRE	Nombre del tipo de caballo	VARCHAR(45)	NOT NULL	Ingresado por el usuario

Figura 3.19: Estructura tabla tb_tipos_caballo

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CODIGO	Código secuencial	NUMBER(12)	PK NOT NULL	Generado por el sistema
USUARIO	Usuario que accede al sistema	VARCHAR(45)	NOT NULL	Generado por el sistema
FECHA	Fecha en la que se accede al sistema	DATE	NOT NULL	Generado por el sistema

Figura 3.20: Estructura tabla tb_log_accesos

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CODIGO	Código secuencial	NUMBER(10)	PK NOT NULL	Generado por el sistema
NOMBRE	Nombre del concurso	VARCHAR(100)	NOT NULL	Ingresado por el usuario
FECHA	Fecha descriptiva	VARCHAR(50)	NOT NULL	Ingresado por el usuario
UBICACION	Descripción del lugar del concurso	VARCHAR(60)	NOT NULL	Ingresado por el usuario
FECHA_CREACION	Fecha en la que se crear el registro	DATE	NOT NULL	Generado por el sistema
FECHA_CALCULO	Fecha de cálculo de edades	DATE	NOT NULL	Ingresado por el usuario
JUEZ_UNICO	Nombre del juez único	VARCHAR(50)	NULL	Ingresado por el usuario
JUEZ_ADJUNTO	Nombre del juez adjunto	VARCHAR(50)	NULL	Ingresado por el usuario
COMISARIO_GENERAL	Nombre del comisario general	VARCHAR(45)	NULL	Ingresado por el usuario
JUEZ_COMPUTO	Nombre del juez de cómputo	VARCHAR(45)	NULL	Ingresado por el usuario
VETERINARIO_OFICIAL	Nombre del veterinario oficial	VARCHAR(45)	NULL	Ingresado por el usuario
CERRADO	Concurso cerrado o no	CHAR(1)	DEFAULT 'N'	Generado por el sistema
INSCRIPCION_CERRADA	Inscripción cerrada o no	CHAR(1)	DEFAULT 'N'	Generado por el sistema
OBSERVACIONES		VARCHAR(200)	NULL	Ingresado por el usuario

Figura 3.21: Estructura tabla tb_concursos

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CODIGO	Código del tipo de calificación	NUMBER(3)	PK NOT NULL	Ingresado por el usuario
DESCRIPCION	Descripción del tipo de calificación	VARCHAR(30)	NOT NULL	Ingresado por el usuario

Figura 3.22: Estructura tabla tb_tipos_calificacion

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CODIGO	Código secuencial	NUMBER(10)	PK NOT NULL	Generado por el sistema
NUMERO	Numero asignado a la categoría	NUMBER(10)	NOT NULL	Ingresado por el usuario
NOMBRE	Nombre de la categoría	VARCHAR(60)	NOT NULL	Ingresado por el usuario
EDADI	Edad inicial	NUMBER(3)	DEFAULT 0	Ingresado por el usuario
EDADF	Edad final	NUMBER(3)	DEFAULT 0	Ingresado por el usuario
PUNTUADA	Otorga puntos	CHAR(1)	DEFAULT 'S'	Ingresado por el usuario
TIPO	Tipo de caballo	CHAR(1)	FK NOT NULL	Ingresado por el usuario
CATEGORIA_CALIFICA	Códigos de categorías que califican	VARCHAR(45)	NULL	Ingresado por el usuario
PUESTO_CALIFICA	Puestos que participan de las categorías de califican	VARCHAR(45)	NULL	Ingresado por el usuario
TIPO_CALIFICACION	Código del tipo de calificación	NUMBER(10)	FK NOT NULL	Ingresado por el usuario

Figura 3.23: Estructura tabla tb_categorias

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CATEGORIA	Código de categoría	NUMBER(10)	PK FK NOT NULL	Generado por el sistema
UBICACION	Número de la ubicación	NUMBER(10)	PK NOT NULL	Ingresado por el usuario
PUNTOS	Número de puntos otorgados	NUMBER(11)	NOT NULL	Ingresado por el usuario
OBSERVACION	Comentarios o recordatorios de los puntos otorgados	VARCHAR(60)	NULL	Ingresado por el usuario

Figura 3.24: Estructura tabla tb_puntos_categoria

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CODIGO	Código secuencial	NUMBER(12)	PK NOT NULL	Generado por el sistema
CONCURSO	Código de concurso	NUMBER(10)	FK NOT NULL	Generado por el sistema
CATEGORIA	Código de categoría	NUMBER(10)	FK NOT NULL	Generado por el sistema
NUMERO_ASIGNADO	Numero asignado a la categoría en l concurso	NUMBER(10)	NULL	Ingresado por el usuario

Figura 3.25: Estructura tabla tb_ categorias_concurso

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
REGISTRO	Código del caballo	VARCHAR(9)	PK NOT NULL	Ingresado por el usuario
PREFIJO	Prefijo asignado a cada caballo	VARCHAR(10)	NULL	Ingresado por el usuario
NOMBRE	Nombre del caballo	VARCHAR(45)	NOT NULL	Ingresado por el usuario
SEXO	Código del tipo de caballo	CHAR(1)	FK NOT NULL	Ingresado por el usuario
FECHA_NACIMIENTO	Fecha de nacimiento del caballo	DATE	NOT NULL	Ingresado por el usuario
PAPA	Registro del padre del caballo	VARCHAR(20)	NOT NULL	Ingresado por el usuario
MAMA	Registro de la madre del caballo	VARCHAR(20)	NOT NULL	Ingresado por el usuario
CRIADOR	Código de persona criador	NUMBER(11)	FK NULL	Ingresado por el usuario
PROPIETARIO	Código de persona propietario	NUMBER(11)	FK NULL	Ingresado por el usuario
REPRODUCTOR	Reproductor Si/No	CHAR(1)	DEFAULT 'N'	Ingresado por el usuario
FECHA_REGISTRO	Fecha de ingreso al sistema	DATE	NOT NULL	Generado por el sistema

Figura 3.26: Estructura tabla tb_ caballos

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CODIGO	Código secuencial	NUMBER(10)	PK NOT NULL	Generado por el sistema
PRIMER_NOMBRE	Primer nombre de la persona	VARCHAR(45)	NOT NULL	Ingresado por el usuario
SEGUNDO_NOMBRE	Segundo nombre de la persona	VARCHAR(45)	NULL	Ingresado por el usuario
PRIMER_APELLIDO	Primer apellido de la persona	VARCHAR(45)	NOT NULL	Ingresado por el usuario
SEGUNDO_APELLIDO	Segundo apellido de la persona	VARCHAR(45)	NULL	Ingresado por el usuario
PREFIJO	Prefijo asignado	VARCHAR(4)	NULL	Ingresado por el usuario

Figura 3.27: Estructura tabla tb_ personas

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
SECUENCIA	Código secuencial	NUMBER(10)	PK NOT NULL	Generado por el sistema
CONCURSO	Código de concurso	NUMBER(10)	FK NOT NULL	Ingresado por el usuario
TIPO	Tipo de computo	NUMBER(2)	NOT NULL	Ingresado por el usuario
PERSONA	Código de persona	NUMBER(10)	NULL	Generado por el sistema
REPRODUCTOR	Código de caballo	VARCHAR(9)	NULL	Generado por el sistema
PUNTOS	Total puntos	NUMBER(5)	NOT NULL	Generado por el sistema
PRIMEROS_PUESTOS	Total de primeros puestos	NUMBER(5)	NOT NULL	Generado por el sistema
FECHA_PROCESO	Fecha de ejecución del proceso	DATE	NOT NULL	Generado por el sistema
USUARIO	Usuario que realiza el cómputo	VARCHAR(45)	NOT NULL	Generado por el sistema

Figura 3.28: Estructura tabla tb_ resultados

Realizado por: Luis Lima

Campo	Descripción	Tipo de Datos	Tipo de Restricción	Procedencia
CODIGO	Código secuencial	NUMBER(10)	PK NOT NULL	Generado por el sistema
CONCURSO	Código de concurso	NUMBER(10)	FK NOT NULL	Ingresado por el usuario
REGISTRO	Código del caballo inscrito	VARCHAR(9)	FK NOT NULL	Ingresado por el usuario
CATEGORIA	Código de categoría	NUMBER(10)	FK NOT NULL	Ingresado por el usuario
FECHA_INSCRIPCION	Fecha de inscripción	DATE	NOT NULL	Generado por el sistema
NUMERO	Número de inscripción	NUMBER(4)	DEFAULT 0	Generado por el sistema
PUESTO	Ubicación del caballo en la categoría	NUMBER(10)	NOT NULL	Ingresado por el usuario
PUNTOS	Puntos obtenidos en función de la ubicación	NUMBER(5)	DEFAULT 0	Generado por el sistema
MENCION_HONORIFICA	Mención honorifica si o no	CHAR(1)	DEFAULT 'N'	Ingresado por el usuario
FECHA_CALIFICACION	Fecha en la que se realiza la calificación	DATE	NOT NULL	Generado por el sistema

CAMPEON_AA	Campeón año anterior Si o No	CHAR(1)	DEFAULT 'N'	Ingresado por el usuario
CALCULO_C	Calculo para mejor criador Si o No	CHAR(1)	DEFAULT 'N'	Ingresado por el usuario
CALCULO_P	Calculo para mejor propietario Si o No	CHAR(1)	DEFAULT 'N'	Ingresado por el usuario
CALCULO_R	Calculo para mejor reproductor Si o No	CHAR(1)	DEFAULT 'N'	Ingresado por el usuario
INSCRIPCION_AUTOMATICA	Inscripción automática Si o No	CHAR(1)	DEFAULT 'N'	Generado por el sistema
USUARIO	Usuario que realiza la inscripción	VARCHAR(45)	NOT NULL	Generado por el sistema

Figura 3.29: Estructura tabla tb_inscripciones
Realizado por: Luis Lima

3.4 DISEÑO Y MODELADO DE INTERFACES

Puesto que los formularios son la unidad básica de una aplicación Web, es importante realizar algunas consideraciones sobre su función y su diseño. El enlace de datos en formularios proporciona los medios necesarios para mostrar y modificar la información procedente de un origen de datos en los controles de un formulario. Puede enlazar tanto con orígenes de datos tradicionales como con casi cualquier estructura que contenga datos.

En CodeIgniter se utiliza formularios HTML generados con PHP los cuales son coordinados por los controladores para establecer el enlace con las fuentes de datos. Estos formularios son complementados y enriquecidos con grillas construidas con JQGRID[6] y Java Script.

A continuación se detalla la estructura básica de las interfaces, formularios así como el patrón de diseño que mantiene:

3.4.1 DISEÑO DE INTERFAZ INGRESO AL SISTEMA

Para el ingreso al sistema los usuarios dispondrán de la siguiente interfaz:

[6] JQGRID: Ajax con Java Script que proporciona soluciones para presentar y manipular datos en la Web.

Figura 3.30: Interfaz Ingreso al Sistema
Realizado por: Luis Lima

Por medio de la interfaz que se muestra en la Figura 3.30 los usuarios van a ingresar al sistema a través de un identificador de cada usuario y su clave, los cuales deben ser escritos en las cajas de texto correspondientes.

3.4.2 DISEÑO DE INTERFAZ PANTALLA PRINCIPAL

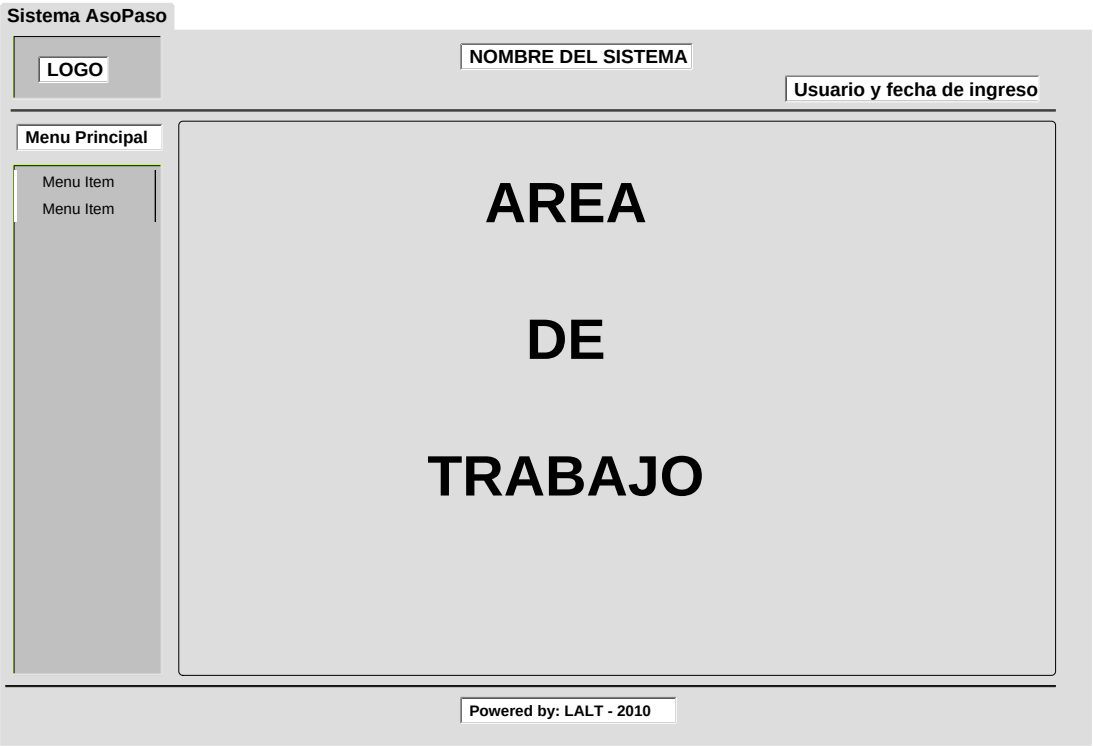


Figura 3.31: Interfaz Pantalla Principal
Realizado por: Luis Lima

En la Figura 3.31 se muestra la pantalla principal del sistema la cual está dividida en las siguientes áreas:

- **Área de cabecera.-** En la cual se muestra: logo del sistema, nombre del sistema y el campo de bienvenida al usuario ingresado con la fecha y hora de registro.
- **Área del menú.-** Es el menú de opciones disponibles.
- **Área de trabajo.-** Es donde se despliega y trabaja la opción activa del sistema.
- **Área de pie de página.-** Se muestra la versión del software y su desarrollador.

3.4.3 DISEÑO DE GRILLAS

Para la mayoría de las interfaces de usuario se utilizará grillas construidas con JQGRID, las cuales tienen la estructura mostrada en la Figura 3.32:

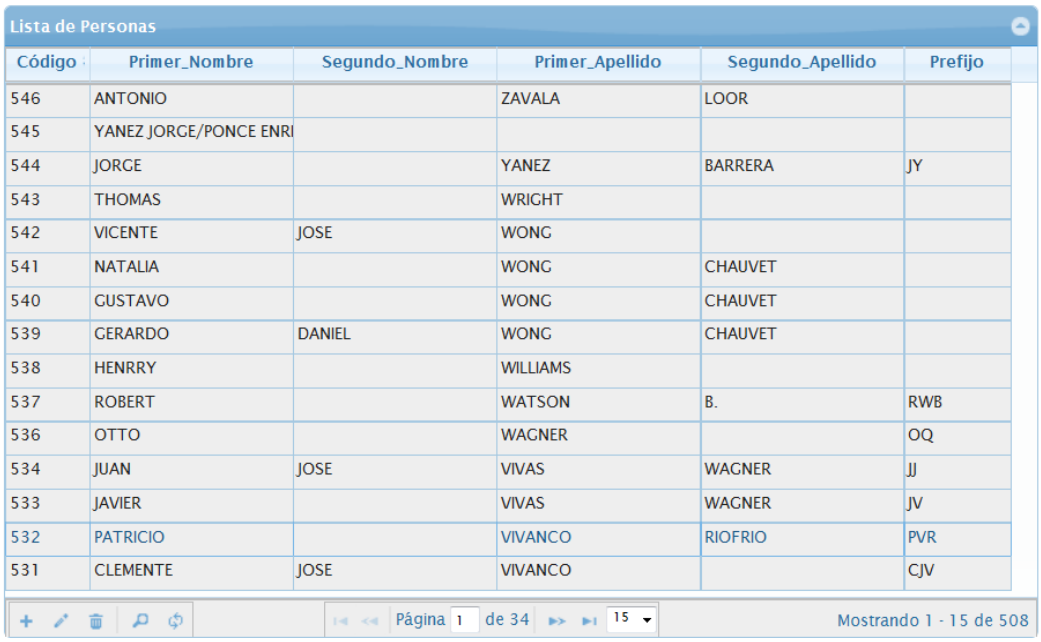


Figura 3.32: Grillas
Realizado por: Luis Lima

La información se muestra de manera tabular y sobre ella se pueden realizar las siguientes acciones en base a los botones disponibles:

- Añadir un registro  .- Al seleccionar este botón se despliega el formulario estándar que permite agregar un registro en la tabla relacionada.
- Modificar un registro  .- Al presionar este botón se despliega un formulario estándar para modificar cualquier campo del registro seleccionado.
- Borrar un registro  .- Al presionar este botón se eliminan el registro seleccionado.
- Buscar  .- Con este botón se despliega un formulario estándar que permite la búsqueda por cualquier patrón de búsqueda sobre los campos desplegados en la grilla.

La grilla representada en la Figura 3.32 es básica, sin embargo, se pueden realizar muchas combinaciones en estructuras maestro detalle de dos y hasta tres niveles.

3.5 MODELO LÓGICO DE LA APLICACIÓN

En base a las estrategias definidas para obtener y administrar la información la aplicación tendrá el modelo lógico representado en la Figura 3.33. En ella se puede apreciar la navegación requerida para ejecutar un proceso en particular. La representada en la figura correspondería a un usuario administrador, para usuarios con otros roles el modelo sería diferente, sin embargo, toman de base el modelo de la figura.

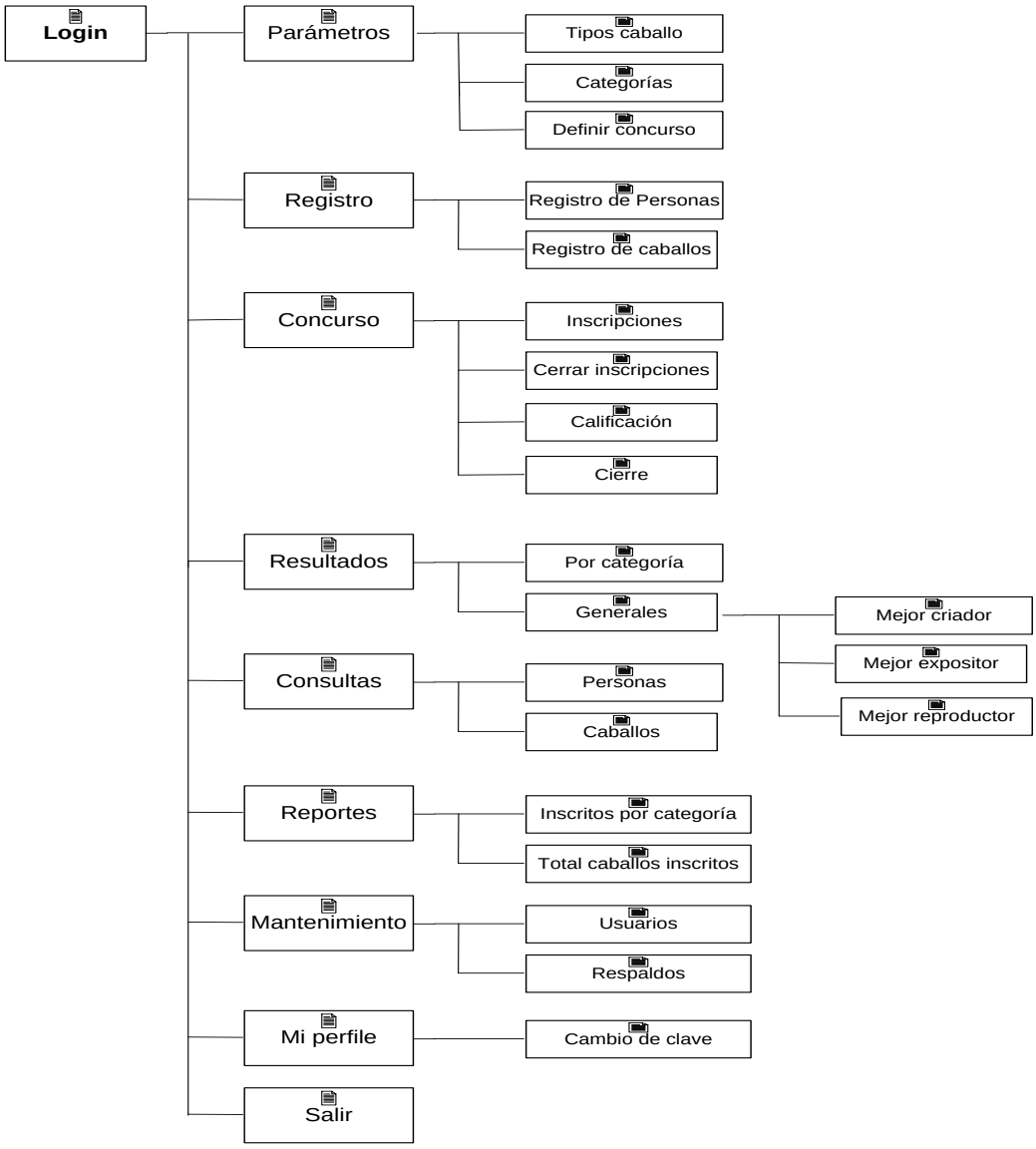


Figura 3.33: Modelo Lógico de la Aplicación
Realizado por: Luis Lima

3.6 RESUMEN

En la metodología XP, este capítulo está relacionado con la fase de diseño. Se realizó la definición de tarjetas CRC en función de las historias de usuario partiendo del escenario macro de la ejecución de un concurso, las clases incluidas en las tarjetas CRC se complementan con los diagramas de clases de la capa 1 y 3 del patrón MVC.

Para cada capa de la arquitectura seleccionada se realiza una descripción detallada de los métodos identificados. CodeIgniter en la capa 2 no crea clases de ningún tipo, únicamente se crean archivos HTML dinámicos generados con PHP, motivo por el cual en la capa 2 tenemos una descripción detallada de las vistas a utilizar y controladores con sus métodos relacionados.

Se utilizan los modelos Entidad – Relación para el diseño de la base de datos, los diagramas realizados son dos: el primero para los temas de seguridad de la aplicación y el segundo para la gestión propia del concurso. Estos diagramas se complementan con el diccionario de datos de todas las tablas involucradas en el proceso.

Para el diseño y modelado de interfaces se inicia con la definición de la interfaz de ingreso al sistema, luego se describe la pantalla principal la cual contempla, entre otras, una área de trabajo sobre la que se desplegará todos los formularios disponibles para los usuarios, los cuales, en la mayoría de las opciones tienen incluidas grillas estándar para la consulta, modificación e inserción de registros.

CAPITULO IV

4. IMPLANTACIÓN DE LA APLICACIÓN

En el presente capitulo se abarcará las fases de codificación y pruebas definidas en la metodología XP, así como los pasos necesarios para la puesta en producción del sistema.

4.1 CODIFICACIÓN

Como se mencionó anteriormente, el cliente es parte integral del equipo de desarrollo y deberá acompañar en los test de unidad que se realizan durante la codificación, sin embargo, dado la complejidad de obtener o construir un software para estas pruebas, en el presente proyecto no se realizaran este tipo de pruebas y básicamente se realizarán las validaciones del programador.

XP de igual forma recomienda la programación en parejas, sin embargo, dado que el presente proyecto está siendo desarrollado por una sola persona no aplicaría.

Para la codificación en PHP, se utilizará el estándar detallado en el Anexo E "PEAR: Estándares de desarrollo para PHP" (DotPress, 2007, p.1). Previo al inicio de la codificación es necesario configurar CodeIgniter, en el Anexo F se detallan los pasos necesarios para la instalación y configuración de este framework.

A continuación se explica cómo se realiza la codificación con CodeIgniter en las tres capas MVC ya explicadas anteriormente.

4.1.1 CREAR UN CONTROLADOR

Se revisará un controlador muy básico de la aplicación en el que se puede apreciar su estructura y codificación tal cual se expone en la Figura 4.1.

```
<?php
class Tipos_caballo_c extends Controller{
	function __construct(){
		parent::Controller();
		//Invoca al modelo
		$this->load->model('mtb_tipos_caballo');
	}

	function index(){
		$datos['contenido_pagina'] = $this->load->view('grilla/tipos_caballo_v', null, true);
		$this->load->view('plantilla', $datos);
	}
```

```
	//Obtiene los datos para la vista
	function obtenerDatos(){
		$page = $this->input->post('page'); // get the requested page
		$limit = $this->input->post('rows'); // get how many rows we want to have into the grid
		$sidx = $this->input->post('sidx'); // get index row - i.e. user click to sort
		$sord = $this->input->post('sord'); // get the direction
		if(!$sidx) $sidx =1; // connect to the database

	   $count = $this->mtb_tipos_caballo->contar();
		if( $count >0 ) {
			$total_pages = ceil($count/$limit);
		}else{
			$total_pages = 0;
		}
		if ($page > $total_pages) $page=$total_pages;
		$start = $limit*$page - $limit;
		$categorias = $this->mtb_tipos_caballo->registros($sidx,$sord,$start,$limit);
		$responce->page = $page;
		$responce->total = $total_pages;
		$responce->records = $count;
		foreach($categorias as $i => $fila){
			$responce->rows[$i]['id']=$fila->CODIGO;
			$responce->rows[$i]['cell']=array($fila->CODIGO,$fila->NOMBRE);
		}
		echo json_encode($responce);
	}

	//Guarda los datos en la BDD
	function guardarDatos(){
		switch($this->input->post('oper')){
			case 'add':
				//insert
				$categoria = array(
					'NOMBRE' => $this->input->post('nombre'),
					'CODIGO' => $this->input->post('codigo')
				);
				$this->mtb_tipos_caballo->insertar($categoria);
				break;
			case 'edit':
				//update
				$categoria = array(
					'NOMBRE' => $this->input->post('nombre'),
					'CODIGO' => $this->input->post('codigo')
				);
				$this->mtb_tipos_caballo->actualizar($this->input->post('id'),$categoria);
				break;
			case 'del':
				//delete
				$this->mtb_tipos_caballo->borrar($this->input->post('id'));
				break;
		}
	}
```

```
        //Selecciona los datos como option value
        function getSelect(){
                $tiposCaballo = $this->mtb_tipos_caballo->getSelect();
                $html = "<select>";
                foreach($tiposCaballo as $i => $fila){
                        $html  .=  '<option  value="'.$fila->CODIGO.'">'.$fila->NOMBRE.'</option>';

                }
                $html .= "</select>";
                print $html;
        }
}
?>
```

Figura 4.1: Controlador tipos_caballo_c.php

Realizado por: Luis Lima

El controlador tipos_caballo_c tiene 5 funciones:

- **function __construct(){}.-** Inicializa el constructor y carga el modelo relacionado.

- **function index(){}.-** Carga la vista relacionada, en este caso es una grilla generada con el archivo php tipos_caballo_v.

- **function obtenerDatos(){}.-** Realiza la consulta a la base de datos, determina el número de páginas y retorna un arreglo tipo json.

- **function guardarDatos(){}.-** Función que realiza la operación de insertar, actualizar o borrar las filas de la tabla relacionada.

- **function getSelect(){}.-** Realiza la consulta de la tabla y retorna un string HTML para ser utilizado como option value en cualquier formulario.

4.1.2 CREAR EL MODELO

En la Figura 4.2 se expone el Modelo utilizado en el controlador descrito anteriormente.

```
<?php
class Mtb_tipos_caballo extends Model{
	function __construct(){
		parent::Model();
	}

	function getSelect(){
		$losRegistros = $this->db->query("SELECT * FROM tb_tipos_caballo ORDER BY nombre")->result();
		return $losRegistros;
	}
	function insertar($losDatos){
		$this->db->insert('tb_tipos_caballo', $losDatos);
	}
	function actualizar($laClave, $losDatos){
		$this->db->where('CODIGO', $laClave);
		$this->db->update('tb_tipos_caballo', $losDatos);
	}
	function borrar($laClave){
		$this->db->delete('tb_tipos_caballo', array('CODIGO' => $laClave));
	}
	function contar(){
		$sql = "SELECT COUNT(*) AS numRegistros from tb_tipos_caballo";
	   $count = $this->db->query($sql)->row()->numRegistros;
	   return $count;
	}
	function registros($vsidx,$vsord,$vstart,$vlimit){
		$SQL = "SELECT  * FROM tb_tipos_caballo ORDER BY $vsidx $vsord LIMIT $vstart , $vlimit";
		$losRegistros = $this->db->query($SQL)->result();
	   return $losRegistros;
	}
}
?>
```

Figura 4.2: Modelo mtb tipos_caballo.php
Realizado por: Luis Lima

El modelo mtb_tipos_caballo tiene 7 funciones:

- **function __construct(){}.-** Inicializa el constructor.
- **function getSelect(){}.-** Función que consulta la tabla relacionada y retorna el resultado.
- **function insertar(){}.-** Función que recibe un arreglo con los datos e inserta en la base de datos.

- **function actualizar(){}.-** Función que recibe un arreglo con los datos y actualiza en la tabla relacionada.

- **function borrar(){}.-** Función que recibe el ID de la fila y elimina los datos.

- **function contar(){}.-** Realiza el conteo del total de registros de la tabla y retorna el valor correspondiente.

- **function registros(){}.-** Función que consulta la tabla relacionada y retorna el resultado. Recibe como parámetros las siguientes variables:

    - **vsidx.-** Campo por el que se ordenará la consulta.
    - **vsord.-** Orden ascendente o descendente.
    - **vstart.-** Límite inicial de la consulta.
    - **vlimit.-** Límite final de la consulta.

4.1.3 CREAR LA VISTA

En la Figura 4.3 se expone la vista creada para interactuar con el controlador que se está revisando.

```
<table id="lista"></table>
<div id="paginador"></div>
<script type="text/javascript">
$(function(){
	$("#lista").jqGrid({
		url: "<?=site_url('tipos_caballo_c/obtenerDatos')?>",
		mtype: "post",
		datatype: "json",
		colNames:['Código','Nombre'],
		colModel:[
			{name:'codigo', index:'codigo', width:55, editable:true},
			{name:'nombre', index:'nombre', width:250, editable:true}
			],
		rowNum: 10,
		rowList: [10, 20, 30],
		pager: '#paginador',
		sortname: 'codigo',
		viewrecords: true,
		sortorder: "desc",
		caption: "Tipo de caballos",
		editurl: "<?=site_url('tipos_caballo_c/guardarDatos')?>"
	}).navGrid('#paginador',{});
});
</script>
```

Figura 4.3: Vista tipos_caballo_v.php

Realizado por: Luis Lima

La vista se cargará en el área de trabajo definida en la plantilla por lo que no es necesario incluir las etiquetas de una página html con su respectivo estilo. La vista tiene las siguientes características:

- Para leer los dato de la tabla relacionada llama al controlador tipos_caballo_c/obtenerDatos.

- Tanto la fila colNames y colModel tienen la correspondencia con los datos de la tabla con sus respectivas reglas. Ej:

  { name:'codigo', index:'codigo', width:55, editable:true}

  Indica: que tanto el **id** y el **index** se llaman código, el ancho del input es 55px y que el campo es editable.

- El valor rowNum:10 determina el número de filas a desplegar, rowList: [10, 20, 30] indica el grupo de registros que se desplegarán en cada página, sortname:"codigo" indica el campo por el que se ordenará la información en este caso por código.

- Finalmente llama al controlador tipos_caballo_c/guardarDatos para que realice la operación respectiva (insertar, actualizar o borrar).

Esta es la forma como se realiza la codificación utilizando CodeIgniter a lo largo del presente proyecto.

4.2 PRUEBAS

Siguiendo las recomendaciones de la metodología de desarrollo seleccionada se utilizará las **pruebas de aceptación** para garantizar que lo codificado se ajusta a lo requerido por el cliente.

Las pruebas de aceptación se realizan al finalizar cada iteración, las cuales están enfocadas básicamente en la funcionalidad del sistema, con ellas se logra determinar de manera ágil, certera y oportuna los errores y cambios a realizar para cumplir satisfactoriamente con las necesidades del cliente.

El formato utilizado para las pruebas de aceptación se encuentra en el Anexo G. A continuación se describen las pruebas desarrolladas para cada iteración.

4.2.1 PRUEBAS DE LA PRIMER ITERACION

PRUEBA DE ACEPTACION	
No. 1	**Historia de usuario: 01 Funcionalidad general**
Descripción: Acceso al sistema a través de cualquier navegador de Internet	
Condiciones de ejecución: • Debe existir el acceso a la página principal de la aplicación. • El usuario debe contar PC con acceso a Internet	
Entrada/Pasos de ejecución: El usuario: • Ingresa url http://localhost/asopaso	
Resultados esperados: • Ingreso exitoso al sistema, se despliega la pantalla principal del sistema.	
Evaluación de la prueba: Satisfactoria	

Figura 4.4: Prueba de aceptación – Funcionalidad general
Realizado por: Luis Lima

PRUEBA DE ACEPTACION	
No. 2	**Historia de usuario: 02 Parámetros generales**
Descripción: El usuario Administrador realiza el mantenimiento de los parámetros generales como son: Tipos de caballos, categorías y concursos.	
Condiciones de ejecución: • Debe existir el usuario Administrador	
Entrada/Pasos de ejecución: El usuario: • Ingresa al menú parámetros y selecciona la opción Tipos caballo. Sobre esta opción realiza operaciones de inserción, actualización y borrado de los registros. • Ingresa al menú parámetros y selecciona la opción Categorías. Sobre esta opción realiza operaciones de inserción, actualización y borrado de los registros. • Ingresa al menú parámetros y selecciona la opción Definir concurso. Sobre esta opción realiza operaciones de inserción, actualización y borrado de los registros. En el formulario de edición se ingresa la información requerida en cada caso y se verifica que sea la adecuada con las validaciones respectivas.	
Resultados esperados: • En los tres casos se refleja operación exitosa. • Se puede observar en los datos consultados las acciones realizadas.	
Evaluación de la prueba: Satisfactoria	

Figura 4.5: Prueba de aceptación – Parámetros generales

Realizado por: Luis Lima

4.2.2 PRUEBAS DE LA SEGUNDA ITERACION

PRUEBA DE ACEPTACION	
No. 3	**Historia de usuario: 03 Registro de personas y caballos**
Descripción: El usuario con rol Registro realiza el mantenimiento de las personas y caballos relacionados con un concurso.	
Condiciones de ejecución: • Debe existir el usuario con el rol Registro. • Debe existir el usuario con el rol Consultas. • Debe existir registrado los tipos de caballos existentes.	
Entrada/Pasos de ejecución: El usuario: • Ingresa al menú Registro y selecciona la opción Registro de personas. Sobre esta opción realiza operaciones de inserción, actualización y borrado de los registros. • Sobre los datos desplegados en pantalla se realiza varias búsquedas por distintos parámetros para lo cual se presiona el botón buscar. • Ingresa al menú Registro y selecciona la opción Registro de caballos. Sobre esta opción realiza operaciones de inserción, actualización y borrado de los registros. • Sobre los datos desplegados en pantalla se realiza varias búsquedas por distintos parámetros para lo cual se presiona el botón buscar. • Ingresa al menú Consultas y selecciona la opción Personas o Caballos. Sobre esta opción realiza operaciones de búsqueda son varios parámetros. En el formulario de edición se ingresa la información requerida en cada caso y se verifica que sea la adecuada con las validaciones respectivas. Se pone especial énfasis en el campo registro que debe cumplir un formato específico.	
Resultados esperados: • En los dos casos se refleja operación exitosa. • Se puede observar en los datos consultados las acciones realizadas. • Las búsquedas son exitosas en función de los parámetros enviados. • Para las consultas se despliega en pantalla según sea el caso: personas o caballos.	
Evaluación de la prueba: Satisfactoria	

Figura 4.6: Prueba de aceptación - Registro de personas y caballos
Realizado por: Luis Lima

4.2.3 PRUEBAS DE LA TERCERA ITERACIÓN

PRUEBA DE ACEPTACION	
No. 4	**Historia de usuario: 04 Gestión de concurso, Inscripciones**
Descripción: El usuario con rol Juez realiza los procesos de inscripción, cierre de inscripción, calificación y cierre del concurso.	
Condiciones de ejecución: • Debe existir el usuario con rol Juez creado en el sistema. • Debe existir registrado los tipos de caballos existentes. • Debe estar parametrizado un concurso con las categorías participantes. • Deben estar registrados en el sistema los caballos que participarán en las pruebas.	
Entrada/Pasos de ejecución: El usuario: • Ingresa al menú Concurso y selecciona la opción Inscripciones. Sobre esta opción realiza operaciones de inserción, actualización y borrado de los registros. • Sobre los datos desplegados en pantalla se realiza varias búsquedas por distintos parámetros para lo cual se presiona el botón buscar. • En el formulario de edición se ingresa el código de registro del caballo y el sistema determina la categoría en la que debe participar. • Ingresa al menú Reportes y selecciona la opción Inscritos por categoría o Total caballos inscritos. Selecciona el concurso y emite los reportes correspondientes. En el formulario de edición se ingresa la información requerida en cada caso y se verifica que sea la adecuada con las validaciones respectivas. Se pone especial énfasis en el campo registro que debe cumplir un formato específico.	
Resultados esperados: • Para cada acción se retorna operación exitosa. • Se puede observar en los datos consultados las acciones realizadas. • Las búsquedas son exitosas en función de los parámetros enviados. • En caso de tener un caballo en la misma categoría se retorna mensaje de error. • Los reportes generados son acordes a los formatos establecidos.	
Evaluación de la prueba: Satisfactoria	

Figura 4.7: Prueba de aceptación – Gestión de concurso, Inscripciones
Realizado por: Luis Lima

PRUEBA DE ACEPTACION	
No. 5	**Historia de usuario: 04 Gestión de concurso, Cerrar inscripciones**
Descripción: El usuario con rol Juez realiza los procesos de inscripción, cierre de inscripción, calificación y cierre del concurso.	
Condiciones de ejecución: • Debe existir el usuario con rol Juez creado en el sistema. • Debe existir registrado los tipos de caballos existentes. • Debe estar parametrizado un concurso con las categorías participantes. • Deben estar registrados en el sistema los caballos que participarán en las pruebas.	
Entrada/Pasos de ejecución: El usuario: • Ingresa al menú Concurso y selecciona la opción Cerrar Inscripciones. Sobre esta opción selecciona el concurso a cerrar y presiona el botón Cerrar.	
Resultados esperados: • El sistema emite mensaje de advertencia previa a la confirmación de la ejecución del proceso • Si el concurso seleccionado ya está cerrado previamente el sistema emite mensaje respectivo y cancela la ejecución. • Al finalizar el proceso el sistema emite mensaje de proceso exitoso.	
Evaluación de la prueba: Satisfactoria	

Figura 4.8: Prueba de aceptación – Gestión de concurso, Cerrar inscripciones
Realizado por: Luis Lima

PRUEBA DE ACEPTACION	
No. 6	**Historia de usuario: 04 Gestión de concurso, Calificación**
Descripción: El usuario con rol Juez realiza los procesos de inscripción, cierre de inscripción, calificación y cierre del concurso.	
Condiciones de ejecución: • Debe existir el usuario con rol Juez creado en el sistema. • Debe existir registrado los tipos de caballos existentes. • Debe estar parametrizado un concurso con las categorías participantes. • Deben estar registrados en el sistema los caballos que participarán en las pruebas. • Debe estar realizada la inscripción de los caballos participantes	
Entrada/Pasos de ejecución: El usuario: • Ingresa al menú Concurso y selecciona la opción Calificación. Sobre esta opción selecciona el concurso y la categoría a calificar. • Sobre los registros desplegados de los caballos inscritos ingresa la ubicación y si tiene mención honorífica o no. • Para registrar caballos ganadores de otras categorías en la actual, presiona el botón Registrar y se cargan automáticamente en función de la parametrización de categorías y puestos participantes.	
Resultados esperados: • Si selecciona un concurso ya cerrado el sistema emite el mensaje correspondiente y cancela el proceso, caso contrario despliega en pantalla los caballos inscritos en la categoría que se está calificando. • Al presionar el botón Registrar el sistema emite mensaje del número de caballos inscritos tardíamente. • Al registrar el puesto se emite mensaje de proceso exitoso y se refleja en pantalla los datos ingresados.	
Evaluación de la prueba: Satisfactoria	

Figura 4.9: Prueba de aceptación – Gestión de concurso, Calificación

Realizado por: Luis Lima

PRUEBA DE ACEPTACION	
No. 7	**Historia de usuario: 04 Gestión de concurso, Cierre**
Descripción: El usuario con rol Juez realiza los procesos de inscripción, cierre de inscripción, calificación y cierre del concurso.	
Condiciones de ejecución: • Debe existir el usuario con rol Juez creado en el sistema. • Debe existir registrado los tipos de caballos existentes. • Debe estar parametrizado un concurso con las categorías participantes. • Deben estar registrados en el sistema los caballos que participarán en las pruebas. • Debe estar realizada la inscripción de los caballos participantes	
Entrada/Pasos de ejecución: El usuario: • Ingresa al menú Concurso y selecciona la opción Cierre. Sobre esta opción selecciona el concurso y presiona el botón Seleccionar.	
Resultados esperados: • El sistema emite mensaje de advertencia previo a la ejecución del proceso. • Si selecciona un concurso ya cerrado el sistema emite el mensaje correspondiente y cancela el proceso. • Al concluir el proceso el sistema emite mensaje de proceso exitoso.	
Evaluación de la prueba: Satisfactoria	

Figura 4.10: Prueba de aceptación – Gestión de concurso, Cierre

Realizado por: Luis Lima

4.2.4 PRUEBAS DE LA CUARTA ITERACIÓN

PRUEBA DE ACEPTACION	
No. 8	**Historia de usuario: 05 Resultados, Por categoría**
Descripción: El usuario con rol Juez genera los resultados del concurso tanto en pantalla como en reportes impresos. Estos pueden ser por categoría o generales para mejor criador, expositor y reproductor.	
Condiciones de ejecución: • Debe existir el usuario con rol Juez creado en el sistema. • Debe existir registrado los tipos de caballos existentes. • Debe estar parametrizado un concurso con las categorías participantes. • Deben estar registrados en el sistema los caballos que participarán en las pruebas. • Debe estar realizada la inscripción de los caballos participantes. • Debe estar cerrado el concurso.	
Entrada/Pasos de ejecución: El usuario: • Ingresa al menú Resultados y selecciona la opción Por categoría. Sobre esta opción selecciona el concurso y presiona el botón Aceptar.	
Resultados esperados: • El sistema genera reporte en pdf por categoría detallando los caballos en las primeras ubicaciones y menciones honrosas obtenidas. • El detalle de los caballos en cada categoría es en orden ascendente por ubicación. • Cada categoría debe ir en una hoja diferente.	
Evaluación de la prueba: Satisfactoria	

Figura 4.11: Prueba de aceptación – Resultados
Realizado por: Luis Lima

PRUEBA DE ACEPTACION	
No. 9	**Historia de usuario: 05 Resultados, Generales**
Descripción: El usuario con rol Juez genera los resultados del concurso tanto en pantalla como en reportes impresos. Estos pueden ser por categoría o generales para mejor criador, expositor y reproductor.	
Condiciones de ejecución: • Debe existir el usuario con rol Juez creado en el sistema. • Debe existir registrado los tipos de caballos existentes. • Debe estar parametrizado un concurso con las categorías participantes. • Deben estar registrados en el sistema los caballos que participarán en las pruebas. • Debe estar realizada la inscripción de los caballos participantes. • Debe estar cerrado el concurso.	
Entrada/Pasos de ejecución: El usuario: • Ingresa al menú Resultados y selecciona la opción Generales. • Sobre esta opción selecciona el concurso y el tipo de consulta: Mejor criador, expositor o reproductor y presiona el botón Consultar. • Para imprimir las ubicaciones o el detalle desplegado en pantalla presiona el botón imprimir en la grilla correspondiente.	
Resultados esperados: • El sistema despliega en pantalla las ubicaciones generales obtenidas para cada caso y en la parte inferior el detalle de los puntos obtenidos por cada caballo. • El detalle de los caballos en cada categoría es en orden ascendente por ubicación y por puntos obtenidos • El reporte de ubicaciones se genera en formato PDF. • El reporte del detalle de puntos obtenidos se genera en formato y es congruente con el reporte de ubicaciones.	
Evaluación de la prueba: Satisfactoria	

Figura 4.12: Prueba de aceptación – Generales

Realizado por: Luis Lima

4.2.5 ANALISIS DE LAS PRUEBAS DE ACEPTACIÓN

La Tabla 4.1 resume los resultados obtenidos en el proceso de las pruebas de aceptación.

ITERACIÓN	HISTORIA DE USUARIO	NOMBRE	PRUEBA DE ACEPTACION	EVALUACIÓN
1	01	Funcionalidad general	01	Satisfactoria
1	02	Parámetros generales	02	Satisfactoria
2	03	Registro de personas y caballos	03	Satisfactoria
3	04	Gestión de concurso, Inscripciones	04	Satisfactoria
3	04	Gestión de concurso, Cerrar inscripciones	05	Satisfactoria
3	04	Gestión de concurso, Calificación	06	Satisfactoria
3	04	Gestión de concurso, Cierre	07	Satisfactoria
4	05	Resultados, Por categoría	08	Satisfactoria
4	05	Resultados, Generales	09	Satisfactoria

Tabla 4.1: Análisis de Pruebas de Aceptación

Realizado por: Luis Lima

En base a los resultados de las pruebas de aceptación, se concluye que el sistema realizado y todos sus procesos son satisfactorios en su mayoría, obteniendo un software de calidad que se ajusta a los requisitos del usuario.

En el Anexo H se encuentran los documentos de trabajo utilizados en las pruebas de aceptación con el usuario.

Una vez recibida la aceptación del usuario del sistema desarrollado se inicia con la instalación de la aplicación en ambiente de producción, iniciamos analizando los datos factibles de migrar de los documentos Excel disponibles.

4.3 MIGRACION DE DATOS EXISTENTES CON EL NUEVO SISTEMA

De la información disponible y una vez realizado el análisis de la calidad de la misma se determina que los únicos datos factibles de migrar son los relacionados con las personas y caballos.

La migración de la información se realizará en base a archivos en formato csv y phpmyadmin con la opción Import. Los archivos de carga fueron validados por el usuario previo a la incorporación al sistema. Los archivos utilizados se detallan en la Tabla 4.2:

ARCHIVO	No. REGISTROS
Personas.csv	508
Caballos.csc	3666

Tabla 4.2: Migración de información
Realizado por: Luis Lima

En el Anexo I se encuentra un ejemplo de los archivos utilizados para migración.

La información referente a categorías, tipos de caballos y concursos se ingresará mediante las opciones disponibles en el sistema, esto debido a que la encontrada en los archivos Excel no tiene ninguna integridad.

La información histórica de los resultados de concursos anteriores no se migrará debido a que no se encuentra archivos de respaldo certificados y en común acuerdo con el cliente se define que el sistema guardará este tipo de registros desde la realización del primer concurso respaldado por el sistema.

Con la información lista para cargar en el sistema se procede con la instalación de la aplicación, que se revisa a continuación.

4.4 PROCEDIMIENTO DE INSTALACION

Para la instalación de la aplicación en ambiente de producción es necesario seguir el siguiente procedimiento:

1. **Instalar de XAMPP.-** En este paso se instala PHP, el servidor de páginas Web Apache y la base de datos MySql. La instalación que se debe realizar es por defecto.

2. **Instalar la Aplicación.-** En este paso se instala todos los archivos necesarios para el funcionamiento de la aplicación. En primer lugar se debe instalar CodeIgniter según lo indicado en el Anexo F. Luego se reemplazan los archivos existentes en: models, views y controllers del directorio system/application/ por sus correspondientes disponibles en el CD de instalación.

```
copy D:\asopaso\system\application\models \public_html\asopaso\system\application\
copy D:\asopaso\system\application\views \public_html\asopaso\system\application\
copy D:\asopaso\system\application\controllers \public_html\asopaso\system\application\
```

3. **Configurar la base de datos.-** En este paso se crea la base de datos (asopaso) y los objetos de base de datos necesarios para la aplicación. Desde phpmyadmin se debe importar el script asopaso.sql disponible en el cd de instalación con lo cual se crean los objetos indicados en la Figura 4.13.

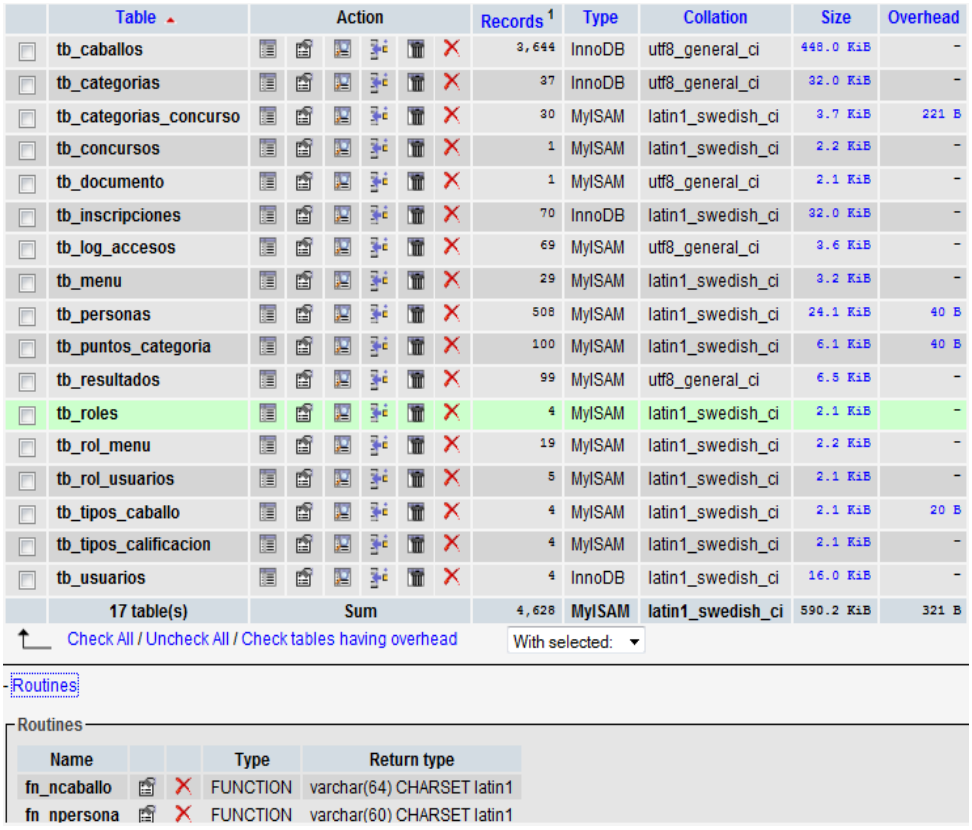


	Table	Action	Records [1]	Type	Collation	Size	Overhead
☐	tb_caballos		3,644	InnoDB	utf8_general_ci	448.0 KiB	-
☐	tb_categorias		37	InnoDB	utf8_general_ci	32.0 KiB	-
☐	tb_categorias_concurso		30	MyISAM	latin1_swedish_ci	3.7 KiB	221 B
☐	tb_concursos		1	MyISAM	latin1_swedish_ci	2.2 KiB	-
☐	tb_documento		1	MyISAM	utf8_general_ci	2.1 KiB	-
☐	tb_inscripciones		70	InnoDB	latin1_swedish_ci	32.0 KiB	-
☐	tb_log_accesos		69	MyISAM	utf8_general_ci	3.6 KiB	-
☐	tb_menu		29	MyISAM	latin1_swedish_ci	3.2 KiB	-
☐	tb_personas		508	MyISAM	latin1_swedish_ci	24.1 KiB	40 B
☐	tb_puntos_categoria		100	MyISAM	latin1_swedish_ci	6.1 KiB	40 B
☐	tb_resultados		99	MyISAM	utf8_general_ci	6.5 KiB	-
☐	tb_roles		4	MyISAM	latin1_swedish_ci	2.1 KiB	-
☐	tb_rol_menu		19	MyISAM	latin1_swedish_ci	2.2 KiB	-
☐	tb_rol_usuarios		5	MyISAM	latin1_swedish_ci	2.1 KiB	-
☐	tb_tipos_caballo		4	MyISAM	latin1_swedish_ci	2.1 KiB	20 B
☐	tb_tipos_calificacion		4	MyISAM	latin1_swedish_ci	2.1 KiB	-
☐	tb_usuarios		4	InnoDB	latin1_swedish_ci	16.0 KiB	-
	17 table(s)	Sum	4,628	MyISAM	latin1_swedish_ci	590.2 KiB	321 B

Check All / Uncheck All / Check tables having overhead With selected:

Routines

Routines

Name			Type	Return type
fn_ncaballo			FUNCTION	varchar(64) CHARSET latin1
fn_npersona			FUNCTION	varchar(60) CHARSET latin1

Figura 4.13 Estructura de la base de datos AsoPaso

Realizado por: Luis Lima

Una vez realizados los tres pasos indicados la aplicación queda lista para su operación.

En el Anexo J se detalla el contenido del CD de instalación de la aplicación.

Se debe considerar que si se utiliza un servicio de alojamiento Web, el servidor Web y la base de datos ya estará configurado por lo que se omite el paso 1.

4.5 TUTORIAS Y ASISTENCIAS POST-FORMACION

Como parte de la entrega del sistema se realiza la capacitación correspondiente al usuario sobre el uso del sistema y se hace la entrega del manual de usuario el cual lo podemos encontrar en el Anexo K. Como asistencia posterior se realizará el acompañamiento en la realización de un concurso real a realizarse próximamente.

4.6 RESUMEN

Los estándares en la codificación tienen mucha importancia, en el presente proyecto se utilizan los definidos para programadores en PHP. Previo al inicio de la codificación es necesario tener el ambiente de desarrollo listo, principalmente instalado y configurado CodeIgniter, lo cual no presenta mayores complicaciones y en pocos pasos se lo puede hacer.

CodeIgniter tiene su forma particular de trabajar desde el punto de vista de la codificación, principalmente en la organización de los archivos PHP en función de las tres capas del patrón Modelo-Vista-Controlador , se revisó a detalle cómo se trabaja en cada capa con lo que se puede ver principalmente la reutilización de código.

Continuando con las prácticas recomendadas por XP se documentan las pruebas de aceptación las cuales se realizan centrándose en la funcionalidad de los procesos desarrollados. Las pruebas se realizaron al final de cada iteración con lo cual se pudo ajustar periódicamente las necesidades precisas del usuario obteniéndose al final de las mismas la aceptación de sistema desarrollado.

Para la puesta en producción del sistema se realizó un análisis de la información existente que se podría migrar al nuevo sistema, sin embargo, se encontró que no existe la garantía de que ésta sea correcta y en común acuerdo con el usuario se decide únicamente migrar la información de personas y caballos existentes en hojas Excel.

Finalmente se detallan los pasos necesarios para la instalación de la aplicación en un ambiente de producción con lo cual se da por finalizado el proyecto.

CAPITULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- En el presente tema investigativo el uso del patrón MVC mediante el framework CodeIgniter permitió reducir los tiempos de desarrollo en un 25% del previsto inicialmente, principalmente por la reutilización de los modelos de la base de datos en muchas opciones del sistema y la validación de formularios propia del framework.

- En las pruebas de usuario del presente proyecto, un 90% de los cambios solicitados se orientan a formularios y reportes los cuales se realizaron en su totalidad sin afectar en gran medida el cronograma establecido gracias a la característica propia de MVC de separación en capas.

- El uso del framework CodeIgniter en aplicativos como el del presente trabajo investigativo permite eliminar del cronograma inicial las actividades y tiempos destinados al desarrollo de la funcionalidad para el **control de sesiones** y **conexión a la base de datos.**

- La combinación del patrón MVC, framework CodeIgniter y la metodología de desarrollo XP fueron elementos claves para el éxito del proyecto fundamentalmente porque los ajustes requeridos de mayor impacto se detectaron en las primeras iteraciones y la flexibilidad dada por el framework utilizado para aplicar estos cambios de manera ágil.

- El framework CodeIgniter garantiza un alto grado de seguridad en la aplicación implementada por las siguientes razones: uso de sesiones encriptadas que protegen la lectura de su contenido en caso de robo de cookies, filtro de seguridad XSS (Cross Site Scripting) en todos los datos enviados por medio de formularios y finalmente por la preferencia de uso del formulario POST en lugar de GET.

- El aplicativo Web del presente trabajo de investigación tiene buenos tiempos de respuesta debido a que en un 90% de las opciones del sistema se implementó grillas de datos, en las cuales se hace uso de las características y propiedades del JSON como protocolo de transmisión de datos conjuntamente con la tecnología AJAX.

- El uso del presente aplicativo Web permite a los organizadores de los concursos reducir a 0 el número de errores en el registro de edades,

propietarios y criadores de los caballos inscritos, el cual en el proceso manual estaba entre el 5% y 7%.

- El uso del presente aplicativo Web permite a los organizadores de los concursos la entrega y distribución de los resultados finales a todo detalle de manera inmediata lo que anteriormente tardaba 5 días aproximadamente.

5.2 RECOMENDACIONES

- Se recomienda el uso del software libre para desarrollo y ejecución de aplicaciones principalmente por la reducción de costos y los buenos resultados de su aplicación.

- Para proyectos similares, es recomendable la utilización de frameworks de desarrollo que combinados con la metodología XP, impulsan a dar soluciones rápidas y de alta calidad.

- Se recomienda la utilización de bases de datos de uso libre como MySQL o Postgres, por cuanto no tienen costo y poseen excelentes características de rendimiento principalmente MySQL que es el estándar de facto para aplicaciones Web.

- Se recomienda contratar un certificado de seguridad SSL (Secure Socket Layer) con la finalidad de complementar la seguridad de la aplicación en la transferencia de datos entre el navegador y el servidor Web.

- Es muy importante mantener actualizado el registro de caballos y personas en el sistema puesto que esta información es utilizada en la mayoría de los procesos que se realizan durante la organización y realización de un concurso.

- Para el correcto funcionamiento del sistema es fundamental que se estandarice las categorías participantes en cada evento con la finalidad de obtener datos históricos claros y coherentes.

- Dado el tipo de usuarios finales que tiene este proceso es recomendable trabajar de inmediato en la siguiente versión de la aplicación la cual considera terminales móviles.

BIBLIOGRAFIA

LIBROS

- Kendall Kenneth E. y Kendall Julie E (2005). *Análisis y Diseño de Sistemas (6ª ed).* México: Pearson Educación.

  Técnicas para aplicar las prácticas esenciales de la programación extrema.

- Gamma Erich, Helm Richard, Jonson Ralph y Vlissides (2006). *Patrones de Diseño: Elementos de software orientado a objetos reutilizable.* Madrid-España: Pearson Educación.

  Introducción a los patrones de diseño en el MVC.
  Catálogo de Patrones de Diseño

- Sommerville Ian (2005). *Ingeniería del Software (7ª ed).* Madrid-España: Pearson Educación.

  Procesos del software.
  Métodos Ágiles.
  Programación Extrema.

- Gutierrez Abraham y Bravo Ginés (2005). *PHP5 a través de ejemplos.* México: Alfaomega Grupo Editor.

  Instalación y configuración del servidor Web Apache, base de datos MySQL y del intérprete de PHP.
  Fundamentos y estructuras básicas de lenguaje
  Modelo de Objetos en PHP.
  Utilización de formularios.
  Modelo DOM.

- Pressman S. Roger (2006). *Ingeniería del Software. Un enfoque práctico (6ª ed).* University of Connecticut: McGrawHill.

  Desarrollo Ágil, programación extrema.
  Aplicación de la Ingeniería Web.

- Jacobson Ivar, Booch Grady y Rumbaugh James (2000). *El Proceso Unificado de Desarrollo de Software.* Madrid – España: Pearson Educación.

El Proceso Unificado: dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

- Chamoun Yamal (2002). *Administración profesional de proyectos la guía*. México:McGrawHill.

  Administración del Riesgo.

- Connolly Thomas M. y Begg Carolyn E. (2005). *Sistemas de bases de datos (4ª ed).* Madrid: Pearson Educación S.A.

  Modelos Entidad - Relación

MANUALES

- Martínez Pablo, Ruiz Díaz Pablo y Waisbrot Sebastián. *Manual de CodeIgniter en español* [En línea]. Disponible en :

  http://lax.franhp.net/CodeIgniter_Spanish_UserGuide.pdf [Consulta 07-07-2010]

  Pasos a seguir para la instalación de CodeIgniter.
  Introducción al framework CodeIgniter.
  Tópicos generales para la utilización, sintaxis a utilizar, librerías, ejemplos prácticos, etc.

INTERNET

- Nicolás Tedeschi (2010). *¿Qué es un patrón de diseño?* [En línea]. Disponible en:

  http://msdn.microsoft.com/es-es/library/bb972240.aspx [Consulta 27-09-2010]

  ¿Qué es un Patrón de Diseño?
  Patrones Creacionales
  Patrones Estructurales
  Patrones de Comportamiento

- Leonardo Caballero, Jesus Lara (2006). *Patrones de diseño y Orientación a Objetos en PHP5* [En línea]. Disponible en:

  http://s3.amazonaws.com/ppt-download/patrones-de-diseo-y-orienacin-a-objetos-en-php52422.pdf [Consulta 30-10-2010]

  Patrones de diseño.
  MVC (Modell View Controller).

- Carlos Reynoso (2008) *Arquitectura de Software* [En línea]. Disponible en:

  http://carlosreynoso.com.ar/arquitectura-de-software/ [Consulta 30-10-2010]

  Introducción a la arquitectura de software.
  Conceptos fundamentales

- Euphoria IT Ltda.(2005 – 2008). *¿Qué es un Framework de desarrollo Web ?* [En línea]. Disponible en :

  http://euphoriait.com/articulos/framework_web [Consulta 07-10-2010]

  Framework de desarrollo Web.
  Patrón MVC y Model 2

- Trirand Inc. (2009). *JQuery Grid* [En línea]. Disponible en:

  http://www.trirand.com/jqgridwiki/doku.php [Consulta 04-09-2010]

  Plugin y tutorial para la utilización de JQGrid en PHP.
  Ejemplos de cómo implementar grillas de todo tipo con formatos, edición, navegación, etc.
  Integración con JQuery UI.

- The jQuery Project and the jQuery UI Team (2010). *JQuery User Interface* [En línea]. Disponible en:

  http://jquery.com/ [Consulta 04-09-2010]

  Plugin y tutorial para la utilización de JQuery.
  Plugin y tutorial para la utilización de JQuery UI.

Demos y documentación.
Tutoriales en español.

- EllisLab, Inc. (2010). *CodeIgniter* [En línea]. Disponible en:

  http://codeigniter.com [Consulta 05-09-2010]

  Tutorial completo de CodeIgniter.
  Comunidad y wiki dedicados a este framework.
  Ejemplos de una gran cantidad de código que tarde o temprano se requiere en la construcción de un sistema.
  Video tutoriales.

- BestWebFrameworks (2010). *Best-Web-Frameworks* [En línea]. Disponible en:

  http://www.bestwebframeworks.com/ [Consulta 02-11-2010]

  Frameworks en PHP
  Frameworks en JAVA
  Frameworks en RUBY

- DotPress (2007). *PEAR: Estándares de desarrollo para PHP* [En línea]. Disponible en:

  http://dotpress.wordpress.com/2007/03/29/pear-estandares-de-desarrollo-para-php/ [Consulta 02-11-2010]

  PEAR
  Estándares

ANEXOS

ANEXO A

REGISTRO DE INSCRIPCIONES

XII CONCURSO REGIONAL DE CABALLOS PERUANOS DE PASO "CIUDAD DE CUENCA"
SABADO 23 DE ENERO Y DOMINGO 24 DE ENERO DEL 2.010
REGISTRO DE INSCRIPCIONES

EJEMPLAR											
CATE-GORIA	SE-XO	PRE-FIJO	NOMBRE	FECHA DE NACIM.	REGISTRO	~~CRC~~	~~CAI~~	PADRE	MADRE	CRIADOR	PROPIETARIO
1	H	EJM	AURORA	Noviembre 17/2007	07-2-0220		SI	GO CANTINERO	AHP TRAVIESA	EDUARDO JACOME	EDUARDO JACOME
5	H	ALA	FRIDA	Enero 04/2004	04-1-0096		SI	GO CANTINERO	AHT MONTONERA	AUGUSTO LUZURIAGA	EDUARDO JACOME
8	H	EJM	ISABELA	Noviembre 09/1999	99-2-0146		SI	EBH DON RAMON	RR ISADORA	EDUARDO JACOME	EDUARDO JACOME
17	C	EJM	CASCABEL	Agosto 18/2005	05-2-0153		SI	ALA CONQUISTADOR	LB CASTAÑUELA	EDUARDO JACOME	EDUARDO JACOME
18	C	GO	INSOMNIO	Diciembre 22/2000	00-2-0097		SI	FPC ENGREIDO	JM ESTIMACION	GALO ORELLANA	EDUARDO JACOME
27	M	EJM	MOJITO	Octubre 05/2005	05-2-0151	SI	SI	GO CANTINERO	AHP TRAVIESA	EDUARDO JACOME	EDUARDO JACOME

CRIADERO "LA ESPERANZA". QUITO

PROPIETARIO: EDUARDO JACOME MERINO

FIRMA

Nota: Se envía por fax el registro con las inscripciones y el certificado con resultado negativo de despistaje de Anemia Infecciosa de los seis ejemplares.

ANEXO B

CATALOGO

ASOCIACION DE CRIADORES Y PROPIETARIOS DE CABALLOS DE PASO DE PICHINCHA
DECIMO SEPTIMO CONCURSO REGIONAL "CIUDAD DE QUITO"
JUEZ UNICO: ROLANDO GARCIA BERTONCINI. JUEZ ADJUNTO: JAVIER CARRERA MORENO.
QUITO TENIS Y GOLF CLUB
SABADO 11 Y DOMINGO 12 DE JULIO DEL 2.009

CATEGORIA POTRANCAS DE TIRO DE 24 A 30 MESES

No.	PRE-FIJO	NOMBRE	FECHA DE NACIMIENTO	REGISTRO	PADRE	MADRE	CRIADOR	PROPIETARIO	Su Calificación	Calificación del Juez
1	LB	VENUS	Abril 08/2007	07-1-0046	FPC QUINTO DE ORO	RYM ANDARES	LUIS BAKKER	LUIS BAKKER		
2	GO	CARICIA	Marzo 14/2007	07-1-0122	PK FLAMENCO	GO MANANTIAL	GALO ORELLANA	GALO ORELLANA		
3	RAH	FAROLERA	Enero 27/2007	07-1-0124	PK FLAMENCO	RAH FRENESI	RAFAEL ARMIJOS	RAFAEL ARMIJOS		

XVIII CONCURSO REGIONAL DE CABALLOS PERUANOS DE PASO "CIUDAD DE QUITO"
PUNTUACION PARA MEJOR CRIADOR Y MEJOR EXPOSITOR

	CATEGORIA	CRIADOR		EXPOSITOR		CRIADOR		EXPOSITOR		CRIADOR		EXPOSITOR	
No.	NOMBRE	Ejemplar	Ptos.	Ejemplar	Ptos.	Ejemplar	Ptos.	Ejemplar	Ptos.	Ejemplar	Ptos.	Ejemplar	Ptos.
1	POTRANCAS DE TIRO DE 24 A 30 MESES												
2	POTRANCAS DE TIRO DE 30 A 36 MESES												
3	CAMPEONA JOVEN DE TIRO POTRANCAS												
4	POTRANCAS DE BOZAL O BARBOQUEJO DE 3 A 4 AÑOS												
5	YEGUAS DE FRENO Y ESPUELAS DE 4 Y 5 AÑOS												
6	YEGUAS DE FRENO Y ESPUELAS DE 6 Y 7 AÑOS												
7	YEGUAS DE FRENO Y ESPUELAS DE 8 Y 9 AÑOS												
8	YEGUAS DE FRENO Y ESPUELAS DE 10 AÑOS EN ADELANTE												
9	ZOOTECNIA YEGUAS												
10	CONJUNTO DE YEGUAS												
11	PISOS YEGUAS												
12	MADRE E HIJA												
13	CAMPEONA DEL AÑO YEGUAS												
14	CAPONES DE BOZAL O BARBOQUEJO												
15	CAPONES DE FRENO Y ESPUELAS DE 4 Y 5 AÑOS												
16	CAPONES DE FRENO Y ESPUELAS DE MAS DE 6 AÑOS												
17	CAMPEON DEL AÑO CAPONES												
18	CONJUNTO DE CAPONES												
19	PISOS CAPONES												
20	POTROS DE TIRO DE 24 A 30 MESES												
21	POTROS DE TIRO DE 30 A 36 MESES												
22	CAMPEON JOVEN DE TIRO POTROS												
23	POTROS DE BOZAL O BARBOQUEJO DE 3 A 4 AÑOS												
24	POTROS DE FRENO Y ESPUELAS DE 4 Y 5 AÑOS												
25	POTROS DE FRENO Y ESPUELAS DE 6 AÑOS EN ADELANTE												
26	ZOOTECNIA POTROS												
27	CONJUNTO DE POTROS												
28	PISOS POTROS												
29	CAMPEON DEL AÑO POTROS												
30	CAMPEON DEL AÑO PISOS (Capones, Yeguas, Potros)												
31	PRUEBAS DE MENORES (Hasta 8 años y de 8 a 15 años)												
32	PROGENIE PADRE												
33	PROGENIE MADRE												
34	PREMIO AFICION												
35	CAMPEON DE CAMPEONES PISOS												
36	CAMPEON DE CAMPEONES CAPONES												
37	CAMPEONA DE CAMPEONAS YEGUAS												
38	CAMPEON DE CAMPEONES POTROS												

PUNTUACION SEGUN EL PUESTO PARA MEJOR CIADOR Y MEJOR EXPOSITOR

PUESTO	PUNTOS
Primero	12
Segundo	10
Tercero	8
Cuarto	6
Quinto	4
Sexto	2

Nota: Los ejemplares al cabestro y caballos de trabajo acreditarán el 50% de puntaje

ANEXO C

RESULTADOS - MEJORES

A	B	C	D	E	F	G	H	I	J	K
Criador	**Puntaje**	**1os. Puestos**		**Expositor**	**Puntaje**	**1os. Puestos**		**Reproductor**	**Puntaje**	**1os. Puestos**
LUIS BAKKER	18	2		LUIS BAKKER	18	2		PK FLAMENCO	13	
GALO ORELLANA	17	1		XAVIER PALACIOS	12	1		FPC QUINTO DE ORO	12	1
ALFONSO ROMERO	10			LUCAS LUCERO	10			FPC ENGREIDO	12	1
LEON FEBRES CORDERO	8			GALO ORELLANA	10			AV EJECUTIVO	10	
ROGER AMORES	5			LEON FEBRES CORDERO	8			LB EXECEPCIONAL	6	1
RODRIGO CELI	4			RODRIGO CELI	4			LV ATILA	5	
RAFAEL ARMIJOS	4			RAFAEL ARMIJOS	4			JRM RETORNO	3	
JOSE SANCHEZ	4			JOSE SANCHEZ	4			MAGISTRAL	2	
ABRAHAM ZABALA	3			JORGE PAEZ	3			PK PRINCIPAL		
FERNANDO ORDOÑEZ	2			MARIO CEDEÑO	2			PK GADITANO		
CARLOS RIBADENEIRA	1			FERNANDO GAME	2			PK ESPARTACO		
XAVIER PALACIOS				CARLOS RIBADENEIRA	1			PK DON SOL		
TEODORO CARRASCO				TEODORO CARRASCO				PG TITANIC		
SYLVIA BAKKER				SYLVIA BAKKER				PG TANGO		
STEVEN BROWN				STEVEN BROWN				PG RODEO		
SOLEDAD BRUZZONE				SOLEDAD BRUZZONE				PG PAVAROTI		
SIDNEY GUERRERO				SIDNEY GUERRERO				PG CATWALK		
SANTIAGO AMBROSI				SANTIAGO AMBROSI				PFC FESTIVO		
SAMUEL GLEISSER				SAMUEL GLEISSER				PC PEREGRINO		

ANEXO D

FORMATO HISTORIAS DE USUARIO

HISTORIA DE USUARIO	
No.	**Usuario responsable:**
Nombre historia:	
Prioridad:	
Iteración asignada:	
Programador responsable:	
Descripción:	
Observaciones:	

ANEXO E

ESTANDARES DE PROGRAMACIÓN

Para los programadores en PHP, existe una guía de estándares de desarrollo y programación que define el modo en que, básicamente, un script PHP debe ser realizado.

Estándar 1: La identación debe ser a cuatro espacios sin caracteres de tabulación. Esto es debido a que ciertos IDE's de desarrollo introducen caracteres de tabulación cuando identan un texto automáticamente. Se recomienda el uso de herramientas o editores generales como EMACS u otros.

Estándar 2: Las estructuras de control deben tener un espacio entre el **keyword** de la estructura y el signo de apertura de paréntesis para distinguir entre las llamadas de las funciones y el signo de llaves debe estar sobre la línea de la estructura.

Estándar 3: Las funciones deben ser llamadas sin espacios entre el nombre de la función, el signo de paréntesis y el primer parámetro; espacios entre cada coma por parámetro y sin espacios entre el ultimo paréntesis, el signo de paréntesis cerrado y el signo de punto y coma (**;**).

Estándar 4: El estilo de los comentarios debe ser como el estilo de comentarios para C (/* */ ó //), no debe de utilizarse el estilo de comentarios de Perl (#).

Estándar 5: Cuando se incluya un archivo de dependencia incondicionalmente utilice **require_once** y cuando sea condicionalmente, utilice **include_once**.

Estándar 6: siempre utilice las etiquetas <?php ?> para abrir un bloque de código. No utilice el método de etiquetas cortas, porque esto depende de las directivas de configuración en el archivo PHP.INI y hace que el script no sea tan portable.

Estándar 7: Los nombres de las clases deben de iniciar con letra mayúscula. Los nombres de las variables y de las funciones pueden iniciar con letra minúscula, pero si estas tienen más de una palabra, cada nueva palabra debe iniciar con letra mayúscula (el nombre puede escribirse separado por signos de guión mayor). Si una función, en una clase, es privada; deberá comenzar con el signo de guión mayor para una fácil

identificación. Las constantes deben de escribirse siempre en mayúsculas y tanto estas como las variables globales deben de tener como prefijo el nombre de la clase a la que pertenecen.

Estándar 8: Los archivos con código PHP, deben de ser guardados en formato ASCII utilizando la codificación ISO-8859-1. **(Actualizado)**. El formato ASCII con codificación ISO-8859-1 es el formato en que se guardan los archivos de texto plano (.txt). La razón de este estándar es que determinados editores HTML (en especial Dreamweaver), agregan códigos de carácter extraño de salto de línea (como si se tratara de un archivo binario) y esto puede ocasionar que el interprete de PHP, encuentre problemas a la hora de leer el script.

Fuente: http://dotpress.wordpress.com/2007/03/29/pear-estandares-de-desarrollo-para-php/

ANEXO F

INSTALACION Y CONFIGURACION DE CODEIGNITER

Los pasos necesarios para instalar y configurar CodeIgniter son los siguientes:

1. Instalar xampp.
2. Descargar CodeIgniter de la url http://codeigniter.com/downloads/
3. Descomprimir la carpeta y renombrar con el nombre “asopaso”.
4. Copiar la carpeta en htdocs ubicada en C:\xampp\
5. Crear la base de datos en MySql.
6. Configurar los archivos necesarios de CodeIgniter:

    a. Configurar el archivo autoload.php

    C:\xampp\htdocs\asopaso\system\application\config\autoload.php:

```
$autoload['libraries'] = array('database'); //Se carga la librería para la conexión a la BDD

$autoload['helper'] = array('url', 'form','date'); //Se carga la librería para la url y formularios
```

    b. Configurar el archivo config.php

    C:\xampp\htdocs\asopaso\system\application\config\config.php:

```
$config['base_url']= "http://localhost/asopaso/"; //Nombre a llamar desde la URL
```

    c. Configurar el archivo database.php

    C:\xampp\htdocs\asopaso\system\application\config\database.php:

```
$db['default']['hostname'] = "localhost";     //nombre del hostname

$db['default']['username'] = "root";          //username de la base de datos

$db['default']['password'] = "";              //contraseña de las base de datos

$db['default']['database'] = "asopaso";       //nombre de la base de datos

$db['default']['dbdriver'] = "mysql";         //driver de la base de datos
```

    d. Configurar el archivo routes.php

    C:\xampp\htdocs\asopaso\system\application\config\routes.php:

```
$route['default_controller'] = "login_c";     //controlador por default
```

e. Mostrar la aplicación desde la url:

```
http://localhost/asopaso
```

ANEXO G

PRUEBAS DE ACEPTACION

PRUEBA DE ACEPTACION	
No.	**Historia de usuario (No. y nombre):**
Descripción:	
Condiciones de ejecución:	
Entrada/Pasos de ejecución:	
Resultados esperados:	
Evaluación de la prueba:	

Descripción:

- **No**.- Identificador único de la prueba
- **Historia de usuario.-** Número y nombre de la historia de usuario considerada para la prueba de aceptación.
- **Descripción.**- Definición corta de la historia de usuario a probarse.
- **Condiciones de ejecución.**- Condiciones previas requeridas a la ejecución de la funcionalidad que se va a probar.
- **Entrada/Pasos de ejecución.**- Pasos por los que el usuario tiene que pasar para ejecutar la acción que se requiere probar.
- **Resultado esperado.**- Respuesta obtenida del sistema posterior a la ejecución de la funcionalidad.
- **Evaluación de la prueba.**- Nivel de satisfacción del cliente con respecto al resultado obtenido del sistema. Es una respuesta directa del usuario.

ANEXO H

DOCUMENTO DE TRABAJO DE PRUEBAS DE ACEPTACION

PRUEBA DE ACEPTACION	
No. 1	**Historia de usuario: 01 Funcionalidad general**
Descripción: Acceso al sistema a través de cualquier navegador de Internet	
Condiciones de ejecución: • Debe existir el acceso a la página principal de la aplicación. • El usuario debe contar PC con acceso a Internet	
Entrada/Pasos de ejecución: El usuario: • Ingresa url http://localhost/asopaso	
Resultados esperados: • Ingreso exitoso al sistema, se despliega la pantalla principal del sistema.	
Evaluación de la prueba: Satisfactoria ING. EDUARDO JACOME M.	

PRUEBA DE ACEPTACION	
No. 2	**Historia de usuario: 02 Parámetros generales**

Descripción:
El usuario Administrador realiza el mantenimiento de los parámetros generales como son: Tipos de caballos, categorías y concursos.

Condiciones de ejecución:

- Debe existir el usuario Administrador

Entrada/Pasos de ejecución:
El usuario:

- Ingresa al menú parámetros y selecciona la opción Tipos caballo. Sobre esta opción realiza operaciones de inserción, actualización y borrado de los registros.
- Ingresa al menú parámetros y selecciona la opción Categorías. Sobre esta opción realiza operaciones de inserción, actualización y borrado de los registros.
- Ingresa al menú parámetros y selecciona la opción Definir concurso. Sobre esta opción realiza operaciones de inserción, actualización y borrado de los registros.

En el formulario de edición se ingresa la información requerida en cada caso y se verifica que sea la adecuada con las validaciones respectivas.

Resultados esperados:

- En los tres casos se refleja operación exitosa.
- Se puede observar en los datos consultados las acciones realizadas.

Evaluación de la prueba:

Satisfactoria
ING. EDUARDO JACOME M.

PRUEBA DE ACEPTACION	
No. 3	**Historia de usuario: 03 Registro de personas y caballos**

Descripción:
El usuario con rol Registro realiza el mantenimiento de las personas y caballos relacionados con un concurso.

Condiciones de ejecución:

- Debe existir el usuario con el rol Registro.
- Debe existir el usuario con el rol Consultas.
- Debe existir registrado los tipos de caballos existentes.

Entrada/Pasos de ejecución:
El usuario:

- Ingresa al menú Registro y selecciona la opción Registro de personas. Sobre esta opción realiza operaciones de inserción, actualización y borrado de los registros.
- Sobre los datos desplegados en pantalla se realiza varias búsquedas por distintos parámetros para lo cual se presiona el botón buscar.
- Ingresa al menú Registro y selecciona la opción Registro de caballos. Sobre esta opción realiza operaciones de inserción, actualización y borrado de los registros.
- Sobre los datos desplegados en pantalla se realiza varias búsquedas por distintos parámetros para lo cual se presiona el botón buscar.
- Ingresa al menú Consultas y selecciona la opción Personas o Caballos. Sobre esta opción realiza operaciones de búsqueda son varios parámetros.

En el formulario de edición se ingresa la información requerida en cada caso y se verifica que sea la adecuada con las validaciones respectivas. Se pone especial énfasis en el campo registro que debe cumplir un formato específico.

Resultados esperados:

- En los dos casos se refleja operación exitosa.
- Se puede observar en los datos consultados las acciones realizadas.
- Las búsquedas son exitosas en función de los parámetros enviados.
- Para las consultas se despliega en pantalla según sea el caso: personas o caballos.

Evaluación de la prueba:

Satisfactoria
ING. EDUARDO JACOME M.

PRUEBA DE ACEPTACION	
No. 4	**Historia de usuario: 04 Gestión de concurso, Inscripciones**

Descripción:
El usuario con rol Juez realiza los procesos de inscripción, cierre de inscripción, calificación y cierre del concurso.

Condiciones de ejecución:

- Debe existir el usuario con rol Juez creado en el sistema.
- Debe existir registrado los tipos de caballos existentes.
- Debe estar parametrizado un concurso con las categorías participantes.
- Deben estar registrados en el sistema los caballos que participarán en las pruebas.

Entrada/Pasos de ejecución:
El usuario:

- Ingresa al menú Concurso y selecciona la opción Inscripciones. Sobre esta opción realiza operaciones de inserción, actualización y borrado de los registros.
- Sobre los datos desplegados en pantalla se realiza varias búsquedas por distintos parámetros para lo cual se presiona el botón buscar.
- En el formulario de edición se ingresa el código de registro del caballo y el sistema determina la categoría en la que debe participar.
- Ingresa al menú Reportes y selecciona la opción Inscritos por categoría o Total caballos inscritos. Selecciona el concurso y emite los reportes correspondientes.

En el formulario de edición se ingresa la información requerida en cada caso y se verifica que sea la adecuada con las validaciones respectivas. Se pone especial énfasis en el campo registro que debe cumplir un formato específico.

Resultados esperados:

- Para cada acción se retorna operación exitosa.
- Se puede observar en los datos consultados las acciones realizadas.
- Las búsquedas son exitosas en función de los parámetros enviados.
- En caso de tener un caballo en la misma categoría se retorna mensaje de error.
- Los reportes generados son acordes a los formatos establecidos.

Evaluación de la prueba:

Satisfactoria

ING. EDUARDO JACOBO M.

PRUEBA DE ACEPTACION	
No. 5	**Historia de usuario: 04 Gestión de concurso, Cerrar inscripciones**

Descripción:
El usuario con rol Juez realiza los procesos de inscripción, cierre de inscripción, calificación y cierre del concurso.

Condiciones de ejecución:

- Debe existir el usuario con rol Juez creado en el sistema.
- Debe existir registrado los tipos de caballos existentes.
- Debe estar parametrizado un concurso con las categorías participantes.
- Deben estar registrados en el sistema los caballos que participarán en las pruebas.

Entrada/Pasos de ejecución:
El usuario:

- Ingresa al menú Concurso y selecciona la opción Cerrar Inscripciones. Sobre esta opción selecciona el concurso a cerrar y presiona el botón Cerrar.

Resultados esperados:

- El sistema emite mensaje de advertencia previa a la confirmación de la ejecución del proceso
- Si el concurso seleccionado ya está cerrado previamente el sistema emite mensaje respectivo y cancela la ejecución.
- Al finalizar el proceso el sistema emite mensaje de proceso exitoso.

Evaluación de la prueba:

Satisfactoria

ING. EDUARDO JACOME H.

PRUEBA DE ACEPTACION	
No. 6	**Historia de usuario: 04 Gestión de concurso, Calificación**

Descripción:
El usuario con rol Juez realiza los procesos de inscripción, cierre de inscripción, calificación y cierre del concurso.

Condiciones de ejecución:

- Debe existir el usuario con rol Juez creado en el sistema.
- Debe existir registrado los tipos de caballos existentes.
- Debe estar parametrizado un concurso con las categorías participantes.
- Deben estar registrados en el sistema los caballos que participarán en las pruebas.
- Debe estar realizada la inscripción de los caballos participantes

Entrada/Pasos de ejecución:
El usuario:

- Ingresa al menú Concurso y selecciona la opción Calificación. Sobre esta opción selecciona el concurso y la categoría a calificar.
- Sobre los registros desplegados de los caballos inscritos ingresa la ubicación y si tiene mención honorífica o no.
- Para registrar caballos ganadores de otras categorías en la actual, presiona el botón Registrar y se cargan automáticamente en función de la parametrización de categorías y puestos participantes.

Resultados esperados:

- Si selecciona un concurso ya cerrado el sistema emite el mensaje correspondiente y cancela el proceso, caso contrario despliega en pantalla los caballos inscritos en la categoría que se está calificando.
- Al presionar el botón Registrar el sistema emite mensaje del número de caballos inscritos tardíamente.
- Al registrar el puesto se emite mensaje de proceso exitoso y se refleja en pantalla los datos ingresados.

Evaluación de la prueba:

Satisfactoria
ING. EDUARDO JACOME M.

PRUEBA DE ACEPTACION	
No. 7	**Historia de usuario: 04 Gestión de concurso, Cierre**
Descripción: El usuario con rol Juez realiza los procesos de inscripción, cierre de inscripción, calificación y cierre del concurso.	
Condiciones de ejecución: • Debe existir el usuario con rol Juez creado en el sistema. • Debe existir registrado los tipos de caballos existentes. • Debe estar parametrizado un concurso con las categorías participantes. • Deben estar registrados en el sistema los caballos que participarán en las pruebas. • Debe estar realizada la inscripción de los caballos participantes	
Entrada/Pasos de ejecución: El usuario: • Ingresa al menú Concurso y selecciona la opción Cierre. Sobre esta opción selecciona el concurso y presiona el botón Seleccionar.	
Resultados esperados: • El sistema emite mensaje de advertencia previo a la ejecución del proceso. • Si selecciona un concurso ya cerrado el sistema emite el mensaje correspondiente y cancela el proceso. • Al concluir el proceso el sistema emite mensaje de proceso exitoso.	
Evaluación de la prueba: Satisfactoria ING. EDUARDO JACOME M.	

PRUEBA DE ACEPTACION	
No. 8	**Historia de usuario: 05 Resultados, Por categoría**

Descripción:
El usuario con rol Juez genera los resultados del concurso tanto en pantalla como en reportes impresos. Estos pueden ser por categoría o generales para mejor criador, expositor y reproductor.

Condiciones de ejecución:

- Debe existir el usuario con rol Juez creado en el sistema.
- Debe existir registrado los tipos de caballos existentes.
- Debe estar parametrizado un concurso con las categorías participantes.
- Deben estar registrados en el sistema los caballos que participarán en las pruebas.
- Debe estar realizada la inscripción de los caballos participantes.
- Debe estar cerrado el concurso.

Entrada/Pasos de ejecución:
El usuario:

- Ingresa al menú Resultados y selecciona la opción Por categoría. Sobre esta opción selecciona el concurso y presiona el botón Aceptar.

Resultados esperados:

- El sistema genera reporte en pdf por categoría detallando los caballos en las primeras ubicaciones y menciones honrosas obtenidas.
- El detalle de los caballos en cada categoría es en orden ascendente por ubicación.
- Cada categoría debe ir en una hoja diferente.

Evaluación de la prueba:

Satisfactoria

ING. EDWARDO JACOME M.

PRUEBA DE ACEPTACION	
No. 9	**Historia de usuario: 05 Resultados, Generales**

Descripción:
El usuario con rol Juez genera los resultados del concurso tanto en pantalla como en reportes impresos. Estos pueden ser por categoría o generales para mejor criador, expositor y reproductor.

Condiciones de ejecución:

- Debe existir el usuario con rol Juez creado en el sistema.
- Debe existir registrado los tipos de caballos existentes.
- Debe estar parametrizado un concurso con las categorías participantes.
- Deben estar registrados en el sistema los caballos que participarán en las pruebas.
- Debe estar realizada la inscripción de los caballos participantes.
- Debe estar cerrado el concurso.

Entrada/Pasos de ejecución:
El usuario:

- Ingresa al menú Resultados y selecciona la opción Generales.
- Sobre esta opción selecciona el concurso y el tipo de consulta: Mejor criador, expositor o reproductor y presiona el botón Consultar.
- Para imprimir las ubicaciones o el detalle desplegado en pantalla presiona el botón imprimir en la grilla correspondiente.

Resultados esperados:

- El sistema despliega en pantalla las ubicaciones generales obtenidas para cada caso y en la parte inferior el detalle de los puntos obtenidos por cada caballo.
- El detalle de los caballos en cada categoría es en orden ascendente por ubicación y por puntos obtenidos
- El reporte de ubicaciones se genera en formato PDF.
- El reporte del detalle de puntos obtenidos se genera en formato y es congruente con el reporte de ubicaciones.

Evaluación de la prueba:

Satisfactoria
ING. EDUARDO IXCOY M.

ANEXO I

ARCHIVOS DE MIGRACIÓN

Personas:

	A	B	C	D	E	F	G
1	CODIGO	PRIMER_NOMBRE	SEGUNDO_NOMBRE	PRIMER_APELLIDO	SEGUNDO APELLIDO	PREFIJO	SI
2	1	RAUL		ACAITURRI			
3	2	ESTEBAN		ACOSTA	E.		
4	3	ALFREDO		ADUM	ZIADE		
5	4	AGRICOLA GANADERA					
6	5	AGRICOLA INDUSTRIAL					
7	6	AGROPECUARIA RWT					
8	7	ADRIANA		AGUAYO	WALDOCK		
9	8	ARTURO		AGUAYO	POZO		
10	9	ARTURO		AGUAYO	WALDOCK		
11	10	JUAN	JOSE	AGUAYO	ARGUELLO		SI

CARGA_PERSONAS

Caballos:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Codigo	Registro	Pref.	Nombre	Sexo	Fecha Nac.	Reg- Padre	Reg-Madre	Criador	Propietario	Reproductor	Fecha.Ins.
2		00-1-0001	RAH	FAISAN	P	09/01/2000	93-2-1132	87-1-0858	43	488	N	30/06/2000
3		00-1-0002	FOV	PALMADA	Y	15/01/2000	88-1-1155	93-2-0900	388	314	N	30/06/2000
4		00-1-0003	JD	SARICO	P	17/03/2000	93-2-1067	85-2-0020	159	#N/A	N	31/10/2000
5		00-1-0004	FOV	COLORINA	Y	07/01/2000	95-2-0055	96-2-0150	388	121	N	30/06/2000
6		00-1-0006	PG	HENNA	Y	12/01/2000	91-2-1136	94-2-0112	241	43	N	30/06/2000
7		00-1-0007	AB	CHAYANE	P	06/05/2000	97-1-0027	93-2-0767	70	71	N	31/10/2000
8		00-1-0008	RAH	CADENCIOSA	Y	18/02/2000	96-1-0053	92-2-0814	43	76	N	30/06/2000
9		00-1-0009	DBG	TRES NOCHE	P	12/01/2000	95-1-0015	88-2-0632	63	63	N	30/11/2000
10		00-1-0010	HM	CA¥ITA	Y	07/06/2000	PN-03378	YN-06441	255	381	N	31/08/2001
11		00-1-0011	LM	BANDOLERO	P	16/02/2000	93-2-0865	95-2-0034	361	361	N	30/11/2000

carga_caballos

ANEXO J

CONTENIDO DEL CD DE INSTALACIÓN

El CD adjunto contiene los archivos necesarios para la instalación de la aplicación así como el código fuente de los programas. A continuación se detalla la estructura del mismo:

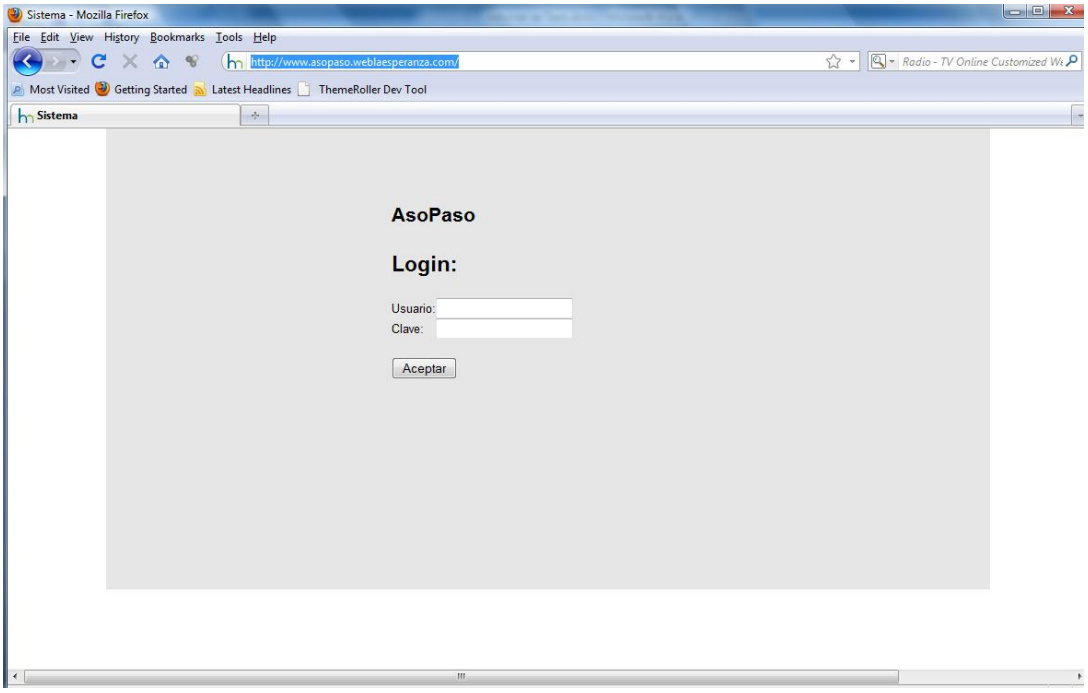
- Software requerido de la aplicación: /asopaso/*.*
- Script de base de datos con información por default: asopaso.sql
- Manual de usuario: manual_usuario.pdf.
- Informe de Tesis: Informe de Tesis.doc

ANEXO K

MANUAL DE USUARIO

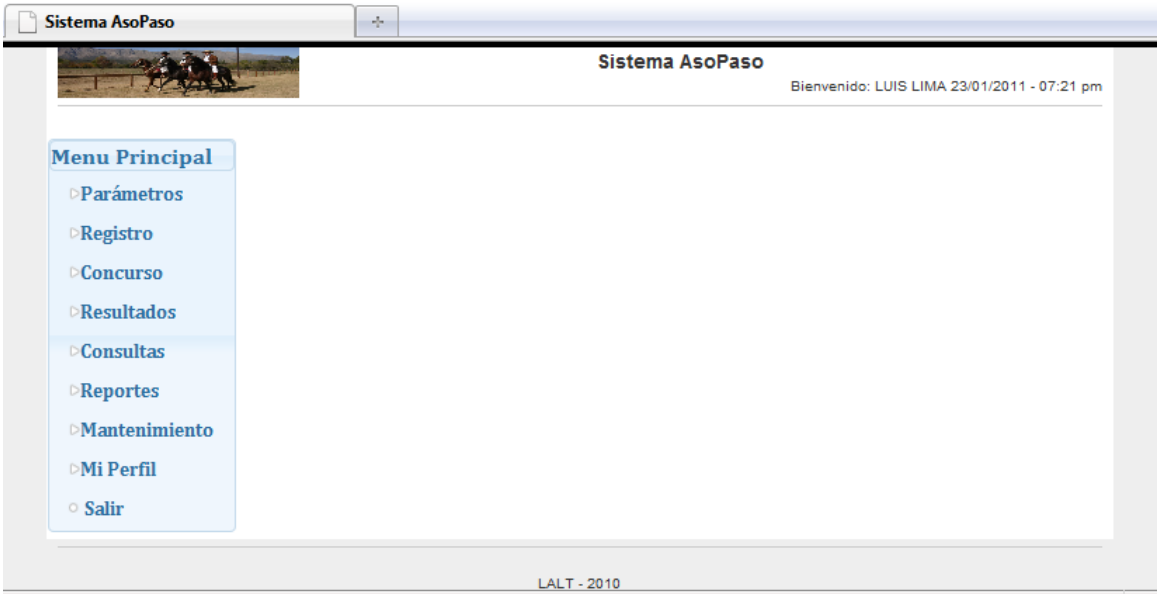
Ingreso al Sistema

Para ingresar al sistema en el browser de Internet se debe digitar la siguiente URL: http://www.asopaso.weblaesperanza.com una vez realizado esto el sistema le solicitará el usuario y clave definidos y debe presionar el botón aceptar.



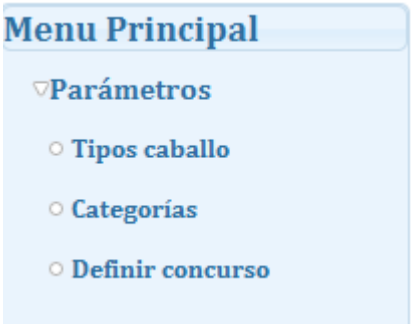
Menú principal

Si el usuario y clave fueron correctos se ingresa a la pantalla principal del sistema en la que se tiene el siguiente menú de opciones:



Parámetros

En este menú constan las opciones básicas de parametrización del sistema como son: Tipos de Caballo, Categorías y Definición del concurso.



Tipos caballo.- En esta opción se debe registrar los tipos de caballo que se contemplan en el sistema como son: yeguas, potros o capones. Se tiene una opción especial N, que está en blanco para calificar categorías especiales.

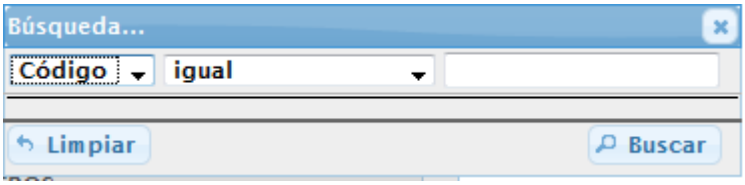
Para ingresar un nuevo tipo se debe presionar el botón  el cual llama al formulario estándar para el ingreso de información.

En este formulario se ingresa la información requerida y se presiona el botón Guardar.

Para modificar un registro ya ingresado se debe presionar el botón  , el cual llama a un formulario similar al indicado en el punto anterior, para grabar se procede de la misma forma que en el caso anterior.

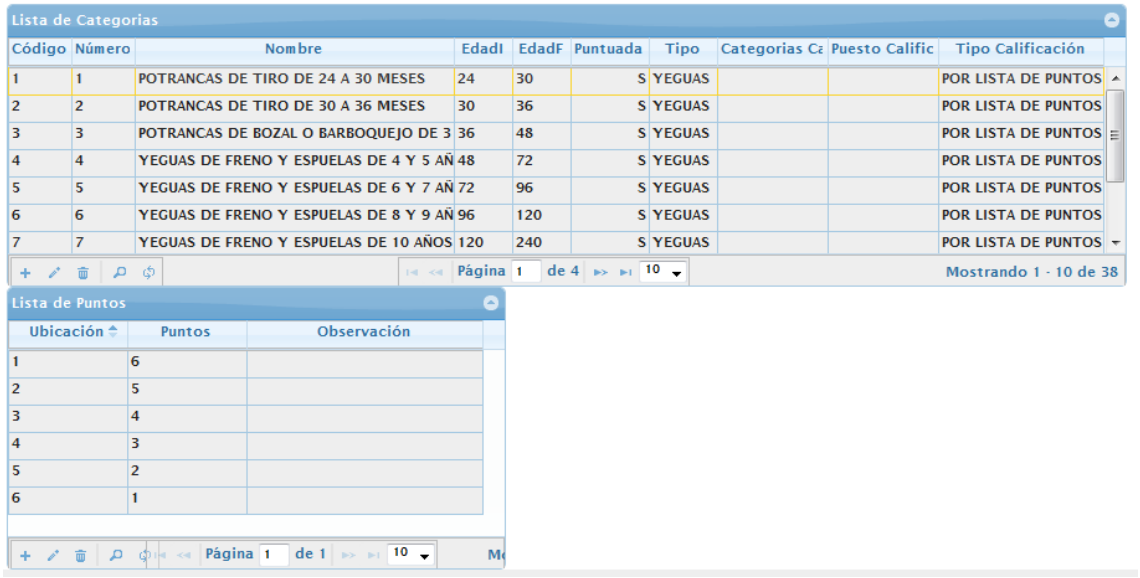
Para borrar se debe en primer lugar seleccionar el registro a borrar y luego se presiona el botón  .

En listados con gran cantidad de registros es necesario buscar uno en particular, para esto se debe presionar el botón  , el cual muestra un formulario para ingresar los parámetros de búsqueda, al presionar el botón Buscar el sistema ubica los registros requeridos.



Los botones  descritos anteriormente se utilizan en la mayoría de las pantallas de edición de información del sistema.

Categorías.- En esta opción se registran las distintas categorías que se consideran en el concurso.



Para las categorías puntuadas se define la puntos que se ganan dependiendo de la ubicación en la que quede el caballo participante.

Al definir una categoría se debe ingresar la siguiente información:

Número.- Corresponde al número asignado por el concurso a la categoría.
Nombre.- Es el nombre dado a la categoría.
EdadI.- Es la edad mínima en meses requerida para los caballos participantes.
EdadF.- Es la edad máxima en meses requerida para los caballos participantes.
Puntuada.- Indica si la categoría otorga puntos a los caballos participantes.
Tipo.- Se debe seleccionar el tipo de caballo que participa.
Categorías Califica.- Se deben ingresar las categorías que participan previamente para conformar los inscritos en esta categoría, se debe ingresar

los códigos de las categorías entre paréntesis y separados por comas. Ej: (4,5,6,7).
Puesto Califica: En complemento al punto anterior, en este campo se indican los puestos de las categorías que califican previamente, se debe ingresar los puestos requeridos entre paréntesis y separados por comas. Ej: (1,2)
Los dos puntos anteriores se utilizan para el proceso de **"inscripción tardía",** es decir, durante el concurso se inscriben en ciertas categorías caballos que hayan obtenido los primeros puestos en las categorías indicadas en el campo Categorías Califica.
Tipo Calificación.- Se debe seleccionar la forma de calificación de cada categoría. Las opciones disponibles son:

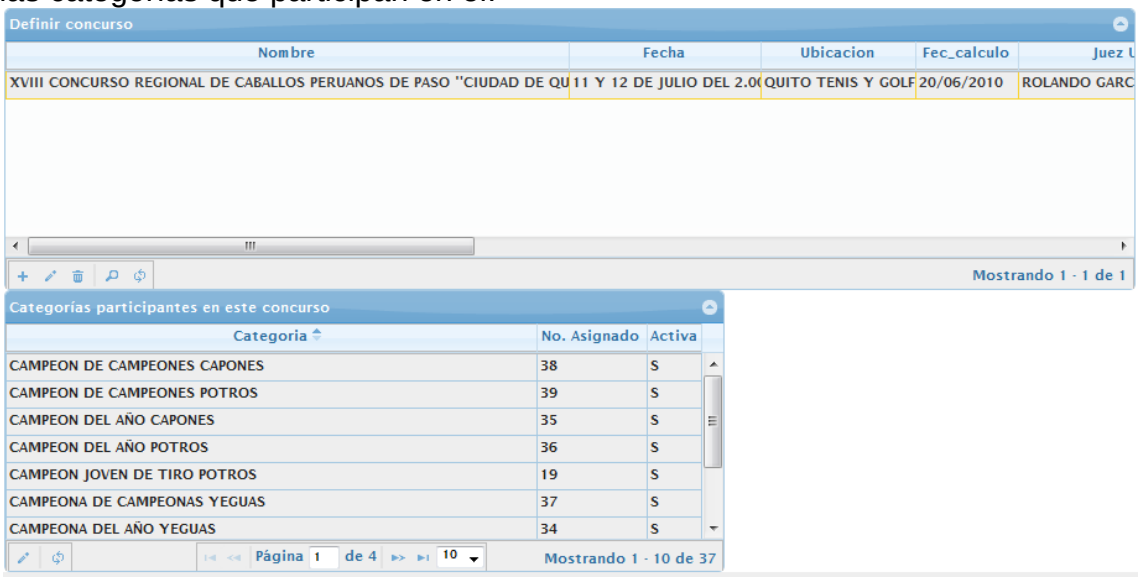
POR LISTA DE PUNTOS.-Los puntos obtenidos son en función de la ubicación y puntos equivalentes definidos en el bloque Lista de puntos.

CAMPEONATO CAPONES.- Si gana la categoría el ganador de la categoría *Campeón del año capones* se otorga 1 punto al ganador y 4 puntos al resto. Si el ganador es el campeón del año anterior se otorga 8 puntos al ganador, al ganador del año 0 puntos y a los demás 4 puntos.

CAMPEONATO YEGUAS Y POTROS.- .- Si gana la categoría el ganador de la categoría *Campeona del año yeguas o Campeón del año potros* se otorga 2 puntos al ganador y 8 puntos al resto. Si el ganador es el campeón del año anterior se otorga 16 puntos al ganador, al ganador del año 0 puntos y a los demás 8 puntos.

NINGUNA.- La categoría no suma puntos.

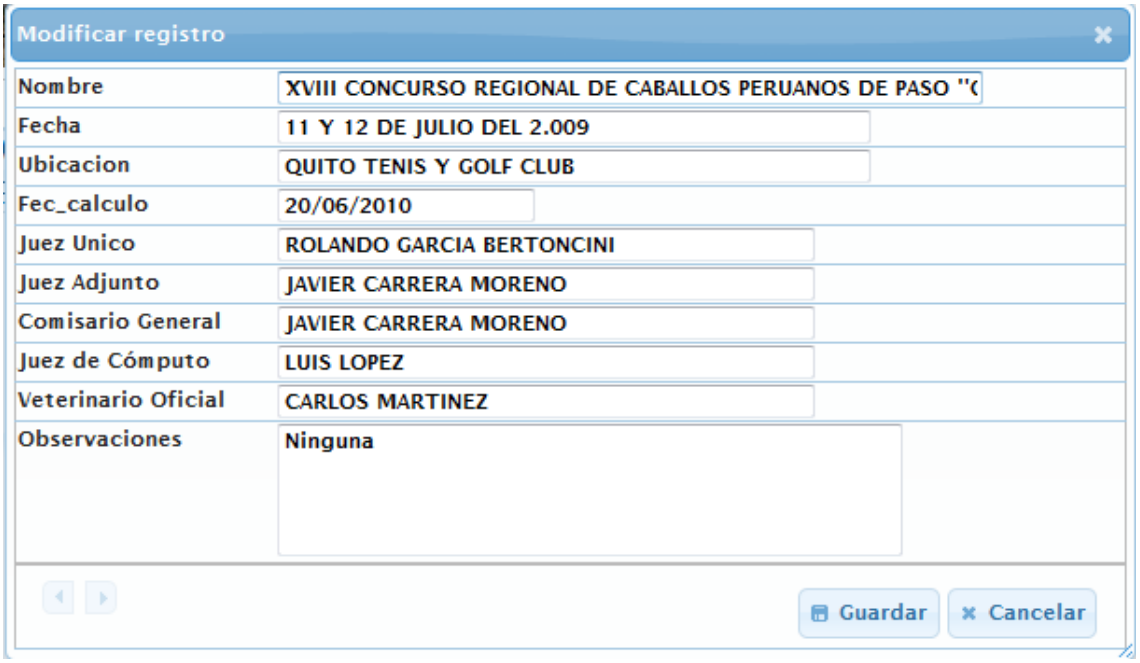
Definir concurso.- Esta opción es sumamente importante ya que se crea el concurso que se realizará. La pantalla esta divida en dos bloques, en el primer bloque se registran los datos generales del concurso y en el segundo las categorías que participan en él.



Para crear un nuevo concurso se presiona el botón de inserción y en el formulario que se muestre se debe registrar la siguiente información:

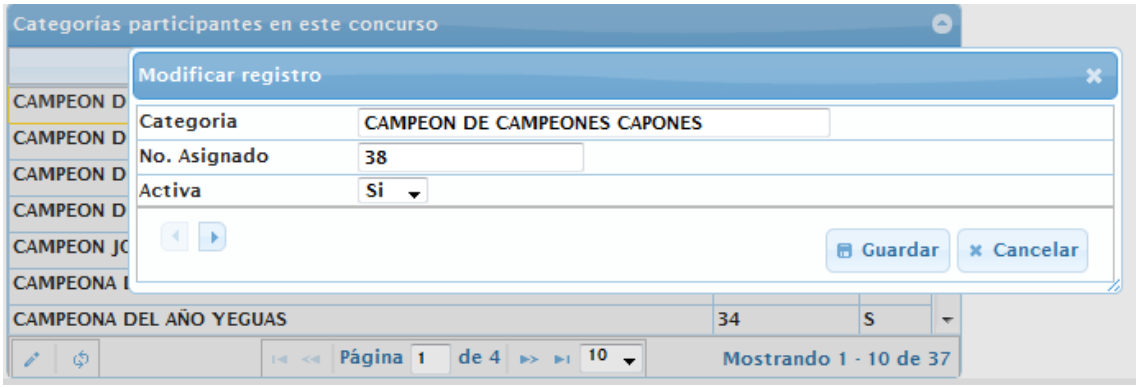
Nombre.- Nombre del concurso.
Fecha.- Información referente a las fechas en las que se realizará el concurso, está en formato abierto.
Ubicación.- Información referente al lugar en el que se realizará el concurso.

Fecha Cálculo.- Se debe registrar la fecha que se tomara de base para el cálculo de la edad de los caballos participantes.
Juez Único.- Nombre del juez único.
Juez Adjunto.- Nombre del juez adjunto.
Comisario General.- Nombre del comisario general.
Juez de Cómputo.- Nombre del juez de cómputo.
Veterinario Oficial.- Nombre del veterinario oficial del concurso.
Observaciones.- Información importante del concurso.



Al finalizar se debe presionar el botón Guardar o Cancelar según sea el caso.

Una vez creado el concurso se debe definir las categorías participantes para lo cual en el bloque de *categorías participantes* de debe presionar el botón de edición y para cada una se debe marcar **Si** en el campo **Activa**, lo que indica que esa categoría es considerada en el concurso creado. También se debe indicar el número asignado de la categoría el mismo que se utiliza en los reportes.



Registro

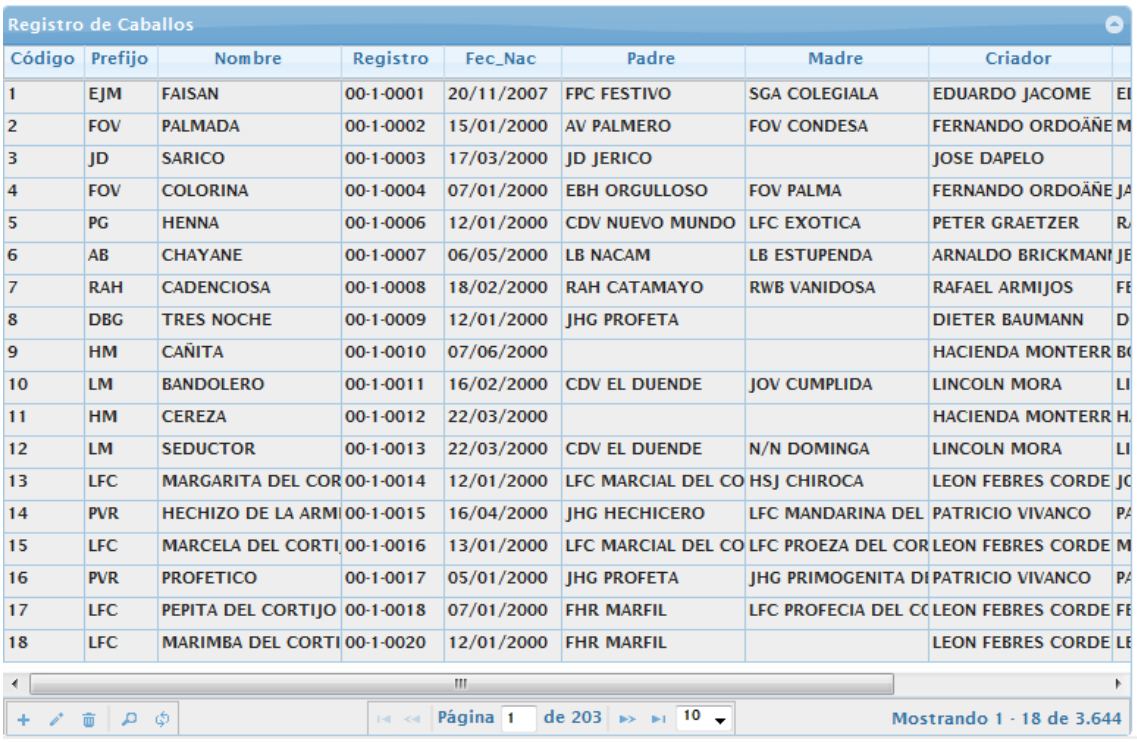
En este menú constan las opciones para el registro de caballos y de las personas que son parte de los concursos.

Registro de Personas.- En esta opción se dispone del formulario que permite el mantenimiento a la base de personas, es decir, crear, actualizar y eliminar personas. Las personas pueden ser naturales o asociaciones así como haciendas.

Al seleccionar la opción de inserción o actualización se muestra el siguiente formulario:

Se ingresa/actualiza la información solicitada y al finalizar se presiona el botón Guardar o Cancelar según el caso.

Registro de Caballos.- En esta opción se dispone del formulario que permite el mantenimiento a la base de caballos, es decir, crear, actualizar y eliminar caballos.



Al seleccionar la opción de inserción o actualización se muestra el siguiente formulario:

En el campo **registro** se debe respetar el formato indicado (##-#-####), caso contrario no se permite el ingreso de esta información.

Se ingresa/actualiza la información solicitada y al finalizar se presiona el botón Guardar o Cancelar según el caso.

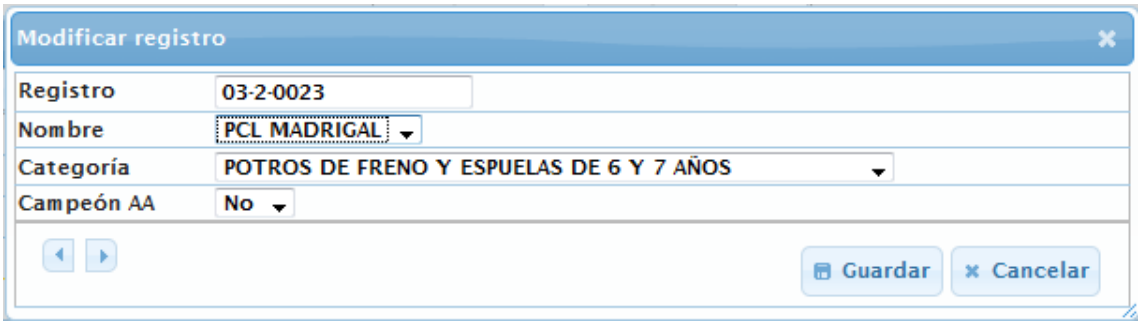
Concurso

En este menú constan las opciones para la gestión propia del concurso.

Inscripciones.- Esta opción consta de dos bloques. En el primero se debe seleccionar el concurso sobre el cual se harán las inscripciones de los caballos en las distintas categorías. En el segundo bloque se realiza el ingreso de los caballos a inscribir.

Si el concurso ya está cerrado no se pueden realizar más inscripciones ni actualizaciones a la información registrada. Un caballo puede estar inscrito en más de una categoría.

Para esto se debe presionar el botón de inserción correspondiente que presenta el siguiente formulario:

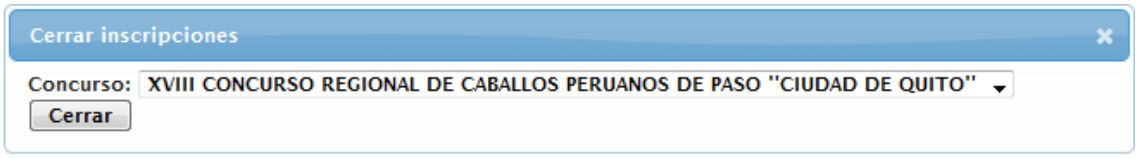


Al ingresar el registro del caballo el sistema carga el nombre y la categoría en la que debe participar en función de la edad, sin embargo, la categoría se puede cambiar. El campo **Campeón AA** indica si el caballo que se está inscribiendo fue el campeón del año anterior lo que se utiliza para el cálculo de los puntos finales obtenidos.

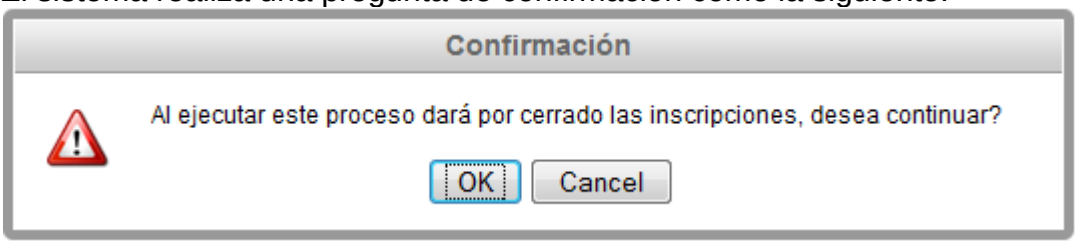
Se ingresa/actualiza la información solicitada y al finalizar se presiona el botón Guardar o Cancelar según el caso.

Cerrar inscripciones.- El cierre de inscripciones asigna el número de participante a todos los caballos inscritos el cual es un solo secuencial con el siguiente orden: yeguas, capones y potros en función de su fecha de nacimiento.

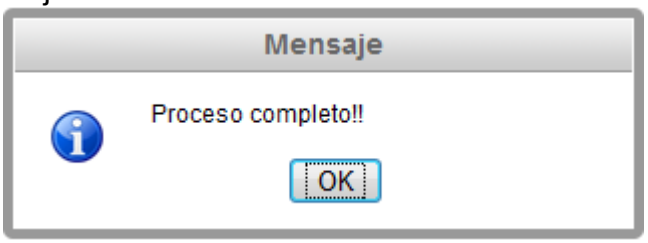
Para realizar el cierre de inscripciones en la pantalla se debe seleccionar en primera instancia el concurso sobre el que se quiere trabajar y luego presionar el botón Cerrar.



El sistema realiza una pregunta de confirmación como la siguiente:

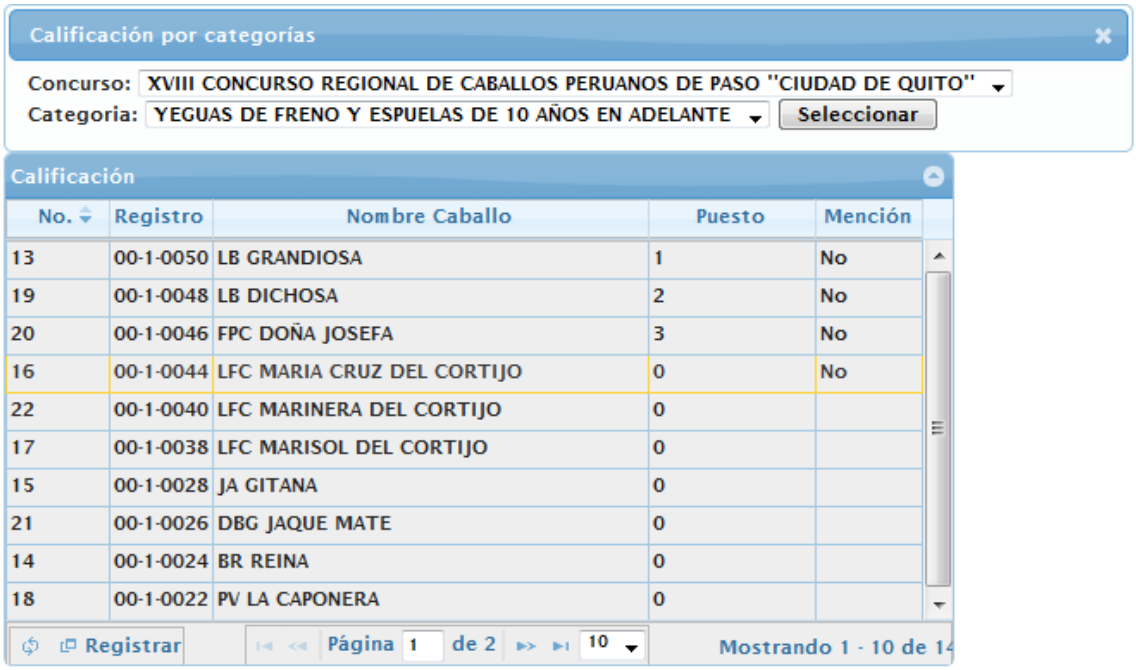


Al presionar el botón Ok se ejecuta el proceso el cual al finalizar presenta el siguiente mensaje.

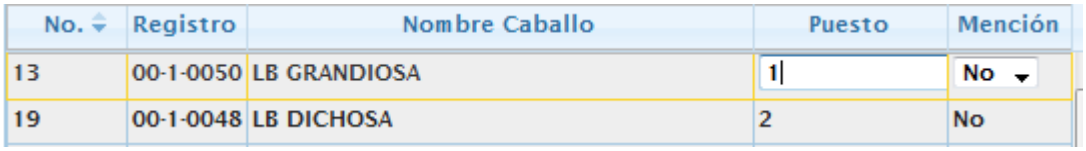


Este proceso únicamente se puede realizar si el concurso no está cerrado ni cerradas las inscripciones.

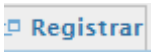
Calificación.- La pantalla de calificación esta divida en dos bloques, en el primero se debe seccionar el concurso y la categoría sobre la que se requiere calificar, una vez hecho esto se debe presiona el botón **Seleccionar** para cargar en el segundo bloque los caballos inscritos.



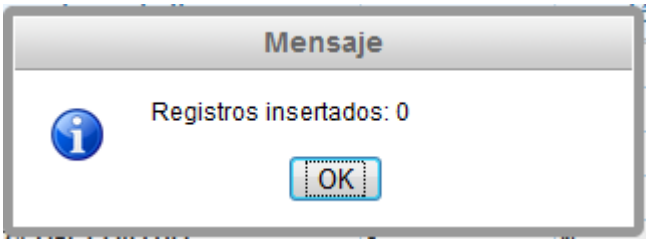
Para calificar un caballo en particular se debe seleccionar el registro correspondiente de manera que se activa para edición el campo Puesto de la siguiente forma:



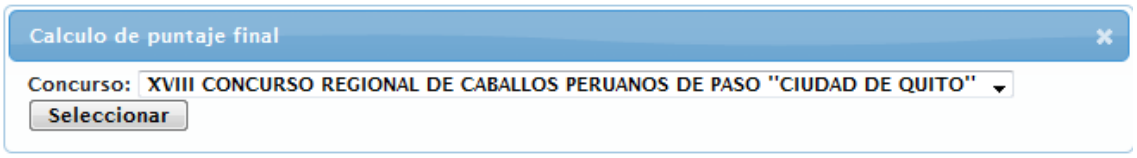
Se ingresa el número correspondiente a la ubicación y se selecciona si tiene o no mención y finalmente se presiona Enter, se repite este proceso por cada caballo. Se debe tomar en cuenta que únicamente se aplican los cambios al presionar la tecla Enter.

En la parte inferior se tiene el botón 

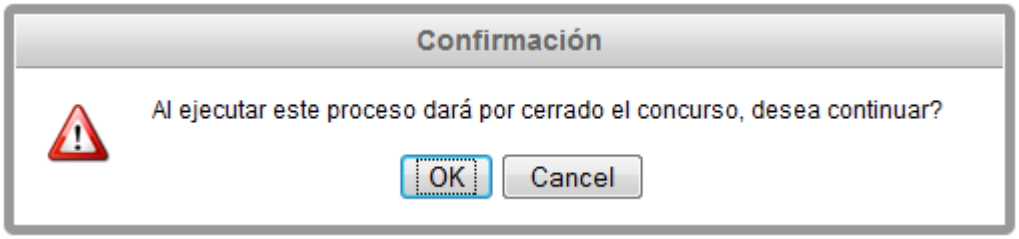
, el cual se lo utiliza para realizar una inscripción tardía de caballos en función de los puestos obtenidos en otras categorías, para este proceso se utiliza la información alimentada en la definición de las categorías (CATEGORIA_CALIFICA, PUESTO_CALIFICA). Únicamente se registran los caballos que no estén inscritos en la categoría seleccionada. Al final de este proceso se presenta un mensaje con el número de caballos inscritos.



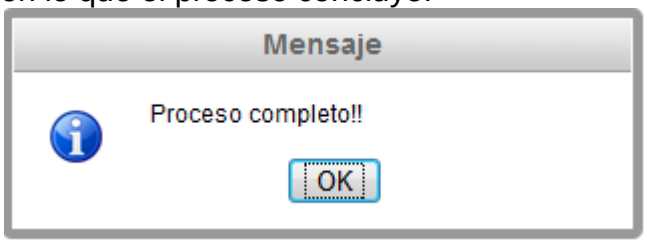
Cierre.- Con esta opción se realiza el cierre del concurso luego de lo cual no se puede realizar ninguna modificación tanto en las ubicaciones como en los caballos inscritos. Como parte del proceso del cierre el sistema realiza el cálculo de los puntajes finales de los caballos, criadores y propietarios en función de las ubicaciones obtenidas en cada categoría. Se tiene la siguiente pantalla:



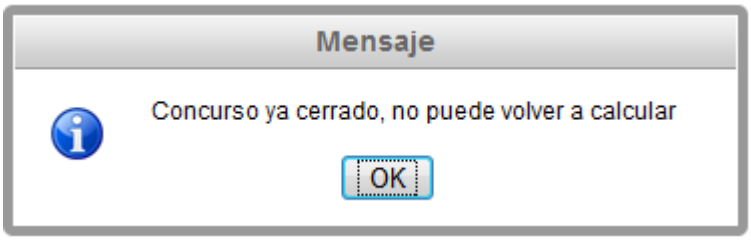
Se debe seleccionar el concurso que se desea cerrar y presionar el botón **Seleccionar,** el sistema hace una pregunta para continuar o no.



Al presionar el botón Ok se ejecuta el cierre del concurso, al final se emite el siguiente mensaje con lo que el proceso concluye.

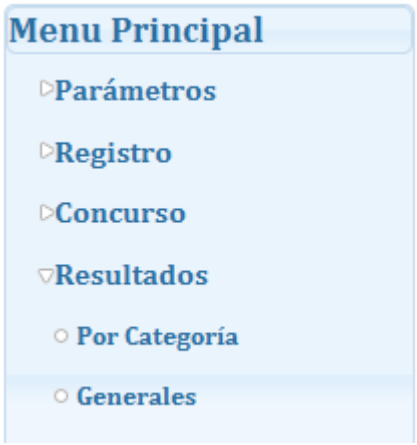


Una vez realizado el cierre no se puede volver a ejecutar este proceso, en caso de intentarlo el sistema emite el siguiente mensaje.

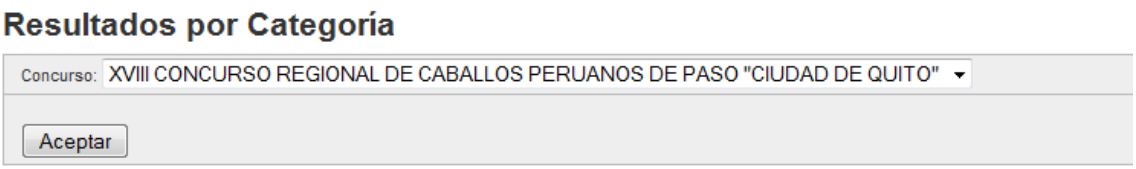


Resultados

En este menú se tiene las opciones necesarias para generar informes sobre los ganadores del concurso.



Por Categoría.- Este es un reporte en PDF que resumen las ubicaciones de cada caballo en las diferentes categorías. Dentro de cada categoría los caballos se ordenan en función de la ubicación de forma ascendente. De igual forma en cada categoría al pie de la página se detallan los caballos que hayan obtenido menciones honrosas. La pantalla permite seleccionar el concurso del que se requiere el informe el mismo que se genera al presionar el botón Aceptar.



Una vez realizado esto se presenta en pantalla el reporte el cual puede ser impreso. El formato del reporte es el siguiente para todas las categorías:

ASOCIACION DE CRIADORES Y PROPIETARIOS DE CABALLOS DE PASO DE PICHINCHA
XVIII CONCURSO REGIONAL DE CABALLOS PERUANOS DE PASO "CIUDAD DE QUITO"

RESULTADOS POR CATEGORIA
Fecha informe: 14-Sep-2010

CATEGORIA YEGUAS DE FRENO Y ESPUELAS DE 4 A 5 AÑOS

No.	PREFIJO	NOMBRE	REGISTRO	CRIADOR	PROPIETARIO	PUESTO
3	JSF	SOLANA	06-1-0132	JORGE FADUL	JORGE FADUL	1
4	JC	BAILARINA	06-1-0118	JACINTO CORDERO	JACINTO CORDERO	2

MENCIONES HONROSAS

5	BOC	LLUVIA	06-1-0104	BOLIVAR OCHOA	

Generales.- Esta opción permite consultar los mejores por tres tipos:

- Mejor criador
- Mejor expositor
- Mejor reproductor

Primero se debe seleccionar el concurso y luego el tipo de consulta.

En la pantalla desplegada en el primer bloque se muestra los ganadores ordenados por el total de puntos obtenidos y en el segundo bloque se detalla los caballos participantes y las calificaciones obtenidas en cada categoría.

Par obtener reportes de los bloques indicados se debe presionar los botones correspondientes de impresión.

Al presionar el botón Imprimir del bloque de ubicaciones obtendremos el siguiente reporte:

ASOCIACION DE CRIADORES Y PROPIETARIOS DE CABALLOS DE PASO DE PICHINCHA
XVIII CONCURSO REGIONAL DE CABALLOS PERUANOS DE PASO "CIUDAD DE QUITO"

Fecha informe: 23-Jan-2011

MEJOR CRIADOR

CRIADOR	PUNTOS	PRIMEROS_PUESTOS
LUIS BAKKER	30	1
PABLO COBO	22	1
LEON FEBRES CORDERO	18	1
JORGE FADUL	12	1
PETER GRAETZER	12	1
JACINTO CORDERO	10	0
CARLOS MARPATIDA	9	0
CARLOS FEIJO	7	2
RAFAEL ARMIJOS	6	1
ESTEBAN QUIROLA	6	1
GALO ORELLANA	5	0
FERNANDO ORDOÃ‚Ã'EZ	0	0
LINCOLN MORA	0	0
PATRICIO VIVANCO	0	0
PEDRO VARAS	0	0
BOLIVAR OCHOA	0	0
BIENVENIDO ROMERO	0	0
DIEGO RIBADENEIRA	0	0
DIETER BAUMANN	0	0
JAIME ALARCON	0	0
JOSE DAPELO	0	1
JULIO HIDALGO	0	0
GERMANICO PAZ	0	0
EDUARDO BALAREZO	0	0
PETER KOECHLIN	0	0
EDUARDO JACOME	0	0

Al presionar el botón Imprimir Detalle se obtendrá el siguiente reporte:

ASOCIACION DE CRIADORES Y PROPIETARIOS DE CABALLOS DE PASO DE PICHINCHA
XVIII CONCURSO REGIONAL DE CABALLOS PERUANOS DE PASO "CIUDAD DE QUITO"

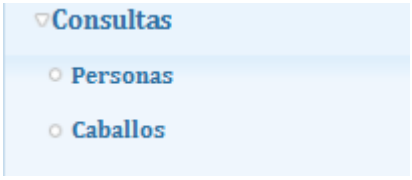
Fecha informe: 23-Jan-2011

CRIADOR: LUIS BAKKER
PUNTOS: 30
PRIMEROS_PUESTOS: 1

REGISTRO	NOMBRE	CATEGORIA	PUNTOS	PUESTO	CALCULO
00-1-0050	LB GRANDIOSA	YEGUAS DE FRENO Y ESPUELAS DE 10 AÑOS EN ADELANTE	12	1	S
00-1-0048	LB DICHOSA	YEGUAS DE FRENO Y ESPUELAS DE 10 AÑOS EN ADELANTE	10	2	N
00-1-0046	FPC DOÑA JOSEFA	YEGUAS DE FRENO Y ESPUELAS DE 10 AÑOS EN ADELANTE	8	3	N
04-2-0088	LB ALTANERA	CAMPEONA DE CAMPEONAS YEGUAS	0	0	S
04-2-0086	LB MELOSA	CAMPEONA DE CAMPEONAS YEGUAS	0	0	N
91-1-1045	LB ABOLENGO	POTROS DE FRENO Y ESPUELAS DE 10 AÑOS EN ADELANTE	0	2	N
00-1-0021	LB TORBELLINO	CAPONES DE FRENO Y ESPUELAS DE 6 AÑOS EN ADELANTE	0	0	N
00-1-0025	LB PODEROSO	POTROS DE FRENO Y ESPUELAS DE 10 AÑOS EN ADELANTE	0	0	S
00-1-0029	LB RECUERDO	CAPONES DE FRENO Y ESPUELAS DE 6 AÑOS EN ADELANTE	0	0	S
00-1-0027	LB LUCERO	POTROS DE FRENO Y ESPUELAS DE 10 AÑOS EN ADELANTE	0	0	N

Consultas

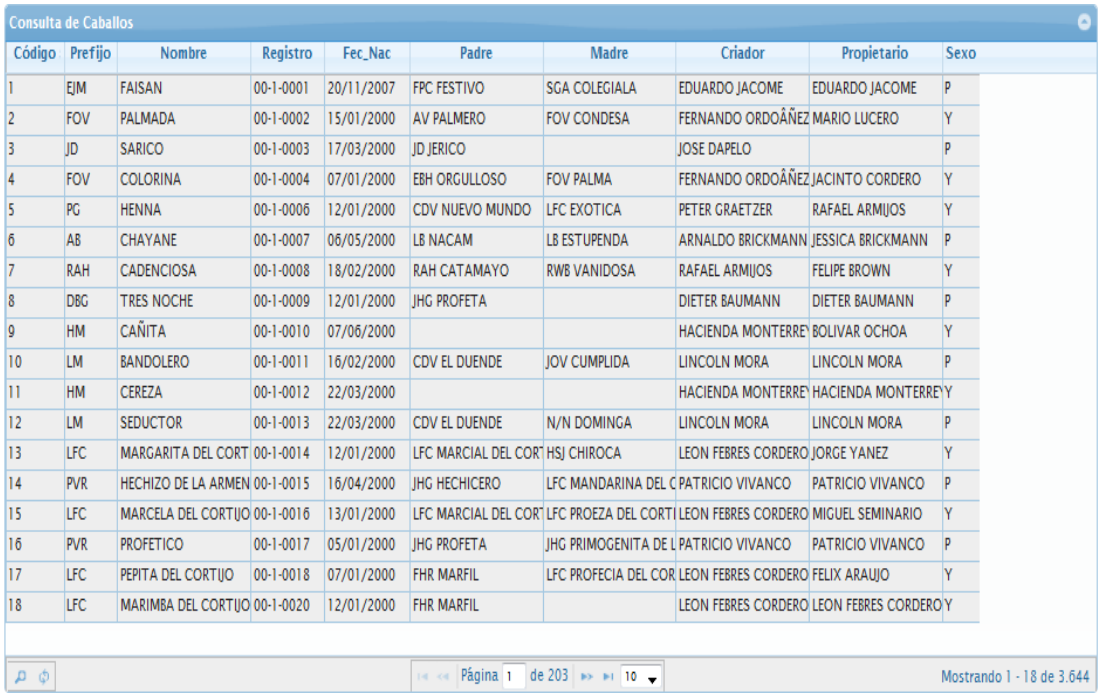
En este menú se tiene las siguientes opciones de consulta:



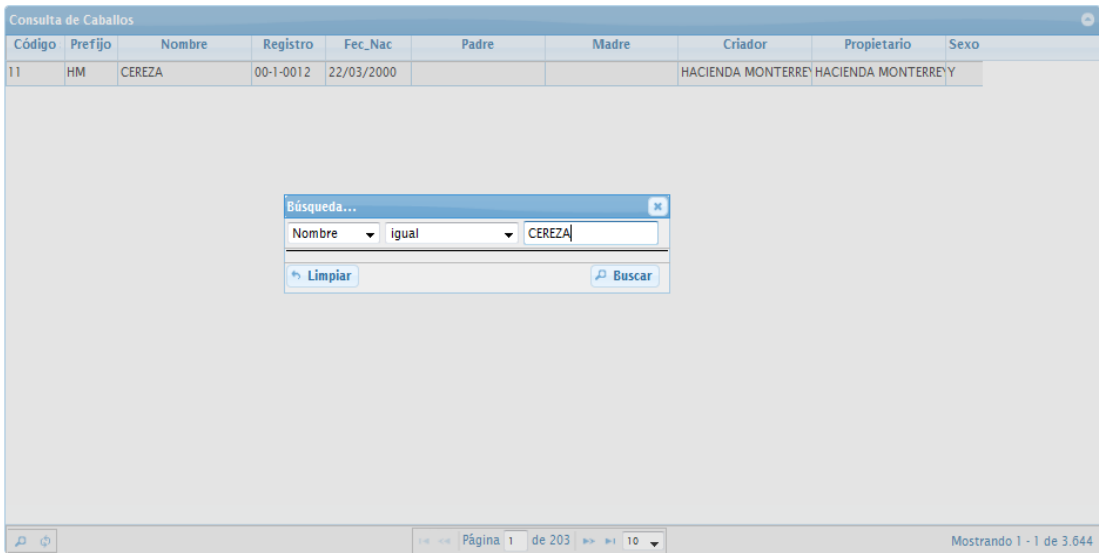
Personas.- Al seleccionar esta opción se carga la pantalla de consulta de las personas registradas, en esta pantalla no se puede modificar la información.

En esta pantalla se dispone de todas las opciones de búsqueda ya explicadas anteriormente para los campos desplegados en la grilla.

Caballos.- Al seleccionar esta opción se carga la pantalla de consulta de los caballos registrados, en esta pantalla no se puede modificar la información.

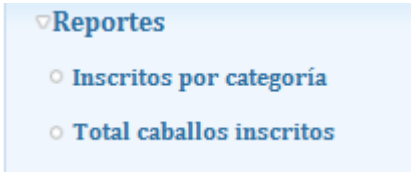


En esta pantalla se dispone de todas las opciones de búsqueda ya explicadas anteriormente para los campos desplegados en la grilla.



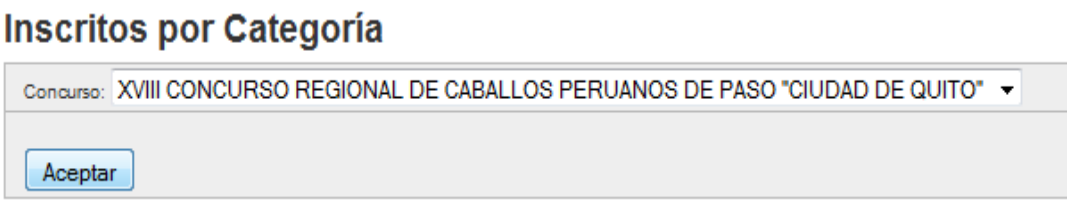
Reportes

En este menú se tiene las siguientes opciones:



Inscritos por categoría.- Esta opción se utiliza para imprimir los caballos inscritos en un determinado concurso. La impresión se realiza por categoría y cada una de ellas en una hoja diferente. Se lo hace así por cuanto este reporte se utiliza para generar los catálogos.

En el formulario inicial se debe seleccionar en primera instancia el concurso y luego presionar el botón Aceptar.



Luego de lo cual se genera el reporte como el que se indica a continuación:

ASOCIACION DE CRIADORES Y PROPIETARIOS DE CABALLOS DE PASO DE PICHINCHA
XVIII CONCURSO REGIONAL DE CABALLOS PERUANOS DE PASO "CIUDAD DE QUITO"
JUEZ UNICO: ROLANDO GARCIA BERTONCINI. JUEZ ADJUNTO: JAVIER CARRERA MORENO
QUITO TENIS Y GOLF CLUB
FECHA: 11 Y 12 DE JULIO DEL 2.009

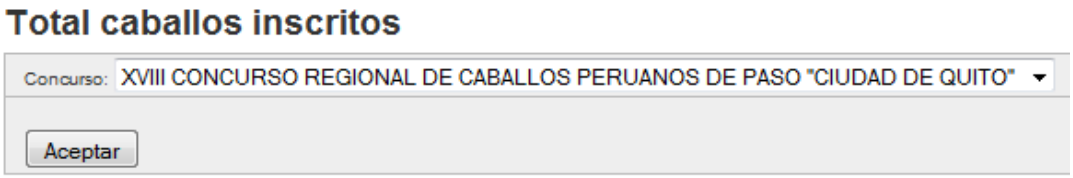
CATEGORIA YEGUAS DE FRENO Y ESPUELAS DE 4 A 5 AÑOS

No.	PREFIJO	NOMBRE	FECHA DE NACIMIENTO	REGISTRO	PADRE	MADRE	CRIADOR	PROPIETARIO	Su Calificación	Calificación del Juez
3	JSF	SOLANA	Abril 12/2006	06-1-0132	JM PRODIGIO	BR REINA	JORGE FADUL	JORGE FADUL		
4	JC	BAILARINA	Marzo 29/2006	06-1-0118	LB VALIENTE II	FOV PALMA	JACINTO CORDERO	JACINTO CORDERO		
5	BOC	LLUVIA	Febrero 26/2006	06-1-0104	GO DESLIZ	DAG MICAELA	BOLIVAR OCHOA			

El orden de las categorías y de los caballos dentro de cada una de ellas se lo realiza por fecha de nacimiento.

Total caballos inscritos.- Esta opción se utiliza para imprimir los caballos inscritos en un determinado concurso.

En el formulario inicial se debe seleccionar en primera instancia el concurso y luego presionar el botón Aceptar.



Luego de lo cual se genera el reporte como el que se indica a continuación:

ASOCIACION DE CRIADORES Y PROPIETARIOS DE CABALLOS DE PASO DE PICHINCHA
XVIII CONCURSO REGIONAL DE CABALLOS PERUANOS DE PASO "CIUDAD DE QUITO"
JUEZ UNICO: ROLANDO GARCIA BERTONCINI. JUEZ ADJUNTO: JAVIER CARRERA MORENO
QUITO TENIS Y GOLF CLUB
FECHA: 11 Y 12 DE JULIO DEL 2.009

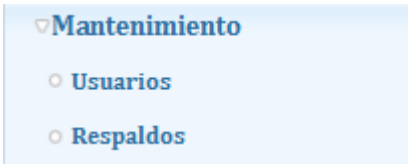
LISTADO TOTAL DE CABALLOS INSCRITOS Página: 1

No.	PREFIJO	NOMBRE	FECHA DE NACIMIENTO	REGISTRO	PADRE	MADRE	CRIADOR	PROPIETARIO	CATEGORIA
1	RAH	FAROLERA	Enero 27/2007	07-1-0124	PK FLAMENCO	RAH FRENESI	RAFAEL ARMIJOS	RAFAEL ARMIJOS	POTRANCAS DE TIRO DE
2	GP	DUDOSA	Junio 26/2006	06-1-0130	GO NAVEGANTE	GP DULCINEA	GERMANICO PAZ	GERMANICO PAZ	CAMPEONA JOVEN DE TI
3	JSF	SOLANA	Abril 12/2006	06-1-0132	JM PRODIGIO	BR REINA	JORGE FADUL	JORGE FADUL	YEGUAS DE FRENO Y ES
4	JC	BAILARINA	Marzo 29/2006	06-1-0116	LB VALIENTE II	FOV PALMA	JACINTO CORDERO	JACINTO CORDERO	YEGUAS DE FRENO Y ES
5	BOC	LLUVIA	Febrero 26/2006	06-1-0104	GO DESLIZ	DAG MICAELA	BOLIVAR OCHOA		YEGUAS DE FRENO Y ES
6	EQF	LUNA	Noviembre 24/2004	04-2-0090	EQF LEON	EQF RECELOSA	ESTEBAN QUIROLA	ESTEBAN QUIROLA	CAMPEONA DE CAMPEONA
7	LB	ALTANERA	Noviembre 13/2004	04-2-0086	JYEP PRIVILEGIO	LB DOÑA SEVILLA	LUIS BAKKER	LUIS BAKKER	CAMPEONA DE CAMPEONA
8	LB	MELOSA	Noviembre 07/2004	04-2-0086	JYEP PRIVILEGIO	LB MAGIA BLANCA	LUIS BAKKER	LUIS CABRERA	CAMPEONA DE CAMPEONA
9	GP	CANTORA	Agosto 07/2004	04-2-0082	GO ESPANTO	PC CASANDRA	GERMANICO PAZ	GERMANICO PAZ	CAMPEONA DE CAMPEONA
10	EBH	LUNA DE PAITA II	Julio 06/2000	00-2-0130	EBH ULISES	EBH RAYO DE LUNA	EDUARDO BALAREZO	EDUARDO BALAREZO	YEGUAS DE FRENO Y ES
10	EBH	LUNA DE PAITA II	Julio 06/2000	00-2-0130	EBH ULISES	EBH RAYO DE LUNA	EDUARDO BALAREZO	EDUARDO BALAREZO	CAMPEONA DE CAMPEONA
12	JHG	CARMELA	Junio 02/2000	00-1-0052	JHG PROFETA	AV ESTERLINA PAIJANE	JULIO HIDALGO	JULIO HIDALGO	CAMPEONA DE CAMPEONA
13	LB	GRANDIOSA	Mayo 19/2000	00-1-0050	FPC QUINTO DE ORO	FPC GACELA	LUIS BAKKER	LUIS BAKKER	YEGUAS DE FRENO Y ES
14	BR	REINA	Mayo 18/2000	00-1-0024	JS ALMENDRO	OVCH REINA	BIENVENIDO ROMERO	JORGE FADUL	YEGUAS DE FRENO Y ES
15	JA	GITANA	Mayo 06/2000	00-1-0028	LFC ESCRITOR DEL COR	CAPP CONSEJERA	JAIME ALARCON		YEGUAS DE FRENO Y ES
16	LFC	MARIA CRUZ DEL CORTI	Abril 11/2000	00-1-0044	FHR MARFIL	LFC PRODIGA DEL CORT	LEON FEBRES CORDERO	LEON FEBRES CORDERO	YEGUAS DE FRENO Y ES
16	LFC	MARIA CRUZ DEL CORTI	Abril 11/2000	00-1-0044	FHR MARFIL	LFC PRODIGA DEL CORT	LEON FEBRES CORDERO	LEON FEBRES CORDERO	CAMPEONA JOVEN DE TI

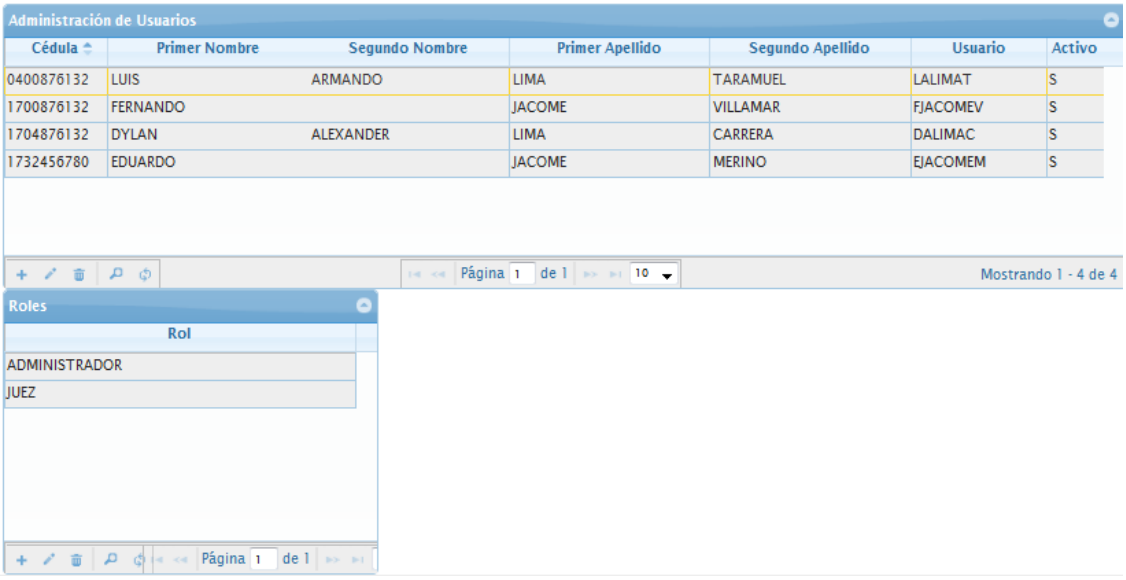
Este reporte esta ordenado por la fecha de nacimiento del caballo inscrito.

Mantenimiento

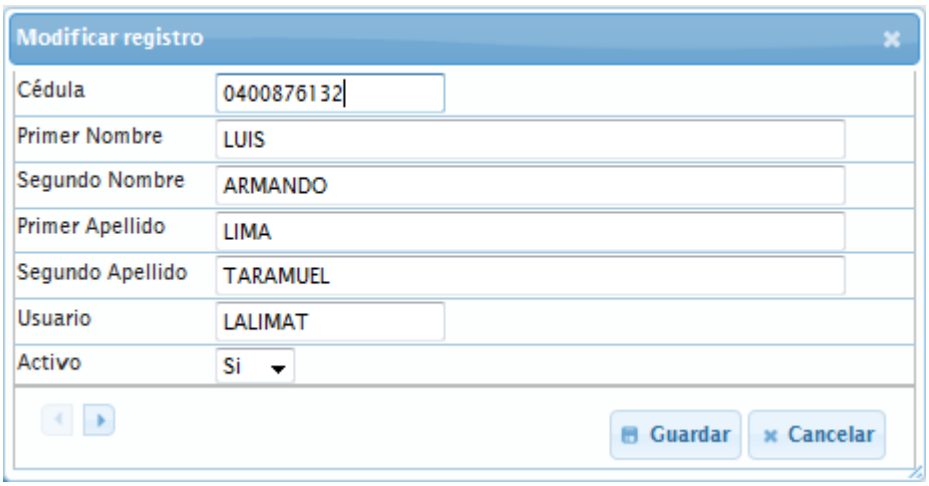
Éste menú está disponible únicamente para el usuario Administrador y en él se tiene las siguientes opciones:



Usuarios.- Esta opción permite gestionar los usuarios y sus roles, es decir, el Administrador del sistema puede crear, eliminar o modificar un usuario. Para realizar esta gestión se dispone de la siguiente pantalla:



Al presionar los botones de inserción, actualización o borrado se permite actualizar la información según se requiera con un formulario estándar de edición como el siguiente:



Descripción:

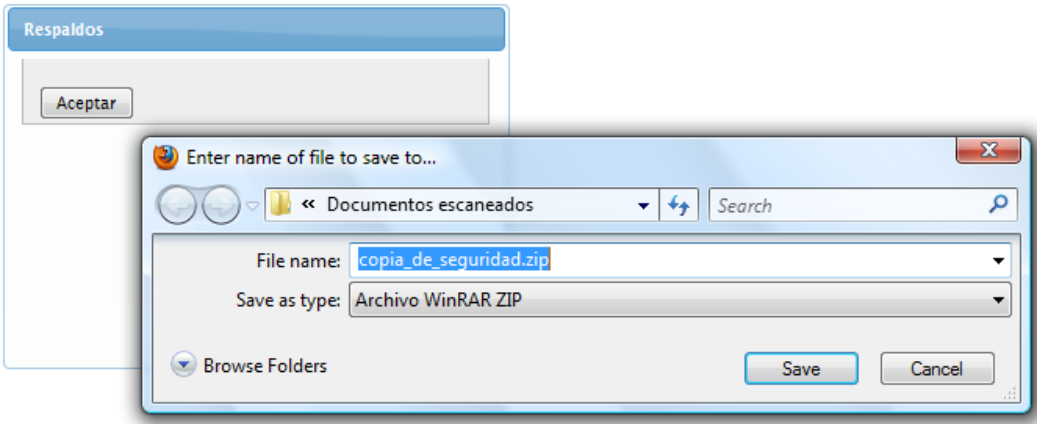
Cédula: Cedula de la persona que se registrará como usuario.
Primer Nombre: Primer nombre del usuario.
Segundo Nombre: Segundo nombre del usuario (opcional).
Primer Apellido: Primer apellido del usuario.
Segundo Apellido: Segundo apellido del usuario (opcional).
Usuario: Es el nombre de usuario el cual debe ser definido por el Administrador.
Activo: Indica si el usuario esta activo o no para el sistema.

En el bloque de Roles se permite asignar el rol requerido por cada usuario, al momento se tienen los siguientes:

- Administrador
- Consultas
- Registro
- Juez

Cada uno de ellos tiene las opciones necesarias para sus funciones.

Respaldos.- Esta opción está disponible para el usuario Administrador y permite obtener respaldos de la base de datos. En la pantalla presentada se presiona el botón Aceptar y el sistema pregunta sobre el directorio y nombre con el que se grabará el respaldo, como se indica a continuación:



Se presiona el botón Save y el proceso del respaldo se concluye. Es responsabilidad del Administrador la custodia y frecuencia de la generación de este archivo y su copia en CDs de respaldo.

Mi perfil

En este menú se tiene la opción que permite el cambio de clave del usuario ingresado en ese momento al sistema.

Cambio de clave.- Esta opción está disponible para todos los usuarios. Para cambiar la clave se tiene el siguiente formulario: