Universidad de las Ciencias Informáticas

UCI.

Facultad 1

Título: Implementación del Sistema de Gestión del Club Bóxer de Cuba.

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Autores:

Addisleidys Castro Osorio.

Orlando Castillo Frómeta.

Tutora:

Ing. Catherine Muñoz Velázquez.

Ciudad de la Habana, 2010.

"Año 52 de la Revolución."

DATOS DE CONTACTO

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Addisleidys Castro Osorio

_____

Firma del autor

Orlando Castillo Frómeta

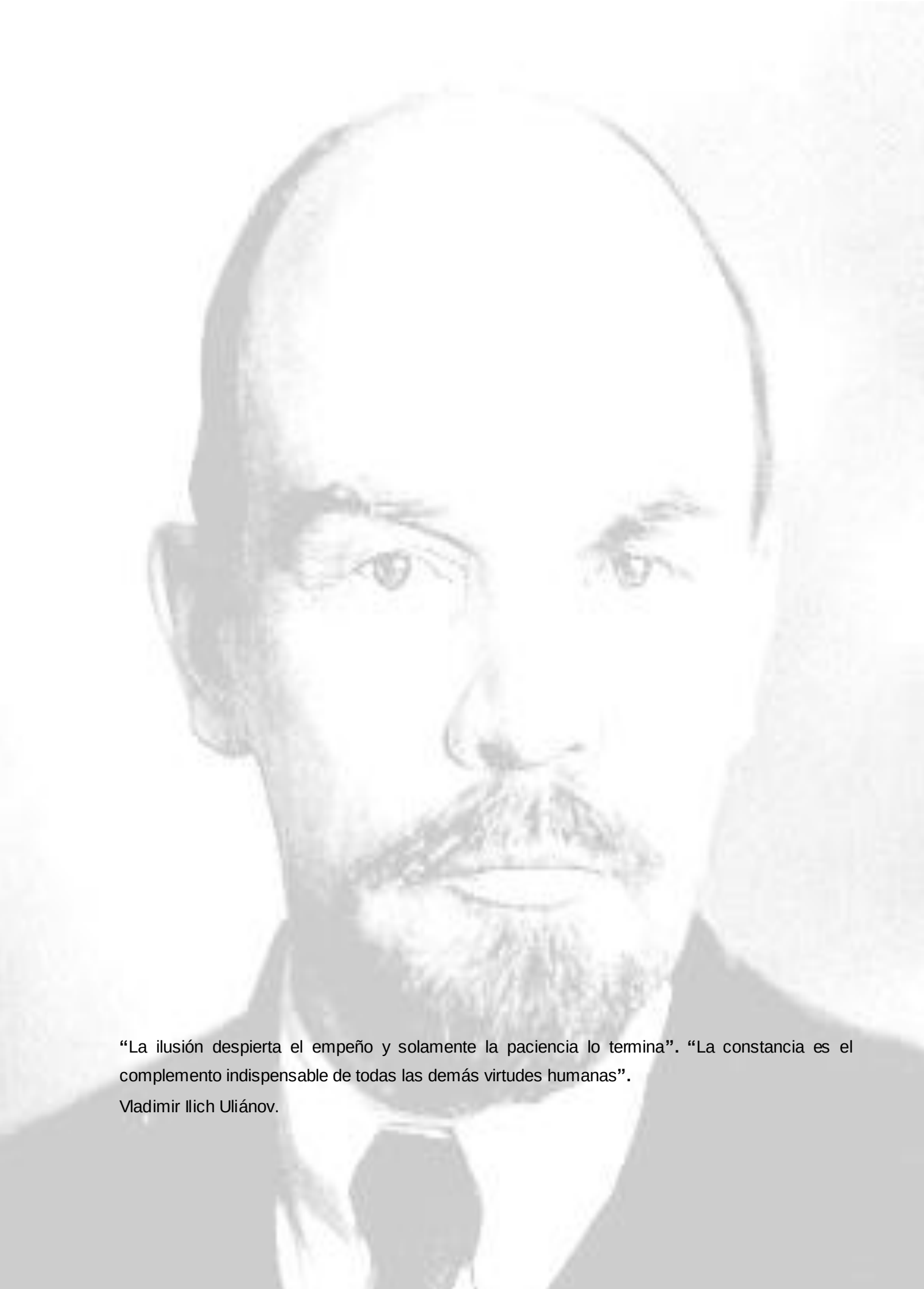
_____

Firma del autor.

Ing. Catherine Muñoz Velázquez

_____

Firma del tutor.



“La ilusión despierta el empeño y solamente la paciencia lo termina**”. “**La constancia es el complemento indispensable de todas las demás virtudes humanas**”.**

Vladimir Ilich Uliánov.

AGRADECIMIENTOS

De Addisleidys:

A mi madre, tu sabes que te amo, gracias por estar siempre, por decirme que si podía aún cuando yo estaba segura que no, por ser mi fortaleza, por todos los sacrificios que has hecho para que yo este hoy aquí, por dedicarme tu vida entera.

A mi tía querida Diosilda, que aunque ya no está entre nosotros, en cualquier parte del cielo que esté, se que hoy se siente orgullosa de mí.

A Yaidy y Daimi por ser mis primis queridas, por ser mis amigas y por ser las hermanas que la vida no me dio, gracias por estar pendiente de mí en toda esta etapa.

A mi tío Orestico por las oraciones y por pedirle a Dios tanto por mí, por siempre darme aliento a seguir.

A Luiso por acompañarnos en los momentos más difíciles de nuestras vidas y por cuidar a mi mamá mientras yo no estaba.

A Made, Sandy y Roliennis, mis amigos del Pre, por confiar en mí, por siempre darme fuerzas para llegar a este día, porque a pesar del tiempo que nos pasamos sin vernos se que hoy están felices por mí.

A Erne, por estar desde primer año soportándome, por toda la ayuda que me diste, sin ti hoy no hubiera logrado mi sueño, por ser mi confidente y por ser más que un amigo, porque para mi eres un hermano, solo espero que aunque la vida nos separe no te olvides de mí, te adoro.

A los que más tiempo pasaron conmigo en esta etapa, a dfresneda por soportar mis pesadeces, por ser una excelente amiga, a yprado la hermana postiza que me conseguí en la universidad, gracias por todos tus consejos, a Y por haber compartido conmigo buena parte de estos 5 años, a Maili porque a pesar de la distancia se conserva la amistad, a Livan por ser tan original y a Enrique por los momentos de alegría que pasamos juntos, fue bueno haberte conocido en esta etapa, porque eres una persona especial para mí.

A mi compañera de cuarto, a Guara por las horas de sueño que le robé, sencillamente porque quería conversar, gracias por escuchar siempre todas mis cosas.

A Michel por su apoyo incondicional, ayuda sin límites, preocupación obsesiva, se que hoy te sientes feliz de verme graduada, porque por mucho tiempo este también fue tu sueño.

A mi compañero de tesis, por soportarme mis perretas, mis quisquillas, fue un placer hacer este trabajo contigo, la tesis me proporcionó un amigo, solo espero que cuando vayas para Guantánamo no me olvides.

A la tía Eloina por ser tan comprensiva, por sus consejos y por la afinidad que surgió entre nosotras, por ser una gran persona.

A mi familia toda, por ser excelentes personas, por ser tan unidos y por quererme tanto.

A la UCI que me acogió sin experiencia alguna de la vida y me devuelve al mundo como profesional, aquí pasé los mejores momentos de mi vida.

A la tutora por su intransigencia, por su empeño y dedicación para que este trabajo saliera con éxito.

A Dole por su excelente labor como oponente, por ser toda una profesional y por su ayuda.

AGRADECIMIENTOS

De Orlando:

A mi madre y a mi padre, que son mi ejemplo a seguir, mis guías, son las personas más importantes y que más quiero en el mundo. Por toda su dedicación, empeño, confianza que depositaron en mí y ayudarme a lograr este sueño de poder regarles esta satisfacción de verme graduado.

A mi hermanito querido, por su preocupación, por compartir tantos momentos de alegría y soportarnos el uno al otro aunque a veces tengamos nuestras diferencias. Por cuidar a mi madre en los dos años que mi papa no estuvo, por los tres.

A mi tía Matha que ha sido en estos 5 años, como mi segunda madre aquí en la Habana, por su preocupación y cariño que me ha brindado. A sus dos hijos que los estimo mucho y sé que a pesar de sus jodederas me hicieron sentir un hermano más

A mis dos abuelitas Hilda y Mercedes a las cuales quiero mucho, por el apoyo que siempre me han brindado, su cariño y por todo lo que me han enseñado. A mis dos abuelos Reinaldo y Fico que a pesar de que hoy no contamos con su presencia física de seguro estarían orgullosos de mí.

A mis compañeros, que compartieron conmigo estos 5 años, por los buenos momentos que pasamos juntos, a todos en general, de todos aprendí algo y siempre los recordaré.

A los socios del vicio, con los cuales compartí grandes momentos, juntos formamos un buen piquete. EL Flaco, Krazz, Sasory, Croqueta, Mulder, Fire, Zeroq, Raro y Enmanuel.

A toda mi familia en general, que los quiero, y no puedo poner nombres porque sino no me alcanza la hojita de los agradecimientos, son muchosssssssssssssssssssssss.

A mi compañera de tesis, que la verdad no sé como me pudo soportar por tanto tiempo, por confiar en mí aún cuando yo le decía que no nos íbamos a graduar, a pesar de sus perretas y en algunas ocasiones sacarme de quicio, la quiero y estimo mucho.

A la tutora, por su ayuda, y empeño en este trabajo, el cuál sin su esfuerzo no se hubiera logrado.

DEDICATORIA

Dedicatoria de Addisleidys:

A mi mamá, tu que fuiste la principal artífice de este sueño, a ti te regalo este triunfo.

A mi tía porque fuiste mi ejemplo, y siempre me alentaste a seguir.

A todos los que confiaron en mí, gracias de todo corazón.

Dedicatoria de Orlando:

A mi madre, a mi padre, a mi hermano y a mi familia en general, a todas las personas que me quieren.

RESUMEN

El Club Bóxer de Cuba, es una institución creada para la cría y selección de la raza bóxer. Actualmente este Club se dedica, en primera instancia, al mejoramiento y perfeccionamiento del estándar de la raza mediante el cruzamiento selectivo.

En este momento en el Club Bóxer de Cuba existe una aplicación de escritorio que permite realizar algunas funcionalidades; pero debido al avance que se ha obtenido en el tema, en este instante se desarrollan muchos procesos y es necesario gestionar una gran cantidad de información para hacer más fácil el trabajo en esta institución.

Por lo que desarrollar una aplicación web que permita automatizar los diferentes procesos que se llevan a cabo en el Club ha sido el objetivo general de está investigación. Se tuvo en cuenta para desarrollar la aplicación la prioridad de las historias de usuario porque fueron realizadas desde la perspectiva del cliente.

Se utilizó SXP como metodología de desarrollo, *Visual Paradigm* como herramienta Case para el modelado, *Symfony* como *framework* de desarrollo, como sistema gestor de base datos PostgreSQL y PHP como lenguaje de programación.

Con el desarrollo de esta aplicación, se eliminó la gestión de información de forma manual, posibilita que exista un control absoluto de todos los ejemplares y asociados que hay en el país. Además, permitirá realizar la búsqueda de ejemplares y asociados de forma rápida, en sentido general se facilita el trabajo en el Club.

ÍNDICE DE CONTENIDOS.

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES.

Introducción.

El origen de la raza canina tuvo lugar en el siglo XIX. Con el avance de los años han surgido hasta la actualidad más de cuatrocientas razas. Una de estas es la raza bóxer que se ha caracterizado por ser una raza de trabajo. En varios países del mundo se han creado clubes dedicados al estudio de está raza, los cuales cuentan con diferentes aplicaciones informáticas para la gestión de sus procesos. En Cuba, hace algunos años se fundó el Club Bóxer, al inicio se contaba con una masa canina pobre, no existían registros genealógicos y se desconocía el estándar europeo. Con el avance de los años el club ha aumentado su capacidad, este se encarga de almacenar toda la información relacionada con este tipo de ejemplar y cuenta hoy con un amplio número de clientes asociados, propietarios y ejemplares. El Club Bóxer actualmente se dedica, en primera instancia, al mejoramiento y perfeccionamiento del estándar de la raza mediante el cruzamiento selectivo, valorando las características fenotípicas y genotípicas del animal y utilizando los procedimientos establecidos, es decir, pruebas estadísticas, pruebas de confirmación (estructura–morfología) y pruebas de caracterología. El Club Bóxer de Cuba se favorece de dos frentes, el frente de genética y el de feria y exposiciones, cada uno se encarga de diferentes actividades pertenecientes al club. En la actualidad en el Club Bóxer Cubano existe una aplicación de escritorio que es la que se está usando en estos momentos; pero hay funcionalidades que se necesitan y el sistema que existe no las tiene implementadas. Actualmente la mayoría de los procesos de gestión de información que se realizan en el Club Bóxer de Cuba son de forma manual, archivándose toda la información en formato duro. El frente de feria y exposiciones organiza competencias nacionales y es el que lleva el control de los ejemplares que participan en los eventos internaciones. El resultado que obtienen los ejemplares en los diferentes eventos que se realizan solo se imprime y archivan. En el club se realizan las pruebas de confirmación a los ejemplares que se encuentran en edad reproductiva para analizar sus características y precisar si están apto para la monta o no y definir qué características debe tener el ejemplar con que se aparee; con este objetivo recientemente se han creado estadísticas para llevar el control de algunas enfermedades hereditarias, que pueden ser valoradas por dos y hasta seis veterinarios, que participan y dan su criterio en las pruebas de confirmación. Igualmente se realizan pruebas de obediencia y temperamento, aunque no muy complejas por falta de recursos y lugares en que se puedan desarrollar; pero se prueba, si el perro es sanguíneo, colérico, melancólico o flemático. En caso de que los ejemplares sean coléricos, está prohibido su cruzamiento con el objetivo de evitar desviaciones negativas de la raza. Los

trabajadores pertenecientes al frente de genética son los encargados de recoger los resultados de las pruebas y archivar la información. Los resultados de cada cruce, se registran en papeles para llevar el control de cuántos cachorros nacieron con defectos; de esta forma se puede en cada cruce, disminuir las posibilidades de defectos del estándar. Respecto a los bóxer blancos, con labio leporino y paladar hendidos, no se permite su nacimiento ni crianza y en caso contrario se practica la eutanasia lo más humanamente posible. Por esta razón es importante llevar el control de cada parto para saber si para la próxima vez se pueden cruzar nuevamente, o si nacieron muchos con características negativas, para no cruzarlos. Expuesta la **situación problémica** existente en el Club Bóxer de Cuba se delimita el siguiente **problema a resolver**: ¿Cómo mejorar la gestión de información en los procesos de control y búsqueda de los ejemplares y asociados en el Club Bóxer de Cuba? En consecuencia, el **Objeto de estudio** es los procesos de gestión de información de los ejemplares y asociados en el Club Bóxer de Cuba y específicamente su **Campo de acción** es las herramientas y tecnologías en el desarrollo de sistemas informáticos de gestión de información para el Club Bóxer de Cuba, para darle respuesta a la situación problémica planteada anteriormente se trazó el siguiente **Objetivo general** de la presente investigación: Implementar un sistema para facilitar la gestión de los principales procesos que se desarrollan en el Club Bóxer de Cuba. Como **objetivos específicos** se tienen:

- Realizar una fundamentación teórica del tema.
- Describir las características del negocio y del sistema.
- Realizar el diseño del sistema a utilizar.
- Implementar la propuesta de solución.
- Realizar las pruebas para verificar el correcto funcionamiento del sistema.

Partiendo de la necesidad que existe en el Club Bóxer de Cuba se trazó la siguiente **Hipótesis:**
Si se implementa una aplicación web centralizada, con un adecuado nivel de acceso, entonces se facilitará la gestión de información en los procesos de control y búsqueda de los ejemplares y asociados en el Club Bóxer de Cuba.
Como **Variable independiente:** se tiene la implementación de una aplicación web centralizada, con un adecuado nivel de acceso y como **Variable dependiente:** gestión de información en los procesos de control y búsqueda de los ejemplares y asociados en el Club Bóxer de Cuba. (Ver anexo1).

Para lograr un eficiente desarrollo de la investigación y darle cumplimiento al objetivo general trazado se ponen en práctica las siguientes **Tareas de la investigación.**

- Estudio del estado del arte relacionado con la gestión de los procesos de cría y selección de especies.
- Estudio de Herramientas de desarrollo.
- Estudio de las Metodologías de desarrollo.
- Identificación de las funcionalidades que tendrá el sistema.
- Diseño de la aplicación web.
- Implementación del sistema.
- Realización de pruebas.

El contenido expuesto en el presente trabajo es el resultado de diversos métodos de investigación, tanto teóricos como empíricos, los cuales se exponen a continuación:

Métodos teóricos.

- **Analítico – sintético**: se ha investigado, leído, buscado documentos y analizados para extraer los elementos más importantes que pudieran estar relacionados con el objeto de estudio y se ha sintetizado la información encontrada, a través del método.
- **Análisis histórico -lógico**: se investigó y se siguió la trayectoria que ha tenido el Club Bóxer de Cuba desde sus inicios, se ha visto su evolución y crecimiento, la esencia del problema, la necesidad de mejorar este sistema.

Métodos empíricos.

- **Entrevista:** se entrevistaron a los trabajadores del Club Bóxer de Cuba, para conocer la información detallada de este centro, se han realizado conversaciones planificadas para obtener la información requerida.
- **Observación:** se ha observado cómo funciona el sistema que existe actualmente en el club, se analizaron cuales son sus principales desventajas y cuales funcionalidades sería conveniente que tuviera implementadas.

El **contenido de esta investigación** se encuentra estructurado de la siguiente forma:

Capítulo I: En este capítulo se efectúa un estudio de sistemas similares existentes, se hace mención de las características de las diferentes herramientas, metodologías, lenguajes y tecnologías a utilizar en el desarrollo de la investigación.

Capítulo II: Este capítulo contiene los principales procesos que se realizan en el Club Bóxer de Cuba así como el flujo normal de los mismos. Aborda los aspectos funcionales para el desarrollo del sistema, se definen los procesos fundamentales por medio de las historias de usuarios creadas por el cliente y se realizó el plan de iteraciones.

Capítulo III: Este capítulo contiene los aspectos relacionados con el diseño, implementación y pruebas del sistema. Se presenta la técnica de tarjetas CRC, el modelo de datos y además las tareas de ingeniería por cada historia de usuario para garantizar la entrega en la planificación establecida. Se realizan las pruebas de aceptación para comprobar el funcionamiento del sistema.

Capítulo IV: Se realiza el estudio de la factibilidad utilizando el método COCOMO II para comprobar si será factible el sistema.

Capítulo1: Fundamentación Teórica.

1.1 Introducción.

En el presente capítulo se realizará un análisis profundo de las diversas metodologías y herramientas, para definir cuáles se van a utilizar, identificando las características que hicieron posible la elección de las que se utilizarán a lo largo del desarrollo del trabajo. Además, se realizará un estudio de sistemas similares existentes vinculados al campo de acción.

1.2 Estado del arte.

El desarrollo actual de la informática ha posibilitado que se automaticen diversos procesos, tal es el caso de los sistemas para el control de la cría y selección de diferentes especies. En el mundo se han desarrollado diversos sistemas con este fin y gracias además a la socialización del conocimiento, los logros alcanzados han sido significativos, posibilitando que día a día sean más eficientes y respondan con más efectividad a los requerimientos de los clientes.

Nivel Internacional.

Gescab: Software para la cría del caballo español.

El software Gescab es para la cría de caballo español, se desarrolló en España y puede ser utilizado por cualquier ganadería equina de caballo de pura raza española, entre sus principales características están que gestiona la información de cualquier ganadería equina de caballo, da a conocer un informe de ascendencia con el cual se puede saber qué porcentaje de cada hierro tiene el ejemplar que se quiera estudiar. Este informe se puede solicitar con un nivel de profundidad en el pedigrí de 16 generaciones, permite conocer un listado detallado de todos los ejemplares que posee una ganadería, además permite gracias al algoritmo Lush(1940)determinar el porcentaje de consanguinidad(sangre común) de cualquier animal, cada ejemplar dispone de un calendario diario donde se puede realizar anotaciones como las fechas de cubriciones, partos, ecografías y tratamientos sanitarios, se caracteriza por tener un informe que va a contar con un aproximado de la cantidad de sementales en función del número de descendientes, determina los ejemplares más idóneos para la compra-venta o para planear el programa de apareamientos (cubriciones) en base las características del ejemplar relacionadas con el nivel de consanguinidad respecto a los antecesores o descendientes, nivel de parentesco entre individuos, porcentaje de ancestros conocidos, porcentaje de los distintos hierros y ganaderías de los antecesores, color de la capa de antecesores y descendientes. Muchas de estas funcionalidades no son de utilidad para nuestro sistema, por ejemplo Gescab dispone de un calendario diario de cada ejemplar para saber las fechas de cubriciones, partos, ecografías y

tratamientos sanitarios, en el sistema a los ejemplares bóxer no se le monitorea la gestación, pues la misma es de un tiempo muy corto, el Club Bóxer de Cuba solo se preocupa por el o los días de la monta, saber si resultó o no gestada la ejemplar, luego de concebir, conocer el estado de los cachorros, la madre y del criadero.

Sistema de la dinámica de rodeo de cría bovino (SDR).

Este sistema de la dinámica de rodeo de cría bovino fue desarrollado por Proyecto Fonsoft Emprendedores, tiene como objetivo describir el grado de avance del desarrollo de un sistema de la dinámica de rodeo de cría bovino (SDR) y de los recursos de alimentación para su aplicación a la planificación productiva y económica, entre sus principales características están que presenta un diseño de la dinámica anual del rodeo se trata de un flujo dinámico, que ocurre en el tiempo, en el cual se van sucediendo eventos y mediciones periódicas como partos, servicio, destete, diagnóstico de gestación, se van logrando eficiencias parciales sobre la tasa de preñez, tasa de parición, tasa de destete y se registran pérdidas, medidas por ejemplo, a través de la tasa de mortandad , tasa porcentual de merma tacto-parto, tasa de merma parto-destete, tasa de descarte, todo lo cual, al acumularse, va aportando al logro de la eficiencia integral del proceso, se trabaja con enfoque de Programación Orientada a Objetos (POO) en el lenguaje Java SE4 usando NetBIOS, este sistema se encuentra parcialmente implementado y continúa avanzando su desarrollo en el marco del proyecto que lo integra, la herramienta fue concebida como un “Lizardi” (o asistente), diseñado como una herramienta de ayuda automática que guía al usuario paso a paso durante su uso, tratando de subsanar las debilidades de usabilidad que otras herramientas del dominio agropecuario presentan, el sistema de la dinámica de rodeo incorpora un módulo de análisis de sensibilidad (en desarrollo) tanto para la variable precio de venta de la hacienda (categorías terneros y vacas) como para las tasas, además participa en la planificación anual de una empresa ganadera (a través del presupuesto económico, financiero y forrajero) y posibilita la formulación de proyectos de inversión al elaborar flujos plurianuales.

-Se caracteriza por la creación de dos tipos de modelos:

- Modelo Rodeo Estabilizado (dinámica anual) Un sistema ganadero estable es aquel que se repite con absoluta regularidad a lo largo de los años, donde se mantiene constante entre años el número de cabezas de cada categoría, su peso y requerimientos, y lo mismo ocurre con las tasas reproductivas.
- Modelo Rodeo en Transición o no estabilizado (proyección plurianual).

Esta dinámica permite considerar tanto situaciones de crecimiento del rodeo (en donde los vientres a servicio van aumentando de un año a otro, es decir, donde la reposición efectiva supera a la reposición de equilibrio, por lo que la diferencia de inventario resulta positiva) como de disminución del mismo (diferencia de inventario negativa, reposición efectiva menor a la de equilibrio).

Software de Genealogía.

Este *software* se desarrolló en la Universidad de Buenos Aires-Argentina, el objetivo que generó esta investigación consistió en analizar la dinámica productiva y reproductiva de la comunidad, específicamente ubicada en lo que es hoy la reserva Quilchamar, entre sus principales características están que se trabaja con programas de ARS (*Software* de Gestión) como el UCINET y PAJET programas muy usados en los análisis de redes sociales, donde las diferentes representaciones generadas pueden almacenarse tanto en archivos de Excel como en archivos de texto con formato nativo directamente compatibles con los programas nombrados, este *software* corre bajo Windows desde una versión 95 en adelante, es fácilmente instalable y no requiere programas adicionales como soporte o complemento, tiene un nivel de profundidad de 3 generaciones como máximo, estas interfaces permiten ingresar información paternal y visualizarla gráficamente de manera inmediata, la simbología usada es muy simple. Un triángulo representa un miembro masculino de la familia y un círculo un miembro femenino y este programa genera una matriz parental directamente procesable por UCINET y PAJET. Este sistema no es multiplataforma, porque solo corre bajo Windows desde una versión 95 en adelante. Este *software* utiliza Access como base dato; pero Access no es recomendable para bases de datos de gran tamaño (en cuanto a volumen de datos o de usuarios), es recomendable usar otros sistemas gestores de base dato, además no crea un compilado real (un .EXE). Siempre necesita tener instalado Access para su funcionamiento.

Nivel Nacional.

***Software* para la gestión del Club Bóxer de Cuba.**

Este *software* se desarrolló en Cuba y actualmente se utiliza en el Club Bóxer de Cuba, entre sus principales características está que permite agregar ejemplares en edad reproductiva, descendientes de un ejemplar, los ejemplares criados y propios de un asociado. Posibilita la gestión de los clientes asociados y los ejemplares, además cuando un cliente asociado pasa a ser propietario de un ejemplar, este sistema permite generar diferentes documentos, como propiedad del ejemplar y folio del ejemplar. Este sistema es una aplicación de escritorio,

desarrollada utilizando *Csharp* como lenguaje de programación, y *Microsoft SQL Server*, como sistema gestor de base dato.

1.2.1 Deficiencias de los sistemas estudiados.

- Los sistemas estudiados están desarrollados con tecnología propietaria, este tipo de *software*, tiene como característica que destacan la manutención de la reserva de derechos sobre su uso, su modificación y redistribución. Para el sistema a desarrollar se utilizarán herramientas basadas en el *software* libre. Cuba tiene el objetivo de migrar al *software* libre poco a poco; pero la cuestión no es declarar al *software* privativo obsoleto e ineficaz cuando en realidad posee diferentes ventajas sobre el *software* libre, sino abogar por el uso de un sistema más justo y equitativo en el plano socio económico de las nuevas tecnologías. El *software* libre, tiene entre sus ventajas que puede ser distribuido, modificado, copiado y usado, es una necesidad impostergable para un país como Cuba que se ve imposibilitado muchas veces de la adquisición de *software* propietario, o de conocer cómo funcionan aquellos que se adquieren.
- Estos sistemas analizados son aplicaciones de escritorio que tiene la particularidad de ejecutarse totalmente con los recursos de la máquina. No utilizan un servidor, necesitan ser instalados en cada máquina donde se pretenda utilizar el *software*. En el caso que se vaya a realizar una actualización al *software*, se tiene que instalar nuevamente en todas las máquinas, a diferencia de las aplicaciones web, que los clientes y usuarios pueden acceder a ella en el momento que lo requieran con el único requisito de tener instalado un navegador web. Además, estas aplicaciones facilitan la actualización y mantenimiento pues no es necesario distribuir e instalar *software* en las PC clientes, también posibilita que la información esté centralizada, y evita dependencia de *software*, de *hardware* así como de sistema operativo.

1.3 Metodologías, herramientas y lenguaje de modelado.

1.3.1 Metodologías de desarrollo de *software*.

La metodología en el desarrollo de un *software* se puede definir como un conjunto de pasos y procedimientos que sirven de apoyo para realizar un *software* con calidad. En un proyecto de *software* la metodología de desarrollo define ¿quién debe hacer qué?, ¿cuándo? y ¿cómo?, debe realizarlo. Todo proceso de desarrollo de *software* es riesgoso y difícil de controlar. Si no se sigue una metodología, aparecen al final del proyecto clientes insatisfechos con el resultado y desarrolladores aún más insatisfechos.

1.3.1.1 Proceso unificado de desarrollo del *software*. (RUP)

El Proceso Unificado de Desarrollo es una propuesta tradicional que se centra especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, las herramientas y notaciones que se usarán, agrupa las actividades en grupos lógicos definidos en nueve flujos de trabajo principales. Los seis primeros son conocidos como flujos de ingeniería y son Modelamiento del negocio, Requerimientos, Análisis y diseño, Implementación, Prueba y Despliegue y los tres últimos como flujos de apoyo y son Gestión de proyecto, Gestión de configuración y Entorno. RUP en dos dimensiones representa el proceso en el que se grafican los flujos de trabajo y las fases y muestra la dinámica expresada en iteraciones y puntos de control. Divide en cuatro fases el desarrollo del *software* Inicio, Elaboración, Construcción y Transición en cada una se obtiene un producto final y cada una está desarrollada mediante el ciclo de iteraciones, las cuales tienen como función la evaluación de las iteraciones precedentes. [1]

Características principales de RUP.

- Guiado por casos de uso.
- Centrado en la arquitectura.
- Iterativo e incremental.

Ventajas de RUP.

- Se basa en las mejores prácticas que se han intentado y se han probado en el campo de la ingeniería.
- Mitigación temprana de posibles riesgos altos.
- Progreso visible en las primeras etapas.
- Temprana retroalimentación que se ajusta a las necesidades reales.
- Gestión de la complejidad.
- Conocimiento adquirido en una iteración puede aplicarse de iteración a iteración.

1.3.1.2 Desarrollo basado en funcionalidades. (FDD)

Desarrollo basado en funcionalidades es una metodología ágil, está pensada para proyectos de desarrollo relativamente cortos (menos de un año). Se basa en un proceso iterativo con iteraciones cortas (aproximadamente dos semanas) que produce un *software* funcional que el cliente y la dirección de la empresa pueden ver y monitorizar. No hace énfasis en la obtención de los requerimientos sino en cómo se realizan las fases de diseño y construcción. Ayuda al equipo

a producir resultados periódicos y tangibles. Esta metodología utiliza pequeños bloques llamados funcionalidades, los cuales contienen la funcionalidad del sistema. Asegura en gran parte la calidad del *software* entregado. Es adaptable, pues permite realizar cambios de último momento debido a nuevos requerimientos y a las necesidades del negocio.

El proyecto que sigue FDD se divide en 5 fases.

- Desarrollo de un modelo general.
- Construcción de la lista de funcionalidades.
- Plan de comunicado sobre la base de las funcionalidades a implementar.
- Diseño sobre la base de las funcionalidades.
- Implementar sobre la base de las funcionalidades.

FDD define métricas para seguir el proceso de desarrollo de la aplicación, útiles para el cliente y la dirección de la empresa y que pueden ayudar además de, para conocer el estado actual del desarrollo, a realizar mejores estimaciones en proyectos futuros.

Características principales de FDD.

- Es ligero, no produce demasiados gastos sobre las actividades de desarrollo, y no impide el avance del proyecto.
- Existe una jerarquía dentro del equipo.
- El código fuente tiene propietario.
- Los equipos varían en función de la funcionalidad a implementar.
- El conocimiento de la aplicación se reparte a través del trabajo en equipo y revisiones.
- Documentación aceptable.

1.3.1.3 SXP.

Esta metodología surgió en Cuba, en la Universidad de las Ciencias informáticas (UCI), con el objetivo de lograr un buen desarrollo de *software*. Es el resultado de la unión de las metodologías ágiles Scrum y XP, estas metodologías se acoplan y retroalimentan perfectamente una dentro de la otra. Scrum modela la interfaz entre el equipo y los clientes y XP entra a funcionar en como el equipo debe hacer su trabajo. De esta manera, se integran efectivamente, surgiendo así la metodología de procedimiento ágil a seguir SXP, la cual cuenta para su desarrollo con cuatro fases, estas son: Planificación-Definición, Desarrollo, Entrega y Mantenimiento. Cada una de estas fases está compuesta por una serie de actividades que son las que generan los artefactos que quedan incluidos en el nuevo expediente de proyecto que se genera.

Planificación–Definición: En esta fase se generan todos los documentos que se encuentran relacionados con la concepción inicial del sistema, así como la definición del mismo. También se incluyen algunos que están vinculados a la primera parte de los procesos de ingeniería de *software* tales como los relacionados con el negocio, los requisitos y el diseño.

Desarrollo: En la primera parte de esta fase se generan todos los documentos relacionados con la planificación de las iteraciones, y además se recogen las principales definiciones que se manejan en la metodología y otros términos de difícil entendimiento para los clientes, así como de las tareas a realizar durante la implementación. Además, se genera el código fuente en la etapa de implementación, los documentos relacionados con las pruebas, como última parte de esta etapa.

Entrega: En esta fase se realiza la entrega del *software* y su documentación, generándose aquellos documentos que son imprescindibles para el entrenamiento y entendimiento del producto.

Mantenimiento: Se realizan las actividades relacionadas con el soporte del *software* y se generan los documentos relacionados con los cambios que puedan ocurrir en el mismo.

Para la definición de los artefactos que se generan en cada una de las fases, se tiene en cuenta como elemento fundamental las características de las metodologías ágiles, las cuales tienen como premisa la no duplicación de esfuerzos. Así como la integración del cliente en el equipo de desarrollo, esto garantiza que no halla necesidad de documentaciones extensas, solo se documenta lo necesario para una futura reutilización.

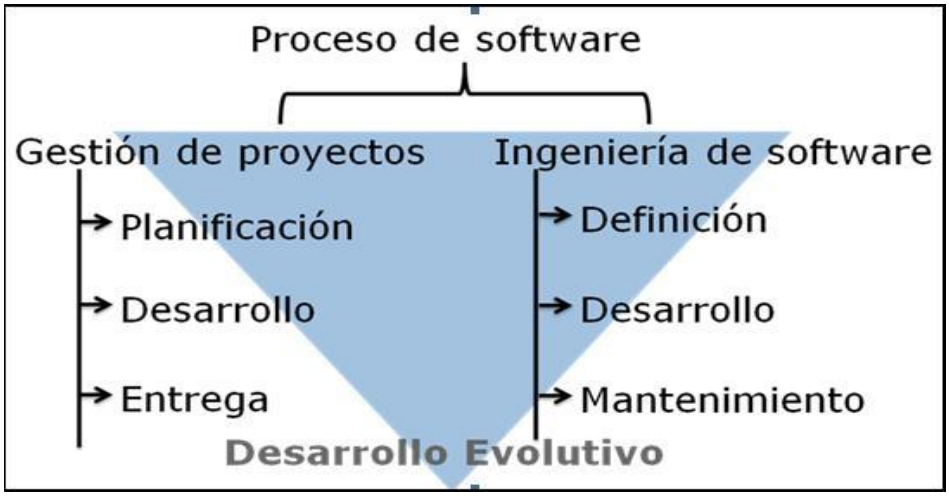


Ilustración 1 Esquema de la metodología SXP

1.3.2 Herramientas Case para el modelado.

Varias herramientas han sido creadas con el propósito de desarrollar la ingeniería de *software*, con el fin de realizar sistemas utilizando técnicas de diseño y metodologías bien definidas, soportadas por herramientas automatizadas.

1.3.2.1 Rational Rose Enterprise Edition.

Es una herramienta Lower CASE, que permite el diseño detallado del *software* y la generación de código fuente e ingeniería inversa basada en modelos con soporte UML, promueve el trabajo en equipo y el desarrollo iterativo. Es una forma de ayuda para la comprensión del sistema y de sus distintos componentes. Su característica más significativa consiste en la creación de componentes, que contengan una serie de archivos dentro de los cuales se encuentran las distintas clases pertenecientes a dicho componente. Es un *software* propietario. La herramienta también mejora la calidad, por el proceso de asignación de Negocios de la arquitectura de *software*. Aumenta la visibilidad y la previsibilidad, por la toma de decisiones de diseño críticas explícitas visualmente. [2]

1.3.2.2 Visual Paradigm Enterprise Edition. (EE_VP_UML)

Es una plataforma para el modelado de sistemas, diseñada para ser usada por arquitectos, desarrolladores, diseñadores, analistas de procesos de negocio y modeladores de datos, con el propósito de acelerar todo el modelo de código para el complejo proceso de desplegar las aplicaciones empresariales a través de la tecnología de modelado visual que facilita el modelado de procesos de negocio. Posibilita la visualización de UML en la última notación UML 2.1 en 13 tipos de diagramas. Además, permite a los desarrolladores crear diagramas con mayor rapidez que cualquier herramienta en el mercado, al contar con una interfaz de usuario potente y fácil de usar, posee motores de generación de código de gran alcance, facilita la programación de bases de datos y cuenta con una excelente interoperabilidad, con mayor apoyo en los IDEs. [3]

Aunque es *software* libre, no es gratuita; pero el centro donde se desarrollará el *software,* la Universidad de las Ciencias Informáticas, posee la licencia de uso de la misma. Esta herramienta tiene unas características gráficas muy cómodas, que facilitan la realización de los diagramas de modelado que siguen el estándar de UML, los mismos son: diagramas de clase, casos de uso, comunicación, secuencia, estado, actividad, componentes, modelo de datos y también se utiliza

para realizar el modelado de procesos del negocio siguiendo el estándar de Notación de gestión de los procesos del negocio (BPMN).

Ventajas de Visual Paradigm.

- Entorno de creación de diagramas para UML 2.0.
- Diseño centrado en casos de uso y enfocado al negocio que genera un *software* de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad en múltiples plataformas.
- Producto de calidad.
- Fácil de instalar y actualizar.
- Compatibilidad entre ediciones.

1.3.3 Lenguajes de Modelado.

1.3.3.1 Lenguaje Unificado de Modelado. (UML)

UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*. Representa decisiones y conocimientos sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está diseñado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar; pero está concebido para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. [4]

Las características más generales de UML son.

- Lenguaje de modelado orientado a objetos.
- Viabilidad en la corrección de errores.

La estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que requieren todos los agentes involucrados en un proyecto

informático. Si se quiere discutir un diseño con alguien más, ambos deben conocer el lenguaje de modelado y no así el proceso que se siguió para obtenerlo.

1.3.3.2 Notación de gestión de los procesos del negocio. (BPMN)

Es un nuevo estándar de modelado de procesos de negocio, en donde se presentan gráficamente las diferentes etapas del proceso del mismo. La notación ha sido diseñada específicamente para coordinar la secuencia de procesos y los mensajes que fluyen entre los diferentes procesos participantes. Es una notación que modela los procesos del negocio, basada en diagramas de flujo fácil de entender. Está diseñada para cubrir varios tipos de modelado, permite la creación tanto de segmentos de procesos, como procesos de negocio de comienzo a fin en diferentes niveles de representatividad. BPMN es gráficamente más rico, con menos símbolos fundamentales, pero con más variaciones de éstos, lo que facilita su comprensión por parte de personas no expertas.

- Provee una notación que es fácilmente entendida por todos los usuarios, desde el analista de negocio, el desarrollador técnico y hasta las propias personas del negocio.
- Crea un puente estandarizado para el vacío existente entre el diseño del proceso de negocio y su implementación.
- Asegura que los lenguajes para la ejecución de los procesos de negocio puedan ser visualizados con una notación común.
- Es usado para comunicar una amplia variedad de información a una amplia variedad de audiencias.

BPMN es un lenguaje gráfico para representar el negocio, facilitando a las organizaciones la habilidad para comunicar esos procedimientos de una manera estándar.

Fundamentación de la selección de Metodologías, herramientas y lenguaje de modelado.

Después de analizar todas las metodologías, sus principales características y los aspectos preponderantes de cada una de ellas, se optó por la metodología ágil SXP por todos los beneficios que está trae consigo y teniendo en cuenta que el sistema a realizar, no es un sistema complejo. Con la utilización de SCRUM para la gestión, se logra una planificación y organización inigualable; mientras que XP respalda con sus prácticas todo el proceso de desarrollo, obteniéndose de esta forma un proceso de *software* completo. Para realizar el modelado se utilizará la herramienta case *Visual Paradigm Enterprise Edition* (EE_VP_UML), usando la

notación UML, excelente lenguaje de modelado visual y BPMN un lenguaje gráfico para representar el negocio, además está herramienta soporta el ciclo de vida completo de un *software* y la Universidad de las Ciencias Informática paga la licencia de su uso.

1.4 Herramientas, tecnologías y lenguajes para el desarrollo.

1.4.1 *Framework*.

El concepto *framework* se emplea en muchos ámbitos del desarrollo de sistemas de *software*, no solo en el ámbito de aplicaciones web. En general, con el término *framework*, se está refiriendo a una estructura de *software* compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un *framework* se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación.

1.4.1.1 CodeIgniter.

Es un programa o aplicación web desarrollada en PHP para la creación de cualquier tipo de aplicación web bajo PHP. Es un producto de código libre, de uso para cualquier aplicación. CodeIgniter contiene una serie de librerías que sirven para el desarrollo de aplicaciones web y además propone una manera de desarrollarlas que se deben seguir para obtener provecho de la aplicación. Esto, marca una manera específica de codificar las páginas web y clasificar sus diferentes *scripts*, que sirve para que el código esté organizado y sea más fácil de crear y mantener. CodeIgniter implementa el proceso de desarrollo utilizando Modelo Vista Controlador (MVC), que es un estándar de programación de aplicaciones, utilizado tanto para hacer sitios web como para programas tradicionales.

Características generales de CodeIgniter.

Versatilidad: Quizás la característica principal de CodeIgniter, en comparación con otros *frameworks* PHP es que CodeIgniter es capaz de trabajar la mayoría de los entornos o servidores, incluso en sistemas de alojamiento compartido, donde sólo se tiene un acceso por FTP para enviar los archivos al servidor y donde no se tiene acceso a su configuración.

Compatibilidad: CodeIgniter, es compatible con la versión PHP 4, lo que hace que se pueda utilizar en cualquier servidor, incluso en algunos antiguos. Por supuesto, funciona correctamente también en PHP 5.

Facilidad de instalación: No es necesario más que una cuenta de FTP para subir CodeIgniter al servidor y su configuración se realiza con apenas la edición de un archivo. Durante la configuración no se necesita acceso a herramientas como la línea de comandos, que no suelen estar disponibles en todos los alojamientos.

Flexibilidad: CodeIgniter es bastante menos rígido que otros *frameworks*. Define una manera de trabajar específica; pero en muchos de los casos se pueden seguir o no y sus reglas de codificación muchas veces se pueden saltar para trabajar como más a gusto le sea al programador. Algunos módulos como el uso de plantillas son totalmente opcionales. Esto ayuda muchas veces también a que la curva de aprendizaje sea más sencilla al principio.

Ligereza: El núcleo de CodeIgniter es bastante ligero, lo que permite que el servidor no se sobrecargue interpretando o ejecutando grandes porciones de código. La mayoría de los módulos o clases que ofrece se pueden cargar de manera opcional, sólo cuando se van a utilizar realmente.

Documentación tutorializada: La documentación de CodeIgniter es fácil de seguir y de asimilar, porque está escrita en modo de tutorial. Esto no facilita mucho la referencia rápida, cuando se sabe acerca del *frameworks* y se quiere consultar sobre una función o un método en concreto; pero para iniciarse en este tema sin duda se agradece mucho.

1.4.1.2 Zend *frameworks.*

Es un *frameworks* de código abierto orientado a componentes para desarrollar aplicaciones y servicios web con PHP 5.0. Está implementado utilizando código totalmente orientado a objetos. Su estructura de componentes es algo única; cada componente se diseña con pocas dependencias en otros componentes. Esta arquitectura débilmente acoplada permite que los desarrolladores utilicen componentes individualmente los cuales forman una potente y extensible aplicación web al ser combinados, todas estas ventajas que brinda y las demás que a continuación se mencionan hacen que dicho *frameworks* sea el más adecuado para programar la capa de negocio del sistema.

Ventajas de Zend *frameworks.*

- Aplicable a variadas tecnologías de desarrollo (adición de múltiples *plugins* externos).
- Cuenta con módulos para manejar archivos PDF, canales RSS, Servicios Web.
- Completa documentación y pruebas de alta calidad.
- Robustas clases para autenticación y filtrado de entrada.

- Marco de trabajo oficial de la comunidad PHP.
- Orientado a componentes.

1.4.1.3 *Symfony*.

La primera versión de *Symfony* fue publicada en Octubre de 2005 por Fabien Potencier, fundador del proyecto y coautor del libro. *Symfony* está desarrollado completamente con PHP5. Ha sido probado en varios proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas Unix, Linux, como en plataformas Windows. Es un completo *frameworks* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Características de *Symfony*.

- Fácil de instalar y configurar en la mayoría de las plataformas (y con la garantía de que funciona correctamente en los sistemas *Windows* y Unix).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de convenir en vez de configurar, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

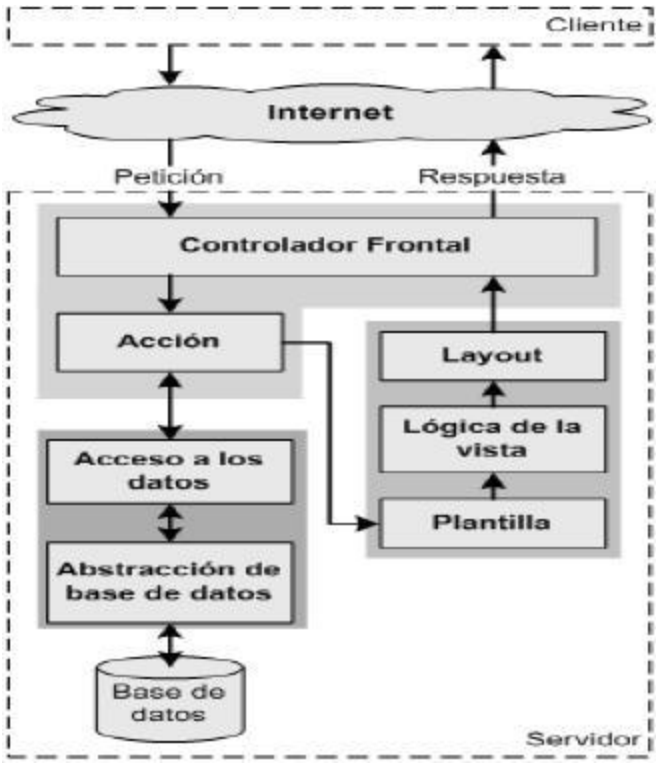


Ilustración 2 Flujo de trabajo de *Symfony*.

1.4.2 Sistema gestor de base datos.

Para el desarrollo de una aplicación, que debe contar con una base de datos, resulta necesario realizar un análisis sobre los diferentes gestores existentes, teniendo en cuenta sus características principales, valorando sus ventajas y desventajas, para determinar cuál de ellos es más factible.

1.4.2.1 MySQL.

MySQL *Database Server* es una de las bases de datos de código fuente abierto más usada. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar. La extensiva reutilización del código dentro del *software* y una aproximación mínima para producir características funcionalmente ricas, han dado lugar a un sistema de administración de base de datos de incomparable velocidad, compactación, estabilidad y facilidad de despliegue. La exclusiva separación del núcleo del servidor del manejador de tablas, permite funcionar a MySQL bajo control estricto de transacciones o con acceso a disco no transaccional rápido.

Características principales.

- Soporta gran variedad de Sistemas Operativos.
- Capacidad de trabajar con servidores con varios procesadores.
- Manejo de la memoria a través de manejo del *buffer* y cache.
- Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.
- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Facilidad de configuración e instalación.
- Conectividad y seguridad.

Desventajas.

- No considera las claves ajenas, ignora la integridad referencial, dejándola en manos del programador de la aplicación.
- Un gran porciento de las utilidades de MySQL no están documentadas.
- No es intuitivo, como otros programas (Access).

1.4.2.2 Firebird.

Es un sistema de administración de base de datos relacional (o RDBMS) SQL de código abierto, fue liberado por Borland en 2000. Su código fue reescrito de C a C++. Es multiplataforma, y actualmente puede ejecutarse en los sistemas operativos: Linux, HP-UX, FreeBSD, Mac OS, Solaris y Microsoft Windows. Presenta un ejecutable pequeño, con requerimientos de *hardware* bajos. Utiliza la arquitectura Cliente/Servidor sobre protocolo TCP/IP y otros, además de que soporta transacciones y claves foráneas. Soporta plenamente el estándar SQL-92, tanto de sintaxis como de tipos de datos. Además, tiene capacidad de almacenar elementos BLOB (objetos binarios grandes).

1.4.2.3 PostgreSQL.

PostgreSQL es un sistema gestor de bases de datos, que no pertenece a ninguna compañía, sino que es dirigido por una comunidad de desarrolladores, que se hace llamar “PostgreSQL Grupo de desarrollo global” y organizaciones comerciales que están a cargo de su desarrollo, cuyo portal es www.postgresql.org. Este robusto sistema gestor de bases de datos es una solución real, a los complejos problemas del mundo empresarial y a la vez mantiene la eficiencia, al consultar los datos. Ha desarrollado y añadido al PostgreSQL las más interesantes y útiles

características, que antes sólo podían hallarse en sistemas manejadores de bases de datos comerciales como Oracle, DB2, por lo que hace honor a su lema: "El manejador de bases de datos de código abierto más avanzado del mundo". Debido a sus características técnicas sobresalientes, el PostgreSQL se ha ganado la admiración y el respeto de sus usuarios, así como el reconocimiento de la industria. PostgreSQL es un producto de código abierto, sin costos de licencia, que se convierte en una alternativa extremadamente atractiva para las empresas, que buscan un ahorro significativo de costos en activos.

Ventajas de PostgreSQL.

- Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.
- Multiplataforma.
- Herramientas gráficas de diseño y administración de bases de datos.
- Por su arquitectura de diseño, escala muy bien al aumentar el número de CPUs y la cantidad de RAM.
- Soporta transacciones y desde la versión 7.0, claves ajenas (con comprobaciones de integridad referencial).

Desventajas:

- Consume bastantes recursos y carga más el sistema.

1.4.3 Mapeado de objetos relacionales. (ORM)

Un ORM o (Mapeado de objeto relacional) es una técnica de programación que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, es decir, las tablas de la base dato pasan a ser clases y los registros objetos que se pueden manejar con facilidad.

Utilizar un ORM tiene una serie de ventajas que facilitan enormemente tareas comunes y de mantenimiento.

- **Reutilización:** La principal ventaja que aporta un ORM es la reutilización permitiendo llamar a los métodos de un objeto de datos desde distintas partes de la aplicación e incluso desde diferentes aplicaciones.
- **Encapsulación:** La capa ORM encapsula la lógica de los datos pudiendo hacer cambios que afectan a toda la aplicación únicamente modificando una función.

- **Seguridad:** Los ORM suelen implementar mecanismos de seguridad que protegen la aplicación de los ataques más comunes como inyecciones SQL.
- **Mantenimiento de código:** Gracias a la correcta ordenación de la capa de datos, modificar y mantener el código es una tarea sencilla.

1.4.3.1 Doctrine

Doctrine es un mapeador de objetos relacionales (ORM) para PHP 5.2.3+ que se sienta encima de una potente capa de abstracción de bases de datos (dbal). Una de sus principales características es la opción de escribir las consultas de base de datos en un objeto de propiedad orientada dialecto SQL llamada Doctrine *Query Language* (DQL) que no es más que el lenguaje que utiliza Doctrine para ejecutar sus consultas. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria.

1.4.3.2 Propel

Propel es un ORM para PHP (Pre-procesador Hipertexto) que facilita la labor de desarrollo de aplicaciones web, gracias a la capa que transforma el tratamiento de la BD mediante objetos, con la que se puede recuperar, insertar y modificar datos. No es necesario preocuparse por las conexiones de la BD y escribir sentencias SQL. Tampoco es necesario escapar datos o realizar *casting*. Tan solo es necesario definir la base de datos en formato XML y obtener la definición desde una base de datos ya existente.

1.4.4 Lenguajes para el desarrollo web.

El desarrollo de aplicaciones web ha tenido un auge en todo el mundo gracias a las ventajas que las mismas ofrecen a empresas e instituciones. Para el desarrollo de estas existen numerosos lenguajes informáticos que se dividen en dos grupos, el primer grupo abarca los lenguajes que corren en el lado del cliente y el segundo los lenguajes que corren en el lado del servidor.

1.4.4.1 Lenguajes del lado del cliente.

Los lenguajes del lado del cliente son aquellos que pueden ser directamente digeridos por el navegador y no necesitan un pre-tratamiento. Son totalmente independientes del servidor.

Lenguaje de marcado de hipertexto. (HTML)

El lenguaje llamado HTML indica al navegador donde colocar cada texto, cada imagen o cada video y la forma que tendrán estos al ser colocados en la página. Es el lenguaje más utilizado para la presentación de textos estructurados en formato hipertexto, estándar de las páginas web. HTML es utilizado por casi la totalidad de navegadores web del mercado con el fin de presentar al visitante de una página web el contenido de la misma tal como el diseñador quiere que se muestre a su público. No es un lenguaje de programación y no tiene compilador alguno, así que si hay algún error que no detecta, lo visualizará de la manera en la que lo ha entendido. Es un sistema de etiquetas que indica al ordenador cuando hay que señalar una cursiva, separar un párrafo o definir el color del texto. De todas maneras, tiene sus limitaciones así que a menudo se utilizan otras herramientas como las hojas de estilo, que le dan mayor libertad al diseñador. En concreto, el HTML le da las indicaciones mencionadas al programa cliente, el *browser* o navegador para que presente el documento en la pantalla de la manera adecuada. El HTML se hizo popular por su sencillez, por lo fácil de aprender y eso lo hace accesible a un mayor número de personas. [5]

JavaScrip.

*JavaScrip*t es un lenguaje de programación utilizado para crear pequeños programas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas y la definición de interactividades con el usuario. *JavaScript* representa, actualmente, el estándar no oficial de referencia para el uso de *script* en documentos hipertextuales. *JavaScript* se interpreta con el código HTML (del que forma parte integrante, y sin el cual no puede existir), sin necesidad de máquinas virtuales o conocimientos profundos de modelos orientados a los objetos. Se dirige a los creadores de sitios web que pretenden obtener resultados apreciables sin necesidad de aprender complejos lenguajes de programación. JavaScript es un lenguaje que se integra directamente en páginas HTML.

Características principales.

- Es interpretado por el cliente.
- Está basado en objetos. No es, como Java, un lenguaje de programación orientada a objetos JavaScript no emplea clases ni herencia.

- No es necesario declarar los tipos de variables que van a utilizarse.
- Las referencias a objetos se comprueban en tiempo de ejecución, por lo tanto, no se compila.
- No puede escribir automáticamente al disco duro.

Hojas de estilo. (CSS)

Las Hojas de estilo son un mecanismo que permiten aplicar formato a los documentos escritos en HTML (y en otros lenguajes estructurados, como XML) separando el contenido de las páginas de su apariencia. Para el diseñador, esto significa que la información estará contenida en la página HTML; pero este archivo no debe definir como será visualizada esa información. Las indicaciones acerca de la composición visual del documento estarán especificadas en el archivo de la CSS.

Ventajas de las Hojas de estilo.

- Alterar la presentación de cada elemento sin tocar el código HTML, ahorrando esfuerzo y tiempo de edición.
- El lenguaje de las Hojas de estilo, aunque es muy potente, es relativamente sencillo y fácil de aprender.
- Los documentos que usan Hojas de estilo generalmente resultan más compactos.
- Las Hojas de estilo pueden aplicarse de varias maneras y combinarse formando una cascada de estilos con la información de cada una.
- Pueden usarse con otros lenguajes de programación (como *JavaScript*) para conseguir efectos dinámicos en las páginas.

***JavaScript* asíncrono *y XML*. (AJAX)**

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet pplications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

AJAX en resumen es la unión de varias tecnologías de la web ya existente que logran una sincronización sorprendente:

- HTML y CSS, para crear una presentación basada en estándares.
- DOM (Modelo de Objetos del Documento), para la interacción y manipulación dinámica de la presentación.

- XML, XSLT2 y JSON (JavaScript Object Notation/ Notación de Objetos de JavaScript), para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías. [6]

Se han creado cientos de aplicaciones basadas en AJAX que en la mayoría de casos pueden sustituir completamente a otras técnicas como Flash y en el caso de las aplicaciones más avanzadas, pueden sustituir a complejas aplicaciones de escritorio.

1.4.4.2 Lenguajes del lado del servidor.

Los lenguajes del lado del servidor son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Cuando se selecciona un enlace hipertexto, en realidad lo que pasa es que se establece una petición de un archivo HTML residente en el servidor (un ordenador que se encuentra continuamente conectado a la red) el cual es enviado e interpretado por navegador (el cliente).

***Active Server Pages.* (ASP)**

ASP dentro de esta tecnología el mayor inconveniente es que se trata de un sistema propietario que es usado nativamente sólo por Microsoft Internet Information Server (IIS). ASP es un lenguaje más lento y pesado que PHP, y también menos estable. Algunas de las ventajas de dicho lenguaje consisten en que debido a que usa principalmente VBScript, es relativamente simple tratar con el lenguaje si ya se conoce cómo programar en *Visual Basic*. El soporte de este lenguaje también se encuentra habilitado por defecto en el servidor IIS, facilitando su instalación y ejecución. Los componentes integrados en ASP son bastante limitados, de modo que si necesita usar características "avanzadas", como interactuar con servidores FTP, necesita comprar componentes adicionales. ASP es un sistema con nula portabilidad pues requiere necesariamente de un servidor Windows, con todas las implicaciones de alto costo, poca flexibilidad y escasa seguridad que estos equipos conllevan.

PHP5.

Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Es un potente lenguaje de *script* del lado del servidor, que se utiliza principalmente para generar páginas de forma dinámica,

es tan potente que se utiliza para muchísimas cosas más: generar imágenes, generar PDF, entre otras. La sintaxis gramática y funciones del lenguaje son muy sencillas, su compatibilidad con otros sistemas se incrementa agregando nuevas funciones. La manera de manejar los datos, las funciones disponibles y la documentación son realmente excepcionales. La mayor parte de su sintaxis ha sido tomada de C, Java y PERL con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. No es un lenguaje de marcas como podría ser HTML, XML o WML.

Ventajas de PHP5.

- Mejor soporte para la Programación Orientada a Objetos, que en versiones anteriores era extremadamente rudimentario.
- Mejoras de rendimiento.
- Manejo de excepciones.
- Soportado por una gran comunidad de desarrolladores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Entre las ventajas de PHP se puede destacar que este lenguaje es completamente expandible, está compuesto de un sistema principal, un conjunto de módulos y una variedad de extensiones de código. Muchas interfaces distintas para cada tipo de servidor.

Fundamentación de la selección de herramientas, tecnologías y lenguajes para el desarrollo.

Se selecciona *Symfony* como *framework* porque es un *framework* maduro, bien documentado y con una gran comunidad que lo apoya. Su estructura modelo vista controlador facilita su estudio. Como sistema gestor de base dato por sus características se utilizará PostgreSQL, puesto que está disponible en cualquier plataforma. PostgreSQL es un gestor magnífico, que posee una gran escalabilidad, haciéndolo idóneo para sitios web. Para la programación del lado del cliente se utilizará como lenguaje, *JavaScrip* para mejorar la gestión de la interfaz cliente/servidor y CSS para realizar la presentación de la página estructurada en HTML, AJAX para aumentar la interactividad, velocidad y usabilidad en la aplicación y para el lado del servidor PHP5

actualmente, es el lenguaje de *software* libre más utilizado en la Internet con una amplia documentación, posee una de las comunidades más grandes en Internet, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente. Es multiplataforma y soporta POO (Programación Orientada a Objetos).

1.5 Servidor web.

Un servidor web es un programa que corre sobre el servidor que escucha las peticiones HTTP (Protocolo de transferencia hipertexto) que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor web buscará una página web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML al cliente o navegador que realizó la petición. [7]

1.5.1 Lighttpd.

Lighttpd es un servidor web diseñado para ser rápido, seguro, flexible, y fiel a los estándares. Está optimizado para entornos donde la velocidad es muy importante, y por eso consume menos CPU y memoria RAM que otros servidores. Por todo lo que ofrece Lighttpd es apropiado para cualquier servidor que tenga problemas de carga. Lighttpd es *software* libre y se distribuye bajo la licencia BSD. Funciona en GNU/Linux y UNIX de forma oficial. Para Microsoft Windows actualmente hay una distribución conocida como "*Lighttpd for Windows*" mantenida por Kevin Worthington.

1.5.2 Servidor web Apache.

El servidor Apache se desarrolla dentro del proyecto HTTP Server de la Apache *Software Foundation*. Presenta características como, mensajes de errores altamente configurables, bases de datos de autenticación y negociado de contenido. Tiene amplia aceptación en la red convirtiéndose en uno de los servidores HTTP más usados.

La arquitectura que utiliza es cliente-servidor, el cliente hace la petición al servidor y este atiende dicha petición. El protocolo utilizado para la transferencia de hipertexto es HTTP (Protocolo de transferencia de hipertexto) que está basado en el envío de mensajes y establece el conjunto de normas mediante las cuales se envían las peticiones de acceso a una web y la respuesta de esa web.

Ventajas de Apache.

- Código abierto.
- Extensible.
- Fácil de conseguir ayuda/soporte.
- Es multiplataforma, aunque idealmente está preparado para funcionar bajo Linux.
- Muy sencillo de configurar.
- Amplias librerías de PHP y PERL a disposición de los programadores.
- Posee diversos módulos que permiten incorporarle nuevas funcionalidades, estos son muy simples de cargar.
- Es capaz de utilizar lenguajes como PHP, TCL, Python, entre otros.

1.6 Entornos Integrados de Desarrollo. (IDE)

Un Entorno Integrado de Desarrollo es un sistema que facilita el trabajo del desarrollador de *software*, integrando sólidamente la edición orientada al lenguaje, la compilación o interpretación, la depuración, las medidas de rendimiento y la incorporación de las fuentes a un sistema de control de fuentes.

1.6.1 Aptana.

Aptana es un IDE para el desarrollo web, incorpora características completas, sincronización, y administración de proyectos. Permite incorporarle funciones mediante *plugins*. Soporte para las plataformas Microsoft Windows, Mac y Linux.

Aptana es una robusta y avanzada interface de desarrollo web, enfocado a JavaScript para el desarrollo de aplicaciones dinámicas.

Características principales.

- Asistente en código JavaScript, HTML y CSS; incluyendo sus propias funciones en JavaScript.
- Vista instantánea de la estructura del código JavaScript, HTML y CSS.
- Soporte para código PHP y ASP.
- Notificación en el código de errores y precauciones.
- Soporte Multiplataforma.
- Código abierto.

Ventajas de Aptana.

➢ Permite comprobar la compatibilidad de las funciones con los diferentes navegadores, multiplataforma, sincronización con carpetas locales y remotas, incluye *plugins* para Eclipse.

1.6.2 NetBeans.

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java; pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso. [8]

Características principales.

➢ El proyecto NetBeans ofrece una versión del IDE a la medida para el desarrollo en PHP de sitios web que abarcan una variedad de secuencias de comandos y lenguajes de marcado.

➢ El IDE NetBeans respalda plenamente el desarrollo iterativo, por lo que los proyectos desarrollados en PHP siguen los patrones clásicos conocidos para los desarrolladores web.

➢ El editor de NetBeans PHP ofrece plantillas de código y la generación (getters y setters), la refactorización, información sobre herramientas de parámetros, consejos y soluciones rápidas (aplicación de métodos abstractos), y autocompletado de código inteligente (incluido el soporte de terminación), formateo de código y marcado de los sucesos y los puntos de salida.

➢ El editor de PHP entiende espacios de nombres y definiciones de tipo variable en los comentarios que mejora la finalización de código y navegación hipervínculo.

Fundamentación de la selección del servidor web y el entorno integrado de desarrollo. (IDE)

Luego de realizar el estudio se decidió utilizar como servidor web, Apache porque es multiplataforma y de código abierto y teniendo en cuenta los IDEs antes caracterizados se decidió utilizar *NetBeans* porque es el único que da soporte para *Symfony*, es un IDE multilenguaje, completo y modular con soporte para Java SE, Java EE, Java ME. Tiene gran cantidad de módulos de terceros (*plugins),* desarrollo intuitivo arrastrar y soltar, y completa código. Es gratis y de código abierto y posee una gran comunidad de usuarios y desarrolladores.

1.7 Arquitectura.

La arquitectura es el esqueleto o base de una aplicación. Representa la organización fundamental de un sistema. Desde los pequeños programas hasta los sistemas más grandes poseen una estructura y un comportamiento que los hace clasificables según su "arquitectura". En la web es muy común la utilización de la arquitectura "3-capas", "n-capas", "MVC", entre otras. El *framework Symfony*, seleccionado para desarrollar la aplicación, implementa a su vez el patrón arquitectónico MVC, es por ello que adopta esta arquitectura para el desarrollo de la propuesta de solución.

Modelo Vista Controlador. (MVC)

Es un patrón de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

- **Modelo:** Representa la información con la que trabaja la aplicación, es decir, su lógica del negocio.
- **Vista:** Presenta el modelo en un formato adecuado, como en una página web que le permite al usuario interactuar con ella, usualmente un elemento de interfaz de usuario.
- **Controlador:** Responde a eventos, usualmente acciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Ventajas principales.

Soporte de múltiples vistas: Dado que la vista se halla separada del modelo y no existe una dependencia directa entre ambos, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación web pueden utilizar el mismo modelo de objetos mostrado de modos diferentes.

Adaptación al cambio: Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas del negocio. Dado que el modelo no depende de la vista, agregar nuevas opciones de presentación generalmente no afecta al mismo.

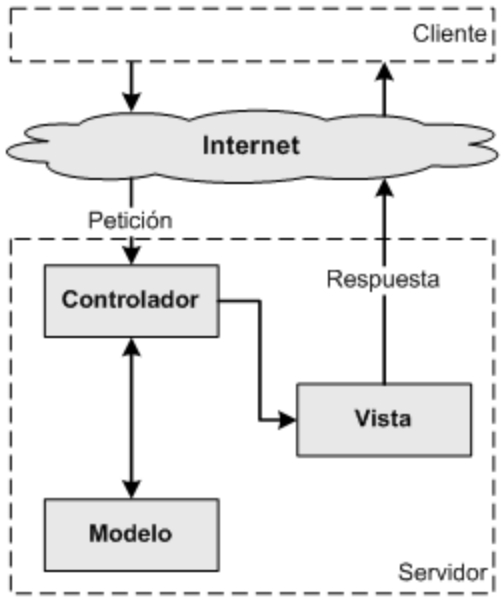


Ilustración 3 Patrón MVC.

1.8 Conclusiones.

Con el estudio realizado en este primer capítulo se logró identificar las diferencias entre los sistemas analizados, notando sus principales características. Se realizó un profundo análisis de varias metodologías, herramientas y lenguajes para fundamentar la elección, de cuáles se decidió utilizar para el desarrollo de la aplicación. Como metodología de desarrollo se optó por la metodología ágil SXP. Para realizar el modelado se utilizará la herramienta case Visual Paradigm Enterprise Edition (EE_VP_UML) y como *framework* se utilizará *Symfony*. Finalmente, se ha llegado a la conclusión de que utilizando correctamente dichas herramientas y cumpliendo estrictamente todas las normas y procedimientos de la implementación se obtendrá como resultado una aplicación web para la gestión de los procesos que se desarrollan en el Club Bóxer de Cuba.

Capítulo 2: Características del sistema.

2.1 Introducción.

En el presente capítulo se define la propuesta de solución y se describen los procesos básicos que influyen en el objeto de estudio y hacen válido el problema a resolver anteriormente puntualizado. Para poseer un mejor conocimiento y concebir el funcionamiento del sistema, se realiza la modelación de los principales procesos del negocio. Además, se enumeran los requisitos funcionales y no funcionales, características y cualidades que el sistema debe cumplir y tener respectivamente. Por último, se muestran las historias de usuarios realizadas desde la perspectiva del cliente.

2.2 Modelación de los principales procesos del negocio.

Gestión de proceso del negocio. (BPM)

Para realizar el modelado de los procesos se utilizó el Modelo de proceso del negocio (BPM). Este es un modelo para la gestión de los procesos de negocio vinculado a la notación BPMN, es una nueva manera de abordar el problema de comunicación entre los clientes y usuarios. La tecnología de los Sistemas de gestión de procesos de negocio permite salvar la distancia con los sistemas, máquinas y aplicaciones que automatizan.

Tabla 1 Proceso realizar inscripción de asociado.

Ficha de proceso	
Proceso:	Realizar inscripción de asociado
Descripción:	El proceso se inicia cuando una persona desea asociarse al club debe ir a la sede del mismo, donde se le entrega un documento de solicitud de ingreso con una serie de datos a llenar. Cuando se acepta a la persona como asociado, se le asigna un número de afiliado y un número de miembro y se le entrega el carné de asociado del Club.
Entradas:	Planilla de solicitud de ingreso.
Salidas:	Carné de asociado.
Actividades principales	
Actividad 1	Dirigirse a la sede del club.
Actividad 2	Entregar la planilla de solicitud de ingreso a la persona.

Actividad 3	Asignar número de afiliado y número de miembro.
Actividad 4	Entregar carné de asociado a la persona.
Diagrama de proceso	

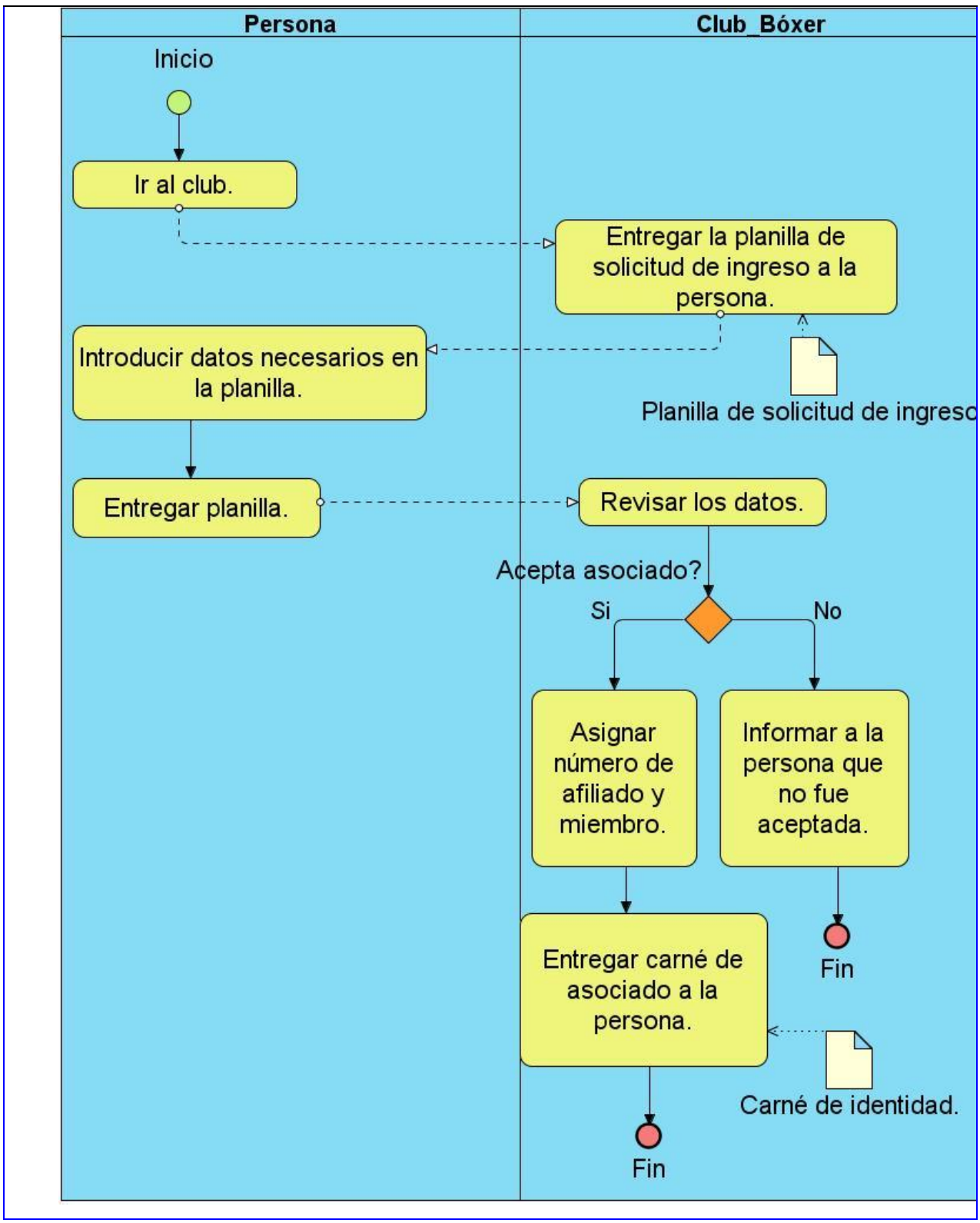

Persona
Club_Bóxer
Inicio
Ir al club.
Entregar la planilla de solicitud de ingreso a la persona.
Introducir datos necesarios en la planilla.
Planilla de solicitud de ingreso
Entregar planilla.
Revisar los datos.
Acepta asociado?
Si
No
Asignar número de afiliado y miembro.
Informar a la persona que no fue aceptada.
Fin
Entregar carné de asociado a la persona.
Carné de identidad.
Fin

Tabla 2 Proceso de apareamiento y entrega de cachorros

Ficha de proceso	
Proceso:	Apareamiento y entrega de cachorros.
Descripción:	El proceso se inicia cuando un asociado del club, propietario de una ejemplar hembra desea aparear la misma. Debe dirigirse al club, para ser orientado de cuáles sementales están disponibles para la cruza. Cuando la ejemplar queda gestada, el proceso de gestación dura alrededor de 2 meses, al parir se evalúa la camada a los 45 días, para entonces registrar los resultados de la monta, luego el propietario tiene 15 días a partir de ese momento para realizar la inscripción de la camada. Cuando se va a entregar el cachorro al propietario se le debe entregar el documento que lo avala como propietario del ejemplar El mismo va a tener los datos del propietario, copropietario y dirección donde va a estar el ejemplar.
Entradas:	Planilla inscripción de camada.
Salidas:	Documento propiedad.
Actividades principales	
Actividad 1	Solicitar que su ejemplar sea apareada.
Actividad 2	Orientar los sementales disponibles para aparearse.
Actividad 3	Realizar el cruzamiento.
Actividad 4	Realizar inscripción de la camada.
Actividad 5	Realizar entrega de cachorro.
Diagrama de proceso	

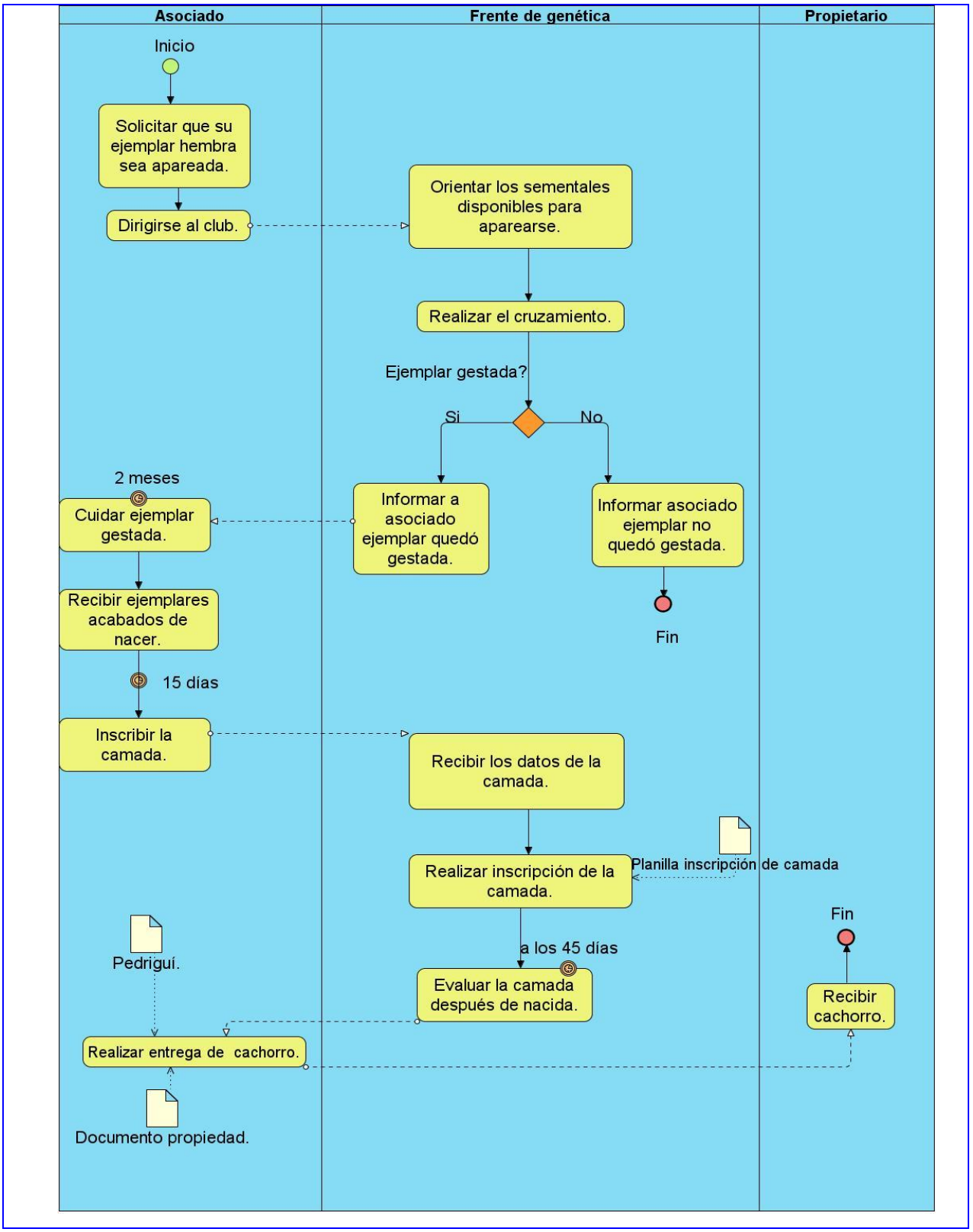

Asociado
Frente de genética
Propietario
Inicio
Solicitar que su ejemplar hembra sea apareada.
Dirigirse al club.
Orientar los sementales disponibles para aparearse.
Realizar el cruzamiento.
Ejemplar gestada?
Si
No
Informar a asociado ejemplar quedó gestada.
Informar asociado ejemplar no quedó gestada.
Fin
2 meses
Cuidar ejemplar gestada.
Recibir ejemplares acabados de nacer.
15 días
Inscribir la camada.
Recibir los datos de la camada.
Realizar inscripción de la camada.
Planilla inscripción de camada
a los 45 días
Evaluar la camada después de nacida.
Fin
Recibir cachorro.
Pedriguí.
Realizar entrega de cachorro.
Documento propiedad.

Tabla 3 Proceso de realizar evento.

Ficha de proceso	
Proceso:	Realizar evento
Descripción:	El proceso se inicia cuando un asociado realiza la inscripción de su ejemplar para participar en los eventos a través del frente de ferias y exposiciones, el cual también es el encargado de recopilar los resultados obtenidos por cada ejemplar cada día del evento. Un evento puede durar varios días, y en cada uno de los días al ejemplar se le entrega una tarjeta donde se van anotando los resultados obtenidos en cada una de las evaluaciones de los jueces.
Entradas:	Planilla de inscripción.
Salidas:	Planilla de resultados.
Actividades principales	
Actividad 1	Divulgar inicio del evento.
Actividad 2	Inscribir su ejemplar.
Actividad 3	Realizar la inscripción del ejemplar al evento.
Actividad 4	Archivar los resultados del evento.
Diagrama de proceso	

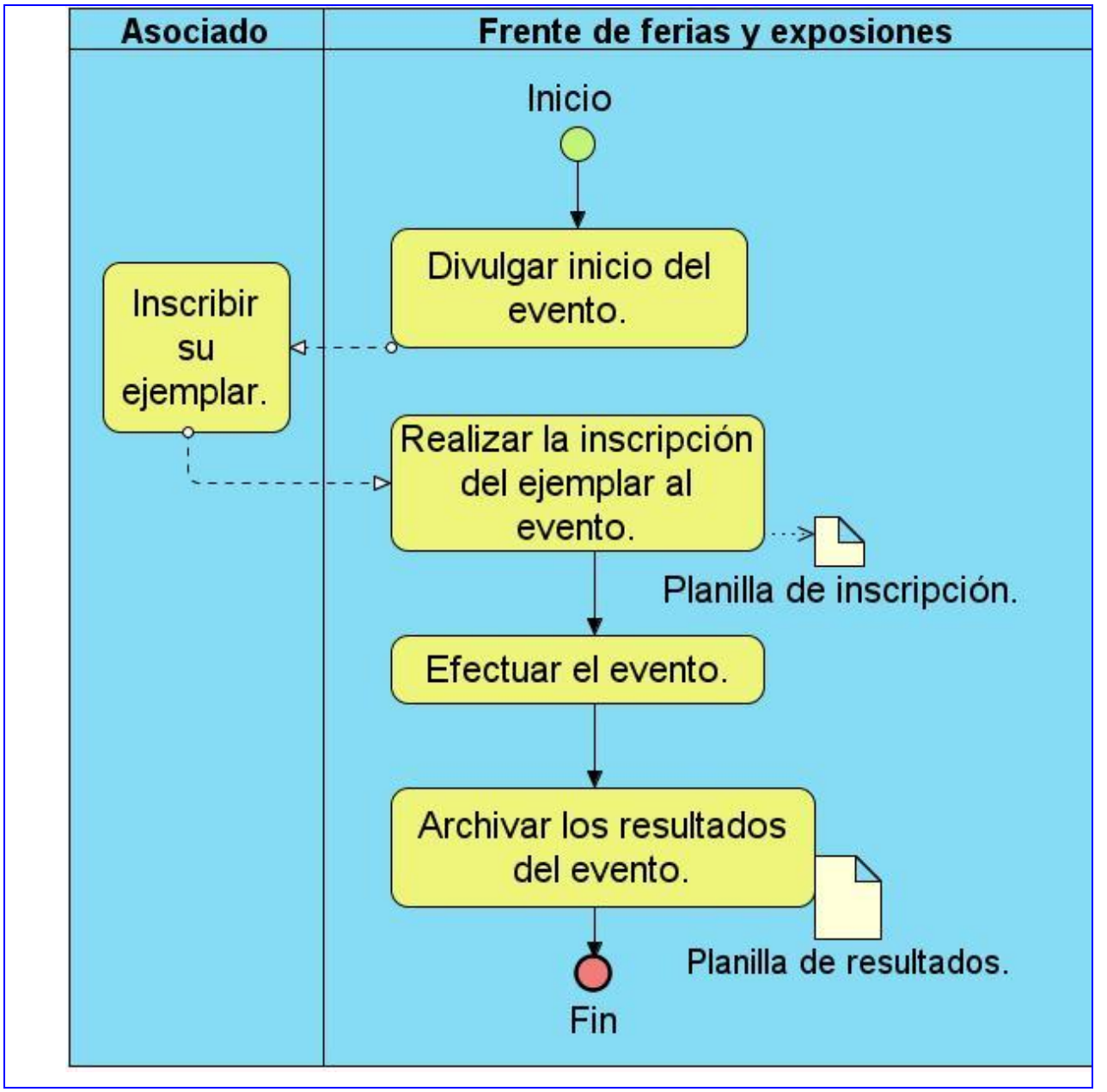
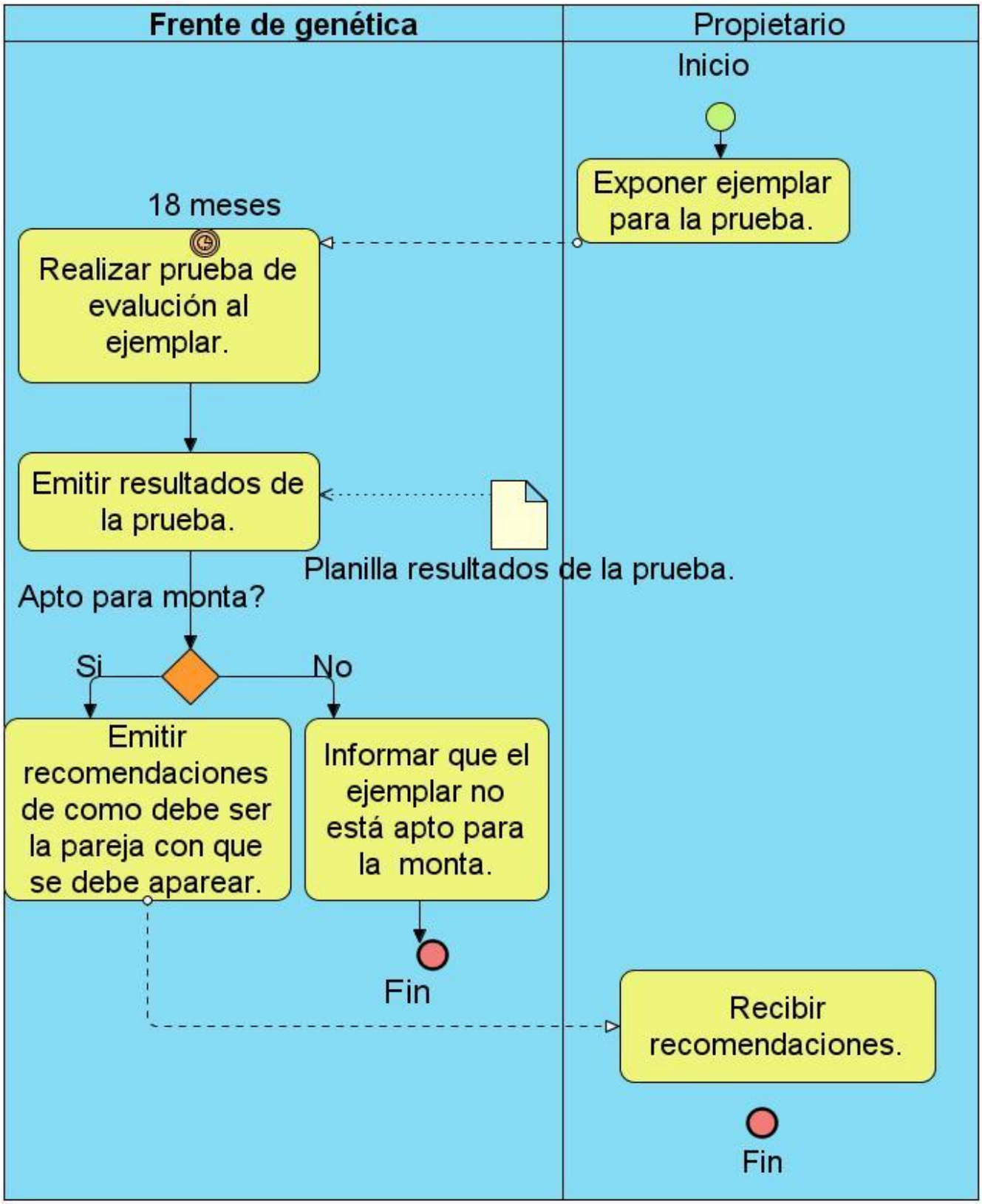

Asociado
Frente de ferias y exposiones
Inicio
Divulgar inicio del evento.
Inscribir su ejemplar.
Realizar la inscripción del ejemplar al evento.
Planilla de inscripción.
Efectuar el evento.
Archivar los resultados del evento.
Planilla de resultados.
Fin

Tabla 4 Proceso de realizar prueba de confirmación.

Ficha de proceso	
Proceso:	Realizar prueba de confirmación.
Descripción:	El propietario expone al ejemplar para que le realicen la prueba a los 18 meses de nacido, prueba conocida como prueba de confirmación. El frente de genética es el responsable de realizar la prueba para evaluar al ejemplar, luego emite el resultado, si está apto, da algunas recomendaciones de las características que deberá tener la pareja con que se decida aparear.
Entradas:	
Salidas:	Planilla resultados de la prueba.
Actividades principales	
Actividad 1	Realizar prueba de evaluación al ejemplar.
Actividad 2	Emitir resultados de la prueba.
Actividad 3	Emitir recomendaciones de como debe ser la pareja con que se desee aparear.
Diagrama de proceso	


Frente de genética
Propietario
Inicio
Exponer ejemplar para la prueba.
18 meses
Realizar prueba de evalución al ejemplar.
Emitir resultados de la prueba.
Planilla resultados de la prueba.
Apto para monta?
Si
No
Emitir recomendaciones de como debe ser la pareja con que se debe aparear.
Informar que el ejemplar no está apto para la monta.
Fin
Recibir recomendaciones.
Fin

2.3 Lista de reserva del producto.

2.3.1 Requerimientos funcionales.

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Los requerimientos funcionales se mantienen invariables sin importar con qué propiedades o cualidades se relacionen.

Código	Descripción	Prioridad
RFR1	Mostrar interfaz para registrar ejemplar.	Alta
RFR2	Realizar traspaso de propiedad del ejemplar.	Alta
RFR3	Mostrar interfaz para modificar ejemplar.	Alta
RFR4	Mostrar interfaz para registrar asociado.	Alta
RFR5	Mostrar opción para generar folio.	Alta
RFR6	Realizar búsqueda de ejemplar.	Alta
RFR7	Realizar búsqueda de asociado.	Alta
RFR8	Mostrar interfaz para registrar prueba de confirmación.	Media
RFR9	Mostrar interfaz para registrar resultados de la cruza.	Media
RFR10	Mostrar interfaz para registrar resultado de los eventos.	Media
RFR11	Mostrar interfaz para modificar asociado.	Media
RFR12	Ver detalles de resultado en los eventos de un ejemplar.	Media
RFR13	Ver detalles de pruebas de confirmación de un ejemplar.	Media
RFR14	Listar posibles reproductores.	Baja
RFR15	Mostrar los ejemplares de un asociado.	Baja
RFR16	Dar baja a ejemplar.	Baja

2.3.2 Requerimientos no funcionales.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

RNF (Requisitos no funcionales)	
	Usabilidad
RFN1	A los usuarios finales de la aplicación se les debe dar un adiestramiento básico

	en el uso de la aplicación web. Estas personas deben tener un nivel de acceso de acuerdos a los permisos que le han sido asignados.
RFN2	Los usuarios finales deben tener un conocimiento básico sobre informática para poder trabajar con el *software* a desarrollar.
	Fiabilidad
RFN3	Políticas de seguridad por usuarios y roles: el sistema debe contar con un grupo de políticas de accesibilidad a las diferentes funcionalidades del mismo en dependencia del nivel de autorización que presente un usuario determinado.
	Eficiencia
RFN4	El sistema debe funcionar con un máximo rendimiento; pero ajustado a las bajas prestaciones de las computadoras debido que no todas las instituciones poseen tecnología de punta.
RFN5	El sistema requiere de un buen rendimiento que se apoya en el mínimo acceso a base de datos, y realización de consultas no redundantes.
RFN6	Para un funcionamiento óptimo de la aplicación se deben seguir las diferentes técnicas de elaboración en la web, que faciliten el rápido acceso a sus páginas.
RFN7	La herramienta propuesta debe ser rápida y el tiempo de respuesta debe ser el mínimo posible, adecuado a la rapidez con que el cliente requiere la respuesta a su petición.
	Soporte
RFN8	Realizar las pruebas, e instalaciones necesarias para lograr el mejoramiento del sistema.
	Restricciones de diseño
RFN9	Como gestor de base de datos se utilizará PostgreeSQL.
RFN10	El servidor de aplicaciones será Apache-2.0.40.
RFN11	El lenguaje de programación utilizado será PHP.
RFN12	Se debe disponer de Sistema Operativo 95 o superior.
RFN13	Para la modelación del sistema se utilizará Visual Paradigm, pues la UCI posee la licencia.
RFN14	Se utilizará como metodología de desarrollo de *software* SXP.
RFN15	Se necesitará una impresora, para la impresión de los pdf generados por el sistema.

RFN16	Como framework se utilizará Symfony.
RFN17	Validar el proceso de la captación de datos para evitar entradas inadecuadas.
	Apariencia o interfaz externa.
RFN18	El sistema debe contar con una interfaz fácil, amigable y sencilla que permita a los usuarios finales interactuar con el sistema, aún teniendo conocimientos básicos.
RFN19	El tamaño y el tipo de letras deberán ser uniforme para todas las interfaces.
RFN20	Las interfaces evitarán cargar diálogos, paneles o ventanas innecesarias.
RFN21	Los iconos y botones que se utilicen en las interfaces, deben de estar relacionados con la función a realizar.
RFN22	El tamaño de los campos debe corresponderse con el tamaño de la información a tratar.
	Interfaces *Hardware*
RFN23	Se requiere de una computadora que se utilice como servidor, debe poseer 512MB de RAM como mínimo; así como estar conectada a la red para permitir el acceso desde las computadoras clientes las cuales deben contar al menos con 256MB de RAM.
	Interfaces *Software*
RFN24	Las computadoras que se utilizarán como servidores deben tener: ➢ Algún sistema operativo. ➢ El servidor Web Apache con PHP y el Sistema Gestor de base de datos PostgreSQL (en el caso de la computadora que funcionará como servidor). ➢ Un navegador Web (en la computadora cliente como por ejemplo Firefox Mozilla o Internet Explorer, entre otros), y debe tener instalado el plugins de documentos pdf.
	Requisitos de Licencia
RFN25	El sistema utiliza la política de *software* libre, todas las herramientas que se utilizan son *software* libre. Para la herramienta Visual Paradigm se utiliza la licencia que la Universidad posee.

2.4 Descripción de la solución propuesta.

Para lograr satisfactoriamente cumplir los objetivos propuestos de este trabajo y además contar con los requerimientos expuestos con anterioridad, los cuales responden directamente a dichos objetivos porque constituyen las características necesarias para lograr las funcionalidades básicas que debe brindar el sistema, se propone el desarrollo de una aplicación web para gestionar toda la información referente al Club Bóxer de Cuba. Se considera la existencia de cuatro roles, el rol de usuario anónimo que solo obtendrá el beneficio de ver la información que este en la aplicación, el rol de administrador del sistema, se encargará de la gestión de la información de todos los procesos que ocurren en el Club Bóxer y será el que tendrá acceso sin restricciones en la aplicación, el rol del Frente genética, será el encargado de registrar las pruebas que se le realizan a los ejemplares, registrar los resultados de la cruza, además de registrar a los ejemplares y puede ver lo mismo a lo que tiene permiso el usuario normal y el último rol es el Frente de ferias y exposiciones, será el encargado de registrar los eventos, registrar los resultados de los eventos además de todo lo que está permitido para el usuario normal.

2.5 Historias de Usuarios.

Las Historias de usuarios (HU) son tarjetas desarrolladas desde la perspectiva del cliente, es el artefacto que sustituye a los casos de uso de la metodología tradicional, su objetivo es especificar en detalle las necesidades del sistema, son descripciones generalmente cortas y sin terminología técnica. Se elaboraron 16 HU cada una de ellas respondiendo a las diferentes funcionalidades que desea el cliente, a continuación se exponen. (Ver anexo2).

Historia de Usuario	
Número: 1	**Nombre historia de usuario:** Registrar ejemplar
Modificación de historia de usuario número: 1	
Referencia: RF1	
Programador: Orlando Castillo	**Iteración asignada: 1**

Prioridad en negocio: Alta	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Medio	**Puntos reales: 1**

Se debe mostrar una interfaz que permita registrar un ejemplar dados los siguientes datos:

1. Nombre
2. Fecha de nacimiento
3. País
4. Sexo (M-H)
5. Color (Atigrado, Leonado, Blanco, Negro)
6. Nombre de padre
7. Nombre de la madre
8. Tomo
9. Folio
10. Tatuaje
11. Número de registro
12. Número de camada.

Se debe mostrar mensaje de verificación para que el usuario verifique los datos que escribió.

Observaciones:

1. Si el criador del ejemplar tiene afijo, se debe adicionar al final del nombre
2. El nomenclador del código debe ser CBOX-AAMMDDn0Camada y es único.
3. El tomo y el folio se debe tomar automáticamente teniendo en cuenta el último tomo y folio asignados, para de esta manera evitar confusiones y debe ser único, se debe permitir su modificación, pues hay ejemplares que no son del club y por tanto no tendrían estos datos.
4. El tatuaje se forma por la unión del tomo y folio, por ejemplo para un Tomo: A y un Folio: 33 el tatuaje sería A33, se debe permitir modificar.
5. El ejemplar puede tener hasta 2 criadores

Tabla 4 HU-1 Registrar ejemplar.

Historia de usuario	
Número: 5	**Nombre historia de usuario:** Registrar asociado.
Modificación de historia de usuario número: 1	

Referencia: RF11	
Programador: Orlando Castillo	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Se debe registrar los datos de los asociados: 1. Nombre 2. Primer apellido 3. Segundo apellido 4. Número de CI 5. Dirección 6. Teléfono 7. Provincia 8. Municipio 9. Ciudadanía 10. Fecha de ingreso 11. Ocupación 12. Nivel Escolar 13. Asociado	
Observaciones:	

Tabla 5 HU-5 Registrar asociado

Historia de usuario	
Número: 6	**Nombre historia de usuario: Buscar ejemplar.**
Modificación de historia de usuario número: 1	
Referencia: RF5	
Programador: Orlando Castillo	**Iteración asignada:** 1

Prioridad en negocio: Alta	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Medio	**Puntos reales: 1**

Se debe permitir la búsqueda de un ejemplar por:

1. Nombre
2. Sexo
3. Nombre del padre
4. Nombre de la madre
5. Criador
6. Propietario
7. Copropietario
8. País

Observaciones:

Tabla 6 HU-6 Buscar ejemplar

Historia de usuario	
Número: 16	**Nombre historia de usuario: Buscar asociado**
Modificación de historia de usuario número: 1	
Referencia: RF15	
Programador: Orlando Castillo	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 1 semana.
Riesgo en desarrollo: Medio	**Puntos reales:** 1

Se debe permitir la búsqueda de un asociado por los siguientes datos:

1. Nombre
2. Primer apellido.
3. Segundo apellido.
4. Número de CI
5. Fecha de ingreso.

6. Ocupación. 7. Nivel escolar.
Observaciones:

Tabla 7 HU-16 Buscar asociado

Historia de usuario	
Número: 7	**Nombre historia de usuario: Registrar prueba de confirmación.**
Modificación de historia de usuario número: 1	
Referencia: RF6	
Programador: Orlando Castillo	**Iteración asignada:** 2
Prioridad en negocio: Media	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Se debe mostrar una interfaz que permita registrar la prueba de confirmación de un ejemplar dados los siguientes datos: 1. Fecha de realización 2. Medida del Ejemplar (altura a la cruz, largo y peso) 3. Fórmula de los Ojos 4. Fórmula Dental 5. Especificar si el ejemplar es apto o no para la monta. 6. Fotografía del ejemplar. 7. Descripción.	
Observaciones: Las pruebas de confirmación se deben realizar cada 2 años.	

Tabla 8 HU-7 Registrar prueba de confirmación

Historia de usuario	
Número: 8	**Nombre historia de usuario: Registrar resultados de la cruza**

Modificación de historia de usuario número: 1	
Referencia: RF8	
Programador: Orlando Castillo	**Iteración asignada:** 2
Prioridad en negocio: Media	**Puntos estimados:** 1 semana
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Se debe mostrar una interfaz que permita registrar los resultados de una cruza dados los siguientes datos: 1. Nombre del ejemplar macho 2. Nombre de la ejemplar hembra 3. Fecha 4. Si la perra quedó gestada. 5. Cantidad de hembras 6. Cantidad de machos 7. Cantidad de blancos 8. Cantidad de negros 9. Cantidad labio leporino 10. Cantidad con falta testicular (se registra a los 6 meses) 11. Cantidad de cachorros con más de 1/3 de color blanco. 12. Cantidad de fallecidos antes de los 7 días 13. Cantidad de hembras criadas 14. Cantidad de machos criados Se debe mostrar mensaje de verificación para que el usuario verifique los datos que escribió.	
Observaciones: 1. La cantidad con falta testicular se registra cuando la camada cumple los 6 meses. 2. Si la perra no queda gestada se debe especificar, y los demás parámetros se podrán en cero.	

Tabla 9 HU-8 Registrar resultados de la cruza

Historia de usuario

Número: 9	**Nombre historia de usuario: Registrar resultado de eventos**
Modificación de historia de usuario número: 1	
Referencia: RF7	
Programador: Orlando Castillo	**Iteración asignada:** 2
Prioridad en negocio: Media	**Puntos estimados :**1 semana
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Se debe registrar el resultado de los eventos por días atendiendo al anexo siguiente: 1. Evento 2. Tipo de evento 3. Número de catálogo 4. Tipo de Clase 5. Fecha 6. Juez 7. Títulos que obtuvo.	
Observaciones: Se debe poder especificar si el perro se convierte en campeón cubano, campeón de Latinoamérica, campeón internacional.	

Tabla 10 HU-9 Registrar resultado de eventos

Historia de usuario	
Número: 10	**Nombre historia de usuario: Modificar asociado**
Modificación de historia de usuario número: 1	
Referencia: RF13	
Programador: Orlando Castillo	**Iteración asignada:** 2
Prioridad en negocio: Media	**Puntos estimados:** 1 semana

Riesgo en desarrollo: Medio	**Puntos reales:** 1
Se debe poder modificar los datos de los asociados: 1. Nombre 2. Primer apellido 3. Segundo apellido 4. Número de CI 5. Dirección 6. Teléfono 7. Provincia 8. Municipio 9. Ciudadanía 10. Fecha de ingreso 11. Ocupación 12. Nivel Escolar 13. Asociado	
Observaciones:	

Tabla 11 HU-10 Modificar asociado

Historia de usuario	
Número: 11	**Nombre historia de usuario: Posibles reproductores**
Modificación de historia de usuario número: 1	
Referencia: RF9	
Programador: Orlando Castillo	**Iteración asignada: 3**
Prioridad en negocio: Baja	**Puntos estimados:** 1día
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Generar un reporte dividido por sexo de los ejemplares de los posibles reproductores, en formato pdf.	

Observaciones: Los posibles reproductores son: Los machos de 18 meses o más y que sean aptos para la monta y que esté vivo. Las hembras entre 18 meses y 7 años, que sean aptas para la monta y que estén vivas.

Tabla 12 HU-11 Posibles reproductores

Historia de usuario	
Número: 12	**Nombre historia de usuario: Ejemplares de un asociado**
Modificación de historia de usuario número: 1	
Referencia: RF14	
Programador: Orlando Castillo	**Iteración asignada: 3**
Prioridad en negocio: Baja	**Puntos estimados:** 1 día
Riesgo en desarrollo: Medio	**Puntos reales: 1**
Generar un reporte dividido por ejemplares propios y ejemplares criados de un asociado en formato pdf.	
Observaciones:	

Tabla 13 HU-12 Ejemplares de un asociado

Historia de usuario	
Número: 13	**Nombre historia de usuario: Obtener resultado de eventos de un ejemplar.**
Modificación de historia de usuario número: 1	
Referencia: RF 10	
Programador: Orlando Castillo	**Iteración asignada:** 2
Prioridad en negocio: Media	**Puntos estimados:** 1 día.

Riesgo en desarrollo: Medio	**Puntos reales:** 1
Permitir obtener un reporte de un ejemplar de todos los eventos en que ha participado con los resultados de cada uno.	
Observaciones: Debe quedar especificado si el ejemplar es campeón cubano, Latinoamérica o internacional en que evento obtuvo el título.	

Tabla 14 HU-13 Obtener resultado de eventos de un ejemplar

Historia de usuario	
Número: 14	**Nombre historia de usuario: Obtener pruebas de confirmación**
Modificación de historia de usuario número: 1	
Referencia: RF11	
Programador: Orlando Castillo	**Iteración asignada:** 2
Prioridad en negocio: Media	**Puntos estimados:** 1 día
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Permitir obtener un reporte de un ejemplar con todos los datos de las pruebas de confirmación que se le han realizado.	
Observaciones:	

Tabla 15 HU-14 Obtener pruebas de confirmación

Historia de usuario	
Número: 15	**Nombre historia de usuario: Dar baja a asociado**
Modificación de historia de usuario número: 1	
Referencia: RF15	

Programador: Orlando Castillo	**Iteración asignada:** 3
Prioridad en negocio: baja	**Puntos estimados:** 2día
Riesgo en desarrollo: Medio	**Puntos reales:** 1
Escoger ejemplar del listado y cambiarle el estado a “baja”.	
Observaciones:	

Tabla 16 HU-15 Dar baja a ejemplar

2.6 Plan de Iteraciones.

Las HU seleccionadas para cada iteración son desarrolladas y probadas en un ciclo de iteración, de acuerdo con el orden preestablecido. Al comienzo de cada ciclo, se realiza una reunión de planificación de la iteración. Cada HU se traduce en tareas específicas de programación. Así mismo, para cada HU se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan; pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores.

Iteración 1: En la primera iteración se implementarán las HU con prioridad alta, obteniendo al final de la misma una primera versión de prueba, dando al sistema las primeras funcionalidades, y centrándose en registrar ejemplares y asociados entre otras.

Iteración 2: En la segunda iteración se implementarán las HU con prioridad de negocio media, relacionada con registrar pruebas de confirmación, registrar resultados de la cruza, registrar resultado de eventos entre otras funcionalidades.

Iteración 3: En la tercera iteración después de haberse implementado las funcionalidades específicas y principales se elaboran las últimas HU con prioridad baja relacionada con generar reportes de los posibles reproductores, de los ejemplares propios y los ejemplares criados de un asociado, entre otras funcionalidades.

Plan de duración de las iteraciones.

Una vez definidas las historias de usuarios a implementar en cada una de las iteraciones, se procede a definir el tiempo de duración de cada una de las historias creando el plan de duración de iteraciones que propone la metodología utilizada. Este plan tiene como finalidad mostrar la duración de cada iteración y el orden en que las historias de usuarios serán implementadas.

Iteración	Orden de las HU a implementar	Duración total de las iteraciones
Iteración 1	**1.** Registrar ejemplar. **2.** Traspaso de Propiedad. **3**. Modificar ejemplar. **4.** Generar Folio. **5.** Registrar asociado. **6.** Buscar ejemplar. **7.** Buscar asociado.	9 semanas
Iteración 2	**1.** Registrar prueba de confirmación. **2.** Registrar resultados de la cruza. **3.** Registrar resultado de eventos. **4.** Modificar asociado. **5.** Obtener resultado de eventos de un ejemplar. **6.** Obtener pruebas de confirmación.	6 semanas
Iteración 3	**1.** Posibles reproductores **2.** Ejemplares de un asociado **3.** Dar baja a ejemplar	3 semanas

Tabla 17 Plan de iteraciones

2.7 Conclusiones

Con la elaboración de este capítulo se realizó el estudio de los procesos del negocio donde se realizó la modelación de los mismos. Se tiene una vista del sistema a desarrollar, se plasmaron los requisitos funcionales y no funcionales a tener en cuenta obteniendo una idea general de las funcionalidades que debe cumplir la aplicación. Se planifica además, el desarrollo de las iteraciones a realizar y el orden en que se implementarán las historias de usuario de acuerdo con la prioridad que se le asignó.

Capítulo 3: Diseño, implementación y pruebas.

3.1 Introducción.

En el presente capítulo se modelan los artefactos que exige la metodología ágil SXP para la fase de desarrollo. Para el diseño de las aplicaciones, esta metodología no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC que en este capítulo se exponen. Además, se realizan las pruebas que es un elemento crítico para la garantía de la calidad del *software* y representa una revisión final de las especificaciones del diseño y de la codificación.

3.2 Patrones usados en el diseño.

El desarrollo del sistema utilizando el marco de trabajo *Symfony* proporciona ventajas significativas para los desarrolladores de *software*. El *framework* mencionado es capaz de fusionar buenas prácticas de trabajo por sí mismo, de forma que los desarrolladores no tengan que preocuparse por implementar varios de los patrones arquitectónicos y de diseño más utilizados en la actualidad, ya que *Symfony* los implementa.

3.2.1 Patrones GRASP.

- **Creador**

En la clase Acción de cada uno de los módulos se encuentran definidas las acciones del sistema y se ejecutan cada una de ellas. En las acciones se crean los objetos de las clases que representan las entidades, evidenciando de este modo que la clase Acción es "creador" de dichas entidades.

- **Experto**

Este es uno de los más utilizados, puesto que *propel* es la librería externa que utiliza *Symfony* para realizar su capa de abstracción en el modelo. *Symfony* divide el modelo en una capa de abstracción a datos y otra de acceso a datos en estas clases se encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

- **Alta Cohesión**

Symfony permite asignar responsabilidades con una alta cohesión, por ejemplo la clase Acción tiene la responsabilidad de definir las acciones para las plantillas y colabora con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades, es decir, está formada por

diferentes funcionalidades que se encuentran estrechamente relacionadas proporcionando que el *software* sea flexible frente a grandes cambios.

- **Controlador**

Todas las peticiones son manejadas por un solo controlador frontal (sfAcción), que es el punto de entrada único de toda la aplicación en un entorno determinado. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario.

- **Bajo Acoplamiento**

Este patrón se manifiesta en cada uno de los módulos del sistema, la clase Acción hereda solamente de sfAcción para lograr un bajo acoplamiento de clases.

3.2.2 Patrones GOF.

En la categoría creacionales:

- ***Singleton* (Instancia única):** Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En el controlador frontal hay una llamada a *sfContext*: *getInstance* (). En una acción, el método *getContext* (), un objeto muy útil que guarda una referencia a todos los objetos del núcleo de *Symfony*.
- ***Abstract Factory* (Fábrica abstracta):** Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Cuando el *framework* necesita por ejemplo crear un nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

En la categoría estructurales:

- ***Decorator* (Envoltorio):** Añade funcionalidad a una clase dinámicamente. El archivo layout.php, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el *layout*, o si se mira desde el otro punto de vista, el *layout* decora la plantilla. La siguiente imagen ilustra lo anteriormente descrito.

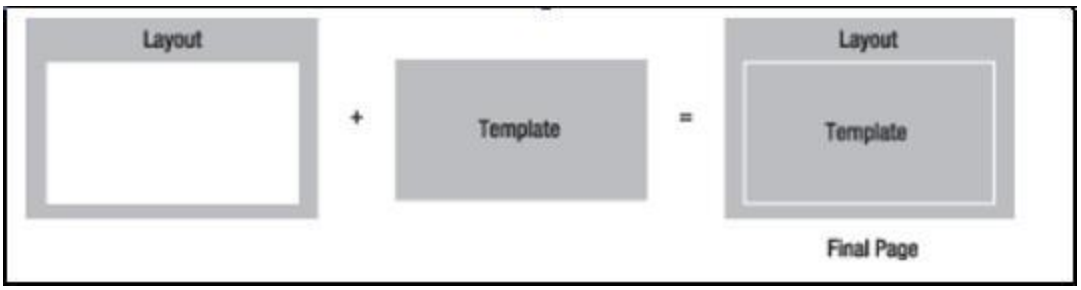


Ilustración 4 Ejemplo del patrón Envoltorio.

- **Composite (Objeto compuesto):** Permite tratar objetos compuestos como si de uno simple se tratase. Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.

En la categoría comportamiento:

- **Command (Acción):** El cual permite que diferentes objetos puedan ejecutar la misma acción sin necesidad de repetir su declaración e implementación.

3.3 Tarjetas CRC.

Las tarjetas CRC son técnicas de modelado creadas para ayudar a los desarrolladores de *software* a crear diseños de clases orientados a responsabilidades. Dichas tarjetas constan de tres secciones, nombre de la clase, responsabilidades y colaboradores. La sección responsabilidades se listan cada una de las funciones o tareas que debe ser capaz cumplir un objeto de dicha clase mientras que la sección colaboradora contiene otras clases del diseño. (Ver anexo3).

Tabla 18 Tarjeta CRC-Módulo1

Módulo 1	
Funcionalidades.	**Colaboraciones.**
ExecuteAddEjemplar() ExecuteUpdateEjemplar() ExecuteDeleteEjemplar() ExecuteBuscarEjemplar() ExecuteRegistrarAfijo() ExecuteAddAsociado() ExecuteBuscarAsociado() ExecuteUpdateAsociado() ExecuteDeleteAsociado() ExecutePruebasConfirmacion() CargardatosPruebas() ExecuteRegistrarResultadoCruza() ExecuteUpdateResultadoCruza() CargarDatosResultadosCruzada() ExecuteEstablecerRelacion() ExecuteResultadoEvento() ExecuteResultadoEventoCatalogo() CargarDatosEvento() CargarDatosEjemplar() CargarDatosAsociado()	sfActions

3.4 Tareas de ingeniería

Las tareas de ingeniería son un conjunto de acciones que se realizan con el objetivo de resolver las historias de usuarios, una historia de usuario puede tener una o más tareas de ingeniería en dependencia de la complejidad de la funcionalidad a desarrollar. (Ver anexo4)

Tabla 19 Tarea4_Registrar ejemplar

Tarea de ingeniería	
Número de la tarea: 1	**Número de la HU**: 1
Nombre de la tarea: Validar campos del formulario para registrar ejemplar.	
Tipo de tarea: Desarrollo	
Fecha Inicio: 10/4/2010	**Fecha Fin:** 10/4/2010
Programador responsable: Orlando Castillo	
Descripción: Verificar que los datos que se inserten en los campos sean correctos.	

Tarea de ingeniería	
Número de la tarea: 2	**Número de la HU**: 1
Nombre de la tarea: Diseñar el formulario para registrar un ejemplar	
Tipo de tarea: Diseño	
Fecha Inicio: 11/4/2010	**Fecha Fin:** 11/4/2010
Programador responsable: Orlando Castillo	
Descripción: Crea un formulario que permitirá registrar ejemplar.	

Tabla 20 Tarea 4_Registrar asociado

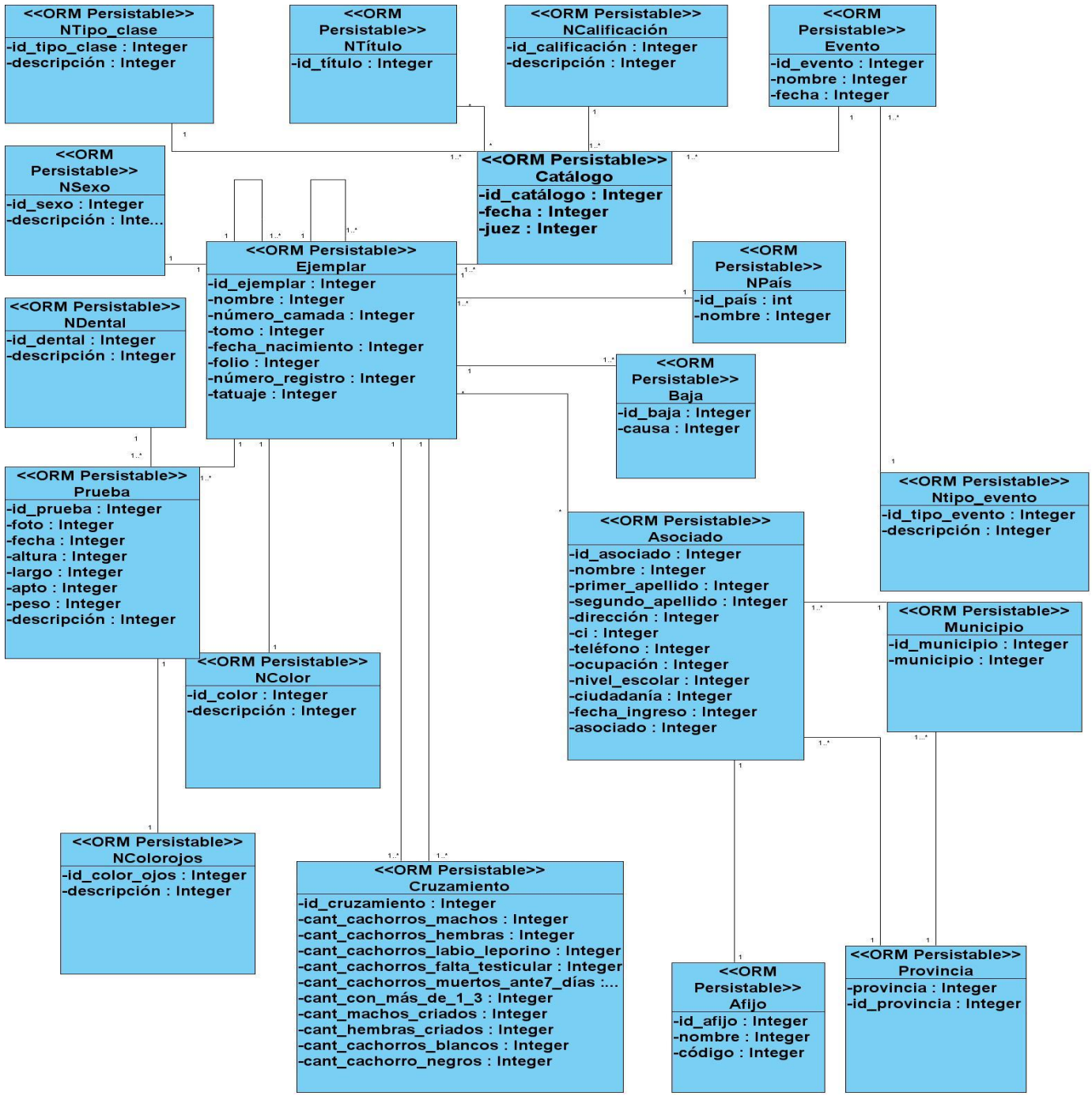
Tarea de ingeniería	
Número de la tarea: 1	**Número de la HU**: 5
Nombre de la tarea: Validar campos del formulario para registrar asociado.	
Tipo de tarea: Desarrollo	
Fecha Inicio: 16/4/2010	**Fecha Fin:** 16/4/2010
Programador responsable: Orlando Castillo	
Descripción: Verificar que los datos que se inserten en los campos sean correctos.	

Tarea de ingeniería	
Número de la tarea: 2	**Número de la HU**: 5
Nombre de la tarea: Diseñar el formulario para registrar un asociado	
Tipo de tarea: Diseño	
Fecha Inicio: 17/4/2010	**Fecha Fin:** 17/4/2010
Programador responsable: Orlando Castillo	
Descripción: Crea un formulario que permitirá registrar asociado.	

3.5 Diseño de la Base Datos.

En el proceso de abstracción que conduce a la creación de una base de datos, desempeña una función prioritaria el modelo de datos. El modelo de datos, como abstracción del universo de discurso, es el enfoque utilizado para la representación de las entidades y sus características dentro de la base de datos. Es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos, semántica asociada a los datos y restricciones de consistencia.

Ilustración 5 Modelo de datos



3.6 Diagrama de Despliegue.

Los diagramas de despliegue muestran nodos, conexiones, componentes y objetos. Los nodos representan objetos físicos con recursos computacionales como procesadores y periféricos;

pueden mostrarse como una clase (e.g una familia de procesadores) o una instancia, por lo que su nombre sigue la misma sintaxis establecida para clases y objetos. Las conexiones son asociaciones de comunicación entre los nodos, y se etiquetan con un estereotipo que identifica el protocolo de comunicación o la red utilizada. Los componentes son archivos de código ejecutable, que residen y se ejecutan dentro de un nodo; se pueden representar relaciones de dependencia entre los componentes que, de manera similar a las dependencias entre paquetes, corresponden al uso de servicios.

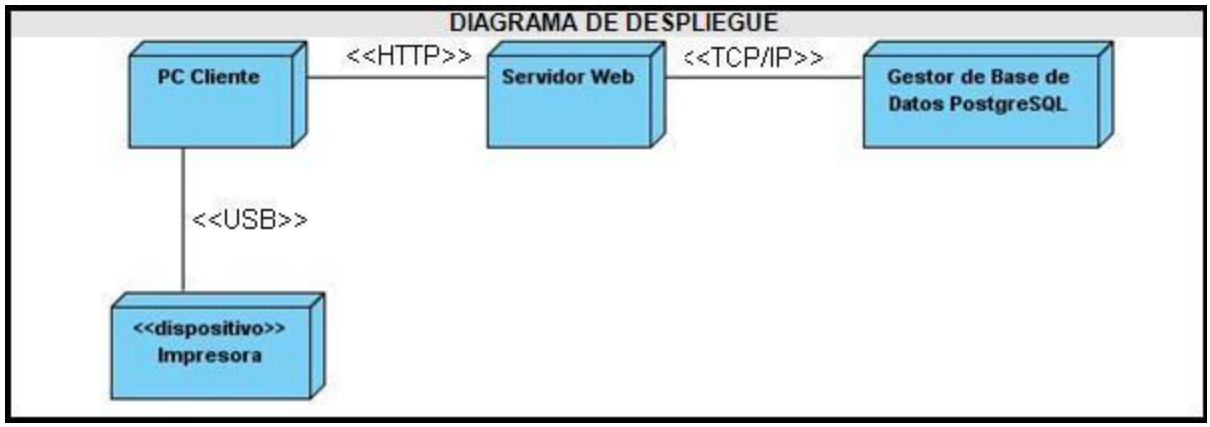


Ilustración 6 Diagrama de despliegue

3.7 Pruebas

Uno de los pilares fundamentales de SXP es el proceso de pruebas, el cual anima a los desarrolladores a probar constantemente tanto como sea posible. Mediante esta filosofía se reduce el número de errores no detectados así como el tiempo entre la introducción de éste en el sistema y su detección, esto contribuye a elevar la calidad de los productos desarrollados y a la seguridad de los programadores a la hora de introducir cambios o modificaciones. Esta metodología divide las pruebas en dos grupos: pruebas unitarias, desarrolladas por los programadores, encargadas de verificar el código de forma automática y las pruebas de aceptación, destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.

3.7.1 Pruebas Unitarias

La producción de código está dirigida por las pruebas unitarias. Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación

del sistema. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial.

3.7.2 Pruebas de Aceptación

Las pruebas de aceptación son pruebas de caja negra que se crean a partir de las historias de usuario, durante las iteraciones las HU (Historias de usuarios) seleccionadas serán traducidas a pruebas de aceptación. En ellas se especifican, desde la perspectiva del cliente, los escenarios para probar que una HU ha sido implementada correctamente. Una HU puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de éstas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable. (Ver anexo5)

Caso de prueba aceptación	
Número de caso de prueba: 1	**Nombre historia de usuario:** Registrar ejemplar
Nombre de la persona que realiza la prueba: Addisleidys Castro	
Descripción de la prueba: En la página principal aparecen varias opciones, al ejecutar el *link* nuevo ejemplar va hacia la interfaz de adicionar ejemplar. Aparecen los datos a llenar, el administrador del sistema al pulsar el botón aceptar quedará registrado el ejemplar.	
Condiciones de ejecución: El ejemplar no puede haber sido registrado antes.	
Entrada/Pasos de ejecución: El administrador debe introducir los datos nombre, seleccionar fecha de nacimiento, seleccionar sexo, seleccionar color, seleccionar país, seleccionar nombre del padre, seleccionar nombre de la madre, introducir tomo, folio, tatuaje es la combinación tomo_folio, número de registro y número de camada y se ejecuta la acción Aceptar.	
Resultado esperado: Los datos fueron insertados correctamente y el ejemplar ha sido registrado en la base de datos.	
Evaluación de la prueba: Satisfactoria	

Tabla 21 Prueba1 Registrar ejemplar

Caso de prueba aceptación	
Número de caso de prueba: 5	**Nombre historia de usuario: Registrar asociado.**
Nombre de la persona que realiza la prueba: Addisleidys Castro	
Descripción de la prueba: En la página principal aparecen varias opciones, al ejecutar el *link* nuevo asociado va hacia la interfaz de adicionar asociado. Aparecen los datos a llenar, el administrador del sistema al pulsar el botón aceptar quedará registrado el asociado.	
Condiciones de ejecución: El asociado no puede haber sido registrado antes.	
Entrada/Pasos de ejecución: El administrador debe introducir el nombre, primer apellido, segundo apellido, carné de identidad, dirección, teléfono, seleccionar provincia, seleccionar municipio, introducir ciudadanía, seleccionar fecha de ingreso, introducir ocupación, introducir nivel escolar, asociado (si o no) y se ejecuta la acción Aceptar.	
Resultado esperado: Los datos fueron insertados correctamente y el asociado ha sido registrado en la base de datos.	
Evaluación de la prueba: Satisfactoria	

Tabla 22 Prueba5 Registrar asociado

Caso de prueba aceptación	
Número de caso de prueba: 6	**Nombre historia de usuario: Buscar ejemplar**
Nombre de la persona que realiza la prueba: Addisleidys Castro	
Descripción de la prueba: En la página principal aparecen varias opciones, al ejecutar el *link* gestionar ejemplar va hacia la interfaz de buscar ejemplar, donde se especifica el criterio de búsqueda por el que se desea buscar al ejemplar, se da clic en el botón buscar y aparecerán abajo los resultados de la búsqueda, con la opción de escoger la cantidad de páginas en los que se quiere ver los resultados de la búsqueda.	
Condiciones de ejecución: El ejemplar tiene que estar registrado con antelación para cumplir objetivo de buscarlo.	

Entrada/Pasos de ejecución: El administrador escoge el criterio de búsqueda que puede ser nombre, sexo, padre, madre, criador, propietario, copropietario, país y ejecuta la acción Buscar.
Resultado esperado: La búsqueda se realizó con éxito de acuerdo con el criterio de búsqueda especificado, se obtuvieron los resultados previstos.
Evaluación de la prueba: Satisfactoria

Tabla 23 Buscar ejemplar

3.8 Conclusiones

En este capítulo se desarrolló el modelo de despliegue del sistema a implementar. Se hizo el diseño del modelo de datos que utilizará la aplicación. Se desarrollaron además las tareas correspondientes para darle solución a las historias de usuarios y las pruebas de aceptación para comprobar que el sistema cumple con los requerimientos del cliente, además de verificar aspectos importantes como es la seguridad de la aplicación. Con este capítulo se culmina la propuesta del sistema a desarrollar.

Capítulo 4: Estudio de factibilidad.

4.1 Introducción

Desde los primeros momentos del desarrollo de un *software*, resulta necesario determinar si el mismo resultará factible o no. Para ello se debe realizar un estudio detallado de los beneficios que este aporta y de las inversiones que implicará tanto en la esfera organizativa (entiéndase estructuras, procesos y personas), como en la económica y técnica (teniendo en cuenta habilidades, experiencias, recursos), para llevar a cabo su implementación. Este estudio incluye una planificación del trabajo a realizar referente al tiempo que demorará el desarrollo del *software* a partir de la cantidad de personas requeridas y del tamaño del mismo.

Una de las tareas de mayor importancia en la planificación de proyectos de *software* es el cálculo de la estimación, la cual consiste en determinar con cierto grado de certeza, los recursos de *hardware* y *software*, costo, tiempo y esfuerzo necesarios para el desarrollo de los mismos. En este capítulo se realiza un estudio de la factibilidad para la realización del sistema propuesto, haciendo una estimación del esfuerzo necesario. Para el cálculo se decidió utilizar el método COCOMO II, este resulta muy útil para estimar un proyecto en forma global, cuando se tiene un conjunto de casos de uso bastante amplio y con escaso nivel de detalle. Así como los beneficios tangibles e intangibles que reportaría la aplicación y se realiza el análisis de costo y beneficio.

4.2 Características del Proyecto.

El primer paso a llevar a cabo para la estimación del proyecto consiste en la obtención de los Puntos de Función desajustados, los cuales están dados por la suma de cada una de las entradas, las salidas y las consultas externas del sistema, así como los archivos lógicos internos y de interfaz externo.

Entradas externas.

Son la entrada de datos del usuario o de control que ingresan desde el exterior del sistema para agregar y/o cambiar datos a un archivo lógico interno. Se definen como un proceso elemental mediante el cual ciertos datos cruzan la frontera del sistema desde afuera hacia adentro.

Entradas externas	Cantidad de archivos referenciados	Cantidad de elementos de datos	Clasificación (baja, media o Alta)
Registrar ejemplar.	1	10	baja
Modificar	1	14	baja

ejemplar			
Traspaso de propiedad	1	3	baja
Registrar asociado.	1	9	baja
Buscar ejemplar.	1	10	baja
Buscar asociado.	1	7	baja
Registrar prueba de confirmación.	1	9	baja
Registrar resultados de la cruza.	1	13	baja
Registrar resultado de eventos.	1	7	baja
Modificar asociado.	1	9	baja

Tabla 24 Entradas externas.

Salidas externas

Se definen como un proceso elemental con componentes de entrada y de salida mediante el cual datos simples y datos derivados cruzan la frontera del sistema desde adentro hacia fuera.

Salidas externas	Cantidad de archivos referenciados	Cantidad de elementos de datos	Clasificación (baja, media o Alta)
Generar folio.	1	1	baja
Obtener resultado de eventos de un ejemplar.	1	1	baja
Obtener pruebas de confirmación.	1	1	baja
Posibles	1	1	baja

reproductores.			
Ejemplares de un asociado.	1	1	baja

Tabla 25 Salidas externas.

Archivos Lógicos Internos

Constituyen un grupo de datos relacionados lógicamente e identificables por el usuario, que residen enteramente dentro de los límites del sistema y se mantienen a través de entradas externas.

Nombre del fichero Interno	Cantidad de archivos referenciados	Cantidad de elementos de datos	Clasificación (baja, media o Alta)
Generar reportes	1	5	baja

Tabla 26 Archivos Lógicos Internos.

4.2.1 Estimación inicial

Puntos de Función sin ajustar (UFP): se obtiene a través de la suma del aporte de todos los elementos.

Elementos	Baja		Media		Alta		Aportes
	Cant	Valor	Cant	Valor	Cant	Valor	
Entradas externas	10	3	0	4	0	6	30
Salidas externas	5	4	0	5	0	7	20

Consultas externas	0	3	0	4	0	6	0
Archivos lógicos internos	1	7	0	10	0	15	7
Archivos de interfaz externos	0	5	0	7	0	10	0
Total	**UFP =57**						

Tabla 27 Puntos de Función desajustados.

Una vez que se han obtenido los Puntos de Función sin ajustar del sistema se puede estimar el esfuerzo, para esto se utilizará el método COCOMO II.

4.3 Cálculo de instrucciones fuertes, esfuerzo, tiempo de desarrollo, cantidad de hombres y costo.

El método COCOMO II consiste básicamente en la aplicación de ecuaciones matemáticas sobre los Puntos de Función sin ajustar estimados para un proyecto. Estas ecuaciones se encuentran ponderadas por ciertos factores de costo que influyen en el esfuerzo requerido para el desarrollo del *software*. La meta es obtener un número que caracterice completamente al sistema.

4.3.1 Cálculo del esfuerzo nominal.

$$PM_{nominal} = A * (Size)^{E}$$

***PM* nominal**: es el esfuerzo nominal requerido en meses-hombre.

***A*:** Es una constante que se utiliza para capturar los efectos multiplicativos en el esfuerzo requerido de acuerdo con el crecimiento del tamaño del *software*. El modelo la calibra con un valor de 2.94.

Size: Es el tamaño estimado del *software*, en Puntos de Función sin ajustar (convertibles a KSLOC). Se calcula el producto de los puntos de función sin ajustar por un factor de conversión que depende del lenguaje a utilizar en el desarrollo del sistema. Se utiliza PHP (factor de conversión = 53 SLOC/UFP).

Size = 53 * 57 = 3021 SLOC

Size = **3.02 KSLOC**

E: Es una constante denominada Factor escalar. Se calcula ponderando las variables escalares, mediante la ecuación:

E = 0.91 + 0.01 * Σ (Wi)

Wi = valor de la variable escalar

Donde las Wi se muestran en la siguiente tabla

Variable	Descripción	Valor
PREC	El sistema es muy familiar	2.25
FLEX	Algo de relajación en cuanto a la flexibilidad del desarrollo	2.50
RESL	La arquitectura es sólida y los riesgos generalmente se mitigan	3.50
TEAM	La interacción del equipo es altamente cooperativa	2.80
PMAT	La madurez del proceso *software* es baja	1.80
Total		**12.85**

Tabla 28 Factor escalar.

E = 0.91 + 0.01 * 12.85 = **1.167**

Por tanto:

PM nominal = A * (Size) E

PM nominal = 2.94 * (3.0 KSLOC) ^ 1.167

PM nominal = **10.54mes-hombre**

4.3.2 Cálculo del esfuerzo ajustado.

PM ajustado = PM nominal * Π (MEi)

Multiplicador	Descripción	Valor
PERS	Se tienen analistas y programadores con alta eficiencia y capacidad de trabajo en equipo.	0.90
RCPX	Las exigencias de confiabilidad, documentación y volumen de datos son moderadas, y la complejidad del producto es baja**.**	1.05
RUSE	No se pretende reutilizar nada.	1.07
PDIF	No existen restricciones en cuando al tiempo de CPU o al consumo de memoria, la plataforma es muy estable.	1.00
PREX	Tanto los analistas como los programadores tienen aproximadamente 6 meses de experiencia en la aplicación, la plataforma, el lenguaje y las herramientas utilizadas.	0.95
SCED	Se requiere terminar el proyecto en el tiempo estimado.	1.00
FCIL	Se tienen herramientas CASE simples e infraestructura de comunicaciones básicas	0.95
Total		**0.96**

Tabla 29 Multiplicadores de esfuerzo.

Π (MEi) = ***0.96.***

***PM* ajustado** = 10.54 * 0.96

***PM* ajustado** = **10.11 Mes-hombre.**

4.3.3 Cálculo del tiempo de desarrollo, cantidad de hombres y costo.

Valores calibrados: A = 2.94**; B** = 0.91**; C** = 3.67**; D** = 0.24

F = D + 0.2 * (E - B)

F= 0.24 + 0.2 * (1.167– 0.91) = 0.29

TDEV (Tiempo de desarrollo) = C * (PM ajustado) ^**F**

TDEV (Tiempo de desarrollo) = 3.67 * (10.11) ^0.29 = **7.17**

CH (Cantidad de Hombres) = PM ajustado / TDVE

CH (Cantidad de Hombres) = 10.11 / 7.17

CH (Cantidad de Hombres) ≈ 1.4 ≈ **1 persona.**

Como la cantidad real de hombres disponibles para el desarrollo de la aplicación es ***2***, al reajustar el tiempo de desarrollo según la cantidad de hombres, resultó un tiempo equivalente a (***7.17/2***)=***5.05*** meses.

Salario promedio: Para determinar el salario promedio se tuvo en cuenta que los desarrolladores son estudiantes de 5to año, por lo que se toma como salario correspondiente $100.00

Costo = 2 * 100 * 10.11 = 2022$

Cálculo de:	Valor
Esfuerzo	10.11 Mes-hombre
Tiempo de desarrollo	5.05 Meses
Cantidad de hombres	2 Personas
Salario medio	$100.00
Costo	$2022.00

Tabla 30 Resultados.

4.3.4 Beneficios tangibles e intangibles.

Tangibles.

Teniendo en cuenta que la aplicación no es un producto desarrollado para la comercialización, no es válido mencionar beneficios económicos. Se puede decir que el costo por desarrollar la aplicación es de $2022.00 MN (moneda nacional), el cual es perfectamente reparable si en un futuro se comercializará.

Intangibles.

Con el uso de la aplicación se podrá reducir considerablemente el tiempo dedicado al desarrollo de las planificaciones, a la vez que se garantiza que no existan errores en las mismas por doble utilización de recursos, algo que pasa con frecuencia si este proceso se desarrolla manualmente como ocurre en la institución. En este lugar solo están automatizadas algunas acciones, esto trae consigo que un grupo de personas se dedique a la captura de datos, teniendo que pasar largas horas en ello, el *software* que se propone evita este tipo de desgaste físico, así como las

dificultades que puede crear no hacerlo de manera correcta, al proveer a los usuarios de un servicio de captura automática que sigue las planificaciones almacenadas en la BD. La tecnología utilizada para el desarrollo del sistema es totalmente libre, por tanto, no hay que incurrir en gastos en el pago de licencias de uso. El sistema es portable por lo que un cambio de plataforma para la implantación del mismo es viable y factible, y no hay que incurrir en muchos cambios. Es una solución a los problemas encontrados en la aplicación que se usa actualmente y propone una interfaz sencilla y fácil de usar.

Análisis de costo.

El desarrollo de la aplicación no constituye un gasto considerable pues todas las herramientas que se han empleado en su desarrollo son libres y de código abierto. El sistema está orientado al trabajador y es de fácil aprendizaje.

4.4 Conclusiones.

En este capítulo se describió el estudio de factibilidad realizado correspondiente al sistema propuesto, teniendo en cuenta el costo estimado y los beneficios que reportará al ser implantado. El estudio realizado, ha proporcionado valiosos argumentos que permiten llegar a la conclusión de que la solución de *software* es factible.

Conclusiones generales.

A partir del análisis realizado en el Club Bóxer de Cuba se determinaron una serie de procesos que se realizaban de forma manualmente. En estos procesos están involucrados datos e informaciones de gran importancia con relación a los ejemplares y asociados del Club. Los trabajadores del Club son los que se encargan de mantener bajo control todo este cúmulo de información, de la cual es necesario obtener reportes y datos efectivos para emitir las evaluaciones correspondientes. Para solucionar todas estas dificultades se desarrolló una aplicación que automatiza los procesos principales que tienen lugar en el Club.

- El empleo de los métodos teóricos y empíricos facilitó conocer el estado del objeto de estudio.
- Se obtuvo a partir del análisis realizado a los sistemas similares existentes tantos nacionales como internacionales, el conocimiento necesario para saber como marcha en el mundo el uso de las aplicaciones informáticas dedicadas al control de la cría y selección de especies.
- Se realizó el estudio de las principales metodologías, lenguajes, y herramientas, lo que permitió determinar las que se consideraron factibles para desarrollar el sistema.
- El desarrollo del trabajo utilizando la metodología SXP permitió documentar el mismo desde el comienzo, lo que facilitó su estudio, permitiendo de esta forma, una comprensión más rápida y fácil de la concepción general del sistema.
- El estudio de factibilidad realizado, proporcionó valiosos argumentos que permiten llegar a la conclusión de que la solución de *software*, es factible pues reportará importantes beneficios sin incurrir en mayores gastos.

Como resultado de la investigación se logra el desarrollo de un *software* en el que se da cumplimiento a las especificidades de los objetivos propuestos. Con el valor fundamental de simplificar la demora que produce el procesamiento manual de la información distribuida, y mejorar la gestión de los procesos.

Recomendaciones.

Como resultado del proceso de investigación y realización de la aplicación, han surgido ideas que serían recomendables tener en cuenta para un futuro perfeccionamiento del sistema, a continuación se listan las mismas.

- Continuar con la implementación agregando nuevas funcionalidades y mejorando las existentes en la medida que sea posible y de acuerdo con las necesidades del Club Bóxer de Cuba
- En el *software* Gescab se encontró una funcionalidad muy importante, que permite determinar el porcentaje de consanguinidad de cualquier animal inscrito en el Stud-Book del PRE utilizando el algoritmo Lush (1940). Siguiendo la misma metodología se puede determinar qué consanguinidad tendrá un ejemplar antes de nacer. Sólo se precisa dar los nombres del semental y la hembra y el programa efectúa el cálculo de forma automática, esta funcionalidad se le podría aplicar al sistema realizado.

Bibliografía referenciada.

[1] **FÉLIX ÓSCAR GARCÍA RUBIO**, C. B. S. "Metodologías de Desarrollo de *Software*". Disponible en: http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/tema3_1xh.pdf

[2] **AUTORES**, C. D. Rational Rose Enterprise, 2009a. [Disponible en: http://www142.ibm.com/software/dre/ecatalog/detail.wss?locale=es_ES&synkey=M221280M46834Z27.

[3] **Paradigma Visual para UML Enterprise Edition**, 2009c. [Disponible en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/

[4] **JAMES RUMBAUGH,** I. J., GRADY BOOCH. El Lenguaje Unificado de Modelado.Manual de Referencia. vol. 1.3, 528 p.

[5] **Antes que nada**. (2007). Obtenido de http://lefaltaazucaramicafe.tzhost.org/ http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html

[6] **Amartino, Mariano.** AJAX un nuevo acercamiento a Aplicaciones Web. [En línea] 2005. http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php

[7] **Apache**. Descarga de Spftware. [En línea] http://www.universidad.edu.uy/software.php?action=fullnews&id=3753.

[8] NetBeans. (2010). Recuperado el 8 de enero de 2010, de http://netbeans.org/index_es.html

Bibliografía consultada.

Micel, Jorge. 2002. *Software* de Genealogias. Barcelona,España : Redes.Revista Hispana para el análisis de redes sociales., 2002.

Melgarejo, M. A. (2000). GESCAB: *Software* para la cría del caballo español. Córdoba, España: Red de revistas científicas de América Latina y el Caribe.

Ponssa, E.E., Machado, C.F and Mangudo, P.A. *Desarrollo de un sistema de la dinámica de rodeo de cría bovina.* Buenos Aires, Argentina : s.n.

1997. HTMLPOINT. [En línea] 1997. [Citado el: 3 de enero de 2010.] http://www.htmlpoint.com/javascript/tutorial/01/index.html.

Informática, Escuela Técnica Superior de Ingeniería. Lenguajes y Sistemas Informáticos. [En línea] [Citado el: 3 de enero de 2010.] http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

Torre, Anibal de la. 2006. Lenguajes del lado servidor o cliente. [En línea] 2006. [Citado el: 2 de enero de 2010.] http://adelat.org/media//docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

libroweb.[En línea] [Citado el: 3 de enero de 2010.] http://www.librosweb.es/symfony/capitulo1/symfony_en_pocas_palabras.html.

Angel Alvarez, Miguel. 2009. desarrolloweb. [En línea] 23 de noviembre de 2009. [Citado el: 4 de enero de 2010.] http://www.desarrolloweb.com/articulos/codeigniter.html.

Cultur, Dirección Provincial. 2007. El portal de la cultura matancera. [En línea] 2007. [Citado el: 5 de enero de 2010.] http://www.atenas.cult.cu/rl/informatica/manuales/sl/introduccion_al_SL/sec-ide.html.

Somos libres. [En línea] [Citado el: 4 de enero de 2010.] http://www.somoslibres.org/modules.php?name=News&file=article&sid=990.

[Enlínea]**maestros del web**. [Online] [Cited: enero 5, 2010.] http://www.maestrosdelweb.com/editorial/editores-web-que-facilitan-tu-trabajo/.

[En línea] **Intercom, Grupo.** MailxMail. [Online] [Cited: enero 6, 2010.] http://www.mailxmail.com/curso-configuracion-apache.

[En línea]**1999.** masadelante. [En línea] 1999. [Citado el: 10 de enero de 2010.] http://www.masadelante.com/faqs/servidor.

Diaz, Luis Carlos, Carrillo, Angela y Alvarado, Deisy. Pontificia Universidad Javeriana. [En línea] [Citado el: 10 de enero de 2010.] http://sophia.javeriana.edu.co/~lcdiaz/ADOO2008-1/IngSoftwareEnADOO%28IS-RUP-UML%29.pdf.

2010. Perros Bóxer. [En línea] 2010. [Citado el: 11 de enero de 2010.] http://perrosboxer.net/estandar/.

Ajax. [En línea] [Citado el: 13 de enero de 2010.] http://www.ajaxya.com.ar/temarios/descripcion.php?cod=34&punto=2.

Departamento de Sistemas y Computación. [En línea] [Citado el: 20 de enero de 2010.] http://sistemas.itlp.edu.mx/tutoriales/basedat1/tema1_4.htm.

Tramulla, Jesus. 1997. Introducción a la Documática. [En línea] 1997. [Citado el: 15 de enero de 2010.] http://tramullas.com/documatica/2-3.html.

Redaccenir, S.L. 2003. Portal programas. [En línea] 2003. [Citado el: 20 de febrero de 2010.] http://gratis.portalprogramas.com/Axure-RP.html.

2009. Entorno virtual de aprendizaje. [Online] 2009. [Cited: febrero 21, 2010.] http://eva.uci.cu/.

NetBeans. (2010). Recuperado el 25 de marzo de 2010, de http://netbeans.org/features/php/

Glosario de términos.

Una prioridad al realizar una investigación es trazarse el objetivo de que toda persona que lea el trabajo entienda cada termino utilizado. Quizás el mayor problema es que los términos usados son entendidos por cada ser humano de una manera diferente. A continuación se menciona una definición de los términos principales que resulta imprescindible saber su definición para entender el problema expuesto.

---A---

Afijo: Los afijos son morfemas que se usan en el proceso de derivación y, en ciertos casos, en el proceso de flexión para formar nuevas palabras a partir de otras primitivas y así ensanchar una familia léxica concreta.

Aplicación Web: Aplicaciones que los usuarios pueden utilizar accediendo a un servidor a través de internet o de una intranet mediante un navegador.

Aplicación de escritorio: Es la aplicación creada para ejecutarse en un ordenador de escritorio sobre un sistema operativo de interfaz visual.

---B---

Bóxer: raza de perros de trabajo y compañía, de tamaño medio, de origen Alemán. Obtenida por medio de la cruza de un *bullenbeiser* y un *bulldog*.

BPMN: notación que modela los procesos de negocio, basada en diagramas de flujo fácil de entender.

---C---

CSS: Las hojas de estilo en cascada son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

Club: un grupo de personas, animales u objetos libremente asociadas, que reúne a un número variable de individuos que coinciden en sus gustos.

CASE: Ingeniería asistida por ordenadores.

---E---

Eutanasia: acción del médico que provoca deliberadamente la muerte del paciente.

---H---

Http: Protocolo de Transferencia de Hipertexto. Modo de comunicación para solicitar páginas web.

Html: Hypertext Markup Language / Lenguaje de marcado de hipertexto, usado para escribir documentos para servidores World Wide Web.

---I---

IDE: *Integrated Development Environment* / Entorno de Desarrollo Integrado. Entorno de programación que ha sido empaquetado como un programa de aplicación, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI.

---L---

Labio leporino: Se denomina labio leporino o fisura labial al defecto congénito que consiste en una hendidura o separación en el labio superior. El labio leporino se origina por fusión incompleta de los procesos maxilares y nasomedial del embrión y es uno de los defectos de nacimiento más frecuentes.

LRP: Lista de reserva del producto. Organiza y prioriza todo el trabajo desarrollado en el proyecto.

---M---

Multiplataforma: Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de *software*, que puedan funcionar en diversas plataformas.

---P---

Patrón: Es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).