Universidad de las Ciencias Informáticas
Facultad 4

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE
INGENIERO EN CIENCIAS INFORMÁTICAS

Título:
“Sistema informático para la gestión de información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito.”

Autor: Rainel Andrés Rivas Blanco

Tutor: Agustín Castillo Cordero

Tutora: Sandra Rodríguez Ramírez

La Habana, junio 2015.

“Año 56 de la Revolución.”

"... Quien no sea capaz de luchar por otros, no será nunca suficientemente capaz de luchar por sí mismo."

Fidel Castro Ruz.

DECLARACIÓN DE AUTORÍA

Declaro ser el autor de la presente tesis que tiene por título: “Sistema informático para la gestión de información enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito“ y admito ceder los derechos patrimoniales del mismo a la Universidad de las Ciencias Informáticas.

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

Rainel Andrés Rivas Blanco (autor)

_____

Lic. Sandra Rodríguez Ramírez (Tutora)

_____

Ing. Agustín Castillo Cordero (Tutor)

_____

A mis padres, por ser lo más especial en este mundo.

Un agradecimiento muy especial para:

Mis padres.

Mis hermanos.

Mis amigos.

Mis tutores.

Resumen

Como resultado de las investigaciones realizadas en el Departamento de Veterinaria del municipio Caimito se conoce que en el mismo se genera y almacena la información de forma manual (escrita en papel), lo cual retrasa los diversos procesos de gestión dado el amplio volumen de datos y la densidad de la información con la que se trabaja. Es por esto que se desea desarrollar un sistema informático para la gestión de información que proporcione un procedimiento estable y confiable de almacenamiento y gestión para dicha información. Una vez desarrollada la aplicación, esta permitirá el almacenamiento de grandes volúmenes de información, la actualización correcta y puntual junto a la generación de recursos de alerta con el objetivo de contener o evitar una actividad epizootiológica emergente.

En el presente trabajo de investigación se desarrolla un sistema Informático para la gestión de Información, mediante el uso de tecnologías libres, con el objetivo de proporcionar numerosas ventajas para este proceso.
La creación del sistema fue guiada en función de la metodología cubana de desarrollo ágil SXP la cual permitió mediante la generación de los diferentes artefactos la correcta creación del mismo. Como lenguajes de programación del lado del cliente se utilizaron HTML v5.0, JavaScript v1.5 y CSS v3.0; en cuanto al lenguaje del lado del servidor se utilizó PHP v5.4, a través del framework de desarrollo Symfony 2.5. Se utilizó como entorno de desarrollo integrado el IDE NetBeans v8.0, Apache v2.4 como servidor web, PostgreSQL v9.3 como gestor de bases de datos y Doctrine v2.3 como ORM. Además se implementaron todos los módulos de gestión de reportes que se manejan en la actualmente en el Departamento de Veterinaria y se sentaron las bases para futuras versiones con el objeto de implementar funcionalidades con inteligencia artificial. Del mismo modo fueron realizadas las pruebas de aceptación a la propuesta de solución con el fin de asegurar la correcta implementación de las funcionalidades y la satisfacción por parte del cliente.

Índice

Índice de tablas

Índice de ilustraciones

Desde el surgimiento de la humanidad la lucha contra las enfermedades ha sido una característica inherente al hombre, ya que las mismas surgieron como resultado del proceso de intercambio de este con diferentes elementos ambientales. Se puede decir que la salud y la enfermedad son parte integral de la vida, del proceso biológico y de las interacciones medioambientales y sociales durante la vida de un ser humano.

En la actualidad, existen numerosas enfermedades que afectan la salud humana, animal y el medio ambiente; entre ellas un grupo muy significativo de enfermedades que se transmiten de los animales a los humanos, conocidas a nivel internacional como Zoonosis o Enfermedades Zoonóticas proveniente del griego ("*zoon*") cuyo significado es animal y ("*nósos*") cuyo significado es enfermedad (1). Estas fueron nombradas de este modo, por ser enfermedades infecciosas transmisibles desde animales vertebrados al ser humano bajo circunstancias naturales.

Los agentes infecciosos involucrados son principalmente bacterias, virus, parásitos, hongos y rickettsias. Estos agentes pueden ser transmitidos por distintos mecanismos, ya sea por contacto directo, ingestión, inhalación, por vectores intermediarios y mordeduras. Como ejemplo de estas enfermedades tenemos La Rabia, Gripe Aviar, Toxoplasmosis, Toxocariasis, H1N1, Leptospirosis, Fiebre amarilla, Encefalitis japonesa, y como elemento más peligroso, la Fiebre Aftosa. (2)

Por tal motivo la Dirección de Veterinaria Nacional en Cuba utiliza un método único de vigilancia y control de dichas enfermedades, conocida internacionalmente como Sistema de Información y Vigilancia Epizootiológica también destacada por sus iniciales SIVE. Esta organización se ocupa de mantener un procedimiento sistemático y continuo para conocer a tiempo los cambios que se operan en las poblaciones animales en un espacio y tiempo determinado, generalmente transmisibles, ya sean estas de carácter emergentes o no.

El mismo fue implantado en 1988 con la colaboración del Centro Panamericano de Fiebre Aftosa (C.P.F.A), con el objetivo de proteger el territorio nacional de las enfermedades exóticas, realizar la vigilancia epizootiológica en el territorio nacional , organizar y aplicar planes y programas de emergencia, determinar y establecer programas de lucha y control contra enfermedades animales y la zoonosis. Además se encarga de realizar el control higiénico sanitario de los alimentos de origen animal destinados al consumo humano, analizar la evolución de las enfermedades sujetas a programas prioritarios de prevención y control, reconocer cualquier incremento anormal de la morbilidad y mortalidad animal, detectar rápidamente la manifestación epizoótica de una enfermedad endémica y el ingreso de enfermedades exóticas, así como facilitar el seguimiento y evaluación de las acciones contra las enfermedades. (3)

Dicha organización se divide en diversas secciones, fragmentadas por todo el territorio nacional, obteniendo fuentes constantes de información directa y puntual con el Instituto de Nutrición e Higiene de los Alimentos, la Defensa Civil (DC), el Ministerio del Interior (MININT), el Ministerio de las Fuerzas Armadas Revolucionarias (MINFAR) y el Ministerio de Salud Pública (MINSAP), es por ello que el sistema maneja un gran volumen de

datos, que crece de manera vertiginosa acumulando información. En el caso de los Departamentos de Veterinaria municipales no existe la informatización en función de este cúmulo de información, es por esto que necesita ser impulsado por la influencia del desarrollo de las nuevas tecnologías de la información y las comunicaciones. (3)

Un ejemplo de uno de estos Departamentos lo constituye el del municipio de Caimito el cual se sitúa en la provincia Artemisa. Este establecimiento forma parte de las secciones intermediarias que suministran información al Instituto de Medicina Veterinaria, es por esto que se encarga de almacenar la información de municipios y organizaciones que se encuentran dentro de su área de trabajo y de efectuar las operaciones preventivas necesarias como parte de las responsabilidades como organización perteneciente al SIVE. Durante el transcurso de los años en el Departamento ha habido un aumento significativo de información y procedimientos, los cuales han ido surgieron como resultado de las diversas operaciones que se efectúan diariamente en esta entidad, es por ello que se realizaron diversas investigaciones para localizar cuáles eran las principales dificultades reales que presentaba el proceso actual y para saber cómo integrar el uso de las nuevas tecnologías. Estas investigaciones se realizaron con los objetivos de informatizar el proceso actual, colaborar con el servicio que brindan los especialistas y mejorar los resultados actuales de esta institución.

Los resultados de las investigaciones pueden sintetizarse en los siguientes apartes:

La información obtenida en este Departamento se genera y almacena de forma manual, lo cual convierte la búsqueda y actualización de la misma en un proceso engorroso, dada la magnitud de los datos que se obtienen y generados por los especialistas. Además, el proceso de registro y toma de decisiones se torna muy arduo y engorroso, ya que al enfrentarse a un volumen incremental de notas y regulaciones el trabajo se hace mucho más complicado.

También el Centro de trabajo necesita un gestor de alerta para los recursos de información; esto se evidencia claramente por diversos motivos: el personal responsabilizado para efectuar el control se encuentre fuera de la organización, exista alguna pérdida de algún recurso de información o atraso con la información puntual. Esta dificultad es una de las causas fundamentales de que una acción preventiva no sea efectiva, ya que no se ejecutó dentro del intervalo de tiempo necesario para detectar a tiempo un brote de enfermedad con alto riesgo epizootiológico.

Por otra parte está el acceso a la amplia gama de datos, regulaciones, enfermedades, decretos y resoluciones empleados para el correcto almacenamiento de la información conjuntamente con el proceso de catálogo e identificación de enfermedades y brotes, cuyos métodos de identificación se tornan muy extensos afectando y retardando la respuesta a una determinada manifestación epizoótica.

Además, el hecho de no poseer un marco de trabajo organizado, provoca que no se ofrezca seguridad y velocidad a la hora de efectuar el control, monitoreo y actualización de la información, conjuntamente a la pérdida de información por agentes naturales (termitas, humedad, deterioro por tiempo) o por otras causas. Lo anterior provoca que se extravíe información valiosa del Departamento de Veterinaria de Caimito y ello traiga consecuencias como incumplimiento con la misión de la institución.

Por la situación anteriormente descrita, se plantea como **problema a resolver**: ¿Cómo realizar el proceso de gestión de la información de las enfermedades zoonóticas de forma eficaz y segura en el Departamento de Veterinaria del municipio Caimito para que contribuya a su seguimiento y control?

En correspondencia con el problema de la investigación señalado se define como **objeto de estudio:** la gestión de información.

El objeto delimita como **campo de acción:** la gestión información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito.

Objetivo General: Desarrollar un sistema informático para la gestión de la información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito.

Dicho objetivo se desglosa en los siguientes **objetivos específicos:**

- Analizar el marco teórico y los antecedentes de los procesos de gestión de información relacionados con la administración y control de las enfermedades zoonóticas.
- Fundamentar la selección de las metodologías, herramientas y tecnologías a utilizar en el desarrollo del sistema informático para la gestión de información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito.
- Realizar el análisis y diseño del sistema para la gestión de Información de enfermedades zoonóticas.
- Implementar el sistema informático para la gestión de Información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito.
- Validar el sistema informático para la gestión de información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito mediante las prueba de aceptación.

Como **hipótesis** se plantea la siguiente: Si se desarrolla un sistema informático para la gestión de información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito posibilitará la gestión de la información de forma eficaz y segura para que contribuya al seguimiento y control de la misma en la institución.

Para dar cumplimiento al objetivo general y a los objetivos específicos se proponen las correspondientes **tareas de investigación:**

- Caracterización y selección de metodologías, herramientas y tecnologías a utilizar en el desarrollo del sistema informático para la gestión de información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito.
- Identificación de los requisitos funcionales y no funcionales que debe cumplir el sistema informático para la gestión de información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito.
- Implementación de las funcionalidades del sistema a partir de los requisitos funcionales y no funcionales identificados que debe cumplir el sistema informático para la gestión de información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito.
- Diseño del sistema a partir de los requisitos funcionales y no funcionales identificados que debe cumplir el sistema informático para la gestión de información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito.
- Diseño de las pruebas de aceptación para validar el correcto funcionamiento del sistema informático para la gestión de información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito.
- Aplicación de las pruebas de aceptación para identificar las no conformidades del sistema informático para la gestión de información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito.

Para dar cumplimiento a las tareas de investigación planteadas, realizando la búsqueda y procesamiento de la información se utilizan los siguientes **métodos de investigación:**
Como **métodos teóricos** se utilizan:

- **Histórico-Lógico:** El uso este método permitió realizar un estudio para entender el proceso de gestión de información de enfermedades zoonóticas y las tendencias existentes de las herramientas que se utilizan para realizar esta operación.
- **Analítico-Sintético:** La práctica de este método permitió obtener una cierta cantidad de información necesaria y suficiente, y al mismo tiempo posibilitó su síntesis para la realización del sistema alcanzando conclusiones importantes acerca del objetivo que pretende cumplir la investigación.
- **Inductivo-Deductivo:** La utilización de este método permitió derivar de casos particulares teorías generales que estuviesen relacionadas con la temática de la investigación, o sea, la gestión de información de enfermedades zoonóticas.

Como **métodos empíricos** se utilizan:

- **Análisis documental:** Es utilizado para examinar y comparar documentos y conceptos relacionados con la gestión de la información de las enfermedades zoonóticas, que respaldan la investigación y que sean los más útiles para el desarrollo de la investigación.
- **Entrevista:** Se utilizó para adquirir información tanto primaria como secundaria que favorece al desarrollo del sistema informático para la gestión de información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito.

Para mejorar la comprensión de la investigación anteriormente descrita, se decidió definir una estructura capitular con los objetivos de lograr una correcta distribución de los contenidos y facilitar el estudio de los mismos. La investigación está desarrollada en 3 capítulos, y a continuación se muestra una breve descripción de cada uno de ellos:

Capítulo 1: Fundamentación teórica de la investigación.

En este capítulo se describen los elementos teóricos presentes en el sistema informático para la gestión de información de enfermedades zoonóticas en el Departamento de Veterinaria en el municipio Caimito. Se incluye el estudio de diferentes sistemas que se utilizan en la actualidad para gestionar la información de enfermedades, se exponen las tendencias y herramientas actuales que se utilizan para el desarrollo de aplicaciones web relacionadas con la gestión de información, además de la fundamentación de la selección de las más adecuadas para ser usadas en el desarrollo de la propuesta de solución.

Capítulo 2: Análisis y Diseño

Durante el desarrollo del Capítulo 2 se describen y especifican los diferentes artefactos que genera la metodología de desarrollo ágil SXP. Ejemplo de estos artefactos están: La plantilla concepción inicial del sistema, Historias de usuario, Lista de reserva del producto, Diagrama entidad relación, entre otros. Estos artefactos poseen gran importancia dado que son utilizados para guiar el proceso de desarrollo y a su vez sirven de apoyo para las futuras investigaciones.

Capítulo 3: Implementación y Prueba

En el Capítulo 3 se realiza la implementación del sistema mediante el uso de los lenguajes y herramientas estudiadas y seleccionadas además de efectuar las siguientes actividades: Se diseña el diagrama de despliegue , se precisan los sprints para la implementación de las funcionalidades, se definen las tareas de la ingeniería asociadas a las diferentes historias de usuario y además se establecen y se aplican las pruebas de aceptación con el objetivo de garantizar que el sistema cumple con los requisitos definidos por el cliente , entre otras.

Capítulo 1: Fundamentación teórica de la investigación.

En el presente capítulo se explican los conceptos fundamentales relacionados con el dominio de la situación problemática y los elementos teóricos esenciales que constituyen la base de la investigación científica planteada. Se comparan las diferentes metodologías y se justifica la elección de la seleccionada utilizando información actualizada. Se procede a la definición y caracterización de las herramientas que se utilizarán para la implementación y diseño del sistema a desarrollar.

1.1 Sistemas de Gestión de Información

Algunos autores conceptualizan los sistemas de gestión de información como: *un sistema integrado y automatizado para proveer la información que sostenga las funciones de operatividad, gestión y toma de decisiones en una organización*. (4)

Como lo define Moreiro González[1] en un artículo del año 1998:

"… el conjunto de políticas y normas relacionadas entre sí que se establecen para el acceso y tratamiento de los recursos de información, incluye los registros administrativos y los archivos, el soporte tecnológico de los recursos y el público a que se destina. En su evolución el sistema puede manejar la función de inteligencia corporativa y generar productos de inteligencia…" (4)

Un sistema de gestión de información permite la gestión de los recursos de información tanto internos como externos. Su finalidad es generar servicios y productos que respondan a las necesidades y sobrepasen las expectativas de los usuarios, posibilitando que el sistema trabaje de forma eficiente y económica a la vez. Aprovecha al máximo los recursos de información en función de la mejora continua y de la toma de decisiones organizacional a todos los niveles jerárquicos desde la cúspide estratégica hasta la base operativa. (4)

La implementación de un sistema de gestión brinda las siguientes ventajas: (5)

- Control efectivo de las actividades de la organización.
- Integración de nuevas tecnologías y herramientas.
- Disminuye errores, tiempo y recursos.
- Disponibilidad de mayor y mejor información para los usuarios en tiempo real.
- Elimina la barrera de la distancia trabajando con un mismo sistema en puntos distantes.

Pero también presentan desventajas como: (5)

- El tiempo que pueda tomar su implementación.
- La resistencia al cambio de los usuarios.
- Los problemas técnicos como fallas de hardware y/o software.

[1] **José Antonio Moreiro González.** Catedrático de Biblioteconomía y Documentación. Decano de la Facultad de Humanidades, Comunicación y Documentación de la Universidad Carlos III de Madrid, ha sido con anterioridad profesor en la UNED, en la Universidad Complutense de Madrid y en la Universidad de Murcia. Es autor de numerosos libros y artículos. (4)

- La inadecuada implementación de las funcionalidades para el apoyo de actividades de la organización.

Los sistemas de gestión de información permiten la actualización, registro y toma de decisiones de los recursos de información. Permite tener un marco de trabajo altamente organizado porque que ofrece seguridad y velocidad a la hora de efectuar consultas en un entorno de trabajo independientemente del paradigma, estado y dimensión de la información. El principal objetivo de un sistema de gestión de información es organizar los servicios que forman parte de una estructura organizativa y para ello establece diversos procesos y procedimientos para la gestión de la información, proporcionando que dicha organización reduzca los costos de la producción y eleve su efectividad en el cumplimiento de su objetivo fundamental. (6)

Como contenido primordial se debe destacar que las estructuras tendrán que adecuar los sistemas de gestión de información a sus recursos y necesidades, esto sucede dado que la tecnología es una parte inherente de la implantación del mismo aunque esto no signifique un modelo de un mejor sistema de gestión de información, porque en términos puede que con tecnologías de información más modestas se satisfagan de igual manera las necesidades de una organización.

Los **objetivos esenciales** de un sistema de gestión de la información consisten en:

- Apoyar los objetivos y estrategias de la organización
- Proporcionar información para el control de las actividades.
- Adaptar las necesidades de información a la evolución de la organización
- Interactuar con todas las piezas de la organización.
- Aumentar la producción y disminución de los costos.
- Dar apoyo a la toma de decisiones.

1.2 Sistemas vinculados a la gestión de la información de Enfermedades Zoonóticas

A continuación se muestran diferentes sistemas vinculados a la gestión de enfermedades con cierto grado de riesgo epidemiológico. Como parte de la investigación realizada dichos sistemas de separaron en 3 grupos: ***Sistemas de notificación***, ***Sistemas de detección*** y ***Software de control.***

1.2.1 Software de notificación

Un software de notificación de enfermedades es un programa diseñado para el registro y notificación por vía electrónica de las enfermedades y eventos sujetos a vigilancia epidemiológica, los cuales se utilizan para investigar casos sospechosos, casos probables de enfermedades y eventos de notificación obligatoria, permitiendo a su vez clasificarlo y monitorizarlo haciendo uso de diversos catálogos de enfermedades que constan en diferentes informes que llegan de organismos públicos de salud, los cuales detectan algún tipo de amenaza con cierto grado de riesgo. (7)

Software NOTISP:

El Software NOTISP (Software oficial de la notificación de daños sujetos a vigilancia epidemiológica) fue establecido para la observación y notificación de las enfermedades, padecimientos, incidentes, acontecimientos y casos sujetos a vigilancia epidemiológica. Actualmente utilizado para realizar la notificación de casos y cuestiones que tienen algún tipo de relación con las enfermedades sujetas a vigilancia epidemiológica. En su versión 3.1 la Dirección General de Epidemiología lo utiliza para procesar, analizar, difundir, monitorear y evaluar los datos recogidos a través del sistema de vigilancia epidemiológica respaldados por el Ministerio de Salud con la finalidad de contribuir a mejorar el sistema de vigilancia epidemiológica en Perú. El software de notificación NOTISP es un sistema diseñado para fortalecer los servicios locales de salud en sus capacidades de respuesta oportuna ante situaciones de emergencias, desastres y brotes epidémicos. (7)

La Dirección General de Epidemiología, a través de la Dirección Ejecutiva de Vigilancia Epidemiológica en el área de notificación viene desarrollando una versión en plataforma Windows del software de la vigilancia epidemiológica – NOTISP , el cual tiene las funciones básicas de sus versiones anteriores y han incorporado la vigilancia para neumonías en mayores de 5 años, vigilancia de febriles para dengue, notificación de brotes epidémicos (con diagnóstico o a través de síndromes), notificación de accidentes de tránsito, coberturas, envío automático de la notificación, así como variables y procesos que facilitan el análisis y tareas en la notificación de casos. (7)

El software NOTISP es una herramienta que permite realizar diversas notificaciones por vía electrónica de diversos casos que aparecen dentro de la sociedad que poseen cierto grado de relevancia dentro de un conjunto de enfermedades epidemiológicas, lo cual evita una propagación archivando todos sus datos para un posterior análisis. Este software tiene muchas ventajas dado que permite a los especialistas tener una comprensión específica de las enfermedades, permite registrar los datos de diversos padecimientos además de vincular notificaciones a los especialistas.

1.2.2 Software de detección

Dentro de los recursos tecnológicos que en la actualidad están disponibles para detectar enfermedades, se encuentra el software de detección. Estos medios especializados se utilizan para analizar los datos con el propósito de detectar anomalías y amenazas a la salud pública, brotes de enfermedades o cualquier penetración o conjunto de eventos que pueden comprometer el estado higiénico-sanitario en las poblaciones humanas. (8)

EpiCenter:

El sistema EpiCenter es utilizado hoy día en su versión 2.23, el cual se utiliza en los Estados Unidos para controlar y analizar los datos de salud pública con el propósito de detectar anomalías y amenazas a las poblaciones humanas, como los brotes de enfermedades y el bioterrorismo.

EpiCenter en su versión 2.23 es accesible tanto por usuarios de salud pública como por los usuarios en los centros de salud individuales. Proporciona notificación automática al personal del Departamento de salud adecuada cuando se detectan anomalías. Todos los usuarios de salud pública son capaces de ver las anomalías, gráficos y opciones de páginas. El acceso puede ser a disposición de las rápidas investigaciones, mapas, informes, etiquetas y páginas clasificadoras personalizadas, además de diversas opciones que brinda el mismo en dependencia del prototipo de padecimiento a la que se le hace el reporte.

También es capaz de procesar mensajes de actualización de condiciones de pacientes además de incluir actualizaciones del mensaje original, y elementos, tales como la temperatura del paciente, secreción, disposición o diagnóstico preliminar de datos. (8)

EpiCenter Software posee una amplia variedad de componentes y mecanismos de análisis de datos que brindan apoyo al control de enfermedades, y los mecanismos de análisis de datos que procesa están: (8)

- Departamento de Emergencia de Registro de Datos
- Centro de Control de Envenenamiento de datos de llamadas.
- Diagnóstico preliminar de datos.
- Casos enfermedad de notificación obligatoria

Además de otras secciones importantes que vinculan el Departamento de salud humana y animal, los cuales permiten el control y clasificación automática de enfermedades antes de un peligro de brote entre las que se encuentran: (8)

- Los clasificadores de Departamento de Emergencia Registro de Datos.
- Los clasificadores para el Control de Envenenamientos
- Los clasificadores de Datos de Enfermedades.

El software EpiCenter es muy abarcador posee secciones de control y seguimiento de enfermedades, procesamiento de enfermedades, mensajes de actualizaciones y emergencias, además de trabajar con información de servidores confiables como por ejemplo National Poison Data System (NPDS)[2]. Este procesa además datos de casos de enfermedades de notificación obligatoria e incluye enfermedades exóticas o sospechas de bioterrorismo que proceden de informes realizados por alguna entidad de salud pública confiable. (8)

Como consecuencia de los estudios realizados al software de notificación EpiCenter, se llega a la conclusión que sistema no cumple con los fundamentos esenciales para darle solución a los procedimientos que se desean desarrollar. El software de notificación EpiCenter fue creado con el objetivo de realizar diversos análisis producto de las informaciones que se almacenan de los diversos de Departamentos de salud, es decir que solo comprende instituciones relacionadas con enfermedades humanas, descartando así las enfermedades que afectan la salud

[2] **NPDS** cuenta con más de 60 millones de registros de casos de exposición y los datos específicos de 390.000 productos que se remontan desde 1983. NPDS pueden rastrear brotes de veneno por todo EE.UU y en muchas situaciones detecta alguna actividad mediante la aplicación de algoritmos de análisis de forma automática o mediante un metodológico paradigma de búsqueda manual. (72)

animal. Este software también presenta la dificultad del idioma, con esto se quiere decir que solamente es aplicable para instituciones de habla inglesa, lo cual dificulta en un alto grado la operatividad de la institución a la cual se desea desarrollar el sistema. En conclusión el software de detección EpiCenter no es aplicable como propuesta de solución, pero si posee ciertas funcionalidades que pueden ser adaptadas a la propuesta de solución. Dichas funcionalidades están descritas más adelante en el presente capítulo.

1.2.3 Software de control

En la actualidad un software de control de enfermedades es utilizado fundamentalmente para coordinar y recopilar información y datos significativos para el correcto funcionamiento de todos los recursos y elementos de una institución que trabaje en las áreas de los cuidados higiénicos-sanitarios con el fin de administrar, ordenar, dirigir y regular la información que posee un alto grado de valor, con el objetivo de reducir las probabilidades de errores y obtener los resultados deseados. (9)

Sistema informático para el control de un banco de sueros animales en áreas de riesgo de entrada de enfermedades exóticas en Cuba.

El sistema informático para el control de un banco de sueros de animales, el cual clasifica como un sistema de gestión de bases de datos, fue desarrollado en Microsoft Access en su versión del 2002. Para su utilización se requiere contar con una computadora con 10 Mb de espacio disponible en la unidad de disco fijo, así como el Microsoft Access XP instalado. El desarrollo del software se ejecutó siguiendo las etapas y fases de la metodología general para el desarrollo de prototipos de sistemas de información propuesta por Laudon[3]. (9)

El sistema informático para el control de un banco de sueros animales permite a los usuarios disponer de una base informatizada de datos históricos que posibilita la investigación retrospectiva, rápida y confiable que declara situaciones epidemiológicas ante una diseminación de una enfermedad con alto grado de peligrosidad. El software ofrece además los procedimientos normalizados de operación para la obtención, documentación y transporte de los sueros de diversidades animales de importancia económica para el país, para así asegurar diferentes medios para la protección de poblaciones animales no-endémicos de Cuba y al mismo tiempo la protección de la biodiversidad dentro del territorio.

El sistema, desarrollado en Microsoft Access XP, se estructuró en tres partes fundamentales:

- Estructura territorial y catálogos, para la captura de la información primaria.
- Planillas de entrada, para la actualización de los registros oficiales.

3 **Kenneth C. Laudon**. Posee el título de bachillerato en Economía de la Universidad de Stanford y un doctorado de la Universidad de Columbia. Es autor de una docena de libros que se ocupan de los sistemas de información, las organizaciones y la sociedad, incluyendo computadoras y reforma burocrática, además de 35 artículos y capítulos de libros en cuestión a los efectos organizativos, sociales y de gestión de los sistemas de información. **(71)**

- Informes de salida, para la consolidación de los datos con vistas al análisis de información y la toma de decisiones.

En el diseño de la estructura de la base de datos y de la interfaz del usuario se consideraron las siguientes bases metodológicas para el establecimiento y funcionamiento del banco de sueros:

- Criterios de selección de animales para la colección de muestras de sueros.
- Ubicación de los Puntos de Animales Centinelas (PAC).
- Caracterización de los Objetivos con Peligro Biológico y Lugares de Alta Densidad Animal
- Sistema de Calidad.

Este recurso informático se dedica fundamentalmente a la recopilación de datos significativos para la creación de sistemas de control de enfermedades poco identificadas en poblaciones animales, el mismo aunque hace uso de software privativo es altamente utilizable porque sus requerimientos funcionales le permiten ejecutarse en prácticamente cualquier ordenador. Sus funcionalidades son altamente prácticas dado que establece grandes y categóricos rasgos que diferencian los procedimientos. El mismo hace uso de parámetros significativos para la selección y recopilación de datos de las poblaciones animales para que así hubiera un plano mayor a la hora de establecer un grado de riesgo mayor en un determinado espacio animal.

El software de control de un banco de sueros animales brinda funcionalidades que son aplicables a la propuesta de solución, pero no son óptimas y poseen muchas incompatibilidades para el cumplimiento del objetivo fundamental del Departamento de Veterinaria de Caimito. También hace uso de software privativo, lo que provoca que dicho sistema no sea útil para el paradigma de desarrollo que se desea establecer en Cuba como parte del proceso de utilización de las TIC, además de que las operaciones que se manejan en dicho sistema tienen notables diferencias, lo cual indica que deben ser modificadas y adaptadas para poder obtener provecho de las mismas.

Análisis de las funcionalidades detectadas

El resultado del estudio realizado al **software de detección** reveló cierta funcionalidad que puede ser añadida e implementada para el desarrollo de la aplicación, dichas funcionalidad tienen cierto grado de complejidad, pero a su vez apoya a la gestión y monitorización de las enfermedades que se detectan a través del propio sistema evitando un brote o aparición de alguna enfermedad. Dicha funcionalidad es la Notificación Automática, las cual puede de clasificarse en dos tipos fundamentales: Notificación Inmediata y Notificación Diaria.

a. Notificación Inmediata:

Fueron creadas para la notificación inmediata de las condiciones relacionadas con la salud que podrían ser el resultado de los análisis de una de las enfermedades con un alto grado de riesgo. Cuando el análisis revela una anomalía en las tendencias de alguno de estos síntomas de prioridad, el sistema hace una notificación inmediata al usuario pre-designado por el Administrador de la organización. (8)

b. Notificación diaria:

Es para las enfermedades que aparecen en las diversas listas de notificación obligatoria pero no crean el mismo nivel de urgencia que las enfermedades de notificación obligatoria inmediata. Sin embargo, una epidemia de cualquiera de estas enfermedades es una preocupación para la salud pública. Las enfermedades en cualquiera de estas categorías de síntomas no prioritarios se informan a los usuarios designados en un estilo de resumen de notificación que se distribuye una vez al día. (8)

Por parte del **software de control** estudiado se identificaron 3 funcionalidades muy importantes que sirven de apoyo al desarrollo de la aplicación. Dado que dicho sistema posee cierto grado de analogía con la aplicación que se desea desarrollar pero aun así no brinda solución a todos los requisitos funcionales que necesita la organización a la cual se le presenta la necesidad de dicho sistema informático.

Dichas funcionalidades son:

a. **Funciones estándares y de formato:** Las funciones estándares y de formato incluyen las opciones relacionadas con los menús y permiten la entrada y salida de información al sistema en diferentes formatos de presentación; así como la búsqueda, selección, y eliminación específica o masiva de datos. (9)
b. **Planillas de entrada para la actualización de los registros oficiales:** Esta sección constituye el núcleo del sistema informático por cuanto agrupa todos los formularios para el registro de los animales en diversas categorías. Estos formularios integran una cantidad significativa de datos tanto para la clasificación, como para la consolidación de los datos para el análisis de información y la toma de decisiones por parte de los especialistas y autoridades fitosanitarias nacionales. (9)
c. **Informes de salida para el análisis de información y la toma de decisiones:** Esta sección muestra los resultados del sistema que pueden ser visualizados, impresos o enviados vía correo electrónico.

   Los informes desarrollados tienen dos objetivos. En primer lugar permiten contar con una versión digital y otra impresa de todos los modelos oficiales establecidos por el sistema de calidad implementado en relación al banco de sueros. En segundo lugar permiten la consolidación de datos relativos a las operaciones realizadas antes y después de la importación y exportación de los animales para el análisis particular de situaciones zoonóticas, así como la realización de estudios retrospectivos y prospectivos para apoyar la toma de decisiones de los organismos reguladores. (9)

1.3 Metodología de Desarrollo de Software

Una metodología de desarrollo de software puede definirse como un marco de trabajo usado durante el proceso de ciclo de vida de un sistema informático para estructurar, proyectar y controlar el ciclo de vida de un software. Tiene como objetivo fundamental aumentar la calidad de un software que se desarrolla a medida que va transitando por cada uno de los niveles de su desarrollo.

La elección de una metodología para el desarrollo de un determinado sistema está definida por varios elementos que definen la más adecuada, entre ellos se encuentran, el tipo de sistema a desarrollar, el equipo de desarrollo,

los recursos disponibles, el tiempo y las necesidades del cliente, lo cual demuestra que no existe una metodología universal a alcanzar, sino que convierte a esta selección en un proceso adaptable y configurable de acuerdo al objetivo trazado por el equipo de desarrollo. Hoy día existen dos tipos fundamentales de metodologías, las que alcanzan los métodos y modelos tradicionales que son generalmente seleccionadas para software de gran alcance (Robustas) y, las que plantean procesos ligeros para el proceso usadas habitualmente en proyectos de menor extensión (Ágiles). (10)

Metodologías robustas

Las metodologías con mayor énfasis en la planificación y control del proyecto, en especificación de requerimientos y modelado, adoptan el título de Metodologías Robustas o Tradicionales. Las metodologías robustas atribuyen indiscutible exactitud en la planificación general de todo el trabajo a ejecutar y una vez completado da inicio al ciclo de desarrollo del software. (11)

Las mismas se centralizan fundamentalmente en el control del proceso, a través una rigurosa definición de actividades, artefactos, herramientas y roles para el modelado y documentación detallada. Las metodologías tradicionales no se adaptan adecuadamente a los cambios, es decir que no son métodos apropiados cuando se trabaja en un entorno variable, donde los requisitos no se pueden pronosticar correctamente o pueden variar en el trascurso del desarrollo del software. (10)

Entre las **metodologías robustas** se encuentran:

- ***RUP*** *(Rational Unified Procces)*
- ***MSF*** *(Microsoft Solution Framework)*
- ***Win-Win*** *Spiral Model*

RUP

El proceso unificado conocido como RUP, es un modelo de software que permite el desarrollo de software a gran escala, mediante un proceso continuo de pruebas y retroalimentación, garantizando el cumplimiento de ciertos estándares de calidad. Aunque con el inconveniente de generar mayor complejidad en los controles de administración del mismo. Sin embargo, los beneficios obtenidos recompensan el esfuerzo invertido en este aspecto.

El proceso de desarrollo constituye un marco metodológico que define en términos de metas estratégicas, objetivos, actividades y artefactos (documentación) requerido en cada fase de desarrollo. Esto permite enfocar esfuerzo de los recursos humanos en términos de habilidades, competencias y capacidades a asumir roles específicos con responsabilidades bien definidas. (12)

Principales desventajas: (13)

- Las iteraciones en cada ciclo pueden tomar mucho más tiempo.
- Exceso de documentación.

- El grado de complejidad puede no resultar muy adecuado para proyectos pequeños.
- Requiere conocimientos del proceso y de UML[4].
- En proyectos pequeños, es posible que no se puedan cubrir los costos de dedicación del equipo de profesionales necesarios

MSF

Microsoft Solutions Framework (MSF) es un conjunto de procesos, principios y prácticas probadas para permitir a los desarrolladores lograr el éxito en el desarrollo del ciclo de vida del software (SDLC). MSF proporciona una guía adaptable, basado en experiencias y mejores prácticas dentro y fuera de Microsoft, para aumentar las posibilidades de realización con éxito de una solución de tecnología de información para el cliente, con el objetivo de realizar el trabajo rápido, disminuyendo el número de personas en el equipo del proyecto, evitando los riesgos, al mismo tiempo generar resultados de alta calidad. (14)

MSF provee un conjunto de principios, modelos, disciplinas, conceptos y lineamientos para la entrega de tecnología de la información utilizando soluciones Microsoft. MSF no se limita sólo al desarrollo de aplicaciones, también es aplicable a otros proyectos como por ejemplo proyectos de implementación de redes o infraestructura. MSF no obliga al desarrollador a utilizar una determinada metodología (Waterfall, Agile), pero les permite decidir qué método utilizar. (15)

Principales desventajas: (16)

- Demasiada documentación en sus fases.
- Retarda el proyecto si el análisis de riesgos es exhaustivo.
- Es creado para que se utilicen las herramientas de Microsoft fundamentalmente.
- Los precios de la licencia, capacitación y soporte de Microsoft son caros.

Win-Win

Esta es una adaptación del modelo de espiral que se hace hincapié explícitamente situados en la participación del cliente en un proceso de negociación en la génesis del desarrollo de productos. Idealmente, el desarrollador simplemente pregunta al cliente lo que se requiere y el cliente proporcionaría los detalles suficientes para proceder. Por desgracia esto rara vez sucede y negociaciones significativas entre ambas partes son necesarias para equilibrar la funcionalidad y rendimiento con los costos y de salida al mercado razones de tiempo. El modelo “win-win” deriva su nombre del objetivo de estas negociaciones, es decir, de "ganar-ganar". El cliente recibe el producto que satisface la mayoría de sus necesidades, y el desarrollador trabaja para alcanzar presupuestos y fechas de entrega. Para lograr este objetivo, el modelo define un conjunto de actividades de negociación al principio de cada paso alrededor de la espiral. (17)

[4] **Lenguaje Unificado de Modelado** (**UML**, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. (70)

Principales desventajas: (18)

- El usuario debe esperar mucho tiempo hasta ver los resultados.
- Resulta difícil convencer a grandes clientes de que el enfoque evolutivo es controlable
- Debido a su elevada complejidad no se aconseja utilizarlo en sistemas pequeños.
- Genera mucho tiempo en el desarrollo del sistema.
- Modelo costoso.
- Requiere experiencia en la identificación de los riesgos.

Metodologías ágiles

Las metodologías cuya prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software donde el diálogo cara a cara es el método más eficiente y efectivo para comunicar información y especialmente preparadas para cambios durante el desarrollo del software adoptan el título de Metodologías Agiles o Metodologías Livianas. Las metodologías ágiles son excelentes cuando se trata con requisitos no definidos o variables. Proporcionan una serie de pautas y principios junto a metodologías pragmáticas que conciben la entrega de un proyecto en un proceso menos complicado y más satisfactorio tanto para los clientes como para los equipos de desarrollo.

Los métodos ágiles se basan en la reutilización de código combinando la preferencia por las comunicaciones físicas-personales dentro del equipo de desarrollo donde el cliente es un integrante más del mismo. Habitualmente los métodos ágiles son criticados por la falta de documentación técnica del proceso y por ser un proceso menos controlado. (19)

Entre las **metodologías agiles** se encuentran:

- ***Scrum***
- ***XP*** *(Extreme Programming)*
- ***SXP***

SCRUM

Desarrollada por Ken Schwaber[5] y Jeff Sutherland[6]. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un

[5] **Ken Schwaber** es presidente de Métodos Avanzados de Desarrollo (ADM), una empresa dedicada a la mejora de la práctica del desarrollo de software. Schwaber inició la revolución de productos de gestión de procesos de la década de 1990 y también trabajó con Jeff Sutherland formular las versiones iniciales del proceso de desarrollo de Scrum. (68)

[6] **Dr. Jeff Sutherland**. Junto con Ken Schwaber, creó Scrum como un proceso formal y ayudó a redactar el Manifiesto Ágil en el año 2001. Autor del libro ¨ *The Scrum Guide*. (69)

incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. (19)

Principales desventajas: (20)

- En general, dificultad de aplicación en grandes proyectos.
- Se requiere de un "agile champion", experto en la metodología que monitorice su cumplimiento.
- Plantea un problema si el desarrollo está restringido por una fecha de entrega y un precio de entrega cerrados por contrato.
- Presupone que los requerimientos cambian, pero no de forma que el cliente acepte un diseño funcional/técnico.
- Presupone que el equipo está muy formado y motivado.
- Presupone que el cliente está muy involucrado en el desarrollo, participa de forma activa y continua, y revisa frecuentemente el avance de la funcionalidad conforme salen a la luz los sprints. Esto sin embargo no parece producirse en la mayoría de los proyectos: el cliente participa, pero no hasta el punto de dedicar tiempo y recursos para revisar pequeños avances en el desarrollo.
- Presupone que el cliente no exige ni necesita toda la documentación que manejan actualmente las empresas y que las diversas normativas internacionales requieren.

XP

Programación Extrema por su nombre en inglés eXtreme Programming (XP). Formulada por el ingeniero de software estadounidense Kent Beck[7], es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (19)

Principales desventajas: (21)

- Es recomendable emplearlo solo en proyectos a corto plazo.
- Altas comisiones en caso de fallar.
- Imposible prever todo antes de programar.

7 **Kent Beck.** Ingeniero de software estadounidense, uno de los fundadores de la metodología de desarrollo de software eXtreme Programming y uno de los 17 firmantes originales del Manifiesto Ágil en 2001. Con Ward Cunningham popularizó la metodología de tarjetas CRC, y con Erich Gamma el framework de pruebas unitarias para Java conocido como JUnit. (4)

- Las desventajas son que no se tiene la definición del costo y el tiempo de desarrollo; el sistema va creciendo después de cada entrega al cliente y nadie puede decir que el cliente no querrá una función más; se necesita de la presencia constante del usuario, y en realidad es muy difícil de lograr. (22)
- Otra desventaja es la programación en parejas, algunos desarrolladores son celosos del código que escriben y no les es grato que alguien más modifique las funciones que realizó o que su código sea desechado por no cubrir el estándar. (22)

1.3.1 SXP

La metodología de desarrollo de software SXP es una metodología híbrida resultado de la unión de las metodologías ágiles XP y SCRUM, las cuales utilizan los mismos principios del desarrollo ágil pero abarcan diferentes panoramas de desarrollo. Esta metodología es muy práctica, porque la misma es aplicable a diversos entornos de trabajo lo cual inicia una nueva forma de producción de software de calidad, utilizando las características fundamentales de sus progenitores. SXP es considerablemente productiva para entornos con equipos de desarrollos pequeños, donde existan cambios en los requerimientos o haya requerimientos aun no definidos, donde riesgo técnico el relativamente elevado y se necesita hacer una entrega rápida con una alta calidad. Beneficia el trabajo en equipo trazando un único objetivo, lo cual facilita el trabajo y el vertiginoso desarrollo de las tareas de una manera tal que el progreso en la construcción de la aplicación informática no se vea obstaculizado por los diferentes inconvenientes que surgen a medida que se transita por las fases de dicha metodología. (23)

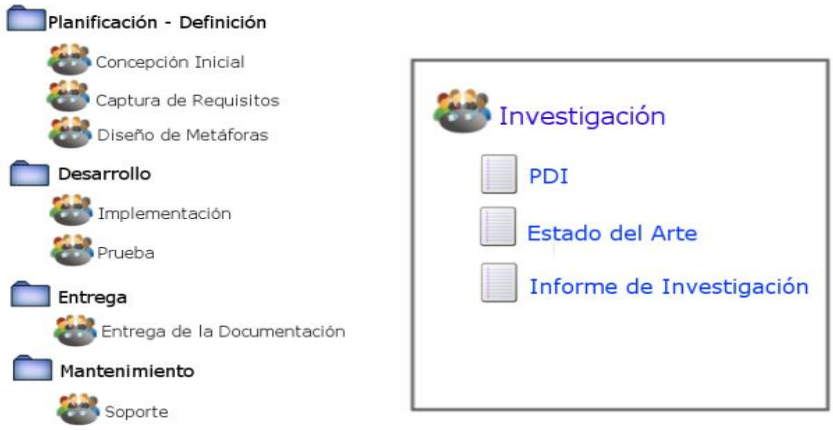

Ilustración 1: Fases de la Metodología SXP

Esta metodología cuenta con 4 fases:

1. **Planificación - Definición:** (24)

Se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto.

Actividades: (24)

- Escribir la visión (Concepción del sistema), presupuesto y reserva del producto con estimaciones iniciales.

- **Artefactos:** (24)
  - Plantilla de Concepción del Sistema.
  - Modelo de Historias de Usuario
  - Lista de Reserva del Producto (LRP).
  - Plantilla de Historia de Usuario.
  - Lista de riesgos
  - Modelo de diseño.

2. Desarrollo:

Se centra en el *¿cómo?* y su propósito es implementar un sistema definiendo como ha de diseñarse las estructuras de datos, como ha de caracterizarse las interfaces y ha de traducirse el diseño en un lenguaje de programación. (25)

- **Actividades:**
  - Reunión de planificación de la iteración.
  - Reuniones de coordinación.
  - Revisión de la iteración.
  - Realización de las pruebas de aceptación del sistema.
- **Artefactos:**
  - Tareas ingenieriles
  - Caso de prueba de aceptación.
  - Cronograma de producción.
  - Estándar de código.
  - Plan de entrega.

3. Entrega:

Es la fase que tiene como propósito la puesta en operación. La entrega contiene el cierre de la iteración, aquí el sistema está listo para ser liberado y es en la que se realiza la integración, la entrega de las pruebas del sistema y la documentación en general. (25)

- **Actividades:** (24)
  - Documentación
  - Entrega
  - Marketing
- **Artefactos:** (24)
  - Manual de desarrollo.
  - Manual de identidad.
  - Manual de usuario

4. **Mantenimiento:**

Centrada en el cambio que va incorporado a la corrección de errores, a las adaptaciones requeridas a medida que va evolucionando el entorno del software y a cambios debido a las mejoras producidas por los requisitos cambiantes del cliente. (25)

- **Actividades:**
  - Soporte
- **Artefactos:**
  - Gestión de cambio (24)

Cada uno de estos artefactos es desarrollado y supervisado por los siguientes roles: (24)

- Líder del Proyecto (Scrum Master).
- Gerente (Management).
- Consultor.
- Cliente (Customer).
- Programadores (Programmers).
- Analista (Analyst).
- Diseñadores (Designers).
- Encargado de Pruebas (Tester).
- Arquitecto (Architect)

1.4 Selección de la metodología

Como resultado del estudio y análisis de las características fundamentales de las metodologías de desarrollo de software se concluyó emplear la metodología ágil SXP, dado que se ajusta a las particularidades del desarrollo del sistema que se desea establecer, respaldando dicha elección con las siguientes **ventajas:**

- Metodología de desarrollo creada en la Universidad de Ciencias Informáticas (UCI)[8] que ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles.
- Metodología orientada para equipos de desarrollo pequeños y proyectos de poca duración.
- Propone un contacto directo entre los desarrolladores y el cliente.
- Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final.
- Metodología ágil y adaptable a los cambios de requisitos.
- Basada completamente en los valores y principios de las metodologías ágiles expuestos en el Manifiesto Ágil.

[8] **Universidad de las Ciencias Informáticas (UCI).** Es un centro de estudios universitarios radicado en La Habana, Cuba. Nacido como un proyecto de la Revolución Cubana, denominado al principio "Proyecto Futuro", con dos objetivos: informatizar el país y desarrollar la industria del software para contribuir al desarrollo económico del mismo. (67)

- Permite realizar entregas rápidas al cliente y así poder satisfacer sus necesidades.
- Asistencia al líder del proyecto a tener un mejor control del mismo.
- Aumenta el nivel de responsabilidad de los miembros del equipo.
- Previo conocimiento de las funciones de la metodología por el equipo de desarrollo.
- Recibe las ventajas ofrecidas por la metodología SCRUM para la gestión de proyectos de forma eficiente y XP para la ingeniería de software.

1.5 Lenguajes de desarrollo

Según la definición teórica, como lenguaje se entiende a un sistema de comunicación que posee una determinada estructura, contenido y uso. La programación es, en el vocabulario propio de la informática, el procedimiento de escritura del código fuente de un software. De esta manera, puede decirse que la programación le indica al programa informático que acción tiene que llevar a cabo y cuál es el modo de concretarla. (26)

Para el desarrollo de aplicaciones web se utilizan dos tipos de lenguajes:

- **Lenguaje de desarrollo del lado del cliente:**
  Los lenguajes del lado cliente (entre los cuales no sólo se encuentra el HTML sino también el Java y el JavaScript los cuales son simplemente incluidos en el código HTML) son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un pre-tratamiento. (27)
- **Lenguaje de desarrollo del lado del servidor:**
  Es una tecnología que consiste en el procesamiento de una petición de un usuario mediante la interpretación en el servidor web para generar páginas HTML dinámicamente como respuesta. Los lenguajes del lado del servidor son reconocidos, ejecutados e interpretados por el propio servidor. (27)

1.5.1 Lenguajes del lado del cliente

HTML 5

HTML (Lenguaje de Marcación de Hipertexto o Hiper Text Markup Language) es un lenguaje que se utiliza fundamentalmente en el desarrollo de páginas web, para establecer la estructura y contenido de un sitio web. (28). Una definición simple de HTML indica que “HTML es lo que se utiliza para crear todas las páginas web de Internet”. Más concretamente, HTML es el lenguaje con el que se “escriben” las páginas web. Los diseñadores de páginas web utilizan el lenguaje HTML para crear sus páginas, los programas que utilizan los diseñadores generan páginas escritas con HTML y los navegadores que utilizan los usuarios muestran las páginas web después de leer su contenido HTML. Aunque HTML es un lenguaje que utilizan los ordenadores y los programas de diseño, es muy fácil de aprender y escribir por parte de las personas. (29)
Para el desarrollo de la propuesta de solución, se utilizará HTML v5.0.

Principales ventajas que brinda HTML 5: (30)

- Tiene una sintaxis más clara.
- Posibilita la inserción de vídeos y audio de forma directa
- La programación es más dinámica, simple y sencilla.
- Tiene la capacidad de ejecutar páginas sin estar conectado.
- El navegador es capaz de dibujar y ejecutar de forma más rápida.
- Es nativo, y por tanto independiente de plugins de terceros.
- Incluye la etiqueta de dibujo canvas, que ofrece más efectos visuales.

CSS 3

Conocido como (Cascading Style Sheets u Hojas de Estilo en Cascada). Es un lenguaje que trabaja junto a HTML para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo y bordes. Es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en el documento a través de un dispositivo de lectura. (31) Para el desarrollo de la propuesta de solución, se utilizará CSS v3.0.

Ventajas que brinda el uso de CSS3: (32)

- Al no usar estilos en línea se hace más legible el código HTML al tener incluido el código CSS.
- CSS3 permite el ahorro del tiempo y trabajo al poder seguir varias técnicas (bordes redondeados, sombra en el texto, sombra en las cajas) sin necesidad de usar un editor gráfico.
- Pueden mostrarse distintas hojas de estilo según el dispositivo que sea utilizado (versión impresa, versión móvil, leída por un sintetizador de voz) o dejar que el usuario elija.
- Se obtiene un mayor control de la presentación del sitio al poder tener todo el código CSS reunido en uno, lo que facilita su modificación.
- Optimización de los tiempos de carga y de tráfico en el servidor
- Conseguimos hacer mucho más legible el código HTML al tener el código CSS aparte.
- Las novedades de CSS3 nos permiten ahorrarnos tiempo y trabajo al poder seguir varias técnicas (bordes redondeados, sombra en el texto, sombra en las cajas, etc.) sin necesidad de usar un editor gráfico.

JavaScript

JavaScript es un lenguaje de programación interpretado, similar en su sintaxis al lenguaje C y también incorpora algunas convenciones del lenguaje Java, orientado a objeto y basado en prototipos. Es una tecnología del lado del cliente por lo que permite interactuar con casi todos los navegadores de manera dinámica y eficaz. Es importante conocer que el lenguaje es muy sensible al uso indebido de mayúsculas. Permite dinamismo en las páginas HTML (11).No requiere compilación, el navegador se encarga de interpretar las secuencias JavaScript

contenidas en una página HTML y ejecutarlas. Para el desarrollo de la propuesta de solución, se utilizará JavaScript v1.5.

Ventajas de usar JavaScript: (33)

- Es un lenguaje de desarrollo de buenas prácticas.
- Fácil de aprender, rápido y potente.
- Es soportado por los navegadores más usados en la actualidad.
- Es un lenguaje sencillo que brinda muchas expectativas a los desarrolladores.
- Presenta una buena cantidad de documentación en Internet con respecto a su aplicación.
- Es totalmente gratuito, es decir, no hay que pagar ningún tipo de licencia para su uso y difusión.
- Se puede utilizar en cualquier ambiente de desarrollo.
- Ligero de carga
- Puede agregar interactividad a elementos web.
- Utiliza poca memoria.
- JavaScript es una excelente solución para poner en práctica la validación de datos de un formulario en el lado del cliente.

1.5.2 Lenguajes del lado del servidor

A continuación se muestra el ranking de los lenguajes de programación más utilizados en la actualidad.

Apr 2015	Apr 2014	Change	Programming Language	Ratings	Change
1	2	▲	Java	16.041%	-1.31%
2	1	▼	C	15.745%	-1.89%
3	4	▲	C++	6.962%	+0.83%
4	3	▼	Objective-C	5.890%	-6.99%
5	5		C#	4.947%	+0.13%
6	9	▲	JavaScript	3.297%	+1.55%
7	7		PHP	3.009%	+0.24%
8	8		Python	2.690%	+0.70%
9	-	⇈	Visual Basic	2.199%	+2.20%
10	10		Visual Basic .NET	2.126%	+0.38%

Ilustración 2: Lenguajes de programación más utilizados

Entre los lenguajes más utilizados en la actualidad se encuentra el Java, C, C++, PHP, Python, entre otros, pero se escoge PHP para el desarrollo de la propuesta de solución y dicha elección se justifica a continuación.

PHP

PHP, siglas de Hypertext Pre-Preprocesor (en español, Preprocesador de Hipertexto), es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en la interpretación del lado del servidor pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt (biblioteca para desarrollar interfaces gráficas de usuario y también para el desarrollo de programas sin interfaz gráfica como herramientas de la consola y servidores) o GTK+ (conjunto de bibliotecas para desarrollar interfaces gráficas de usuario). (34)

Para el desarrollo de la propuesta de solución, se utilizará PHP v5.4.

Ventajas que brinda PHP: (34)

- PHP es un lenguaje libre y de código abierto.
- Este lenguaje puede ser utilizado en los SO más utilizados (OSX, Linux, Windows).
- Posee una amplia documentación.
- Es un lenguaje de Programación Orientada a Objetos.
- Presenta capacidad de conexión con la mayoría de los motores de base de datos en la actualidad.
- El código fuente es transparente al navegador y al usuario.
- Posee una alta variedad de funciones que pueden ser utilizadas para mejorar el rendimiento de los programas.
- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una Base de Datos.
- El framework que se utiliza es básicamente un framework para PHP.

1.5.3 Framework de desarrollo del lado del servidor

Un framework de aplicaciones web es un tipo de framework que permite el desarrollo de sitios web dinámicos, web services (servicios web) y aplicaciones web (35). Entre los framework de desarrollo más populares para la creación de aplicaciones web que se utilizan en la actualidad están Ruby on Rails, Codelgniter, Django, Zend Framework, Symfony entre otros. Entre todos los framework mencionados, se selecciona el Symfony en su versión 2.5, y dicha elección se justifica a continuación.

Framework Symfony

Symfony es un framework completo diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Para empezar, 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 (36)

Ventajas del Framework Symfony: (36)

- Separa la lógica de negocio, de servidor y la presentación de la aplicación web.
- Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL entre otros.
- Fácil uso y aplicación
- Se puede ejecutar tanto en plataformas *nix (Unix, Linux) como en plataformas Windows.
- Automatiza las tareas más comunes.
- Independiente del sistema gestor de bases de datos.
- Posee numerosas fuentes de información en internet en Español
- Posee un paradigma de desarrollo flexible y adaptable a diversos entornos.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.
- Existe cierto grado de experiencia con el uso del mismo por parte del equipo de desarrollo.
- Symfony es un framework de PHP especialmente orientado al desarrollo web. Se basa en la arquitectura Model-View-Controller (MVC).

1.5.4 Mapeador de objetos relacional

El mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational Mapping, o sus siglas O/RM, ORM, y O/R Mapping) es una técnica de programación para 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, utilizando un motor de persistencia. (37)

Doctrine

Doctrine es un asignador objeto-relacional (ORM) para las versiones PHPv5.2.3 en adelante que proporciona persistencia transparente de objetos PHP. Se sitúa en la parte superior de una poderosa capa de abstracción de base de datos (DBAL por DataBase Abstraction Layer). (38) Para el desarrollo de la propuesta de solución, se utilizará Doctrine v2.3.

Ventajas del ORM Doctrine: (38)

- Rapidez durante el desarrollo de las aplicaciones.
- Creación automática de un modelo adecuado a partir del esquema de tablas y relaciones.
- Posee lenguaje propio para consultas a la base de datos.
- Bajo nivel de configuración que necesita para empezar un proyecto.
- Excelente diseño y entendimiento de los códigos generados.
- Integración totalmente transparente con el framework Symfony.
- Posibilidad de escribir consultas de base de datos utilizando un dialecto de SQL denominado DQL

1.5.5 Servidor web

Un servidor web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de Internet. A continuación se muestra una imagen de los servidores web más utilizados.

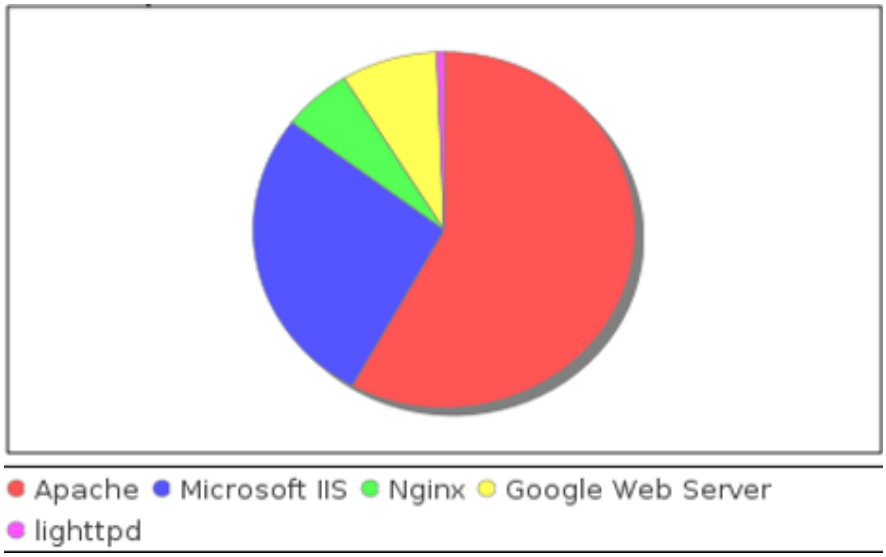

Ilustración 3: Servidores web más utilizados

Entre los servidores web más utilizados se encuentran Apache, Microsoft IIS, Nginx, entre otros, pero en la propuesta de solución se utilizará el servidor Web Apache v2.4 y su selección se justifica a continuación.

Servidor Web Apache

El servidor Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Este actualmente es unos de los servidores más usados en la creación de sitios web. (39) Apache es el servidor web hecho por excelencia, su robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. La licencia Apache es una descendiente de la licencias BSD. Esta licencia te permite hacer lo que quieras con el código fuente siempre que les reconozcas su trabajo.

Ventajas del Servidor Web Apache: (40)

- Ofrece una perfecta combinación entre estabilidad y sencillez
- Es un software de código libre y constituye una tecnología gratuita.
- Se puede desarrollar con diversos lenguajes, ya sea Java, Perl, PHP entre otros.
- Es una herramienta multiplataforma.
- Existe cierto grado de experiencia con el uso del mismo por parte del equipo de desarrollo.
- Es fácil de configurar
- Consta de una buena cantidad de información actualizada sobre de su uso.

1.5.6 Herramientas de desarrollo

Un entorno de desarrollo integrado o IDE (por sus siglas en inglés, Integrated Development Environment) es una herramienta destinada para escribir, compilar, depurar y ejecutar programas.

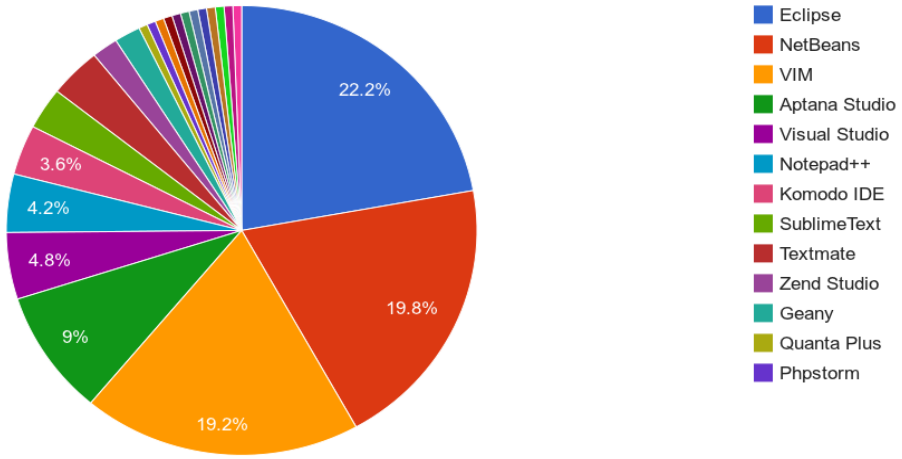

Ilustración 4: IDEs más utilizados

En la actualidad se utilizan una amplia variedad de IDE para el desarrollo, entre los cuales se encuentran: Eclipse, NetBeans, PhpStorm, Visual Studio, entre otros pero para el desarrollo de la propuesta de solución se selecciona el IDE NetBeans v8.0, y dicha selección se justifica a continuación.

Entorno de desarrollo integrado NetBeans

Este es un producto libre y gratuito sin restricciones de uso. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Estos módulos contienen clases de Java escritas para interactuar con las API[9] (interfaz de programación de aplicaciones.) de NetBeans y un archivo especial que lo identifica como módulo. NetBeans es un proyecto de código abierto que ha alcanzado mucho éxito. (41)

Ventajas del IDE NetBeans: (41)

- El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.
- La plataforma Netbeans puede ser usada para desarrollar cualquier tipo de aplicación.
- Existe cierto grado de experiencia con el uso del mismo por parte del equipo de desarrollo.
- Disponible para diferentes plataformas (Windows XP/Vista, Linux, Mac OS, Solaris).
- Soporta lenguajes como C/C++, JavaScript, Ruby, Groovy, Phython y PHP.
- Documentación y entorno de desarrollo disponible en diversos idiomas, el español entre ellos.

[9] La **interfaz de programación de aplicaciones** (**API)** del inglés (*Application Programming Interface*), es el conjunto de funciones, procedimientos o métodos que permiten a dos programas de software comunicarse entre sí. Típicamente, las API se liberan para el desarrollo de terceros como parte de un kit de desarrollo de software (SDK) o como una API abierta publicada en Internet. (41)

- Permite desarrollar aplicaciones de forma más sencilla y productiva.
- Instalación y actualización simple.

1.5.7 Sistema gestor de base de datos

Un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. En el presente existen una amplia variedad de los mismos entre los cuales se encuentran PostgreSQL, MySQL, Firebird, SQLite, Apache Derby, entre otros. Cada uno presenta diferentes características, pero para la propuesta de solución se selecciona el SGBD PostgreSQL v9.3 y dicha selección se justifica a continuación.

PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada que se ha ganado una sólida reputación de fiabilidad, integridad y corrección de los datos. Se ejecuta en los principales sistemas operativos, incluyendo Linux, UNIX, Mac OS X, Solaris y Windows. Cuenta con interfaces nativas de programación para C/C++, Java, Net, Perl, Python, Ruby y posee una documentación excepcional. Es altamente escalable, tanto en la enorme cantidad de datos que puede manejar, como en el número de usuarios concurrentes que puede acomodar. (42)

Ventajas del PostgreSQL: (42)

- Es un sistema que posee variedad de bibliografía en diferentes idiomas.
- Es altamente adaptable a diversos entornos de desarrollo.
- Existe cierto grado de experiencia con el uso del mismo por parte del equipo de desarrollo.
- Es uno de los gestores de base de datos más robustos y potentes en la actualidad.
- Permite manejar una considerable cantidad de datos sin riesgo de pérdida de información.
- Posee licencia liberal, lo cual permite que sea modificado y distribuido y utilizado para cualquier fin, ya sea privativo, lucrativo o en proyectos académicos.

1.5.8 Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering) por sus siglas en inglés (Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costos de las mismas en términos de tiempo y de dinero. (43) Como ejemplo de estas herramientas están el JDeveloper, MagicDraw, Visual Paradigm, Microsoft Visio, CASE Studio, entre otros. Las herramientas CASE se han venido ampliando y desarrollando pero para la propuesta de solución se selecciona al Visual Paradigm v8.0 como herramienta case de modelado y dicha elección se justifica a continuación.

Visual Paradigm

Visual Paradigm es una herramienta de diseño, que hace uso del Lenguaje Unificado de Modelado. Este soporta todos los diagramas UML y el diagrama de entidad-relación. Produce documentación del sistema en varios formatos como PDF y HTML. Los desarrolladores pueden diseñar documentación del sistema con una plantilla de diseño. Los analistas de sistemas pueden estimar las consecuencias de los cambios con los diagramas de análisis de impacto, tales como la matriz y el diagrama de análisis. (44)

Ventajas del Visual Paradigm: (44)

- Permite expresar de una forma gráfica un sistema para perfeccionar el entendimiento del mismo.
- Permite especificar las características del sistema que se desea modelar.
- Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño, construcción, pruebas y despliegue.
- Posibilita la captura de requisitos con sus respectivos diagramas
- Posee amplia cantidad tutoriales y demostraciones interactivas.
- Es una herramienta multiplataforma (capacidad de funcionar en varios sistemas operativos como Windows, Linux y Unix).
- Fácil de usar.

1.5.9 Lenguaje Unificado de Modelado (UML)

UML

El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales.

Tiene partes estáticas, dinámicas, de entorno y organizativas. 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 pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (45) Para el desarrollo de la propuesta de solución, se utilizará UML v2.0.

Ventajas de programar usando UML: (45)

- Modela sistemas (no sólo de software) utilizando conceptos orientados a objetos.
- Establece conceptos y artefactos ejecutables.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.

- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos para el desarrollo.

1.6 Conclusiones del capítulo

Como resultado de las investigaciones acerca de diferentes sistemas que gestionan la información de enfermedades zoonóticas, o manejan diferentes tecnologías relacionadas con la gestión de información de enfermedades altamente trasmisibles de poblaciones animales a los seres humanos, se ha podido llegar a la conclusión que los mismos no cumplen con las características necesarias para una solución completa al problema planteado, dado que el estudio desarrollado permitió conocer los diversos sistemas que de una manera u otra no se ajustan a las necesidades y características de la institución a la cual se le desea construir dicha herramienta informática. Es vital reconocer ciertas analogías con algunos de los sistemas de gestión de información de enfermedades estudiadas, pero sus características y funcionalidades hacen que solo sean aplicables al entorno para el cual fueron creados.

Producto del estudio de las herramientas y metodologías, se profundizó en los conocimientos necesarios para desarrollar los diferentes módulos en el sistema de gestión de enfermedades zoonóticas, es por ello que se decide desarrollar dicho sistema utilizando la metodología de desarrollo SXP, como marco de trabajo Symfony v2.5, PHP v5.4 como lenguaje de programación por parte del servidor y HTML v5.0 , JavaScript v1.5 y CSS v3.0 como lenguajes por parte del cliente, como entorno de desarrollo integrado IDE NetBeans v8.0, Visual Paradigm v8.0 como herramienta de modelado y UML v2.0 como lenguaje de modelado, como servidor web Apache v2.4 , PostgreSQL v9.3 como gestor de base de datos y Doctrine v2.3 como ORM . Las herramientas establecidas por el equipo de desarrollo permitirá la elaboración de un software de calidad dentro de las normas de desarrollo establecidas por Cuba y a nivel internacional.

Capítulo 2: Análisis y diseño de la propuesta de solución

En el presente capítulo se analizan y describen las características fundamentales de la aplicación que se desea desarrollar, proceso guiado por la metodología de desarrollo ágil SXP. Para facilitar el análisis de dicho sistema se describen y especifican los requisitos funcionales y no funcionales, la plantilla de concepción inicial del sistema, la descripción del modelo de datos e historias de usuario del negocio, la lista de reserva del producto además de la estructura que debe tener la base de datos la cual posee el papel fundamental para la gestión de la información. En general, se presenta la estructura a utilizar y se especifican y describen las primeras etapas que se transitan al aplicar la metodología ágil SXP para el correcto desarrollo de una aplicación.

2.1 Breve descripción del problema

El Departamento de Veterinaria del municipio Caimito genera y almacena la información de forma manual, lo cual convierte a los diversos procesos de gestión, incluidos los propios procesos de almacenamiento y conservación en un asunto complejo dado el volumen de datos y la extensión de la información. Se ha demostrado que la toma de decisiones manual realizando el correcto análisis de los volúmenes de información se torna complicada y engorrosa incluyendo diversos errores en los resultados del proceso. Al no existir un control dinámico, hace que dicha institución no posea un marco de trabajo organizado, no ofreciendo seguridad y velocidad a la hora de gestionar información puntual para la toma de decisiones recopilada de diversos sectores de la producción. Esto constituye la mayor desventaja del proceso actual y es lo fundamental que se desea solucionar con el desarrollo de la aplicación.

2.2 Solución propuesta

Se desea desarrollar un Sistema informático de gestión de información que proporcione un método seguro de almacenamiento y control de los datos. Que sea capaz de ajustarse a diferentes situaciones y que permita almacenar los criterios de los especialistas. Al mismo tiempo debe proporcionar todas las funcionalidades necesarias para gestionar recursos de alerta, los cuales son el resultado de los análisis de la información de los reportes que se gestionan y que dichas operaciones solo sean accesibles para los especialistas. Conjuntamente, al sistema se accederá solamente con usuarios registrados previamente por un único administrador, el cual será el encargado de la gestión de los mismos. Como tarea fundamental es necesario mencionar todas las funcionalidades relacionadas con la gestión de la información y la implementación de dichas funcionalidades constituyen el objetivo fundamental. Una vez desarrollada la aplicación, con sus diferentes módulos, permitirá el procesamiento de grandes volúmenes de información, el correcto almacenamiento de los datos, la actualización correcta y puntual de enfermedades zoonóticas, y la generación de recursos de alerta en caso de una actividad epizootiológica emergente.

2.3 Concepción inicial del sistema

Dentro de los flujos de trabajo se encuentra el de “Concepción Inicial” que es donde se realiza el primer encuentro con los clientes, se analizan cada uno de los procesos y las expectativas de desarrollo de los interesados, qué va a tener el futuro producto, análisis del presupuesto, herramientas a utilizar, visión y alcance del proyecto así como la organización y distribución del equipo de trabajo. (23)

2.3.1 Plantilla de concepción inicial del sistema

Tabla 1: Plantilla Concepción inicial del sistema

Plantilla Concepción Inicial del Sistema
Polo productivo: *Macro proyecto de investigación del Departamento de Veterinaria del municipio Caimito: Sistema para la gestión de información de enfermedades zoonóticas*. **Clasificación del proyecto***: Desarrollo de aplicación.* **Tipo de proyecto:** *Nacional*.
Resumen: *En el presente documento se describen los diferentes aspectos de la estructura general del sistema a implementar, además detalla los diferentes roles que intervendrán durante el ciclo de vida del desarrollo del software, así como las responsabilidades que tendrán en dicho proceso. Se justifica el tipo de proyecto al que pertenece así como la descripción del polo productivo y del mismo modo su respectiva categoría. Se explica detalladamente los métodos y herramientas que serán utilizados para el desarrollo de la aplicación, el alcance de la misma, la descripción de los actores implicados en el negocio, los factores que provocaron el desarrollo de la aplicación y la propuesta de solución.*
Surgimiento: *La necesidad de concebir el sistema nace posteriormente de haber realizado diversas investigaciones en las diversas áreas de investigación del Departamento de Veterinaria del municipio de Caimito, donde el control y monitorización de los recursos de información de las enfermedades zoonóticas se vinculan directamente con la toma de decisiones, las cuales tienen un papel fundamental. Estos vastos terrenos donde el decisor tiene que examinar y analizar grandes cantidades de información utilizando la forma tradicional (anotaciones en papel) hicieron que paulatinamente se fuera percibiendo la necesidad de un sistema que ayudara a los especialistas en dicho proceso. En muchas ocasiones este se realizaba bajo condiciones y parámetros inexactos, como resultado del proceso de escritura de forma manual y en muchas ocasiones por la amplia cantidad de información a procesar dentro de un intervalo de tiempo corto. Esto trae como consecuencia que el proceso lejos de brindar criterios eficientes y confiables, se volvía dudoso, lo que provocaba que en muchas ocasiones no cumplía con su objetivo fundamental.*

Definición:

El sistema de gestión de información de enfermedades zoonóticas, es un software para la gestión de información creado con el objetivo de dominar los principales elementos relacionados con los reportes que se realizan en el Departamento de Veterinaria del municipio Caimito. El sistema además incluye las funcionalidades necesarias para gestionar información de las enfermedades registradas y las que puedan aparecer a medida que dicha institución vaya haciendo uso del mismo.

También permite gestionar recursos de alerta, los cuales se hacen con el objetivo de reportar cualquier enfermedad de notificación de carácter urgente y obligatorio brindando la posibilidad de imprimirla en caso de ejecutar alguna medida con alguna entidad. El sistema permitirá además la gestión de todos los usuarios que hagan uso del mismo, para ello se designara un único administrador, el cual será el único con los permisos necesarios para la gestión de los mismos. En resumen, sistema para la gestión de información de enfermedades zoonóticas permite almacenar grandes volúmenes de datos y gestionar recursos de alerta que posteriormente se utilizan como apoyo y soporte a la toma de decisiones de la institución. El mismo podrá adaptarse y ser utilizado en diferentes entornos donde sea necesario controlar y monitorizar grandes cantidades de información relacionadas con el trabajo que se ejecuta actualmente en el SIVE.

Metodología a utilizar:

SXP, es un híbrido cubano de metodologías ágiles, que ofrece una estrategia tecnológica, a partir de la introducción de procedimientos ágiles que permitan actualizar los procesos de software para el mejoramiento de la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo, ayudando al líder del proyecto a tener un mejor control del mismo. Consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar el éxito del proyecto. Basada completamente en los valores y principios de las metodologías ágiles expuestos en el Manifiesto Ágil. Como método de estimación se utiliza la opinión de expertos y constan con métricas o indicadores para lograr una eficiente calidad. **(46)**

Involucrados:

Rainel Andrés Rivas Blanco: ***Líder, programador, diseñador, analista, probador, arquitecto.***

Dr. Raydel Rodríguez Zamora: ***Cliente***

Dr. Ricardo Alexis Rivas Blanco: ***Consultor.***

Misión:

Construir un entorno de trabajo que sea capaz de almacenar y preservar la información generada por los especialistas estableciendo un campo de acción estable que brinde apoyo a la toma de decisiones en la institución.

Alcance:

El desarrollo de las funcionalidades necesarias para gestionar la información de las enfermedades zoonóticas, la generación de alertas sobre enfermedades de primer orden y ofrecer un espacio correcto para la gestión, almacenamiento y protección de la información.

Roles

Tabla 2: Roles involucrados en el proceso de desarrollo.

Rol	Responsabilidad	Nombre
Líder de proyecto	Es un rol de administración que debe asegurar que el proyecto se esté llevando a cabo de acuerdo con las prácticas y que todo funcione según lo planeado. Es el encargado de coordinar y facilitar las reuniones, asegurar que se consigan los objetivos de la reunión de planificación de la iteración y determinar cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración. (23)	Rainel Andrés Rivas Blanco
Gerente	Es el responsable de tomar las decisiones finales, acerca de estándares y convenciones a seguir durante el proyecto. Participa en la selección de objetivos y requerimientos y en la selección del Usuario Interno. Tiene la responsabilidad de controlar el progreso y trabaja junto con el Jefe de Proyecto en la reducción de la Lista de Reserva del Producto.	Rainel Andrés Rivas Blanco
Cliente	El cliente participa en las tareas que involucran la lista de reserva del producto. Sus funciones son presentar la reserva del producto al equipo, enfatizando el valor y prioridades del mismo, definir la meta de la iteración, aprobar las modificaciones en la Reserva del producto y en el alcance de la iteración. (23)	Dr. Raydel Rodríguez Zamora
Programador	Es el encargado de producir el código y escribir las pruebas unitarias. El programador define las tareas de ingeniería y produce el código del sistema. Además selecciona el estándar de programación a utilizar, controlando incluso la gestión recambios. (23)	Rainel Andrés Rivas Blanco
Analista	Es el encargado de escribir las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio, todo esto lo realiza junto con el cliente. Escribe la concepción del sistema y las historias de usuario. Crea el Modelo de historia de usuario del negocio y la LRP. (23)	Rainel Andrés Rivas Blanco

Diseñador	Encargados del diseño del sistema; así como el de los prototipos de interfaces, máximos responsables de la realización del diseño de las metáforas y supervisan el proceso de construcción. (23)	Rainel Andrés Rivas Blanco
Encargado de Pruebas	Es el encargado de ayudar al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas. Escribe los casos de prueba de aceptación y ejecuta las pruebas regularmente. (23)	Rainel Andrés Rivas Blanco
Arquitecto	Se vincula directamente con el analista y el diseñador debido a que su trabajo tiene que ver con la estructura y el diseño en grande del sistema. Ayuda en el diseño de las metáforas. (23)	Rainel Andrés Rivas Blanco
Consultor	Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas, además aportan ideas y experiencias para el beneficio del sistema en desarrollo. (47)	Ricardo Alexis Rivas Blanco

2.4 Modelo de Historias de Usuario del Negocio.

La metodología de desarrollo de software SXP define una serie de actividades para guiar el proceso de desarrollo de un software, entre una de las más importantes se encuentra la definición del Modelo de Historias de Usuario del Negocio. Este modelo permite realizar una descripción detallada del negocio y por consenso entre clientes y los desarrolladores se le llama de igual forma como Modelo de Dominio. En la siguiente figura se muestra el Modelo de Dominio para la propuesta de solución.

Ilustración 5: Modelo de Dominio

Clase Información: Representa la información mensual, trimestral y anual que se recibe en el Departamento de Veterinaria.

Clase Enfermedad: Representa la enfermedad a la cual hace referencia la información.

Clase Especialista: Representa al encargado de recibir la información y de gestionar los reportes con la misma.

Clase Reporte: Representa el reporte creado por el especialista, el cual se hace con la información que llega de diferentes organizaciones nacionales.

Clase Especialista Superior: Es el encargado de revisar los reportes y el único autorizado para realizar un registro de alerta en caso de una enfermedad con alto riesgo.

Alerta: Representa el recurso que utiliza un especialista superior para indicar que un reporte posee cierto grado de peligrosidad para la población humana y animal.

2.5 Captura de requisitos

A través de los años se ha podido constatar que los requerimientos o requisitos son la pieza fundamental en un proyecto de desarrollo de software, porque marcan el punto de partida para actividades como la planeación, básicamente en lo que se refiere a las estimaciones de tiempos y costos, así como la definición de recursos necesarios y la elaboración de cronogramas que será uno de los principales mecanismos de control con los que se contará durante la etapa de desarrollo. (48)

Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir sin alterar la funcionalidad del producto y se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Los mismos están detallados a continuación en el presente documento.

Requisitos no Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener; especifican propiedades como restricciones del entorno o de la implementación, rendimiento, dependencia de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad. Los requisitos de manera general recogen las necesidades del cliente añadiendo el valor esperado por estos. Deben ser descritos de modo que sean comprendidos por los usuarios y clientes. (10). Los mismos están detallados a continuación en el presente documento.

Entre las técnicas y métodos existentes para la captura de requisitos se utilizó la **entrevista** y la **tormenta de ideas** (BrainStorming.)[10].

[10] **BrainStorming** o Tormenta de ideas es una técnica de creatividad en grupo. Los miembros del grupo aportan, durante un tiempo previamente establecido el mayor número de ideas posibles sobre un tema o problema determinado. (73)

2.5.1 Lista de reserva del producto (LRP)

La lista de reserva es un artefacto generado en la captura de requisitos, cuyo objetivo fundamental es el de describir los requisitos como funcionalidades que el sistema debe cumplir para su correcto desarrollo.

Como parte de la identificación de los mismos se detectaron 140 requisitos funcionales de prioridad muy alta, 6 requisitos funcionales de prioridad alta, 10 requisitos funcionales de prioridad media y 7 requisitos funcionales de prioridad baja. Por parte de los requisitos no funcionales se detectaron un total de 20 RNF. Los mismos están completamente detallados en el **anexo # 1** del presente documento. A continuación se muestran 25 requisitos funcionales, de prioridad muy alta.

Lista de Reserva del Producto

Tabla 3: Lista de reserva del producto

Lista de reserva del producto					
Requisitos Funcionales					
Prioridad	**Ítem**	**Descripción**	**Est.(Días)**	**Est. Por:**	**Asig. A:**
Muy Alta					
	1	Crear Cierre de Mes (CM)	1	Analista	Rainel Andrés Rivas Blanco
	2	Editar Cierre de Mes (CM)	1	Analista	Rainel Andrés Rivas Blanco
	3	Mostrar Cierre de Mes (CM)	1	Analista	Rainel Andrés Rivas Blanco
	4	Eliminar Cierre de Mes (CM)	1	Analista	Rainel Andrés Rivas Blanco
	5	Crear Macropatologías(CM)	1	Analista	Rainel Andrés Rivas Blanco
	6	Editar Macropatologías(CM)	1	Analista	Rainel Andrés Rivas Blanco
	7	Mostrar Macropatologías(CM)	1	Analista	Rainel Andrés Rivas Blanco
	8	Eliminar Macropatologías(CM)	1	Analista	Rainel Andrés Rivas Blanco
	9	Crear Acciones estatales (CM)	1	Analista	Rainel Andrés Rivas Blanco
	10	Editar Acciones estatales (CM)	1	Analista	Rainel Andrés Rivas Blanco

	11	Mostrar Acciones estatales (CM)	1	Analista	Rainel Andrés Rivas Blanco
	12	Eliminar Acciones estatales (CM)	1	Analista	Rainel Andrés Rivas Blanco
	13	Crear Certificados (CM)	1	Analista	Rainel Andrés Rivas Blanco
	14	Editar Certificados (CM)	1	Analista	Rainel Andrés Rivas Blanco
	15	Mostrar Certificados (CM)	1	Analista	Rainel Andrés Rivas Blanco
	16	Eliminar Certificados (CM)	1	Analista	Rainel Andrés Rivas Blanco
	17	Crear Sacrificios (CM)	1	Analista	Rainel Andrés Rivas Blanco
	18	Editar Sacrificios (CM)	1	Analista	Rainel Andrés Rivas Blanco
	19	Mostrar Sacrificios (CM)	1	Analista	Rainel Andrés Rivas Blanco
	20	Eliminar Sacrificios (CM)	1	Analista	Rainel Andrés Rivas Blanco
	21	Crear Traslados (CM)	1	Analista	Rainel Andrés Rivas Blanco
	22	Editar Traslados (CM)	1	Analista	Rainel Andrés Rivas Blanco
	23	Mostrar Traslados (CM)	1	Analista	Rainel Andrés Rivas Blanco
	24	Eliminar Traslados (CM)	1	Analista	Rainel Andrés Rivas Blanco
	25	Crear M621-Mortalidad (M621)	1	Analista	Rainel Andrés Rivas Blanco

2.5.2 Plantilla Historia de Usuario.

Las plantillas de historias de usuario son un instrumento para el levantamiento de requerimientos para el desarrollo de un software, que ha surgido con la aparición de los nuevos marcos de trabajo de desarrollo ágil. Las

historias de usuarios están ubicadas en el en el **anexo # 2** del presenten documento y como ejemplo se muestra la historia de usuario Registrar Cierre de Mes.

Tabla 4: Historia de Usuario: Registrar cierre de mes.

Historia de usuario

Número : HU_1	**Nombre de historia de usuario:** Crear Cierre de Mes
Modificación de historia de usuario : Ninguna	
Usuario: Ricardo Alexis Rivas Blanco	**Iteración asignada** : Sprint 1
Prioridad en el negocio: Muy alta	**Puntos estimados:** 1 días
Riesgo en el desarrollo: Alto	**Puntos Reales :** 1 días

Descripción :

Se inserta en la base de datos un reporte de cierre de mes, el mismo tiene asociados a su creación los datos de los reportes [Macropatologías] [Sacrificios] [Certificados de Muerte] [Traslados] [Acciones Estatales] los cuales completan en conjunto la creación del mismo, contiguo con las funcionalidades necesarias para su gestión.

Observaciones: Ninguna

Prototipo Interface:

2.6 Lista de riesgos

Un riesgo es aquel factor que influye negativamente en el éxito del proyecto. El riesgo en un proyecto de desarrollo de software incluye componentes técnicos y de conocimiento del mismo.

El análisis de riesgos informáticos es un proceso que comprende la identificación de activos informáticos, sus vulnerabilidades y amenazas a los que se encuentran expuestos así como su probabilidad de ocurrencia y el impacto de las mismas, a fin de determinar los controles adecuados para aceptar, disminuir, transferir o evitar la ocurrencia del riesgo. (49)

Lista de Riesgos

Tabla 5: Lista de riesgos

Lista de Riesgos						
Riesgo	**Tipo de Riesgo**	**Impacto**	**Descripción**	**Probabilidad**	**Efectos**	**Mitigación**
R_1	*Personal*	*Desarrollo*	*No se posee la suficiente experiencia con el framework de desarrollo Symfony v2.5*	*Alta*	*Serios*	*Consultar capacitaciones, bibliografías y tutoriales acerca del framework v2.5*
R_2	*Tecnológico*	*Todo el sistema*	*Las computadoras destinadas para la implementación del sistema no poseen las prestaciones necesarias.*	*Alta*	*Catastróficos*	*Priorizar el desarrollo de la aplicación en computadoras que presenten buenas prestaciones.*
R_3	*Tecnológico*	*Todo el sistema*	*Fallos eléctricos.*	*Alta*	*Tolerable*	*Salvar cada cierto periodo de tiempo, los cambios a la aplicación y al documento.*
R_4	*Tecnológico*	*Desarrollo*	*Falta de los recursos necesarios para la investigación.(Internet)*	*Alta*	*Serios*	*Evitar el mal uso de la cuota de Internet y realizar consultas en el mismo, una vez que se tiene una ruta fija de los recursos de información.*
R_5	*Tecnológico*	*Todo el sistema*	*Rotura o fallos técnicos en la pc de desarrollo.*	*Alta*	*Catastróficos*	*Mantener copias de seguridad en diversos dispositivos extraíbles.*

2.7 Arquitectura

Las técnicas metodológicas desarrolladas con el fin de facilitar la programación se engloban dentro de la llamada Arquitectura de Software o Arquitectura lógica. Se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiar el desarrollo de software dentro de un sistema informático. Así, los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo una línea común que les posibilite lograr el objetivo deseado. (50).

Una arquitectura software consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. En si es la estructura de ese sistema, que incluye componentes de software, las propiedades visibles externas de esos componentes, y las relaciones entre estos. El término también puede incluir la documentación sobre la arquitectura de software del sistema (51).

2.7.1 Estilos y patrones arquitectónicos

Estilos Arquitectónicos

Desde los inicios de la arquitectura de software, se observó que en la práctica del diseño y la implementación ciertas regularidades de configuración aparecían una y otra vez como respuesta a similares demandas. El número de esas formas no parecía ser muy grande. Muy pronto se las llamó estilos, por analogía con el uso del término en arquitectura de edificios. Un estilo describe entonces una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas. Esas piezas se encuentran repetidamente en la práctica, sintetizando la existencia de decisiones estructurales coherentes. Una vez que se han identificado los estilos, es lógico y natural pensar en re-utilizarlos en situaciones semejantes que se presenten en el futuro (52). Un estilo arquitectónico expresa un esquema de organización estructural para sistemas de software. Proveen un conjunto de modelos de elementos predefinidos y especifica sus responsabilidades, e incluye reglas y guías para organizar las relaciones entre ellos. (53)

Patrones Arquitectónicos

Los patrones arquitectónicos ofrecen soluciones a problemas de arquitectura de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. Aunque un patrón arquitectónico comunica una imagen de un sistema, no es una arquitectura como tal. Un patrón arquitectónico es más un concepto que captura elementos esenciales de una arquitectura de software. Muchas arquitecturas diferentes pueden implementar el mismo patrón y por lo tanto compartir las mismas características. (52)

2.7.2 Modelo Vista Controlador (MVC)

MVC es una propuesta de diseño de software utilizada para implementar sistemas donde se requiere el uso de interfaces de usuario. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado,

donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos. Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores, o lo que es lo mismo. (54)

Symfony2 basa su funcionamiento interno en la arquitectura Modelo - Vista – Controlador (MVC), utilizada por la mayoría de frameworks web. No obstante, según su creador Fabien Potencier[11]: "Symfony2 no es un framework MVC. Symfony2 sólo proporciona herramientas para la parte del Controlador y de la Vista. La parte del Modelo es responsabilidad tuya, aunque existen librerías para integrar fácilmente los ORM más conocidos, como Doctrine y Propel". (55)

En **Symfony2** cuando el usuario solicita ver la portada del sitio, internamente sucede lo siguiente: (55)

1. El sistema de enrutamiento determina qué **Controlador** está asociado con la página de la portada.

2. Symfony2 ejecuta el **Controlador** asociado a la portada. Un controlador no es más que una clase PHP en la que puedes ejecutar cualquier código que quieras.

3. El **Controlador** solicita al **Modelo** los datos de la oferta del día. El **Modelo** no es más que una clase PHP especializada en obtener información, normalmente de una base de datos (en este caso, el **Modelo** está formado por las entidades de Doctrine).

4. Con los datos devueltos por el **Modelo**, el **Controlador** solicita a la **Vista** que cree una página mediante una plantilla y que inserte los datos del **Modelo.**

5. El **Controlador** entrega al servidor la página creada por la **Vista**.

Modelo

Es la capa donde se trabaja con los datos, por tanto contendrá mecanismos para acceder a la información y también para actualizar su estado. Los datos los estarán habitualmente en una base de datos, por lo que en los modelos se ubicaran todas las funciones de acceso a las tablas y serán los encargados de ejecutar las funciones correspondientes de gestión de la información.

No obstante, cabe mencionar que cuando se trabaja con MVC lo habitual también es utilizar otras librerías como PDO o algún Mapeador de Objetos Relacional (ORM) como Doctrine, que permiten trabajar con abstracción de bases de datos y persistencia en objetos. Por ello, en vez de usar directamente sentencias SQL, que suelen depender del motor de base de datos con el que se esté trabajando, se utiliza un dialecto de acceso a datos basado en clases y objetos. (54)

11 **Fabien Potencier** fundador del proyecto Symfony en 2004 y la compañía de software SENSIO LABS en 1998. Creador de varios proyectos Open-Source, escritor y protagonista de numerosas conferencias internacionales. (74)

En symfony 2 El modelo no es más que una clase PHP especializada en obtener información, normalmente de una base de datos (en este caso, el modelo está formado por las entidades de Doctrine). Las consultas de Doctrine2 se realizan a través de un objeto de tipo EntityRepository obtenido mediante el método getRepository () del entity manager: (55)

Vista

Las vistas, como su nombre hace entender, contienen el código de la aplicación que va a producir la visualización de las interfaces de usuario, o sea, el código que permitirá renderizar los estados de la aplicación en HTML. En las vistas nada más están los códigos HTML y PHP que permite mostrar la salida. En la vista generalmente se trabaja con los datos, sin embargo, no se realiza un acceso directo a éstos. Las vistas requerirán los datos a los modelos y ellas se generarán la salida, tanto como la aplicación requiera. (54)

En Symfony 2 la Vista crea páginas con plantillas y datos para entregar al usuario además todo el código HTLM y Twig común se guarda en una plantilla base de la que heredan el resto de plantillas. Esta plantilla base suele denominarse layout y su uso es imprescindible en cualquier aplicación web real. (50)

Controladores

Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, entre otras. Se define como una capa que sirve de enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de nuestra aplicación. (54)

En Symfony 2 el Controlador manda y ordena. El controlador es la parte de la aplicación que contiene lo que se llama la lógica de negocio, que es una forma elegante de decir que cada controlador se encarga de una funcionalidad completa de la aplicación. Los controladores hacen uso de otros componentes para obtener la información y para generar las páginas, por lo que su código no suele ser muy largo. (55)

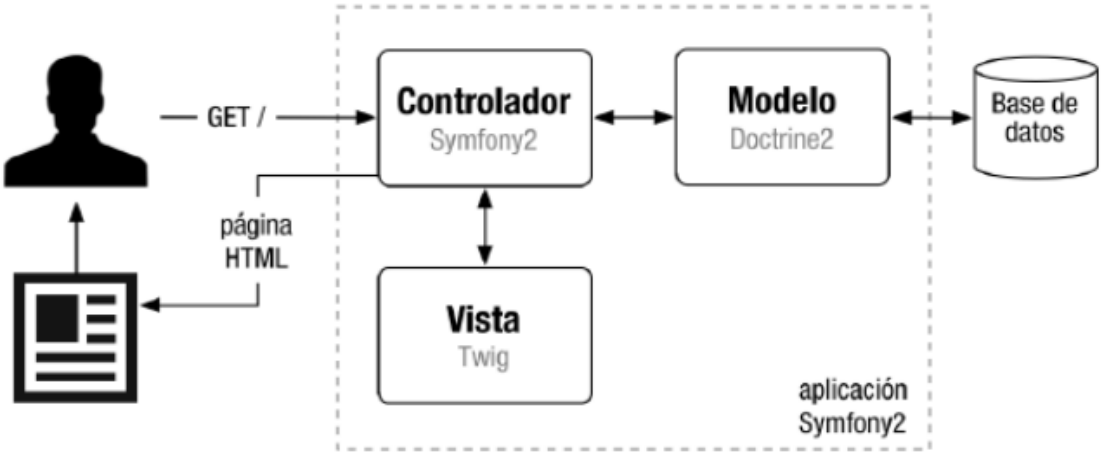

Ilustración 6: Modelo Vista Controlador Symfony2

2.7.3 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. GRASP es un acrónimo que significa (General Responsibility Assignment Software Patterns). El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos. (56)

Definición formal:

I. **Experto en información:** Responde a la pregunta ¿De qué forma se puede saber qué responsabilidad delegar a cada objeto? y su objetivo es asignar una responsabilidad al experto en información; la clase que tiene la información necesaria para llevar a cabo la responsabilidad. (56). En Symfony es utilizado en la capa de abstracción del modelo de datos. Con el uso del ORM Doctrine, Symfony genera automáticamente las clases que representan las entidades del modelo de datos. Asociado a cada una de estas clases son generadas un conjunto de funcionalidades que las relacionan de forma directa con la entidad que representan. Estas clases contienen toda la información necesaria de la tabla que representan en la base de datos. (57)

```php
namespace Lancelot\mainBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * CierreMes
 *
 * @ORM\Table(name="cierre_mes", indexes={@ORM\Index(name="IDX_DFB923337B4DD5A0", columns={"usuarioid"})})
 * @ORM\Entity
 */
class CierreMes
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer", nullable=false)
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="SEQUENCE")
     * @ORM\SequenceGenerator(sequenceName="cierre_mes_id_seq", allocationSize=1, initialValue=1)
     */
    private $id;
```

Ilustración 7: Declaración de la clase Cierre de Mes

II. **Creador:** Responde a la pregunta ¿Quién debería ser responsable de crear una nueva instancia? Y su objetivo es crear una nueva instancia por la clase que tenga la información necesaria para la creación del objeto, que use directamente las instancias creadas del objeto, que a su vez almacene varias instancias de la clase y de igual modo contenga o agregue la clase. (56) En Symfony es utilizado en los controladores, en ellos se encuentran las acciones definidas para el sistema. En la implementación de las acciones se crean instancias de las clases del modelo y de los formularios que representan a estas clases. (57)

```php
public function createAction(Request $request)
{
    $entity = new CierreMes();
    $form = $this->createCreateForm($entity);
    $form->handleRequest($request);

    if ($form->isValid()) {
        $em = $this->getDoctrine()->getManager();
        $em->persist($entity);
        $em->flush();
        $this->get('session')->getFlashBag()->set('success', 'Reporte insertado satisfactoriamente a la base de

        return $this->redirect($this->generateUrl('cierremes_show', array('id' => $entity->getId())));
    }

    return $this->render('mainBundle:CierreMes:new.html.twig', array(
        'entity' => $entity,
        'form'   => $form->createView(),
    ));
```

Ilustración 8: Funcionalidad donde se crea un Cierre de Mes

III. **Controlador:** Este patrón resuelve el problema de asignar la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase. Un controlador sirve como intermediario entre una interfaz y la acción que se desee ejecutar. Dentro del framework el patrón se evidencia en las clases que forman la capa Controlador del patrón arquitectónico MVC y quienes heredan de la misma. En Symfony todas las peticiones son procesadas por un solo controlador frontal, este es el único punto de entrada de una aplicación en un entorno determinado (57)

```php
<?php

namespace Symfony\Bundle\FrameworkBundle\Controller;

use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\HttpFoundation\StreamedResponse;
use Symfony\Component\DependencyInjection\ContainerAware;
use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
use Symfony\Component\HttpKernel\HttpKernelInterface;
use Symfony\Component\Security\Core\Exception\AccessDeniedException;
use Symfony\Component\Form\FormTypeInterface;
use Symfony\Component\Form\Form;
use Symfony\Component\Form\FormBuilder;
use Symfony\Component\Routing\Generator\UrlGeneratorInterface;
use Doctrine\Bundle\DoctrineBundle\Registry;

/**
 * Controller is a simple implementation of a Controller.
 * @author Fabien Potencier <fabien@symfony.com>
 */
class Controller extends ContainerAware
{
    /**
     * Generates a URL from the given parameters.
     *
     * @param string         $route         The name of the route
     * @param mixed          $parameters    An array of parameters
     * @param bool|string    $referenceType The type of reference (one of the constants in UrlGeneratorInterface
     * @return string The generated URL
```

Ilustración 9: Clase controladora Controller

IV. **Alta cohesión:** Responde a la pregunta ¿Cómo mantener manejable la complejidad? y su objetivo es asignar responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase. (56) La alta cohesión se evidencia en los controladores que poseen un conjunto de funcionalidades, existiendo estrecha relación entre algunas. Ejemplo de ello lo constituyen las acciones create () y update () que al crear o actualizar un objeto realizan validaciones. (57)

```
public function updateAction(Request $request, $id)
{
    $em = $this->getDoctrine()->getManager();
    $editForm = $this->createEditForm($entity);
    $editForm->handleRequest($request);

    if ($editForm->isValid()) {
        $em->flush();
        $this->get('session')->getFlashBag()->set('success', 'Se han realizado los ca

        return $this->redirect($this->generateUrl('cierremes_show', array('id' => $ic
    }

    return $this->render('mainBundle:CierreMes:edit.html.twig', array(
        'entity'      => $entity,
        'edit_form'   => $editForm->createView(),
```

Ilustración 10: Validación de la función updateAction en CierreMesController

V. **Bajo acoplamiento:** Responde a la pregunta ¿Cómo dar soporte a las bajas dependencias y al incremento de la reutilización? y su objetivo es diseñar con el objetivo de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. (56) El bajo acoplamiento se evidencia en el hecho de que los controladores heredan únicamente de la clase Controller. Además las clases que implementan la lógica del negocio y de acceso a datos no tienen asociaciones con las de la vista o el controlador. (57)

```
namespace Lancelot\mainBundle\Controller;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Lancelot\mainBundle\Entity\CierreMes;
use Lancelot\mainBundle\Form\CierreMesType;

class CierreMesController extends Controller
{
    public function indexAction()
    {
        $em = $this->getDoctrine()->getManager();

        $entities = $em->getRepository('mainBundle:CierreMes')->findAll();
         $cant= count($entities);
```

Ilustración 11: Controlador de la clase CierreMes

2.8 Diseño con metáforas

En la metodología SXP el diseño con metáforas es una práctica que tiene como objetivo proporcionar una visión única del sistema y de su arquitectura general. Es un diseño más simple de la propuesta de solución, que se utiliza para agilizar el proceso de implementación de la misma. Como parte de este proceso se genera el **modelo de diseño**, el cual está compuesto por el **diagrama de paquetes.**

Modelo de diseño:

El diseño del software desarrolla un modelo de instrumentación o implantación basado en los modelos conceptuales desarrollados durante el análisis del sistema. Implica diseñar la decisión sobre la distribución de datos y procesos (10). Es por esto que durante la etapa de análisis y diseño se realiza el modelo de diseño. El propósito fundamental de dicho modelo es crear una entrada apropiada y un punto de partida para actividades de implementación.

2.8.1 Diagrama de paquetes

En el Lenguaje Unificado de Modelado, un diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Un diagrama de paquetes muestra los elementos físicos del sistema así como las relaciones existentes entre ellos muestran las dependencias lógicas entre paquetes de software, ya se trate de componentes de código fuente, librerías, entre otros. A continuación como ejemplo se muestra el diagrama de paquetes específico para la entidad **Cierre de Mes** de la propuesta de solución.

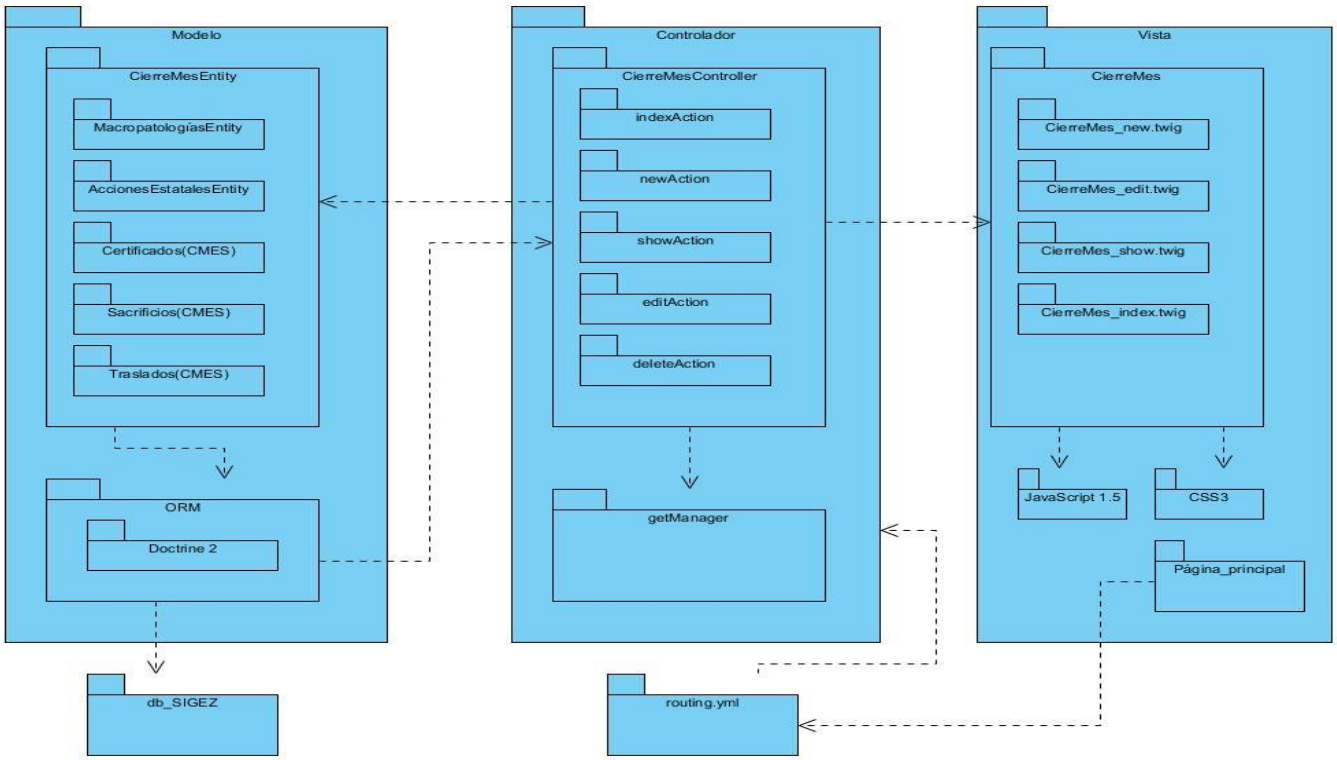

Ilustración 12: Diagrama de paquetes - Cierre de Mes

2.9 Modelo de Datos

Definición:

Un modelo de datos es un conjunto de herramientas conceptuales para describir datos, sus relaciones, su significado y sus restricciones de consistencia. (58)

Un **modelo de datos** es una combinación de tres componentes: (59)

I. Una colección de estructuras de datos, donde se especifican la cantidad, el tipo, la categoría y la forma en que se relacionan con otros.

II. Una colección de operadores o reglas de inferencia, los cuales pueden ser aplicados a cualquier instancia para consultar o derivar datos de cualquier parte de estas estructuras en cualquier combinación deseada.

III. Una colección de reglas generales de integridad, las cuales explícita o implícitamente definen un conjunto de estados consistentes –estas reglas algunas veces son expresadas como reglas de insertar, actualizar o borrar.

Tipos de modelado de datos (58)

- **Conceptual:** Muy general y abstracto, visión general del negocio/institución.
- **Lógico:** Versión completa que incluye todos los detalles acerca de los datos.
- **Físico:** Modelo que se implementara en un sistema gestor de bases de datos.

2.9.1 Diagrama Entidad Relación (DER)

El Diagrama de Entidad Relación es un modelo físico de datos basado en una percepción del mundo real que describe la distribución de los datos de un sistema de forma abstracta. Se caracteriza por utilizar una serie de elementos y reglas para representar los datos y sus relaciones con el objetivo de representar de manera gráfica la estructura lógica de una base de datos. (60)

Este modelo representa a la realidad a través de un esquema gráfico empleando la terminología de entidades, atributos y relaciones. Las entidades son los elementos fundamentales que se identifican en el problema a resolver con el diagramado, los atributos definen o identifican las características de las entidades y la relación es un vínculo que nos permite definir una dependencia entre varias entidades, es decir, nos permite exigir que varias entidades compartan ciertos atributos de forma indispensable. (60)

A continuación se muestra el modelo físico correspondiente a la base de datos, elaborado como propuesta de solución; el mismo fue modelado usando la Herramienta Case Visual Paradigm v8.0 haciendo uso del lenguaje de modelado UML v2.0. El diagrama entidad relación está completamente detallado en el **anexo # 3** del presente

documento, y como ejemplo se muestra un sub-modulo que contiene la Historia de Usuario **Cierre de Mes**, con sus respectivas entidades relacionadas.

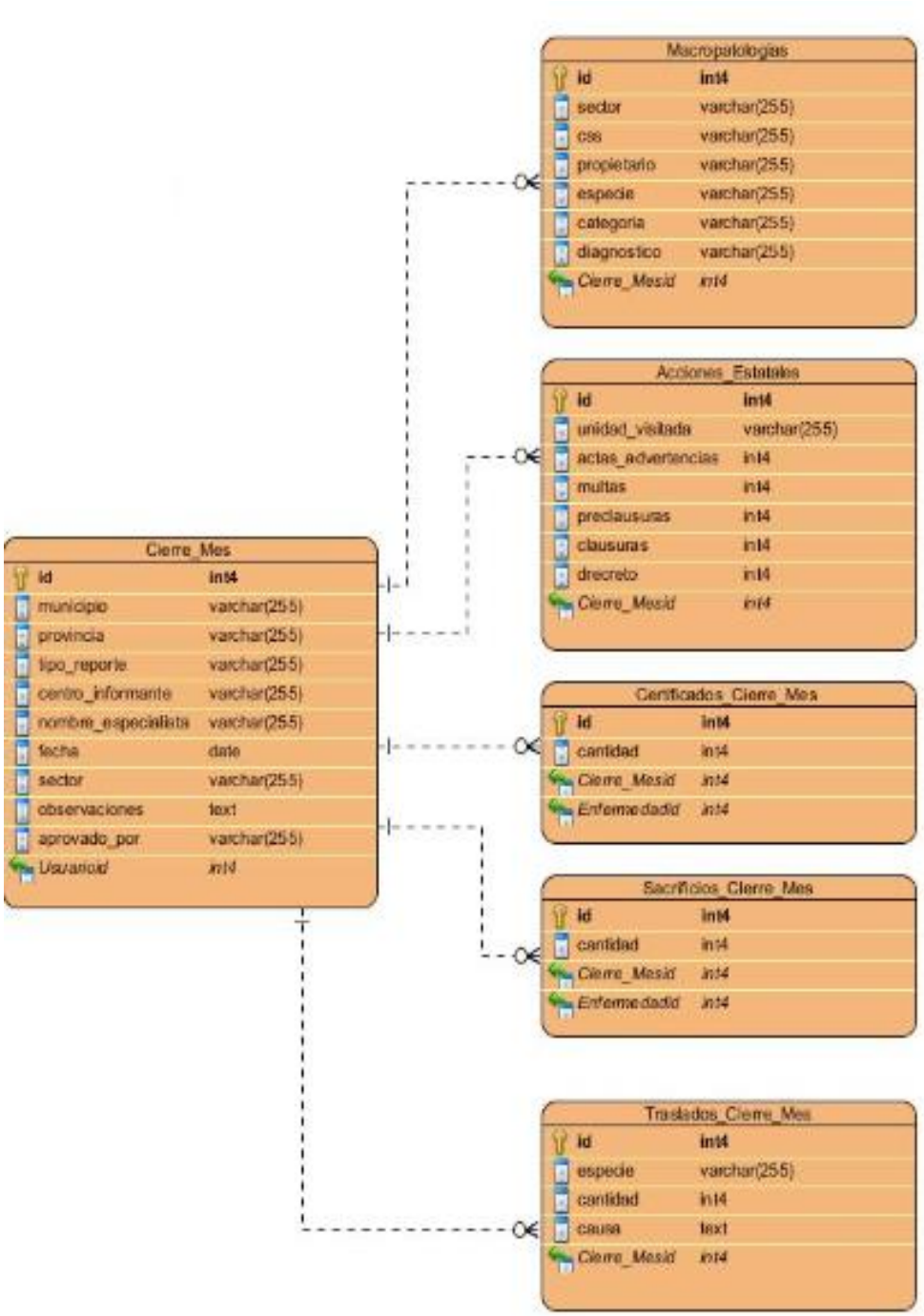

Ilustración 13: Modelo de Datos: Sub-Módulo: Cierre de Mes

2.9.2 Descripción del modelo de datos

La descripción del modelo de datos está completamente detallado en el **anexo # 4** del presenten documento y como ejemplo se muestra a continuación la descripción del modelo de datos de la tabla Cierre_Mes.

Descripción del Modelo de Datos

Tabla 6: Descripción del modelo de datos: Cierre de Mes

Nombre Tabla: Cierre_Mes		
Descripción: Tabla creada para almacenar los datos de un Cierre de Mes		
Atributos:	**Tipo:**	**Descripción:**
id	integer	Es el identificador del cierre de mes y la llave primaria de la tabla.
municipio	string	Nombre del municipio
provincia	string	Nombre de la provincia
tipo_reporte	string	Tipo de reporte de cierre de mes
centro_informante	string	Centro que reporta el cierre de mes
nombre _especialista	string	Nombre del especialista encargado
fecha	date	Fecha
sector	string	Sector al cual está dirigido el cierre de mes
observaciones	string	Observaciones del cierre de mes
aprobado_por	string	Persona encargada de aprobar el cierre
usuarioid	integer	Identificador del usuario que registra el cierre de mes

2.10 Conclusiones del capítulo

Durante el desarrollo del presente capítulo se explican y exponen los artefactos plantilla concepción inicial del sistema , historias de usuario del negocio, lista de reserva del producto (LRP), plantilla historias de usuario y el modelo de datos, generados como resultado del uso de la metodología ágil SXP. Se realizó la organización y distribución de los roles para el desarrollo de la aplicación con el objetivo distribuir las tareas y garantizar la obtención de un trabajo organizado. Además se justificaron y documentaron los requisitos funcionales y no funcionales necesarios para el desarrollo de la aplicación y se describió la arquitectura de desarrollo MVC utilizada por el framework de desarrollo Symfony 2 con el objetivo de demostrar las ventajas que brinda como modelo para la arquitectura de las aplicaciones web.

Capítulo 3: Implementación y Prueba

En el presente capítulo se describen los elementos necesarios para la etapa de implementación, partiendo de los resultados obtenidos en el capítulo anterior durante la etapa de análisis y diseño del sistema. También se muestran las tareas ingenieriles definidas por el equipo de desarrollo para darle cumplimiento a las Historias de Usuario definidas en el capítulo anterior y del mismo modo se presenta el diagrama de despliegue determinado para mostrar las relaciones físicas de los distintos nodos que componen el sistema. Además se exponen las pruebas de aceptación que se realizaron para verificar y confirmar el cumplimiento de los requisitos funcionales establecidos y se valora el resultado de las mismas.

3.1 Diagrama de despliegue

El diagrama de despliegue se utiliza para organizar los elementos de hardware que se utilizan en las implementaciones de sistemas y las relaciones entre sus componentes, los cuales se conocen como (nodos). Los nodos son objetos físicos que existen en tiempo de ejecución y que representan algún tipo de recurso computacional, también pueden ser dispositivos del sistema. El usuario accede, desde su puesto de trabajo al sistema que se encuentra instalado en el servidor de aplicaciones y dicho servidor se conecta al servidor base de datos, para realizar las consultas y obtener los resultados deseados (61). A continuación muestra el diagrama de despliegue relacionado con la propuesta de solución.

Ilustración 14: Diagrama de despliegue

3.2 Iteraciones

El desarrollo iterativo es una característica fundamental de las metodologías de desarrollo ágiles, las mismas se planifican con el objetivo de liberar software funcional cuanto antes, para lograr una versión final eficiente sin

errores y brindar la posibilidad al cliente de ir añadiendo nuevas funcionalidades, o de igual manera eliminar algunas que no sean de utilidad.

Dichas actividades permiten ofrecer una versión parcial del producto en un periodo determinado de tiempo que ha sido estimado previamente por el analista del proyecto y a su vez permite que el cliente vaya interactuando con la aplicación, dado que el resultado de cada sprint es un incremento ejecutable que se muestra al cliente para su evaluación. Se realizaron 4 sprints para la implementación y dicho procedimiento está completamente detallado en el **anexo # 5** del presente documento. A continuación se muestra un fragmento del mismo.

Tabla de Sprints para la implementación.

Tabla 7: Sprints para la implementación

Sprint	Historias de Usuario	Tiempo(Días)
Sprint #1	**HU_1 :** Crear Cierre de Mes (CM) **HU_2 :** Editar Cierre de Mes (CM) **HU_3 :** Mostrar Cierre de Mes (CM) **HU_4 :** Eliminar Cierre de Mes (CM) **HU_5 :** Crear Macropatologías(CM) **HU_6 :** Editar Macropatologías(CM) **HU_7 :** Mostrar Macropatologías(CM) **HU_8 :** Eliminar Macropatologías(CM) **HU_9 :** Crear Acciones estatales (CM) **HU_10 :** Editar Acciones estatales (CM) **HU_11:** Mostrar Acciones estatales (CM) **HU_12:** Eliminar Acciones estatales (CM) **HU_13:** Crear Certificados (CM) **HU_14:** Editar Certificados (CM) **HU_15:** Mostrar Certificados (CM) **HU_16:** Eliminar Certificados (CM) **HU_17:** Crear Sacrificios (CM) **HU_18:** Editar Sacrificios (CM) **HU_19:** Mostrar Sacrificios (CM) **HU_20:** Eliminar Sacrificios (CM) **HU_21:** Crear *Traslados (CM)* **HU_22:** Editar *Traslados (CM)* **HU_23:** Mostrar *Traslados (CM)* **HU_24:** Eliminar *Traslados (CM)* **HU_25:** Crear M621-Mortalidad (M621) **HU_26:** Editar M621-Mortalidad (M621) **HU_27:** Mostrar M621-Mortalidad (M621) **HU_28:** Eliminar M621-Mortalidad (M621) **HU_29:** Crear *Sacrificios (M621)* **HU_30:** Editar *Sacrificios (M621)* **HU_31:** Mostrar *Sacrificios (M621)* **HU_32 :** Eliminar *Sacrificios (M621)*	**30 días**

3.3 Tareas de ingeniería

Como resultado de los diversos análisis de las historias de usuarios, se construyen un grupo de tareas ingenieriles para cada una de ellas, utilizando las mismas como patrón para el posterior desarrollo de la solución propuesta. Las tareas ingenieriles están ubicadas en el en el **anexo # 6** del presente documento y a continuación se muestran las tareas relacionadas con la preparación del ambiente de desarrollo, en este caso, las relacionadas con la generación de la base de datos.

Tareas de ingeniería

Tabla 8: Tarea de Ingeniería # 1

Tarea de Ingeniería	
Número : T_1	**Número HU:**
Nombre Tarea: Generación de la Base de Datos	
Tipo Tarea: Desarrollo	**Puntos Estimados:** 2 días
Fecha Inicio: 25 enero 2015	**Fecha Fin :** 27 de enero 2015
Programador: Rainel Andrés Rivas Blanco	
Descripción: *La generación de la base de datos comprende desde el análisis de la situación planteada hasta la implementación. En este proceso se realiza un diseño manual en papel, para después formalizar su arquitectura utilizando la herramienta de modelado Visual Paradigm para posteriormente crearla usando el sistema gestor de Base de Datos PostgreSQL 9.3*	

Tabla 9: Tarea de Ingeniería # 2

Tarea de Ingeniería	
Número : T_2	**Número HU:**
Nombre Tarea: Realizar el mapeo de la Base de Datos	
Tipo Tarea: Desarrollo	**Puntos Estimados:** 1 días
Fecha Inicio: 28 de enero 2015	**Fecha Fin :** 29 de enero 2015
Programador: Rainel Andrés Rivas Blanco	
Descripción: *Realizar el mapeo de la base de datos hacia el framework Symfony v2.5.*	

3.4 Plan de entrega

El plan de entregas es un documento oficial, de carácter verídico por el cual los clientes exigen a los equipos de desarrollo, la entrega de las distintas versiones que se vayan produciendo, estos se hacen teniendo en cuenta las prioridades planteadas por el mismo, y las estimaciones realizadas por los analistas del proyecto. Para la elaboración del mismo se tienen en cuenta 2 paramentos fundamentales: tiempo de desarrollo ideal y el grado de importancia que posee una funcionalidad para el cliente (62). El **plan de entrega** del proyecto SIGEZ se muestra a continuación:

Plan de entregas

Tabla 10: Plan de Entregas

Plan de Entregas					
Entregable	**1ra Entrega(1ra Semana de Marzo)**	**2ra Entrega(3ra Semana de Marzo)**	**3ra Entrega(2da Semana de Abril)**	**4ra Entrega(3ra Semana de abril)**	**Versión Final(1ra Semana de Mayo)**
Sistema Informático para la gestión de Información de Enfermedades Zoonóticas. (SIGEZ)	Versión 0.1	Versión 0.2	Versión 0.3	Versión 0.4	Versión 1.0

3.4.1 Funcionalidades disponibles por entregas

Las funcionalidades por entregas están completamente detallada en el **anexo # 7** del documento, y a continuación se muestran las funcionalidades disponibles por entrega para las HU relacionadas con entidad Cierre de Mes.

Funcionalidades disponibles por entregas

Tabla 11: Funcionalidades disponibles por entregas

Funcionalidades por entrega						
HU	**Descripción**	1ra Entrega	2da Entrega	3ra Entrega	4ta Entrega	V. Final
1	Crear Cierre de Mes (CMES)	X	X	X	X	X
2	Editar Cierre de Mes (CMES)	X	X	X	X	X
3	Mostrar Cierre de Mes (CMES)	X	X	X	X	X
4	Eliminar Cierre de Mes (CMES)	X	X	X	X	X
5	Crear Macropatologías(CMES)	X	X	X	X	X
6	Editar Macropatologías(CMES)	X	X	X	X	X
7	Mostrar Macropatologías(CMES)	X	X	X	X	X
8	Eliminar Macropatologías(CMES)	X	X	X	X	X

9	Crear Acciones Estatales (CMES)	X	X	X	X	X
10	Editar Acciones Estatales (CMES)	X	X	X	X	X
11	Mostrar Acciones Estatales (CMES)	X	X	X	X	X
12	Eliminar Acciones Estatales (CMES)	X	X	X	X	X
13	Crear Certificados (CMES)	X	X	X	X	X
14	Editar Certificados (CMES)	X	X	X	X	X
15	Mostrar Certificados (CMES)	X	X	X	X	X
16	Eliminar Certificados (CMES)	X	X	X	X	X
17	Crear Sacrificios (CMES)	X	X	X	X	X
18	Editar Sacrificios (CMES)	X	X	X	X	X
19	Mostrar Sacrificios (CMES)	X	X	X	X	X
20	Eliminar Sacrificios (CMES)	X	X	X	X	X
21	Crear Traslados (CMES)	X	X	X	X	X
22	Editar Traslados (CMES)	X	X	X	X	X
23	Mostrar Traslados (CMES)	X	X	X	X	X
24	Eliminar Traslados (CMES)	X	X	X	X	X

3.5 Estilos y estándares de codificación.

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. El uso de estándares de codificación permite lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su mantenimiento a lo largo del tiempo. (63)

1. Los comentarios de varias líneas se escriben comenzando con los caracteres “/*” y terminando con “*/”, los comentarios de una sola línea comienzan con los caracteres “//”.
2. La notación de los nombres de los controladores debe terminar en Controller.
3. La notación de los nombres de los formularios deben terminar en Type.
4. La utilización de la notación “Camel Case” para las funciones, las cuales comienzan con letra inicial minúscula, no deben estar compuestas de lenguaje reservado.
5. Los comentarios en las plantillas twig se escriben comenzando con los caracteres “{#” , y se terminan con los caracteres “#}”
6. Los atributos de las entidades presentan carácter privado, y el acceso a los mismos se hace mediante los métodos get () y set ().
7. Para nombrar las entidades, controladores y enrutadores no se pueden utilizar palabras reservadas del lenguaje.

3.6 Pruebas de software

Las pruebas son un elemento crítico para la calidad del software. La importancia de los costos asociados a los errores, promueve la definición y aplicación de un proceso de pruebas minuciosas y bien planificadas. Las pruebas permiten validar y verificar el software, entendiendo como validación del software el proceso, externo al equipo de desarrollo, que determina si el software satisface los requisitos y verificación como el proceso interno que determina si los productos de una fase satisfacen las condiciones de dicha fase. (10)

Este sistema, como en cualquier otro en ingeniería de software puede probarse de dos formas: conociendo la función específica para la que fue diseñado y conociendo el funcionamiento del producto. El primer enfoque se centra en las llamadas pruebas de caja negra y el segundo en las pruebas de caja blanca. (10)

3.6.1 Pruebas de caja blanca

Son pruebas estructurales. Conociendo el código y siguiendo su estructura lógica, se pueden diseñar pruebas destinadas a comprobar que el código hace correctamente lo que el diseño de bajo nivel indica y otras que demuestren que no se comporta adecuadamente ante determinadas situaciones. Ejemplos típicos de ello son las pruebas unitarias. Se centran en lo que hay codificado o diseñado a bajo nivel por lo que no es necesario conocer la especificación de requisitos, que por otra parte será difícil de relacionar con partes diseñadas a muy bajo nivel. (64)

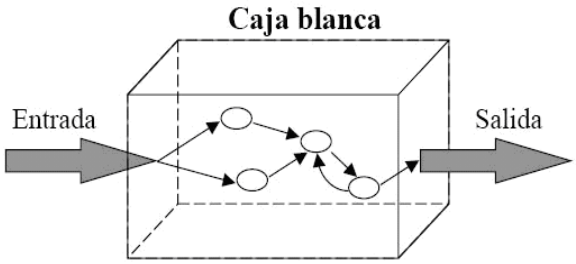

Ilustración 15: Pruebas de Caja Blanca

Ventajas: Es recomendable hacer estas pruebas si se tiene la oportunidad dado que por estas pruebas se conoce el funcionamiento interno de un sistema lo que si se aíslan los componentes de un sistema en cajas negras puede hacerse un prueba de caja blanca a cada componente lo que en consecuencia puede llegar a dar el funcionamiento de ese sistema o como es que trabaja. (64)

3.6.2 Pruebas de caja negra

Son pruebas funcionales. Se parte de los requisitos funcionales, a muy alto nivel, para diseñar pruebas que se aplican sobre el sistema sin necesidad de conocer como está construido por dentro. Las pruebas se aplican sobre el sistema empleando un determinado conjunto de datos de entrada y observando las salidas que se producen para determinar si la función se está desempeñando correctamente por el sistema bajo prueba. Las herramientas básicas son observar la funcionalidad y contrastar con la especificación. (64)

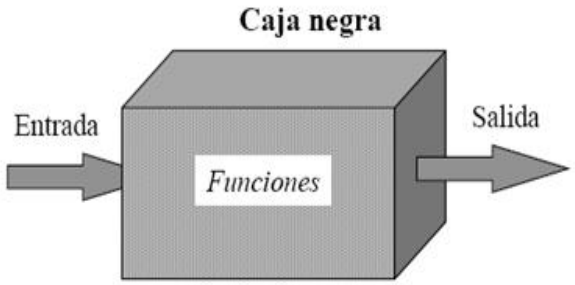

Ilustración 16: Pruebas de Caja Negra

Ventajas: Con ellas es fácil encontrar la función de un sistema y poder probar esa capacidad, además de ver hasta qué punto es capaz de desarrollarse. Las mismas pueden ser realizadas tanto por el equipo de desarrollo como por el cliente o simplemente por ambos. (64)

Niveles de Prueba

Tabla 12: Niveles de Prueba

Niveles de Prueba				
Test	**Objetivo**	**Participantes**	**Ambiente**	**Método**
Unitario	*Detectar errores en los datos, lógica y algoritmos.*	*Programadores*	*Desarrollo*	*Caja Blanca*
Integración	*Detectar errores de interfaces y relaciones entre componentes.*	*Programadores*	*Desarrollo*	*Caja Blanca, Top Down, Bottom Up.*
Funcional	*Detectar errores en la implementación de requerimientos.*	*Probadores y Analistas*	*Desarrollo*	*Caja Negra*
Sistema	*Detectar fallas en el cubrimiento de los requerimientos.*	*Probadores y Analistas*	*Desarrollo*	*Caja Negra*
Aceptación	*Detectar fallas en la implementación del sistema.*	*Probadores , Analistas y Cliente*	*Productivo*	*Caja Negra*

3.6.3 Pruebas de aceptación

Las pruebas de aceptación son aquellas realizadas por los usuarios con carácter previo al paso a producción de una nueva versión del producto. Se trata de pruebas de caja negra en un entorno de preproducción en la que se verifican si las funcionalidades pactadas para la entrega y recogidas en catálogos de requisitos, casos de uso, historias de usuario u otro hito documental, cumplen las expectativas del usuario. (65)

Durante el proceso de prueba fueron elaborados un total de 53 casos de prueba, a continuación solo se muestran los casos de prueba que se realizaron a las funcionalidades de gestión implementadas de un reporte de Cierre de Mes. Los restantes se encuentran en el **anexo # 8** del presente documento.

Casos de prueba de aceptación

Tabla 13: Caso de Prueba: HU1_CP1

Caso de Prueba de Aceptación	
Código: HU1_CP1	**Historia de usuario:** HU_1
Nombre: Mostrar listado de reportes **Cierre de Mes (CMES)**	
Condiciones de ejecución: El usuario debe estar autenticado con el rol Especialista o Especialista Superior.	
Entrada/ Pasos de ejecución: **Entrada:** **Pasos de ejecución:** *Para ver el listado de cierre de mes el usuario debe seleccionar la pestaña CMES y el sistema muestra el listado con todos los registros tipo cierre de mes que hay en el sistema, junto con los botones NUEVO y la Barra de Búsqueda.*	
Resultado esperado: *El sistema muestra el listado satisfactoriamente*	
Evaluación de la prueba: *Prueba satisfactoria*	

Tabla 14: Caso de Prueba: HU1_CP2

Caso de Prueba de Aceptación	
Código: HU1_CP2	**Historia de usuario:** HU_1
Nombre: Crear reporte **Cierre de Mes (CMES)**	
Condiciones de ejecución: El usuario debe estar autenticado con el rol Especialista o Especialista Superior.	
Entrada/ Pasos de ejecución: **Entrada:** *La entrada consiste en los datos de un cierre de mes que el usuario debe introducir para ser que sea insertado en la base de datos.* **Pasos de ejecución:** *Para insertar los datos debe ir al menú principal y seleccionar la pestaña CMES. Luego seleccionar la opción NUEVO. Una vez seleccionado, el navegador muestra un formulario para adicionar los datos del mismo junto a los botones REGISTRAR y CANCELAR.*	
Resultado esperado: *El reporte cierre de mes es creado satisfactoriamente*	
Evaluación de la prueba: *Prueba satisfactoria*	

Tabla 15: Caso de Prueba: HU1_CP3

Caso de Prueba de Aceptación	
Código: HU1_CP3	**Historia de usuario:** HU_1
Nombre: Mostrar reporte **Cierre de Mes (CMES**	
Condiciones de ejecución: El usuario debe estar autenticado con el rol Especialista o Especialista Superior y debe existir el registro de cierre de mes en la base de datos.	
Entrada/ Pasos de ejecución: **Entrada:** **Pasos de ejecución:** *Consiste en seleccionar la pestaña CMES, después que se muestra el listado el usuario debe seleccionar la opción MOSTRAR o dar clic en el identificador de cierre de mes. El sistema muestra todos los datos relacionados con el cierre de mes, junto a los botones ELIMINAR y CANCELAR.*	
Resultado esperado: *El reporte cierre de mes es mostrado satisfactoriamente.*	
Evaluación de la prueba: *Prueba satisfactoria*	

Tabla 16: Caso de Prueba: HU1_CP4

Caso de Prueba de Aceptación	
Código: HU1_CP4	**Historia de usuario:** HU_1
Nombre: Editar reporte **Cierre de Mes (CMES)**	
Condiciones de ejecución: El usuario debe estar autenticado con el rol Especialista o Especialista Superior y debe existir el registro de cierre de mes en la base de datos.	
Entrada/ Pasos de ejecución: **Entrada:** *Los datos del cierre de mes que se desea modificar.* **Pasos de ejecución:** *Para editar un registro de cierre de mes, el usuario debe seleccionar la pestaña CMES. Una vez que se muestra el listado de cierre de mes, debe seleccionar la opción EDITAR correspondiente al cierre de mes que desea editar. El sistema muestra el formulario para editar el cierre de mes, y el usuario modifica los datos. El sistema muestra los botones ACTUALIZAR y CANCELAR. Después de cambiar los datos se usa el botón ACTUALIZAR y el sistema muestra un mensaje de confirmación, y actualiza los datos en la base de datos.*	
Resultado esperado: *El reporte cierre de mes es editado satisfactoriamente.*	
Evaluación de la prueba: *Prueba satisfactoria*	

Tabla 17: Caso de Prueba: HU1_CP5

Caso de Prueba de Aceptación	
Código: HU1_CP5	**Historia de usuario:** HU_1
Nombre: Buscar reporte **Cierre de Mes (CMES)**	
Condiciones de ejecución: El usuario debe estar autenticado con el rol Especialista o Especialista Superior y el reporte existe en la base de datos.	
Entrada/ Pasos de ejecución: **Entrada:** El nombre del sector al cual está asociado el reporte que se desea encontrar **Pasos de ejecución:** *Para buscar un reporte tipo cierre de mes, el usuario va a la pestaña CMES, entonces inserta en la barra de búsqueda el nombre del sector que tiene asociado el reporte y pulsa el botón BUSCAR. El sistema muestra el cierre de mes, si existe, asociado al usuario junto con las opciones REGRESAR y BUSCAR.*	
Resultado esperado: *El reporte cierre de mes es encontrado satisfactoriamente.*	
Evaluación de la prueba: *Prueba satisfactoria*	

Tabla 18: Caso de Prueba: HU1_CP6

Caso de Prueba de Aceptación	
Código: HU1_CP6	**Historia de usuario:** HU_1
Nombre: Eliminar reporte **Cierre de Mes (CMES)**	
Condiciones de ejecución: El usuario debe estar autenticado con el rol Especialista o Especialista Superior y el reporte debe existir en la Base de Datos	
Entrada/ Pasos de ejecución: **Entrada:** **Pasos de ejecución:** *Para buscar el reporte tipo cierre de mes, el usuario va a la pestaña CMES. El usuario selecciona el cierre de mes que desea eliminar o escoge el botón MOSTRAR. Una vez que se muestran los datos del cierre de mes, el usuario selecciona el botón ELIMINAR, el sistema muestra un mensaje de confirmación, y de ser afirmativa la selección, el sistema elimina el reporte tipo cierre de mes elegido de la base de datos.*	
Resultado esperado: *El reporte cierre de mes es eliminado satisfactoriamente*	
Evaluación de la prueba: *Prueba satisfactoria*	

3.7 Análisis de los resultados

Al terminar el proceso de prueba y transitar cada una de las iteraciones de prueba planificadas para el desarrollo de la propuesta de solución, se ejecutaron las pruebas adecuadas para realizar las entregas condicionadas con el cliente. Como resultado de se obtuvieron 16 no conformidades significativas, 40 no conformidades no-significativas y un total de 28 recomendaciones. Queda resaltar que todas las no conformidades fueron solucionadas, lo cual indica que se realizó un buen trabajo durante en proceso de prueba de la aplicación y se comprobó por el cliente la calidad de la propuesta de solución, cumpliendo así con las expectativas planteadas al inicio del proceso de desarrollo del sistema. Dichos resultados se muestran a continuación.

Resultados de las pruebas de aceptación

Tabla 19: Resultados de las pruebas de aceptación

No-Conformidades Detectadas				
	Iteración # 1	*Iteración # 2*	*Iteración # 3*	*Iteración # 4*
Cantidad	**28**	**16**	**8**	**4**

Análisis de los resultados

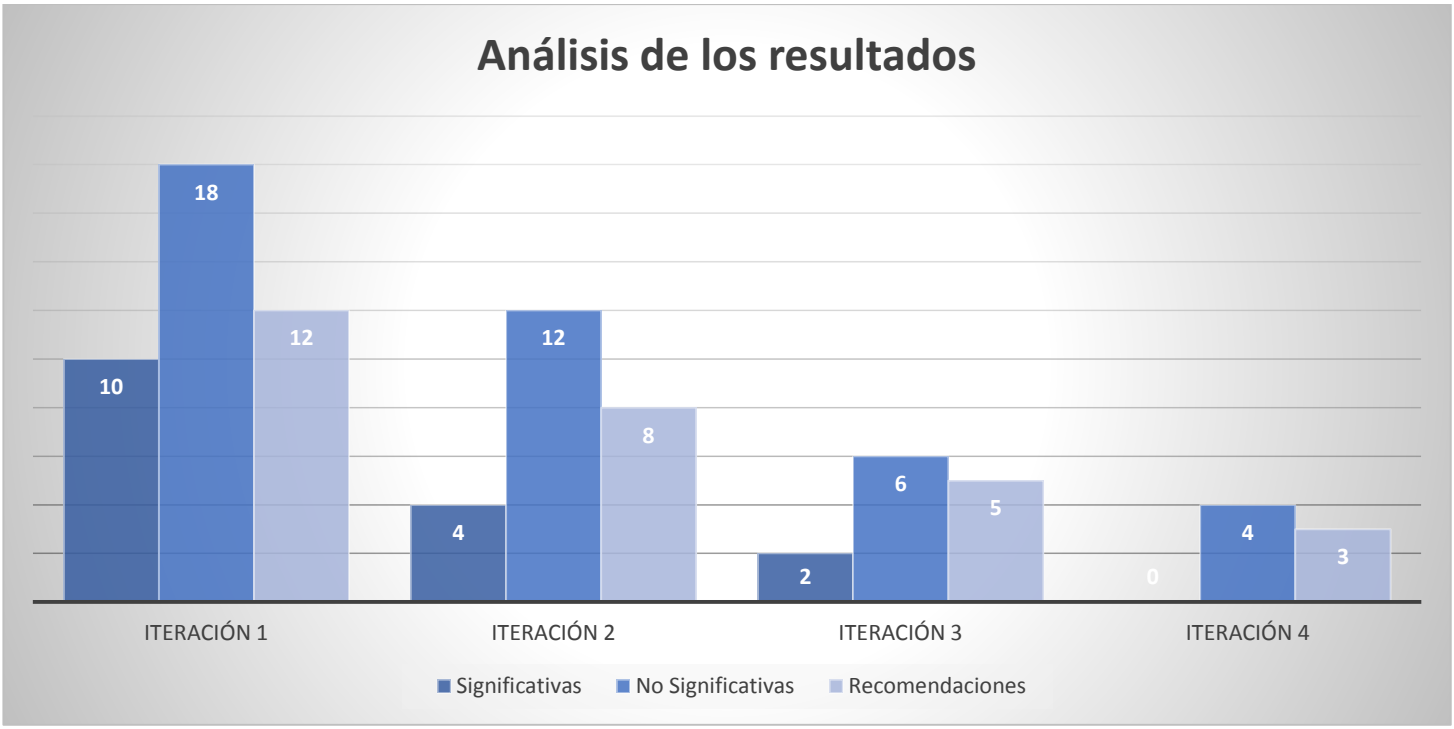

Ilustración 17: Gráfico para el análisis de los resultados

3.8 Conclusiones del capítulo

En el presente capítulo se han identificado y modelado diversos artefactos para el correcto desarrollo de la aplicación. Inicialmente se presentaron las tareas ingenieriles definidas con el objetivo de facilitar el trabajo con las historias de usuario definidas en el capítulo anterior para proporcionar un mejor análisis y realizar la estimación de su tiempo de desarrollo. También se presentó el diagrama de despliegue utilizado para representar las relaciones físicas entre los componentes hardware y software presentes en el ambiente de trabajo de la propuesta de solución. Conjuntamente se plantearon diferentes conceptos relacionados con las estrategias de pruebas utilizadas por el equipo de desarrollo y el resultado de las mismas, con el objetivo de demostrar que las pruebas de software ayudan a entregar el producto con la calidad suficiente para satisfacer las necesidades del cliente y con la certeza de que el producto cumple con las expectativas.

Conclusiones generales

A partir de los resultados alcanzados durante el proceso de investigación y desarrollo del sistema informático para la gestión de la información de enfermedades zoonóticas en el Departamento de Veterinaria del municipio Caimito se pudo arribar a las siguientes conclusiones:

- Los sistemas de información de enfermedades analizados, no obstante de sus limitaciones, contribuyeron como referencia en cuanto a conocimientos y procedimientos para la elaboración del sistema informático para la gestión de la información de enfermedades zoonóticas.
- Mediante la metodología de desarrollo de software SXP se obtuvieron los artefactos necesarios para brindar soporte a la investigación, ya sea actual o futura.
- La utilización de las herramientas y lenguajes de programación analizados y seleccionados permitió la correcta implementación de los requisitos funcionales y no funcionales.
- La utilización de las pruebas de aceptación permitió comprobar la correcta funcionalidad del software obteniendo resultados satisfactorios. Este proceso demostró que las funciones del software desarrollado se aplican de acuerdo con las especificaciones y requisitos del cliente.

Recomendaciones

- Implementar funcionalidades con inteligencia artificial que brinden apoyo en la toma de decisiones de la institución.
- Desarrollar una aplicación escritorio del sistema, con el objetivo de poder utilizarla en diferentes entornos donde no exista una red estable.
- Realizar funcionalidades integradas al sistema de Alertas, las cuales brinden notificaciones directamente al servicio de correos.

Referencias Bibliográficas

1. Definicion.de. Definición.de. *Definición.de.* [En línea] Definicion.de. [Citado el: 25 de abril de 2015.] http://definicion.de/enfermedad.

2. *FUNCION ESTATAL ESPECIFICA No. 6 SALUD ANIMAL.* Dr.Ernesto Mendoza Mainegra, Dr.Cristóbal A. Arredondo Alfonso,Dra Regnit Ferrer Rodríguez. Habana : s.n., 2014.

3. *Sistema de vigilancia epizootiologica.* Habana : s.n., 2014.

4. WikiSysop. Ecured. *Ecured.* [En línea] WikiSysop. [Citado el: 5 de Febrero de 2015.] http://www.ecured.cu/index.php/.

5. Orquidia Ferrer Rodríguez, Rayner Bestard Díaz. *Sistema de Gestión de la Información para las Ópticas .* Ciudad Habana : s.n., 2013.

6. Prendes, Lic. Lourdes M. Dueñas. *Caracterización de un Sistema de Gestión de Información Científico.* Ciudad Habana, Palacio de las Convenciones : Congreso Internacional de Información, 2004.

7. Olaechea, Jr. Daniel. Herramientas para la vigilancia. *Herramientas para la vigilancia.* [En línea] 23 de febrero de 2015. [Citado el: 26 de abril de 2015.] http://www.dge.gob.pe/portal/.

8. Epicenter. *Epicenter User Manual.* EE:UU : s.n., 2015.

9. Centro Nacional de Sanidad Agropecuaria. Revista de Salud Animal. *Revista de Salud Animal.* [En línea] Centro Nacional de Sanidad Agropecuaria. [Citado el: 05 de Febrero de 2015.] http://scielo.sld.cu/scielo.php?pid=S0253-570X2007000200005&script=sci_arttext.

10. Pressman, Roger S. *Ingeniería del Software. Un enfoque práctico. 5ta Edición.*

11. LibrosWeb.es. LibrosWeb.es. *LibrosWeb.es.* [En línea] [Citado el: 09 de febrero de 2015.] http://www.librosweb.es/.

12. Zaragoza, María de Lourdes Santiago. Desarrollando aplicaciones informáticas con el Proceso de Desarrollo Unificado. *Desarrollando aplicaciones informáticas con el Proceso de Desarrollo Unificado.* [En línea] [Citado el: 2 de mayo de 2015.] http://www.utvm.edu.mx/OrganoInformativo/orgJul07/RUP.htm.

13. Mind Meister. Mindmeister. *Mindmeister.* [En línea] [Citado el: 5 de mayo de 2015.] https://www.mindmeister.com/261074538/modelo-rup.

14. UNAD. Universidad Nacional Abierta y a Distancia. *Universidad Nacional Abierta y a Distancia.* [En línea] [Citado el: 1 de mayo de 2015.] http://datateca.unad.edu.co/contenidos/201493/CONTENIDO%20DIDACTICO%20EXE1/leccin_77_metodologa_msf_microsoft_solution_framework.html.

15. Beymar Jimenez Ruiz, Sandra Peña Perez, Yamile Valverde Perez, Aracely J. Aramayo Cuellar, Iver Salazar Zorrilla. *Metodología de desarrollo de software MSF.* Santra Cruz - Bolivia : s.n.

16. Prezi Inc. Prezi. *Prezi.* [En línea] [Citado el: 6 de mayo de 2015.] https://prezi.com/9qwgjiv_b77l/metodologia-msf/.

17. HananTek. Hanatek. *Hanatek.* [En línea] [Citado el: 3 de mayo de 2015.] http://www.hanantek.com/es/win-win.

18. Prezi Inc. Prezi. *Prezi.* [En línea] [Citado el: 6 de mayo de 2015.] https://prezi.com/nehnidpd3l31/espiral-win-win/.

19. Penadés, Patricio Letelier y Mª Carmen. *Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP).* Valencia : Universidad Politécnica de Valencia.

20. Optimus Software y Consultoria C.A. Optimus- Software y Consultoria CA. *Optimus- Software y Consultoria CA.* [En línea] [Citado el: 5 de mayo de 2015.] http://www.optimus-software.com/noticias/2011/11/28/desventajas-del-scrum/.

21. Google Sites. Metodologia xp. *Metodologia xp.* [En línea] [Citado el: 4 de mayo de 2015.] https://sites.google.com/site/xpmetodologia/marco-teorico/ventajas.

22. Universidad Veracruzana. Universidad Veracruzana. *Universidad Veracruzana.* [En línea] [Citado el: 4 de mayo de 2015.] http://www.uv.mx/universo/486/infgral/infgral_15.html.

23. *SXP, metodología de desarrollo de software.* Gladys Marsi Peñalver Romero, Sergio Jesús García de la Puente, Abel Meneses Abad. La Habana : Serie Científica de la Universidad de las Ciencias Informáticas, 2011.

24. Ailema Sisinta Ayra, Adolis Daniel Rodriguez Reyes. *Propuesta de técnicas de estimacion y metricas para la metodología ágil SXP.* Cuidad Habana : s.n., 2009.

25. Ana Yensi Fiallo De Coro, Javier González La Nuez. *Desarrollo del módulo Gestión de Tesis del Subsistema de Gestión Académica de Pregrado en la Universidad de las Ciencias Informáticas.* Ciudad de la Habana : s.n., 2010.

26. Definicion.org. Definicion. *Definicion.* [En línea] Definicion.org. [Citado el: 06 de Febrero de 2015.] http://www.definicion.org/lenguaje-de-programacion.

27. Velazco, Jose Evaristo Pacheco. Lenguaje de desarrollo del lado del cliente. *Lenguaje de desarrollo del lado del cliente.* [En línea] 12 de abril de 2015. [Citado el: 07 de Febrero de 2015.] http://www.paginasprodigy.com.mx/evaristopacheco/pweb/clienteServidor.html.

28. W3C. W3C. *W3C.* [En línea] 17 de Mayo de 2013. [Citado el: 08 de Febrero de 2015.] http://www.w3.org/html/.

29. Pérez, Javier Eguíluz. *Introducción a XHTML.*

30. Gauchat, Juan Diego. *Gran libro de HTML5, CSS3 y JavaScript.* Barcelona : s.n., 2012.

31. Pérez, Javier Eguíluz. *Introducción a CSS.*

32. HLML.net. HLML.net. *HLML.net.* [En línea] [Citado el: 20 de Enero de 2015.] http://es.html.net/tutorials/css/lesson1.php.

33. ClubEnsayos.com. ClubEnsayos.com. *ClubEnsayos.com.* [En línea] [Citado el: 03 de marzo de 2015.] http://www.clubensayos.com/Temas-Variados/JAVA-SCRIPT-VENTAJAS-Y-DESVENTAJAS/222066.html.

34. Hinostroza, Raúl Rodas. LinuxCentro.net - Características de PHP. *LinuxCentro.net - Características de PHP.* [En línea] 25 de Enero de 2015. http://www.linuxcentro.net/linux/staticpages/index.php.

35. Alcalde, Alejandro. El baul del programador. *El baul del programador.* [En línea] [Citado el: 25 de mayo de 2015.] http://elbauldelprogramador.com.

36. Fabien Potencier, François Zaninotto. *Symfony 1.4, La guía definitiva.* 2008.

37. Doctrine Team. Doctrine. *Doctrine.* [En línea] [Citado el: 12 de abril de 2015.] http://www.doctrine-project.org/projects/orm.html.

38. ServerGrove. sf2-es. *sf2-es.* [En línea] [Citado el: 02 de 2015 de Mayo.] http://sf2-es.net16.net/doctrine/orm-documentation/reference/introduction.html.

39. mistock.lcompras.biz. SAD. *SAD.* [En línea] 2011. [Citado el: 02 de febrero de 2015.] http://mistock.lcompras.biz/tallersoftware/1259 -apache-una-alternativa-viable-para-el-servidor-web.

40. Blog ibrugor. Ibrugor. *Ibrugor.* [En línea] [Citado el: 23 de mayo de 2015.] http://www.ibrugor.com/blog/apache-http-server-que-es-como-funciona-y-para-que-sirve/.

41. NetBeans. Netbeans.org. *Netbeans.org.* [En línea] [Citado el: 02 de Febrero de 2015.] http://netbeans.org.

42. Group, PostgreSQL Global Development. PostgreSQL. *PostgreSQL.* [En línea] [Citado el: 22 de Enero de 2015.] http://www.postgresql.org/.

43. Google sites. Gestion 2 STI. *Gestion 2 STI.* [En línea] [Citado el: 25 de mayo de 2015.] https://sites.google.com/site/gestion2osti/tema-3/8.

44. visualparadigm.com. Visual Paradigm for UML. *Visual Paradigm for UML.* [En línea] [Citado el: 26 de Enero de 2015.] http://www.visual-paradigm.com/aboutus/newsreleases/vpuml80.jsp.

45. James Rumbaugh, Ivar Jacobson,Grady Booch. *EL Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison Wesley - Object Technology Series.

46. *SXP, METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE.* Peñalver G. Meneses A. García, S. La Habana : Universidad de las Ciencias Informáticas.

47. Táramo, Yanet Peña. *Implementación del método TOPSIS en un sistema de ayuda en el proceso de Toma de Decisiones Multicriterio.* La Habana : s.n., 2013.

48. Chaves, Michael Arias. *La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software.* Costa Rica : Revista InterSedes, 2007.

49. Vargas, Ariel. Ingenieria de software. *Ingenieria de software.* [En línea] [Citado el: 3 de mayo de 2015.] http://arielvargasu.blogspot.com/2010/10/analisis-y-gestion-de-riesgo_18.html.

50. Guglielmetti, Marcos. mastermagazine. *mastermagazine.* [En línea] [Citado el: 31 de marzo de 2015.] http://www.mastermagazine.info/termino/3916.php.

51. ALEGSA. ALEGSA. *ALEGSA.* [En línea] [Citado el: 31 de marzo de 2015.] http://www.alegsa.com.ar/.

52. Microsoft. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* 2014.

53. Ruiz, Francisco. *Introducción a la Ingeniería de Software I.* s.l. : Univ. Cantabria – Fac. de Ciencias, 2014.

54. Álvarez, Miguel Angel. Desarrollo web. *Desarrollo web.* [En línea] [Citado el: 26 de marzo de 2015.] www.desarrolloweb.com.

55. Eguiluz, Javier. *Desarrollo Ágil en Symfony 2.* 2014.

56. Practicas de Software. Practicas de Software. *Practicas de Software.* [En línea] [Citado el: 31 de marzo de 2015.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

57. Alex García Hernández, Roanny Lamas López. *Sistema para la planificación y control del entrenamiento en deportes de combate.* Ciudad de la Habana : s.n., 2014.

58. H, Alfredo Sánchez. Interactive and Cooperative technologies Lab. *Interactive and Cooperative technologies Lab.* [En línea] [Citado el: 12 de abril de 2015.] http://ict.udlap.mx/people/carlos/is341/bases02.html.

59. Departamento de Ciencia de la Computación e IA. *Modelado de Datos.* Granada : Universidad de Granada.

60. Guillermo Storti, Gladys Ríos,Gabriel Campodónico. *Base de datos - Modelo Entidad Relacion.* s.l. : Tecnología de la Información y la Comunicación.

61. Pérez, Celso Darien Montero. *Herramienta para la gestión de licencias de software para las soluciones del Centro de Informatización de la Gestión de Entidades (CEIGE).* La Habana : s.n., 2013.

62. Tangient LLC. Procesosoftware. *Procesosoftware.* [En línea] [Citado el: 6 de junio de 2015.] http://procesosoftware.wikispaces.com/M%C3%A9todo+%C3%81gil+XP.

63. Alfonso, Damian. *Los Estándares de Codificación.*

64. S, Javier Zapata. Pruebas del software. *Pruebas del software.* [En línea] [Citado el: 03 de mayo de 2015.] https://pruebasdelsoftware.wordpress.com/.

65. Jummp. Jummp. *Jummp.* [En línea] [Citado el: 13 de mayo de 2015.] https://jummp.wordpress.com/2011/08/14/desarrollo-de-software-pruebas-de-aceptacion/.

66. *Sistema de Vigilancia Epizootiologica.* Instituto de Medicina Veterinaria. La Habana : s.n., 2014.

67. Universidad de las Ciencias Informáticas. Universidad de las Ciencias Informáticas. *Universidad de las Ciencias Informáticas.* [En línea] [Citado el: 6 de junio de 2015.] http://www.uci.cu/.

68. Scrum.org. Scrum.org. *Scrum.org.* [En línea] [Citado el: 3 de junio de 2015.] http://courses.scrum.org/about/ken-schwaber.

69. Scrum Training Institute. Scrum Fundation. *Scrum Fundation.* [En línea] [Citado el: 6 de junio de 2015.] http://scrumfoundation.com/about/jeff-sutherland.

70. IBM. Unified Modeling Language (UML). *Unified Modeling Language (UML).* [En línea] [Citado el: 7 de junio de 2015.] http://www-01.ibm.com/software/rational/uml/.

71. NYUISTERN. NYUISTERN. *NYUISTERN.* [En línea] [Citado el: 7 de junio de 2015.] http://pages.stern.nyu.edu/~klaudon/.

72. AAPCC. National Poison Data System. *National Poison Data System.* [En línea] [Citado el: 7 de junio de 2015.] http://www.aapcc.org/data-system/.

73. Mind Tools Ltd. Mind Tools. *Mind Tools.* [En línea] [Citado el: 8 de junio de 2015.] http://www.mindtools.com/brainstm.html.

74. Potencier, Fabien. Fabien Potencier. *Fabien Potencier.* [En línea] [Citado el: 8 de junio de 2015.] http://fabien.potencier.org/.

75. LibrosWeb.es. LibrosWeb.es. *LibrosWeb.es.* [En línea] [Citado el: 09 de febrero de 2015.] http://www.librosweb.es/javascript/.

76. visualparadigm.com. Visual Paradigm for UML. *Visual Paradigm for UML.* [En línea] [Citado el: 26 de Enero de 2015.] http://www.visualparadigm.com/product/vpuml/editions/modeler.jsp.