Universidad de las Ciencias Informáticas

Facultad 7

Título: Implementación del Sistema de Detección y Control de Vectores. Versión 1.0.1

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autores:

Neda María Bravo Rodríguez

Williams Rojas Cárdenas

Tutor:

Ing. Daybert Hernández Hernández

Ciudad de La Habana, Julio de 2010

"Año 52 de la Revolución"

"La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica."

Aristóteles

384 AC-322 AC. Filósofo griego.

DECLARACIÓN DE AUTORÍA

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

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

Neda María Bravo Rodríguez

_____

Firma del Autor

Williams Rojas Cárdenas

_____

Firma del Autor

Ing. Daybert Hernández Hernández

_____

Firma del Tutor

DATOS DE CONTACTO

Ing. Daybert Hernández Hernández. Graduado de Ingeniería en Ciencias Informáticas, en la UCI, en el curso 2007-2008. Instructor recién graduado en adiestramiento. Durante su trabajo como profesor ha impartido la asignatura Idioma Extranjero II.

En la vinculación con la producción pertenece al Departamento de Sistemas Especializados en Medicina del Centro Especializado en Soluciones de Informática Médica y específicamente trabaja en el desarrollo del proyecto alas Telemedicina donde se desempeña como Líder de Proyecto.

Correo electrónico: dhhernandez@uci.cu

AGRADECIMIENTOS

DEDICATORIA

RESUMEN

Con la realización de este trabajo se desarrolló la implementación de la versión 1.0.1 del Sistema de Detección y Control de Vectores. La realización del mismo surgió por la necesidad de mejorar una aplicación ya existente al migrar esta hacia las tecnologías definidas por el grupo de arquitectura MINSAP-MIC y la Facultad 7 de la Universidad de las Ciencias Informáticas.

Para realizar el software se utilizó como lenguaje de programación PHP 5 y el gestor de base de datos PostgreSQL 8.3, como framework el Symfony 1.4.3 y se hace rehúso de librerías como es la YUI Library 2.8. Para su confección se utilizó el NetBeans 6.8 como IDE de desarrollo por las grandes prestaciones que brinda.

La aplicación cuenta con funcionalidades para llevar un control sistemático de las inspecciones diarias a locales y los locales positivos encontrados. Permite gestionar los datos utilizados para realizar estas inspecciones como son: los nomencladores, la configuración de ciclos y de Áreas de salud.

Con la utilización de este sistema informático se espera mejorar el flujo de información actualizada, minimizar la introducción de errores humanos, evitar la pérdida de datos importantes y aumentar el tiempo de respuesta ante la presencia de epidemias.

Palabras claves: Implementación, Control Sanitario Internacional, Arquitectura, Base de Datos, Symfony.

TABLA DE CONTENIDOS

Tabla de Contenidos

INTRODUCCIÓN

Las Tecnologías de la Información y Comunicación (TIC) se han convertido en la base fundamental e imprescindible sobre la que se apoya todo el desarrollo de la humanidad actualmente. Estas favorecen la interconexión entre personas y entidades a nivel mundial, y eliminan barreras espaciales y temporales.

A nivel mundial la introducción de las TIC es un proceso en constante ascenso, y aunque es gradual existen grandes diferencias entre países dependiendo de su condición económica.

Cuba ha identificado la magnitud que posee la informatización de los distintos sectores a través de las nuevas Tecnologías de la Información y las Comunicaciones. Esto facilita a la sociedad acercarse más hacia el objetivo de un desarrollo sostenible, donde la Salud Pública constituye un punto sensible para el incremento de la calidad de vida de la sociedad cubana.

El Ministerio de Salud Pública (MINSAP) es el Organismo rector del Sistema Nacional de Salud. Encargado de dirigir, ejecutar y controlar la aplicación de la política del Estado y del Gobierno en cuanto a la Salud Pública, el desarrollo de las Ciencias Médicas y la Industria Médico Farmacéutica.[1]

El MINSAP brinda varios servicios a la población, los cuales necesitan ser perfeccionados en vía de elevar la calidad asistencial y la satisfacción de los pacientes. Entre las principales prioridades que tiene este organismo se encuentra la informatización del sector de la salud.

Gracias a la aplicación de las tecnologías de la información hoy existen una buena cantidad de centros asistenciales con actividades económicas y administrativas automatizadas: entre ellas INFOMED (Red de Información de las Ciencias Médicas). INFOMED, posibilita el intercambio entre personas e instituciones dedicadas a mejorar la salud de Cuba y el mundo; es uno de los tantos ejemplos que podrían contarse.

Control Sanitario Internacional es otro de los programas que cubre el sector de salud de nuestro país. El gran interés por este programa está dado por el continuo crecimiento del tráfico internacional tanto de viajeros como de aeronaves, buques y cargas con otros países. Esta situación trae consigo el riesgo de la entrada de personas enfermas o portadoras y de vectores u hospederos intermediarios de enfermedades desconocidas o erradicadas en Cuba, aumentando el índice de transmisión y propagación.

Este programa está conformado por tres subprogramas: el programa de Higiene y Epidemiología que se encarga de la vigilancia epidemiológica al viajero, el programa de Salud Ambiental que estudia los factores

del ambiente y del entorno que afectan la salud de humanos, animales y vegetales; y el programa de Vigilancia y Lucha Antivectorial que tiene la misión de contribuir a evitar la introducción y/o propagación de enfermedades introducidas por vectores e incrementar los niveles de salud y satisfacción de la población cubana.

La investigación en curso está centrada en este último subprograma el cual mantiene un alto control de datos en diferentes niveles como son áreas de salud, municipios, provincias y nación, debido al riesgo que representa para Cuba la propagación de enfermedades trasmitidas por vectores; ya sean roedores, cucarachas, moscas o mosquitos.

Prestándole mayor atención al Aedes Aegypti, el cual tiene su origen en la Región Etiópica de África, por ser el principal agente transmisor del dengue enfermedad que ha tenido un incremento registrado en los últimos años. Ancestralmente, desde esas áreas, este vector inició una dispersión efectuada por el hombre, que lo ha llevado hasta la actualidad a una constante propagación y reinfestación de diversas áreas de las Américas entre las que se encuentra Cuba; motivo por el cual surge la necesidad de su control.

Esta actividad se realiza en las unidades de vigilancia y lucha antivectorial de las áreas de salud, luego de la recogida de información en las diferentes zonas. Posteriormente en los diferentes niveles del Sistema Nacional de Salud (SNS) se hace el análisis de todos estos datos bajo la dirección del Viceministerio de Higiene y Epidemiología que radica en Ciudad de La Habana.

Generalmente la recogida, procesamiento y transmisión de los datos en los diferentes niveles se realiza manualmente. Otras de las vías que se emplean son: la entrega personal, por correo electrónico y teléfonos. Estos medios son poco seguros, lentos y no confiables; y su uso puede causar la introducción de errores humanos, la tardía detección de epidemias, retraso en los análisis necesarios de los reportes e información estadística.

Durante el curso 2007-2008 en la Universidad de las Ciencias Informáticas (UCI) se desarrolló una primera versión de un producto informático para la gestión de la información relacionada a la detección y control del vector Aedes Aegypti. El sistema se desarrolló sobre el framework CodeIgniter, que a pesar de ser fácil de configurar y sencillo de utilizar, carece de un conjunto de funcionalidades que pudieran limitar el desarrollo óptimo de nuevas versiones del sistema actual. Además presenta una interfaz gráfica poco amigable y segura.

En el curso 2008-2009 se realizó un nuevo análisis y diseño de este software al cual se añadieron nuevos requerimientos definidos por el cliente para un mejor control de los datos, debido a que la primera versión no cumplía con los nuevos lineamientos de la arquitectura definida para el Departamento de Sistemas Especializados en Medicina.

Sobre la base de lo anteriormente planteado se identifica como **problema a resolver**: ¿Cómo obtener un sistema funcional a partir del diseño realizado para viabilizar el proceso de gestión de la información relacionada con el control de focos de mosquitos Aedes Aegypti en Cuba?

Donde el **Objeto de estudio** es el proceso de gestión de la información referente a la detección y control de vectores en Cuba. El **Campo de acción** se enfoca en el Proceso de gestión de la información relacionada con el control de focos de mosquitos Aedes Aegypti en Cuba. Además se define como **Objetivo general:** Implementar la versión 1.0.1 del Sistema de Detección y Control de Vectores.

Para alcanzar el objetivo mencionado anteriormente y lograr el adecuado desarrollo del trabajo se plantean las siguientes tareas de investigación:

1. Realizar un análisis acerca de los procesos de negocio asociados al control de focos de mosquitos Aedes Aegypti en Cuba para la detección y control de vectores.
2. Asimilar la arquitectura definida por el Departamento de Sistemas Especializados en Medicina de conjunto con el cliente para el desarrollo del Sistema de Control Sanitario Internacional.
3. Actualizar el diseño de la base de datos de los procesos de negocio asociados al control de focos de mosquitos Aedes Aegypti en Cuba para la detección y control de vectores.
4. Realizar la actualización del modelo de datos.
5. Realizar la implementación de los procesos de negocio asociados al control de focos de mosquitos Aedes Aegypti en Cuba para la detección y control de vectores utilizando los patrones establecidos en el Análisis y Diseño.

El presente trabajo está estructurado en tres capítulos donde se refleja todo el trabajo investigativo, así como todo lo referente al diseño de la base de datos y la implementación de la solución propuesta, distribuido de la siguiente manera:

Capítulo 1. Fundamentación teórica: Estudio crítico y valorativo de la metodología, tecnologías y herramientas, plataforma y librería a utilizar en el desarrollo del módulo descrito en la tesis anterior.

Capítulo 2. Elementos de arquitectura: Se realiza una explicación de la arquitectura del sistema, para lo que se exponen los requisitos no funcionales del mismo, se realiza un análisis de los patrones o estilos arquitectónicos presentes en la propuesta de solución, además de un análisis de la estrategia de integración con otros módulos o sistemas, así como de la seguridad, una explicación de la vista de despliegue y de los estándares y estilos que se utilizaron.

Capítulo 3. Descripción y análisis de la solución propuesta: Se realiza un análisis del diseño propuesto por el analista, refinando los diagramas de clases del diseño y diagramas de interacción, así como el modelo de datos con una descripción de las tablas del mismo y por último una descripción de la vista de implementación.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA.

En el presente capítulo se realiza un análisis de los lenguajes de programación web actuales utilizados en el desarrollo de sistemas informáticos, y se presentan las metodologías de software y las herramientas a utilizar para llevar a cabo la solución del problema que se enfrenta; las cuales fueron previamente definidas en el documento de arquitectura de la Facultad 7 de la Universidad de la Ciencias Informáticas.

1.1 Sistema de Detección y Control de Vectores (Versión 1.0)

Construido en la Universidad de las Ciencias Informáticas, por un equipo de desarrollo del Polo de la Informática para la Salud perteneciente a la Facultad 7, fue desarrollado con los lineamientos arquitectónicos diseñados por el Grupo de Arquitectura – Facultad 7 en el curso 2007-2008, a los cuales se le han hecho modificaciones. Destinado a gestionar toda la información relacionada con el control de focos de mosquitos Aedes Aegypti en Cuba, el cual actualmente no se encuentra desplegado.

1.2 Flujo actual de los procesos

En la campaña de Vigilancia y Lucha Antivectorial contra los mosquitos Aedes Aegypti tienen lugar varios procesos importantes que serán descritos a continuación los cuales comienzan en la brigada, célula principal de información y terminando en el Viceministerio de Higiene y Epidemiología, donde se determinan las principales estrategias a nivel nacional para enfrentar al vector:

Inspecciones diarias

Al comenzar el día el Responsable del Área de Salud (AS) le indica a cada uno de los Jefe de brigadas el itinerario del día y de esta manera guían a sus respectivos Operarios “A” el trabajo que deben realizar, estos visitan (centros de trabajo, viviendas particulares, centros especializados, terrenos baldíos y solares yermos) que están en su plan de trabajo.

Al llegar un operario a un local determinado registra en un modelo los datos particulares del mismo y procede a inspeccionarlo. En estas visitas se examinan los lugares y depósitos posiblemente vulnerables

a presentar focos de mosquitos. También se le realiza un tratamiento a los depósitos que puede ser: destrucción, flameo y abatización.

Una vez culminada la jornada laboral cada operario le entrega la información recopilada a su Jefe de Brigada. En el Área de Salud los jefes de brigada recolectan estos modelos y los transmiten a través de entrega personal, por teléfono, correo electrónico u otros medios al Responsable del AS. Lo que trae consigo que no se gestione toda la información de forma rápida y eficiente pues existe la posibilidad de ocurrir pérdida de la misma en su trayectoria y en ocasiones no se cuentan con datos reales y precisos pues pueden ser cambiados accidentalmente durante su traslado.

Provoca, al ser lenta la gestión, que no se proporcione la ubicación rápida y exacta de los focos, pues el Responsable del AS se puede retrasar en su trabajo. Esta información lejos de ser inmediata es tardía.

Recogida y análisis de muestras

Si en alguno de estos depósitos examinados se encuentra un posible foco de mosquito, el Operario A recolecta una muestra y la entrega al Jefe de Brigada. Las muestras son enviadas al laboratorio para el análisis. En caso de resultar positiva, el Laboratorista informa el resultado al Responsable de la Vigilancia y Lucha Antivectorial del Área de Salud (AS) y envía las muestras al Laboratorista Provincial donde fue recogida la muestra.

Recopilar información por niveles

Además del resultado de las muestras enviadas del laboratorio, el Estadístico del AS recolecta toda la información de las brigadas, hace un resumen y lo entrega al responsable del AS el cual lo envía a la Unidad Municipal Vigilancia y Lucha Antivectorial (UMVLA). En esta unidad se efectúa un sumario de las áreas de salud asociadas al municipio y se transmite a la Unidad Provincial Vigilancia y Lucha Antivectorial (UPVLA) y esta a su vez a la Dirección Nacional de Vigilancia y Lucha Antivectorial (DNVLA).

Al concluir todos los procesos el Viceministro de Higiene y Epidemiología obtiene reportes estadísticos generales a nivel nacional, provincial, municipal y por áreas de salud, con el objetivo de controlar todas las viviendas inspeccionadas, incluyendo las positivas, el índice de infestación y las manzanas positivas reiterativas, posibilitándose así que se tomen las medidas pertinentes en las zonas más críticas.

1.3 Metodología de Desarrollo de Software

Una Metodología de Desarrollo de Software es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software. Una metodología puede seguir uno o varios modelos del ciclo de vida, es decir, indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo.

1.3.1 Proceso Unificado de Desarrollo (RUP)

RUP es una metodología de desarrollo de software, que intenta integrar todos los aspectos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de asegurar la producción de software de calidad, dentro de plazos y presupuestos predecibles. [2] Se utilizó esta metodología ya que define claramente quién, cómo, cuándo y qué debe hacerse en el proyecto. Además porque se caracteriza por ser guiado por los casos de uso, estar centrado en la arquitectura e iterativo e incremental.

RUP consta de 4 fases: Inicio, Elaboración, Construcción y Transición; las cuales finalizan con un hito donde se debe tomar una decisión importante. Además agrupa las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los tres últimos como de apoyo. (Figura 1.1)

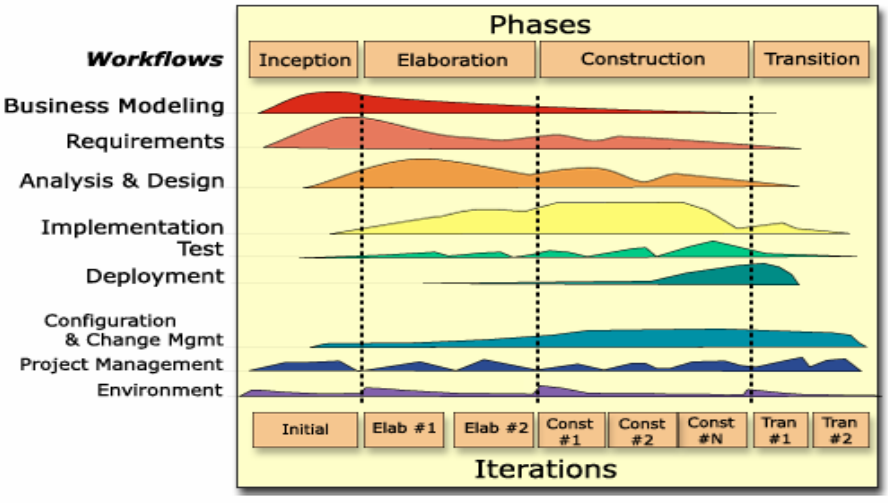

Figura 1: Proceso Unificado de Desarrollo

RUP y el Lenguaje Unificado de Modelado unidos conforman la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

1.4 Lenguaje Unificado de Modelado (UML) 2.0

Se utilizó UML debido a que es un lenguaje que permite especificar, construir, visualizar y documentar los artefactos de un sistema que involucra una gran cantidad de software; además permite la modelación de sistemas con tecnología orientada a objetos. Este lenguaje intenta darle solución al problema de propiedad de código al implementar un lenguaje de modelado común para todos los desarrollos, permitiendo que cualquier desarrollador con conocimientos de UML lo pueda entender, independientemente del lenguaje utilizado para este.

En lo que corresponde al desarrollo de programas, posee elementos gráficos para soportar la captura de requisitos, el análisis, el diseño, la implementación, y las pruebas. Sin embargo es necesario recalcar que UML es una notación y no un proceso/método, es decir, es una herramienta útil para representar los modelos del sistema en desarrollo, más no ofrece ningún tipo de guía o criterios acerca de cómo obtener esos modelos.

1.5 Herramientas Case

Case (Computer Aided Software Engineering o Ingeniería de Software Asistida por Computación) es la aplicación de métodos y técnicas a través de las cuales se les hace útil a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación. Por lo que se puede decir que las herramientas Case representan una forma que permite Modelar los Procesos de Negocios de las empresas y desarrollar los Sistemas de Información Gerenciales.

1.5.1 Visual Paradigm 6.4

Como herramienta Case se utilizó el Visual Paradigm debido a que es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos,

construcción, pruebas y despliegue. Apoya los estándares más altos de las notaciones de Java y de UML. Genera productos de calidad, soporta aplicaciones web y es fácil de instalar y actualizar. Está orientada a la creación de diseños y se usa el paradigma de programación orientada a objetos.

Visual Paradigm tiene características que son muy útiles que facilitaron el análisis y diseño del sistema, entre ellas se destaca que permite crear diagramas para UML 2.0, está diseñado para ser centrado en Casos de Uso y enfocado al negocio. Permite usar un lenguaje estándar común y la ingeniería directa e inversa. Se integra a los principales Entorno Integrado de Desarrollo (IDE) y está disponible para utilizarse en múltiples plataformas.

1.6 Lenguajes de programación web

La programación Web, parte de las siglas WWW, que significa World Wide Web o telaraña mundial. En la actualidad existen diferentes lenguajes para desarrollar en la web, estos han ido surgiendo debido a las tendencias y necesidades de las plataformas. Estos lenguajes se dividen en:

- ✓ Lenguajes del lado del Servidor: son los lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y son enviados al cliente en un lenguaje comprensible.
- ✓ Lenguajes del lado del Cliente: independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio.

1.6.1 PHP 5

PHP es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Es tecnología de código abierto muy útil para diseñar de forma rápida y eficaz aplicaciones web conectadas a bases de datos.

Potente lenguaje de secuencia de comandos diseñado específicamente para permitir a los programadores crear aplicaciones web con distintas prestaciones de forma rápida; el cual puede ser incluido con facilidad dentro del código HTML.

PHP es un lenguaje que posee tanto ventajas como desventajas para los programadores, entre sus ventajas se puede destacar que es muy fácil de aprender, es rápido y multiplataforma; además posee

conexión con la mayoría de los manejadores de base de datos y documentación en su página oficial con descripción y ejemplos de cada una de sus funciones.

Entre sus desventajas se encuentra que se necesita instalar un servidor web; donde el trabajo lo realiza el servidor y no delega al cliente. También la legibilidad del código se afecta al mezclar sentencias HTML y PHP y la programación orientada a objetos es aún muy deficiente para aplicaciones grandes.

1.6.2 Javascript

Es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los que soportan la carga de procesamiento.

Javascript es un lenguaje interpretado, basado en prototipos con el que se pueden crear pequeños programas que luego son insertados en una página web y en programas más grandes. Además es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera y Mozilla Firefox.

Se utilizó este lenguaje porque posee varias características, entre ellas se puede mencionar que es un lenguaje basado en acciones que posee menos restricciones. Gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas, entre otros.

1.7 Framework

Un Framework es una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Este se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Un framework Web, por tanto, se puede definir como un conjunto de componentes que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web.

1.7.1 Symfony 1.4.3

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. [3]

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix y Linux) como en plataformas Windows. [4]

Symfony se diseñó para que se ajustara a los siguientes requisitos: [5]

- ✓ Fácil de instalar y configurar en la mayoría de las plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- ✓ Independiente del sistema gestor de bases de datos.
- ✓ Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- ✓ Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- ✓ Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- ✓ Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- ✓ Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- ✓ Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

1.7.2 Yahoo User Interface (YUI) 2.8

Yahoo User Interface (YUI) Library es un conjunto de utilidades y controles. Estas están escritas en Javascript para desarrollar aplicaciones web interactivas usando técnicas DOM (Document Object Model), DHTML (Dynamic HTML) y AJAX (Asynchronous Javascript And XML). YUI contiene también un conjunto de fuentes CSS (Cascading Styles Sheets). Los componentes que YUI brinda están en Open Source bajo licencia BSD y está disponible para todo tipo de uso.

En el sistema se utilizó YUI por contar con un conjunto de componentes que están agrupados en Utilidades y Controles. Las utilidades simplifican el desarrollo para la compatibilidad entre navegadores basados en scripts DOM, DHTML y AJAX; mientras que los controles brindan elementos de diseño altamente interactivos para las páginas webs.

1.8 Entorno Integrado de Desarrollo (IDE)

Un Entorno Integrado de Desarrollo (IDE, Integrated Development Environment) es un sistema que facilita el trabajo del desarrollador de software, integrando sólidamente la edición orientada al lenguaje, la compilación o interpretación, la depuración, las medidas de rendimiento, la incorporación de las fuentes a un sistema de control de fuentes; normalmente de forma modular. [6]

1.8.1 NetBeans 6.8

El NetBeans IDE se utilizó ya que es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans, la cual es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso.

Aunque está escrito en Java puede servir para cualquier otro lenguaje de programación como es el PHP, brindando completamiento de código y está pensada para escribir, compilar, depurar y ejecutar programas. Existe además un número importante de módulos para extender el IDE NetBeans. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones.

1.8.2 Dreamweaver 8

Dreamweaver es un editor visual profesional para la creación de sitios y páginas web, con el cual resulta fácil crear y editar páginas compatibles con cualquier explorador y plataforma. La gran ventaja de este editor sobre otros es su gran poder de ampliación y personalización del mismo, puesto que en este programa, sus rutinas están hechas en Javascript, lo que le ofrece una gran flexibilidad en estas materias.

Entre las novedades presentes en Dreamweaver está la validación dinámica en distintos navegadores, es decir que las etiquetas HTML y las reglas CSS, siempre estarán legibles, siendo éstas compatibles con cualquier navegador del mercado. El soporte para CSS es amplio y la edición gráfica, como por ejemplo rotar, recortar o redimensionar, integrada a la interfaz de Dreamweaver, sin salir del programa.

Esta herramienta posee otras características como es el soporte para posicionamiento absoluto, cliente de FTP (File Transfer Protocol o Protocolo de Transferencia de Archivos) integrado (con soporte Firewall), soporte XML, plantillas, interfaz optimizada y personalizada, así como herramientas de codificación mejoradas.

1.9 Servidor Web

Con el surgimiento de Internet, más conocido como la “La Red de Redes” se puede acceder a la información a través de páginas web basadas en el modelo Cliente/Servidor. La estrategia Cliente/Servidor, es un paradigma de división del trabajo informático en el que las tareas se reparten entre un número de clientes que efectúan peticiones de servicios de acuerdo con un protocolo, y un número de servidores que las atienden.

Para que esto sea posible debe existir el servidor web, el cual no es más que un programa que corre sobre el servidor que escucha las peticiones HTTP (HyperText Transfer Protocol o Protocolo de Transferencia de Hipertexto) que le llegan y las satisface. Dependiendo del tipo de la petición, el servidor web buscará una página web o bien ejecutará un programa en el servidor. De cualquier modo, siempre devolverá algún tipo de resultado HTML (HyperText Markup Language o Lenguaje de Marcas de Hipertexto) al cliente o navegador que realizó la petición. [7]

1.9.1 Servidor Web Apache 2.2

Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. [8] La licencia Apache es una descendiente de la licencias BSD (Berkeley Software Distribution), no es GPL (General Public License). Esta licencia permite hacer lo que se desee con el código fuente (incluso productos propietarios). [9] Algunas razones por las que este software libre es grandemente reconocido en muchos ámbitos empresariales y tecnológicos son:

- ✓ Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- ✓ Apache es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si se quiere ver que es lo que se está instalando como servidor se puede saber, sin ningún secreto, sin ninguna puerta trasera.
- ✓ Es un servidor altamente configurable de diseño modular. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que sean instalados cuando sea necesario.
- ✓ Trabaja con gran cantidad de Perl, PHP (Hypertext Preprocessor) y otros lenguajes de script. También trabaja con Java y páginas jsp (Java Server Pages o Páginas de Servidor Java). Teniendo todo el soporte que se necesita para tener páginas dinámicas.
- ✓ Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- ✓ Tiene una alta configurabilidad en la creación y gestión de logs. Permite la creación de ficheros de log a medida del administrador y de este modo se puede tener un mayor control sobre lo que sucede en el servidor. [10]

1.10 Sistema Gestor de Bases de Datos

Un Sistema Gestor o Manejador de Bases de Datos (SGBD) es un conjunto de programas que permite a los usuarios crear y mantener una Base de Datos (BD), asegurando su integridad, confidencialidad y seguridad. El SGBD es un software que facilita el proceso de definir, construir y manipular la BD para diversas aplicaciones.

1.10.1 PostgreSQL 8.3

Es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. Está considerado como la base de datos de código abierto más avanzada del mundo, liberado bajo la licencia BSD (Berkeley Software Distribution) y proporciona un gran número de características.

Se utilizó PostgresSQL por contar entre sus características con aproximación de los datos a un modelo objeto-relacional, y que es capaz de manejar complejas rutinas y reglas. Es altamente extensible ya que soporta operadores, funciones, métodos de acceso y tipos de datos definidos por el usuario. PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos. Además de poseer Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), tecnología usada para evitar bloqueos innecesarios.

En este capítulo se realizó un análisis de las herramientas, metodologías y tecnologías que se han propuesto para el desarrollo de la aplicación con el objetivo de dar una solución informática a la situación existente. Estas se encuentran definidas en el documento de arquitectura de la Facultad 7 de la Universidad de la Ciencias Informáticas, por lo que se determinó utilizar:

Como servidor web el Apache 2.2, el cual es libre de código abierto y bajo la licencia Apache/GPL cuya descripción y aplicaciones ya fue dada anteriormente. Como lenguaje de programación PHP en su versión 5, como gestor de bases de datos se usará el ya conocido PostgreSQL en la versión 8.3.

Para la elaboración de la aplicación, se usará el IDE de programación NetBeans por las grandes prestaciones que este brinda. El sistema además está implementado usando el framework Symfony y se hace rehúso de librerías como es la YUI Library 2.8.

CAPÍTULO 2. ELEMENTOS DE ARQUITECTURA

En el siguiente capítulo se expone la arquitectura que ha sido determinada para el desarrollo de la aplicación de Vectores, mostrando para ello los requisitos no funcionales, los patrones arquitectónicos seguidos en el desarrollo de la aplicación, una explicación del modelo de despliegue y las estrategias de codificación a utilizar. Además de realizar un análisis de las posibles estrategias de integración.

2.1 Requerimientos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Un requisito no funcional especifica restricciones físicas sobre un requisito funcional. [11]

A continuación se muestran los requerimientos no funcionales correspondientes al sistema en cuestión:

2.1.1 Requerimientos de Apariencia o Interfaz Externa

- ✓ Se podrán distinguir colores atractivos y acordes con los recomendados para los software de salud.
- ✓ Debe poseer un ambiente amigable, intuitivo, sencillo y de fácil navegación, tratando así de impedir el rechazo por parte del usuario al tener que interactuar con un sistema no conocido.
- ✓ Paginación de reportes de búsqueda, y listados.
- ✓ Diseño encuadrado para resoluciones de 1024 x 768, pero preparado para verse en otras resoluciones.

2.1.2 Requerimientos de Usabilidad

- ✓ La aplicación Web debe ser flexible y de fácil aprendizaje, pues se trata en todo lo posible de mantener un estándar de operabilidad que logre que las interacciones del usuario con el sistema sean predecibles y familiares.

- ✓ El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web en sentido general.

2.1.3 Requerimientos de Soporte

- ✓ El sistema contará con una ayuda para facilitar al usuario el manejo de la aplicación.
- ✓ Estará bien documentado para garantizar futuros mantenimientos.

2.1.4 Requerimientos de Portabilidad

- ✓ El sistema podrá ejecutarse sobre plataforma Linux, Windows 98 ó superior.

2.1.5 Requerimientos de Seguridad

- ✓ La autenticación de los usuarios en el sistema será garantizada por el Sistema de Autenticación, Autorización y Auditoría (SAAA) al cual estará conectada la aplicación.
- ✓ Las funcionalidades deberán estar restringidas de acuerdo a los diferentes roles con el objetivo de garantizar que el acceso a la información sea según el nivel de autorización.
- ✓ El sistema deberá estar protegido ante el acceso no autorizado a la información.

2.1.6 Requerimientos de Software

Del lado del cliente:

- ✓ El cliente podrá tener acceso al sistema a través de los navegadores Internet Explorer 1.5 o superior y Mozilla Firefox 2.0 o superior.
- ✓ Sistema operativo Linux o Windows 98 ó Superior.

Del lado del servidor:

- ✓ Sistema operativo Linux.

- ✓ Servidor Web Apache 2.2 y PHP 5.
- ✓ Servidor de Base de Datos PostgreSQL 8.3.
- ✓ Framework Symfony 1.4.3.

2.1.7 Requerimientos de Hardware

Del lado del cliente:

- ✓ Procesador Pentium III o superior.
- ✓ 256 MB de memoria RAM o superior.
- ✓ Monitor VGA o superior.
- ✓ Tarjeta de red.

Del lado del servidor:

- ✓ Procesador Pentium IV o superior.
- ✓ 2 GB de memoria RAM o superior.
- ✓ Disco Duro de 80 GB.
- ✓ Monitor tipo VGA o superior.
- ✓ Tarjeta de red.

2.1.8 Requerimientos de Diseño e Implementación

- ✓ Se utilizarán los patrones de diseño GRASP.
- ✓ Para la implementación se utilizará como lenguaje de programación PHP.

2.1.9 Requerimientos de Ayuda y Documentación en línea

- ✓ Contará con un manual de usuario para que se pueda explotar al máximo.

- ✓ Contará con una ayuda digital, a la cual se podrá acceder desde cualquier parte de la aplicación.

2.2 Arquitectura de Software y patrones arquitectónicos

En la actualidad la Arquitectura de Software (AS) posee un gran número de definiciones, se decidió utilizar la que brinda el documento del Instituto de Ingenieros Electricistas y Electrónicos (IEEE) 1471-2000 que expresa:

"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución."

Partiendo de que la Arquitectura de Software se inscribe en el campo de la Ingeniería de Software, vale la pena comparar sus definiciones, entonces conforme al estándar IEEE 610.12.1990, se entiende como Ingeniería de Software a:

"La aplicación de una estrategia sistemática, disciplinada y cuantificable al desarrollo, aplicación y mantenimiento del software; esto es, la aplicación de la ingeniería al software."

Se puede notar que la noción clave de la arquitectura es la organización (un concepto cualitativo o estructural), mientras que la ingeniería tiende fundamentalmente a ser una noción sistemática susceptible de cuantificarse. [12]

El sistema basa su arquitectura en una elaborada por el grupo de arquitectura MINSAP-MIC para el desarrollo de todas las aplicaciones para Infomed. Existen numerosos patrones y estilos de arquitecturas de software o variantes arquitectónicas que definen a una familia de sistemas informáticos en términos de su organización estructural.

Los patrones arquitectónicos especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes. Describen interacciones amplias de elementos abstractos de diseño que permiten al arquitecto o diseñador pensar en un problema complejo mediante una abreviatura intuitiva. [13]

A continuación se mencionan los patrones y estilos de arquitecturas de software que se utilizarán para el desarrollo de la aplicación.

2.2.1 Arquitectura en capas

La arquitectura en capas es la generalización de la arquitectura Cliente-Servidor, es un estilo de programación, su objetivo primordial es la separación de la capa de presentación, capa de negocio y la capa de datos.

Los 3 niveles o capas son:

- ✓ **Capa de presentación**: Presenta el sistema al usuario, comunica y captura la información del usuario dando un mínimo de proceso. Esta capa se comunica únicamente con la capa de negocio.
- ✓ **Capa de negocio**: La capa de negocio es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio o incluso de lógica del negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para almacenar o recuperar los mismos.
- ✓ **Capa de datos**: La capa de acceso a datos contiene clases que interactúan con la base de datos, estas clases altamente especializadas permiten, utilizando los procedimientos almacenados (funciones para interactuar con la base de datos) generados, realizar todas las operaciones con la base de datos de forma transparente para la capa de negocio. [14]

La utilización de esta arquitectura en el caso del módulo Vectores es debido a que se logra separar la lógica de presentación de la lógica del negocio, aunque esta última se encuentra un tanto acoplada a la lógica de acceso a datos, sin embargo se logra una gran abstracción en cuanto al Sistema Gestor de Base Datos a utilizar por lo que si este es reemplazado, el impacto sobre la lógica del acceso a datos será mínimo.

2.2.2 Arquitectura basada en componentes

La Arquitectura Basada en componentes es una rama de la Ingeniería de Software en la cual se trata con énfasis la descomposición del software en componentes funcionales. Esta descomposición permite convertir componentes pre-existentes en piezas más grandes de software.

El proceso de construcción de un software con componentes ya existentes, da origen al principio de reutilización del software, a través de este los componentes son implementados de una forma que permita su utilización funcional sobre diferentes sistemas en el futuro.

El uso de esta arquitectura posee algunas ventajas:

- ✓ Reutilización del software. Nos lleva a alcanzar un mayor nivel de reutilización de software.
- ✓ Simplifica las pruebas. Permitir que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados.
- ✓ Simplifica el mantenimiento del sistema. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema.
- ✓ Mayor calidad. Dado que un componente puede ser construido y luego mejorado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo. [15]

Se debe tener en cuenta que este tipo de arquitectura permite la integración de múltiples sistemas, lo que implica una dependencia en la información gestionada por cada uno de ellos.

Se emplea esta arquitectura porque el sistema estará integrado con componentes del Sistema de Información para la Salud (SISalud) de los que consumirán algunos servicios, estos son:

- ✓ Sistema de Autenticación, Autorización y Auditoría. (SAAA).
- ✓ Registro de Unidades de Salud (RUS).
- ✓ Registro de Ubicación (RU).
- ✓ Registro de Localidades (RL).

2.2.3 Modelo-Vista-Controlador (MVC)

El Modelo Vista Controlador es un patrón de diseño de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Este patrón se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la

página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica de negocio. [16]

Los elementos de este patrón son:

- ✓ **Modelo*:*** Es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.
- ✓ **Vista*:*** Es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.
- ✓ **Controlador**: Es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo. [17]

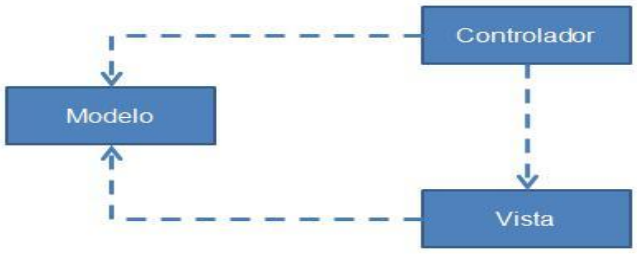

Figura 2: Patrón Modelo Vista Controlador.

Entre las ventajas del uso del patrón Modelo-Vista-Controlador están las siguientes:

- ✓ La separación del Modelo de la Vista, es decir, separar los datos de la representación visual de los mismos.

- ✓ Facilita agregar nuevos tipos de datos según sea requerido por la aplicación ya que son independientes del funcionamiento de las otras capas.
- ✓ Facilita el mantenimiento en caso de errores. [18]

En el caso del módulo de Vectores se utilizó este patrón ya que el framework utilizado (Symfony) se basa en el mismo y/o implementa de forma que el desarrollo de las aplicaciones sea rápido y sencillo.

2.3 Estrategias de Integración

El Sistema de Detección y Control de Vectores para lograr un completo funcionamiento necesitará consumir varios servicios desarrollados con anterioridad en el SISalud. Los sistemas externos y los aspectos que se solicitarán de ellos son:

- ✓ El SAAA se utilizará con el objetivo de conocer el usuario que está autenticado en el sistema, a qué nivel pertenece, qué tipo de usuario es y a qué módulos tiene acceso.
- ✓ El RUS para conocer cuál es el nombre de la Unidad de Salud (US) en la que se encuentra el usuario.
- ✓ El RU para listar todas las provincias, municipios y las localidades de cada municipio. Además para listar las manzanas por localidad, las calles por manzana y las calles por localidad.
- ✓ El RL se necesitará para listar los consejos populares por localidad y las circunscripciones por consejo popular.

Además se integrará con el módulo Higiene y Epidemiología de CSI para un mejor control de las enfermedades transmisibles.

2.4 Seguridad

La seguridad informática es tanto la investigación como la ejecución de políticas de protección de datos en ordenadores con el propósito de asegurar que los recursos del sistema sean utilizados de la manera que se decidió y que el acceso a la información allí contenida así como su modificación sólo sea posible a las personas que se encuentren acreditadas y dentro de los límites de su autorización.

Symfony posee herramientas que permiten la creación de aplicaciones seguras, en las que los usuarios necesitan estar autenticados antes de acceder a alguna característica o a partes de la aplicación. Añadir esta seguridad a una aplicación requiere dos pasos: declarar los requerimientos de seguridad para cada acción y autenticar a los usuarios con privilegios para que puedan acceder estas acciones seguras.

En Symfony, los privilegios están compuestos por dos partes:

- ✓ Las acciones seguras requieren que los usuarios estén autenticados.
- ✓ Las credenciales son privilegios de seguridad agrupados bajo un nombre y que permiten organizar la seguridad en grupos.

Para restringir el acceso a una acción se crea y se edita un archivo de configuración YML llamado security.yml en el directorio config/ del módulo. En este archivo, se pueden especificar los requerimientos de seguridad que los usuarios deberán satisfacer para cada acción o para todas las acciones.

Las acciones no incluyen restricciones de seguridad por defecto, así que cuando no existe el archivo security.yml o no se indica ninguna acción en ese archivo, todas las acciones son accesibles por todos los usuarios. [19]

En el caso del sistema de Vectores las tareas de autenticación serán confiadas al componente de seguridad del Sistema de Información para la Salud (SISalud), el SAAA; para garantizar la autenticación de los usuarios y para tratar la autorización se utilizará un sistema de roles y permisos previamente definido que verificará los privilegios de cada uno de los usuarios en determinadas áreas.

2.5 Modelo de Despliegue

Los diagramas de despliegue muestran la disposición física de los distintos nodos que entran en la composición de un sistema y el reparto de los programas ejecutables sobre estos nodos. [20] En general un nodo será una unidad de computación de algún tipo y las instancias de componentes software pueden estar unidas por relaciones de dependencia.

Para el despliegue del Sistema de Detección y Control de Vectores del Proyecto CSI se contará con un Servidor de Aplicaciones donde se encontrará el código fuente de la aplicación, el mismo se comunicará mediante protocolo SOAP con el Servidor Web RIS donde se encontrarán los diferentes componentes del

SISalud que se necesitarán utilizar para llevar a cabo el desarrollo del sistema de Vectores como parte de su propia arquitectura basada en componentes.

Además contará con un Servidor de Base de Datos donde estará la base de datos de la aplicación, el cual se comunicará mediante protocolo TCP/IP con el Servidor de Aplicaciones; y por último contará con las PC Cliente que serán las encargadas de permitirle al usuario el acceso al sistema mediante protocolo HTTP, las cuales se comunicaran con una impresora.

A continuación se presenta el Diagrama de Despliegue correspondiente al Módulo Vectores del Sistema de Control Sanitario Internacional.

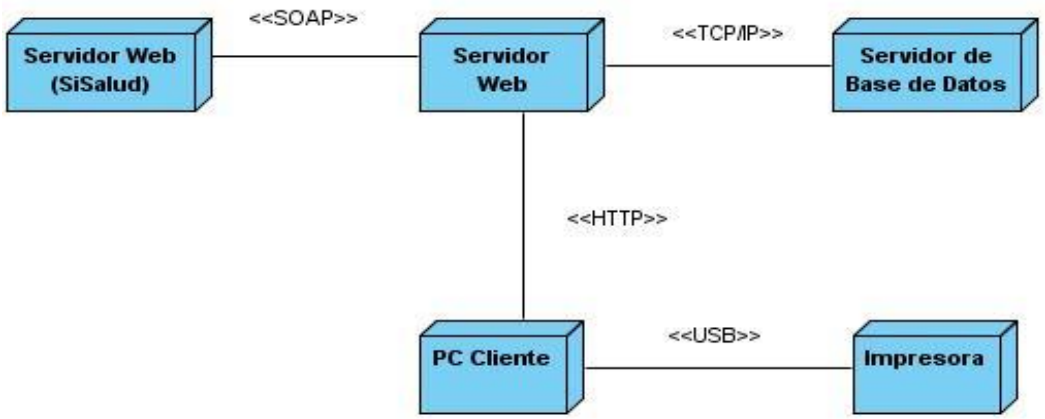

Figura 3: Diagrama de Despliegue.

2.6 Estrategias de Codificación. Estándares y estilos.

Para la realización de la aplicación se tuvieron en cuenta las restricciones de nomenclaturas y los estándares de codificación definidos por el Grupo de Arquitectura y Tecnologías de la Facultad 7.

Restricciones de nomenclatura

Idioma: Se debe utilizar como idioma el español, las palabras no se acentuarán.

Identación
Objetivo: Lograr una estructura uniforme para los bloques de código así como para los diferentes

<table>
<tr><td colspan="3">niveles de anidamiento.</td></tr>
<tr><td>Inicio y fin de bloque</td><td></td><td>Se recomienda dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones if, else, for, while, do while, switch, foreach.</td></tr>
<tr><td>Aspectos Generales</td><td colspan="2">El identado debe ser de dos espacios por bloque de código. No se debe usar el tabulador; ya que este puede variar según la PC o la configuración de dicha tecla.

Los inicios ({) y cierre (}) de ámbito deber estar alineados debajo de la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción.

Nunca colocar {en la línea de un código cualquiera, esto requiere una línea propia.</td></tr>
<tr><td colspan="3">Comentarios, separadores, líneas, espacios en blanco y márgenes.

Objetivo: Establecer un modo común para comentar el código de forma tal que sea comprensible con sólo leerlo una vez.</td></tr>
<tr><td>Ubicación de comentarios</td><td>Al inicio de cada clase o función y al final de cada bloque de código.</td><td>Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro) entre otras cosas.</td></tr>
<tr><td>Líneas en blanco</td><td>Se emplean antes y después de métodos, clases y</td><td>Se recomienda dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función</td></tr>
</table>

	estructuras.	
Espacios en blanco	Entre operadores lógicos y aritméticos.	Se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código. Ejemplo: producto = nomproducto
Aspectos generales	Sobre el comentario	Se debe evitar comentar cada línea de código. Cuando el comentario se aplica a un grupo de instrucciones debe estar seguido de una línea en blanco. En caso de que se necesite comentar una sola instrucción se suprime la línea en blanco o se escribe a continuación de la instrucción
	Sobre los espacios en blanco	No se debe usar espacio en blanco: Después del corchete abierto y antes del cerrado de un arreglo. Después del paréntesis abierto y antes del cerrado. Antes de un punto y coma.

Variables y constantes		
Apariencia de variables	Las variables tendrán un prefijo para el tipo de datos en minúscula.	El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, la cual identificara el tipo de datos al que se refiere (ver tabla 1.1), en caso de que sea un nombre compuesto se empleará notación CamellCasing**. Ejemplo: sNombrePaciente
Apariencia de constantes	Todas sus letras en mayúscula	Se deben declarar las constantes con todas sus letras en mayúscula.

Aspectos generales	Nombres de las variables y constantes	El nombre empleado, debe permitir que con sólo leerlo se conozca el propósito de la misma.
Clases y Objeto **Objetivo:** Nombrar las clases e instancias de forma estándar para todas las aplicaciones.		
Apariencia de clases y objetos	Primera letra en mayúscula	Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing*. Ejemplo: MiClase(). Para el caso de las instancias se comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.
Apariencia de atributos	Primera letra en minúscula	El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, la cual estará en correspondencia al tipo de dato al que se refiere, en caso de que sea un nombre compuesto se empleará notación CamellCasing**.
Apariencia de las funciones	Primera letra en mayúscula	Para nombrar las funciones se debe tratar de utilizar verbos que denoten la acción que hace la función. Se empleará notación PascalCasing*. Ejemplo: functionBuscarUnidad(). Si son funciones que obtienen un dato se emplea el prefijo get y si fijan algún valor se emplea el prefijo set
Declaración de parámetro en	Agrupados por tipos Poner los string 1	Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que

funciones	numéricos 2, además, agrupar según valores por defecto.	estén agrupados por el tipo de dato que contienen, especificando el tipo de datos (tabla 1.1).
Aspectos generales	Sobre las clases, los objetos, los atributos y las funciones.	El nombre empleado para las clases, objetos, atributos y funciones debe permitir que con sólo leerlo se conozca el propósito de los mismos.
Bases de Datos, Tablas, esquemas y Campos		
Apariencia de la Base de Datos.	Las 2 primeras letras representan el tipo.	Los nombres de las Bases de Datos deben comenzar con el prefijo bd a continuación underscoard y luego el nombre completamente en minúscula, en caso de que sea un nombre compuesto se empleará notación. Los nombres serán cortos y descriptivos. Ejemplo: bd_balancematerial.
Apariencia de las vistas	Las 2 primeras letras representan el tipo. Todas las letras en minúscula	El nombre a emplear para las vistas deben comenzar con el prefijo vt seguido de underscoard y el nombre debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor. Ejemplo: createview ‘vt_finanzas’;

Apariencia de las tablas	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para las tablas debe comenzar con el prefijo tb seguido de underscoard y luego debe escribirse todas las letras en minúscula, en caso de que sea un nombre compuesto se utilizará underscoard para separarlo. Ejemplo: 'tb_producto';
Tablas que representen Relaciones	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tr seguido de underscoard y el nombre será la concatenación del nombre de las dos tablas que la generaron separados por uderscoard todo en minúscula. Ejemplo: 'tr_paciente_enfermedad'
Tablas que representen nomencladores.	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tn seguido de underscoard. El nombre será corto y descriptivo todo en minúscula. Ejemplo: 'tn_color_piel'
Apariencia de los procedimientos almacenados.	Las 2 primeras letras representan el tipo. Todas las letras en minúscula.	El nombre a emplear para los procedimientos debe comenzar con el prefijo pa seguido de underscoard y luego debe escribirse todas las letras en minúscula en caso de que sea un nombre compuesto se utilizará el underscoard para separarlo.

		Ejemplo: pa _ paciente_especialidad.
Apariencia de los campos	Todas las letras en minúscula.	El nombre a emplear para los campos debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor. Ejemplo: 'idproducto';
Nombre de los campos	En caso de identificadores.	Todos los campos identificadores van a comenzar con el identificador id seguido de underscoard y posteriormente el nombre del campo Ejemplo: id_municipio.
Sentencias SQL	Todas las letras en mayúscula.	Las palabras correspondientes a las sentencias SQL y sus parámetros deben ir en mayúsculas.
Aspectos generales	Sobre las BD, vistas, tablas atributos y procedimientos.	El nombre empleado para las Bases de Datos, las vistas, las tablas, los campos y los procedimientos almacenados, deben permitir que con sólo leerlos se conozca el propósito de los mismos.
Controles		
Apariencia de los	Los controles tendrán un prefijo para el tipo de	El nombre que se le da a los controles deben comenzar con las primeras letras en minúscula,

controles.	datos en minúscula.	las cuales identificarán el tipo de datos al que se refiere (ver tabla 1.2), en caso de que sea un nombre compuesto se empleará notación CamellCasing**. Ejemplo: btnAceptar

- ***Notación PascalCasing:** Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula. Ejemplo: NotacionPascalCasing.
- ****Notación CamellCasing:** Los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula excepto la primera palabra que debe iniciar con minúscula. Ejemplo: notacionCamellCasing.

Estándares de codificación

Para el lenguaje de programación del lado del servidor PHP se utilizaron los estándares de codificación definidos en http://pear.php.net/manual/en/standards.php.

En este capítulo se definió los RNF con los que contará el sistema entre los que se encuentran los de Apariencia o Interfaz Externa, Usabilidad y Seguridad entre otros. Se realizó un análisis de los elementos de arquitectura presentes en el desarrollo del software; definiéndose como patrones arquitectónicos el Modelo-Vista-Controlador, la Arquitectura en Capas y la basada en Componentes.

Se definió la autenticación del sistema que será principalmente a través del SAAA y la distribución del hardware en la cual se desplegará el sistema teniéndose un servidor de aplicaciones, un servidor de aplicaciones del SISalud, uno de BD, una PC cliente y una impresora.

CAPÍTULO 3. DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

El capítulo que a continuación se presenta, abordará primeramente una valoración sobre el diseño propuesto por el analista, además de la descripción de las clases del sistema y de las tablas de la Base de Datos; observándose también el Modelo de Datos de la misma. Por último en el mismo también se presenta la Vista de Componentes para describir la vista de implementación estática del sistema.

3.1 Valoración crítica del diseño propuesto por el analista

El diseño propuesto por los analistas aportó un punto de partida para comenzar las actividades de implementación, a pesar de que hubo que realizar cambios en el mismo durante el transcurso del desarrollo del sistema. Los diagramas de clases permitieron tener una mejor visión de las relaciones entre las clases que involucra el sistema.

Los diagramas de interacción están compuestos por los diagramas de secuencia y los de colaboración. Los de secuencia mostraron en detalles los escenarios pertenecientes a cada caso de uso, permitiendo ver con mayor claridad el intercambio dinámico del sistema.

Uno de los cambios que se realizó fue en el prototipo no funcional de la aplicación debido a que el Centro Especializado en Soluciones de Informática Médica (CESIM) definió un diseño estándar para todas sus aplicaciones y el anterior no estaba en correspondencia con el mismo.

3.2 Descripción de las nuevas clases u operaciones necesarias

A partir del enfoque MVC del Symfony, las nuevas clases a describir serán de tipo Controladora, Modelo o Vista (Interfaz). Seguidamente se exponen solo las controladoras y los modelos de los casos de uso críticos del sistema.

Nombre: inspecciónActions.php
Tipo de clase: Controladora

Para cada responsabilidad:	
Nombre:	executeIndex()
Descripción:	Muestra la vista principal para las acciones inspecciones a viviendas inspeccionadas por brigadas.
Nombre:	Add()
Descripción:	Adiciona una nueva inspección diaria.
Nombre:	Delete()
Descripción:	Elimina una inspección diaria existente.
Nombre:	Update()
Descripción:	Modifica una inspección diaria.
Nombre:	List()
Descripción:	Mostrar todos los datos de las inspecciones diarias.
Nombre:	Search()
Descripción:	Busca los datos de las inspecciones diarias.
Nombre:	Acumulado()
Descripción:	Realiza el acumulado de todas las inspecciones diarias para un día.

Tabla 1: Descripción de la clase controladora inspecciónActions.php

Nombre: positivaActions.php

Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	executeIndex()
Descripción:	Muestra la vista principal para las acciones viviendas positivas por brigadas.
Nombre:	Add()
Descripción:	Adiciona una nueva vivienda positiva.
Nombre:	Delete()
Descripción:	Elimina una vivienda positiva existente.
Nombre:	Update()
Descripción:	Modifica una vivienda positiva.
Nombre:	List()
Descripción:	Muestra los datos de las viviendas positivas.
Nombre:	Search()
Descripción:	Busca los datos de las viviendas positivas.

Tabla 2: Descripción de la clase controladora positivaActions.php

Nombre: area_saludActions.php
Tipo de clase: Controladora
Para cada responsabilidad:

Nombre:	executeIndex()
Descripción:	Muestra la vista principal para las acciones de las áreas de salud.
Nombre:	Search()
Descripción:	Busca todas las áreas de salud.
Nombre:	List()
Descripción:	Muestra todas las áreas de salud.
Nombre:	Update()
Descripción:	Modifica los datos de un área de salud.

Tabla 3: Descripción de la clase controladora area_saludActions.php

Nombre: cicloActions.php	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	executeIndex()
Descripción:	Muestra la vista principal para las acciones de los ciclos.
Nombre:	Add()
Descripción:	Adiciona un nuevo ciclo.
Nombre:	Delete()
Descripción:	Elimina un ciclo existente.

Nombre:	Update()
Descripción:	Modifica los datos de un ciclo.
Nombre:	Search()
Descripción:	Busca los datos de todos los ciclos.
Nombre:	List()
Descripción:	Muestra los datos de todos los ciclos.

Tabla 4: Descripción de la clase controladora cicloActions.php

Nombre: TbInspeccionDiaria.class.php	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Tb_Inspeccion_Diaria()
Descripción:	Constructor de la clase.
Nombre:	Add($inspeccion_diaria)
Descripción:	Método que se encarga de insertar los datos de una inspección en la base de datos.
Nombre:	Delete($id)
Descripción:	Método que se encarga de eliminar una inspección determinada.
Nombre:	Update($inspeccion_diaria)

Descripción:	Método que se encarga de actualizar los datos de una inspección determinada.
Nombre:	Search()
Descripción:	Busca los datos de todos los ciclos.
Nombre:	Calcular_Acumulado($inspeccion)
Descripción:	Método que se encarga de calcular los acumulados de los locales inspeccionados, locales cerrados, locales recuperados, locales inspeccionados, depósitos inspeccionados, destruidos, tratados y flameados hasta ese momento de una brigada en ese ciclo.
Nombre:	Inspeccion_Diaria($id)
Descripción:	Método que devuelve los datos de una inspección determinada.
Nombre:	Listar_Inspeccion_Diaria_Brigadas($query)
Descripción:	Método que devuelve los nombres de las brigadas de una lista de inspecciones.
Nombre:	Listar_Inspeccion_Diaria_Localidades($query)
Descripción:	Método que devuelve los nombres de las localidades de una lista de inspecciones.
Nombre:	Listar_Inspeccion_Diaria_Manzanas($query)
Descripción:	Método que devuelve los nombres de las manzanas de una lista de inspecciones.

Tabla 5: Descripción de la clase modeloTbInspeccionDiaria.class.php

Nombre: TbLocalPositivo.class.php

Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Tb_Local_Positivo()
Descripción:	Constructor de la clase.
Nombre:	Insertar_Vivienda($local_positivo,$numero_vivienda,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de insertar los datos de un local positivo en caso de que sea una vivienda.
Nombre:	Actualizar_vivienda($local_positivo,$numero_vivienda,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de actualizar los datos de un local positivo en caso de que sea una vivienda.
Nombre:	Insertar_Edificio($local_positivo,$edificio,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de insertar los datos de un local positivo en caso de que sea un edificio.
Nombre:	Actualizar_Edificio($local_positivo,$edificio,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de actualizar los datos de un local positivo en caso de que sea un edificio.
Nombre:	Insertar_Centro_Trabajo($local_positivo,$centro_trabajo,$depositos,$especies_asoci

	adas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de insertar los datos de un local positivo en caso de que sea un centro de trabajo.
Nombre:	Actualizar_Centro_Trabajo($local_positivo,$centro_trabajo,$depositos,$especies_asociadas,$circunscripcion,$consejo_popular,$calle)
Descripción:	Método que se encarga de actualizar los datos de un local positivo en caso de que sea un centro de trabajo.
Nombre:	Insertar_Consejo_Popular($consejo_popular)
Descripción:	Método que se encarga de insertar los datos de un consejo popular.
Nombre:	Insertar_Circunscripcion($circunscripcion)
Descripción:	Método que se encarga de insertar los datos de una circunscripción.
Nombre:	Insertar_Calle($calle)
Descripción:	Método que se encarga de insertar los datos de una calle.
Nombre:	Insertar_Depositos($id_local_positivo, $depositos, $especies_asociadas)
Descripción:	Método que se encarga de insertar los depósitos correspondientes a un local positivo.
Nombre:	Actualizar_Depositos($id_local_positivo, $depositos, $especies_asociadas)
Descripción:	Método que se encarga de actualizar los depósitos correspondientes a un local positivo.
Nombre:	Insertar_Especies_Asociadas($id_deposito_ficticio,$id_deposito, $id_local_positivo,

	$especies_asociadas)
Descripción:	Método que se encarga de insertar las especies asociadas a un depósito.
Nombre:	Actualizar_Especies_Asociadas($id_deposito_ficticio,$id_deposito, $id_local_positivo, $especies_asociadas)
Descripción:	Método que se encarga de actualizar las especies asociadas a un depósito.
Nombre:	Eliminar_Local_Positivo($id)
Descripción:	Método que se encarga de eliminar un local positivo.
Nombre:	Listar_Local_Positivo($id_inspeccion_diaria,$per_page, $offset)
Descripción:	Método que lista los locales positivos de tipo terreno baldío correspondientes a una inspección.
Nombre:	Listar_Local_Positivo_Casa($id_inspeccion_diaria)
Descripción:	Método que lista los locales positivos de tipo casa correspondientes a una inspección.
Nombre:	Listar_Local_Positivo_Edificio($id_inspeccion_diaria)
Descripción:	Método que lista los locales positivos de tipo edificio correspondientes a una inspección.
Nombre:	Listar_Local_Positivo_Centro_Trabajo($id_inspeccion_diaria)
Descripción:	Método que lista los locales positivos de tipo centro de trabajo correspondiente a una inspección.
Nombre:	Listar_Local_Positivo_Calles($query)

Descripción:	Método que lista el nombre de las calles de un conjunto de locales positivos.
Nombre:	Local_Positivo($id)
Descripción:	Método que devuelve los datos de un local positivo determinado.
Nombre:	Listar_Depositos($id_local_positivo,$per_page, $offset)
Descripción:	Método que lista los depósitos pertenecientes a un local positivo determinado.
Nombre:	Listar_Especies_Asociadas_A_Deposito($id_deposito,$per_page, $offset)
Descripción:	Método que lista las especies asociadas a un depósito determinado.
Nombre:	Nombre_Consejo_Popular($id)
Descripción:	Método que devuelve el nombre de un consejo popular.
Nombre:	Nombre_Circunscripcion($id)
Descripción:	Método que devuelve el nombre de una circunscripción determinada.
Nombre:	Nombre_Calle($id)
Descripción:	Método que devuelve el nombre de una calle determinada.

Tabla 6: Descripción de la clase modeloTbLocalPositivo.class.php

Nombre: TbAreaSalud.class.php	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Tb_Area_Salud()

Descripción:	Constructor de la clase.
Nombre:	Insertar_Area_Salud($areas_salud,$cantidad)
Descripción:	Método que se encarga de insertar los datos de las áreas de salud en la base de datos.
Nombre:	Buscar_Areas_Configuradas($provincia,$municipios)
Descripción:	Método que se encarga de buscar las áreas de salud que han sido configuradas.
Nombre:	Obtener_Ciclo_Actual($id_area_salud)
Descripción:	Método para obtener el ciclo actual de un área de salud.

Tabla 7: Descripción de la clase modeloTbAreaSalud.class.php

Nombre: TbCiclo.class.php	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	Tb_Ciclo()
Descripción:	Constructor de la clase
Nombre:	Nombre_Ciclo($ciclo)
Descripción:	Método que determina el nombre del ciclo.
Nombre:	Insertar($ciclo)
Descripción:	Método que se encarga de insertar un ciclo en la base de datos.

Nombre:	Eliminar($id)
Descripción:	Método que se encarga de eliminar un ciclo.
Nombre:	Listar_Ciclos_Por_Tipo($tipo_ciclo,$per_page, $offset)
Descripción:	Método que devuelve los ciclos según el tipo especificado.
Nombre:	Validar_Ciclos_Completados()
Descripción:	Método que se encarga de validar los ciclos que ya se han completado, dando lugar a los ciclos nuevos.

Tabla 8: Descripción de la clase modeloTbCiclo.class.php

3.3 Modelo de datos

Las bases de datos permiten el almacenamiento de un conjunto de información en máquinas, de manera tal que los datos que la conforman puedan ser utilizados en forma fragmentada cuando sea necesario.Los cambios en datos no implican cambio en programas y viceversa, procurando un menor coste de mantenimiento. Posibilita la reducción de la redundancia aumentando así la consistencia de los datos.

A continuación se presenta una vista del modelo de datos del sistema de Vectores.

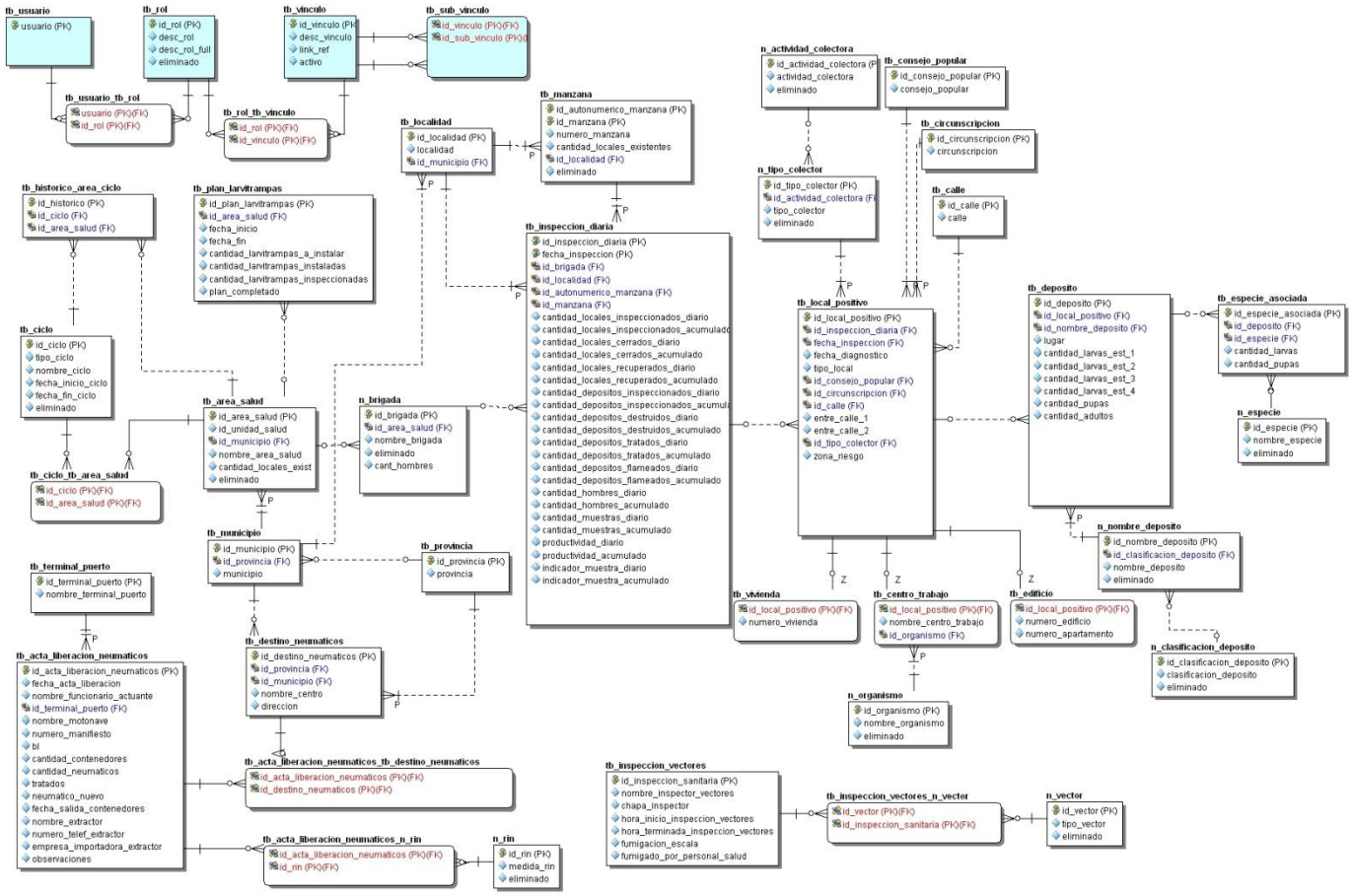

Figura 4: Modelo de datos

3.4 Descripción de las tablas.

Nombre: tb_area_salud		
Descripción: Contiene los datos de las áreas de salud		
Atributo	**Tipo**	**Descripción**
id_area_salud	INTEGER	Llave primaria, guarda el id de la unidad de salud.

id_unidad_salud	INTEGER	Guarda el id de la unidad de salud.
id_municipio	INTEGER	Guarda el id del municipio al que pertenece
id_provincia	INTEGER	Guarda el id de la provincia a la que pertenece
nombre_area_salud	VARCHAR	Guarda el nombre del área de salud formado por el tipo de ciclo y el nombre de la unidad de salud
cantidad_locales_exist	INTEGER	Guarda la cantidad de locales existentes
eliminado	TINYINT	Guarda un valor para conocer su estado: 1 eliminado, 0 no eliminado.

Tabla 9. Descripción de la tablatb_area_salud

Nombre: tb_ciclo		
Descripción: Guarda los datos de los ciclos		
Atributo	**Tipo**	**Descripción**
id_ciclo	INTEGER	Llave primaria de la tabla.
tipo_ciclo	INTEGER	Guarda el tipo de ciclo, que puede ser de dos semanas, mensual, bimestral.
nombre_ciclo	VARCHAR	Guarda el nombre del ciclo.
fecha_inicio_ciclo	DATE	Guarda la fecha de inicio de un ciclo.
fecha_fin_ciclo	DATE	Guarda la fecha de fin de un ciclo.
eliminado	TINYIN	Guarda un valor para conocer su estado: 1 eliminado, 2 no

		eliminado.

Tabla 10. Descripción de la tabla tb_ciclo

Nombre: tb_ciclo_tb_area_salud		
Descripción: Guarda los datos esenciales de las áreas de salud actualizadas. Es el resultado de la unión entre las tablas tb_ciclo y tb_area_salud.		
Atributo	**Tipo**	**Descripción**
id_ciclo	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_ciclo.
id_area_salud	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_area_salud

Tabla 11. Descripción de la tabla tb_ciclo_tb_area_salud

Nombre: tb_inspeccion_diaria		
Descripción: Almacena los datos de las inspecciones diarias		
Atributo	**Tipo**	**Descripción**
id_inspeccion_diaria		Forma parte de la llave primaria de la tabla.
fecha_inspeccion	DATE	Forma parte de la llave primaria. Importada de la tabla tb_inspeccion_diaria.
id_brigada	INTEGER	Forma parte de la llave primaria. Importada de la tabla n_brigada.

id_localidad	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_localidad.
id_autonumerico_manzana	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_manzana.
id_manzana	INTEGER	Forma parte de la llave primaria. Importada de la tabla tb_manzana.
cantidad_locales_inspeccionados_diario	INTEGER	Guarda la cantidad de locales inspeccionados en un día por una brigada.
cantidad_locales_inspeccionados_acumulado	INTEGER	Guarda la cantidad de locales inspeccionados hasta por una brigada hasta un día determinado dentro del ciclo.
cantidad_locales_cerrados_diario	INTEGER	Guarda la cantidad de locales que no pudieron ser inspeccionados por una brigada en un día.
cantidad_locales_cerrados_acumulado	INTEGER	Guarda la cantidad de locales que no han podido ser inspeccionados por determinada brigada hasta un día determinado dentro del ciclo.

cantidad_locales_recuperados_diario	INTEGER	Guarda la cantidad de locales que son recuperados por una brigada en un día.
cantidad_locales_recuperados_acumulado	INTEGER	Guarda la cantidad de locales que han sido recuperados por una brigada hasta un día determinado dentro del ciclo.
cantidad_depositos_inspeccionados_diario	INTEGER	Guarda la cantidad de depósitos que son inspeccionados por una brigada en un día.
cantidad_depositos_inspeccionados_acumulado	INTEGER	Guarda la cantidad de depósitos que han sido inspeccionados por una brigada hasta un día determinado dentro del ciclo.
cantidad_depositos_destruidos_diario	INTEGER	Guarda la cantidad de depósitos destruidos por una brigada en un día.
cantidad_depositos_destruidos_acumulado	INTEGER	Guarda la cantidad de depósitos que han sido destruidos por una brigada hasta un día determinado dentro del ciclo.
cantidad_depositos_tratados_diario	INTEGER	Guarda la cantidad de depósitos que son tratados por una brigada en un día.
cantidad_depositos_tratados_acumulado	INTEGER	Guarda la cantidad de depósitos

		que han sido tratados por una brigada hasta un día determinado dentro del ciclo.
cantidad_depositos_flameados_diario	INTEGER	Guarda la cantidad de depósitos que son flameados por una brigada en un día.
cantidad_depositos_flamedos_acumulado	INTEGER	Guarda la cantidad de depósitos que han sido flameados por una brigada hasta un día determinado dentro del ciclo.
cantidad_hombres_diario	INTEGER	Guarda la cantidad de hombres que tiene una brigada en un día.
cantidad_hombres_acumulado	INTEGER	Guarda la cantidad de hombres que tuvo uno brigada hasta un día determinado dentro del ciclo.
cantidad_muestras_diario	INTEGER	Guarda la cantidad de muestras que son recogidas por una brigada en un día.
cantidad_muestras_acumulado	INTEGER	Guarda la cantidad de muestras que han sido recogidas por una brigada hasta un día determinado dentro del ciclo.
productividad_diario	INTEGER	Guarda la cantidad de locales entre la cantidad de hombres en un día.

productividad_acumulado	INTEGER	Guarda la cantidad de locales entre la cantidad de hombres hasta un día determinado dentro del ciclo.
indicador_muestra_diario	INTEGER	Guarda la cantidad de locales inspeccionados entre la cantidad de muestras colectadas en un día.
indicador_muestra_acumulado	INTEGER	Guarda la cantidad de locales inspeccionados entre la cantidad de muestras colectadas hasta un día determinado dentro del ciclo.

Tabla 12. Descripción de la tabla tb_inspeccion_diaria

Nombre: tb_local_positivo		
Descripción: Se guardan los datos de los locales encontrados positivos		
Atributo	**Tipo**	**Descripción**
id_local_positivo	INTEGER	Forma parte de la llave primaria de la tabla.
id_inspeccion_diaria	INTEGER	Forma parte de la llave primaria de la tabla. Es importada de tb_inspeccion_diaria.
fecha_inspeccion	DATE	Forma parte de la llave primaria de la tabla. Es importada de tb_inspeccion_diaria.
fecha_diagnostico	DATE	Guarda la fecha de diagnóstico del local positivo.

tipo_local	VARCHAR	Especifica el tipo de local encontrado positivo (casa, edificio, terreno baldío)
id_consejo_popular	INTEGER	Guarda el id del consejo popular.
id_circunscripcion	INTEGER	Guarda el id de la circunscripción.
id_calle	INTEGER	Guarda el id de la calle de la dirección.
entre_calle_1	VARCHAR	Guarda el primer número de la entrecalle.
entre_calle_2	VARCHAR	Guarda el segundo número de la entrecalle.
id_tipo_colector	INTEGER	Almacena el id del tipo de colector. Importado de la tabla n_tipo_colector.
zona_riesgo	TINYINT	Especifica si es una zona de riesgo o no.

Tabla 13. Descripción de la tabla tb_local_positivo

3.5 Vista de Implementación (Diagrama de Componentes)

El diagrama de componentes como bien se enuncia muestra un conjunto de componentes y sus relaciones que se utilizan para describir la vista de implementación estática del sistema, en el mismo se evidencia la utilización del patrón MVC.

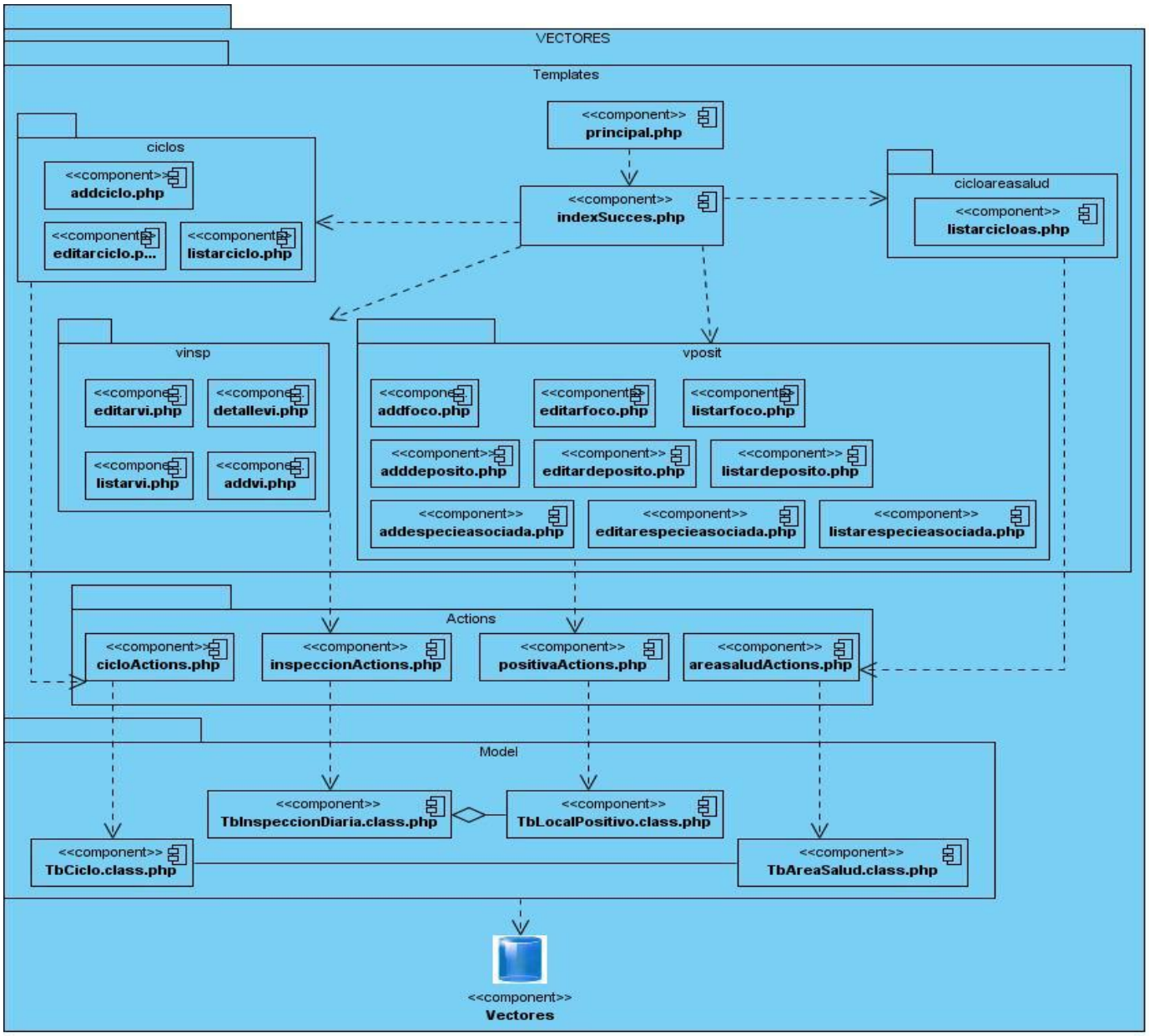

Figura 5: Diagrama de Componentes General

A continuación se muestran con más detalle las diferentes partes por las que está compuesto el sistema: Vistas, Controladoras y Modelos.

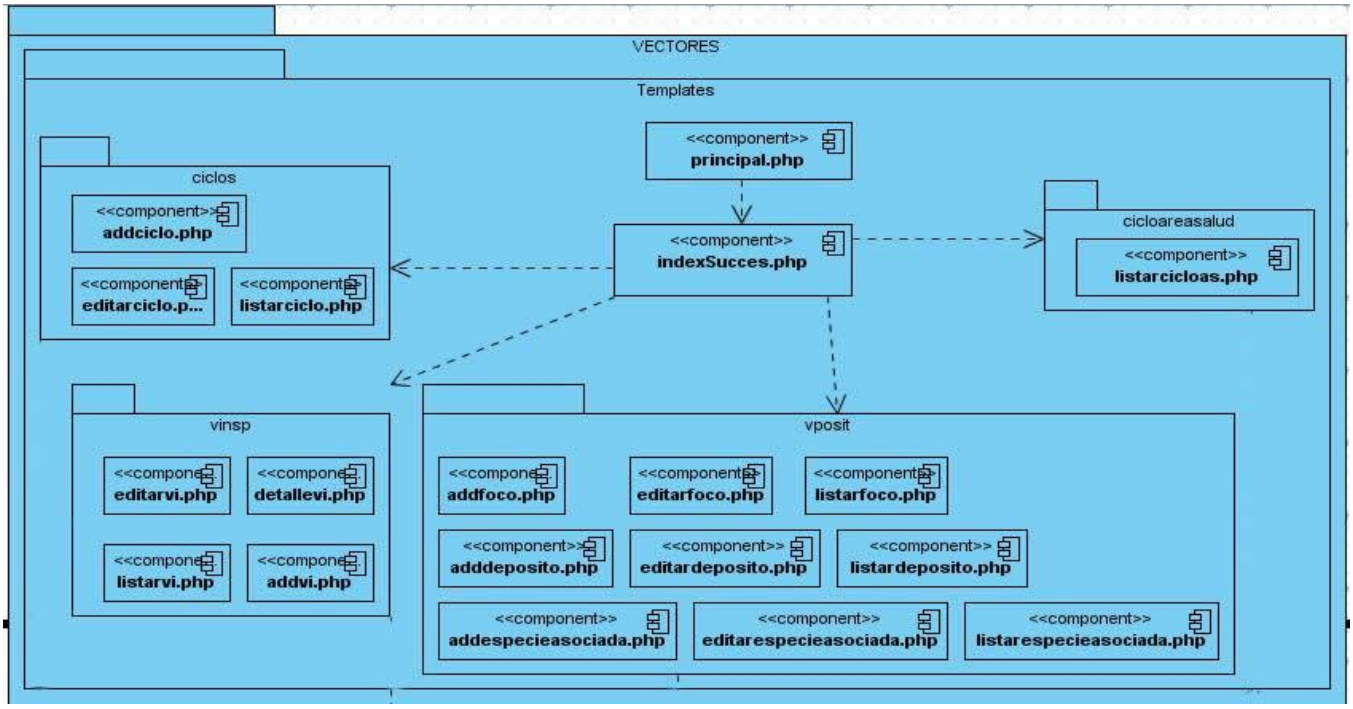

Figura 6: Diagrama para las Clases Vistas de Implementación

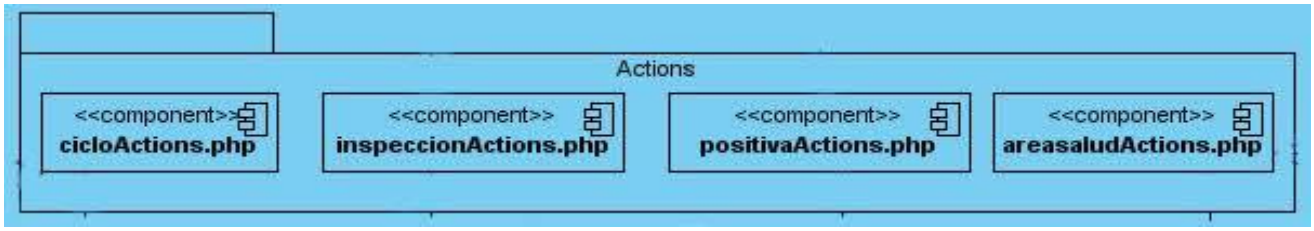

Figura 7: Diagrama para las Clases Controladoras de Implementación

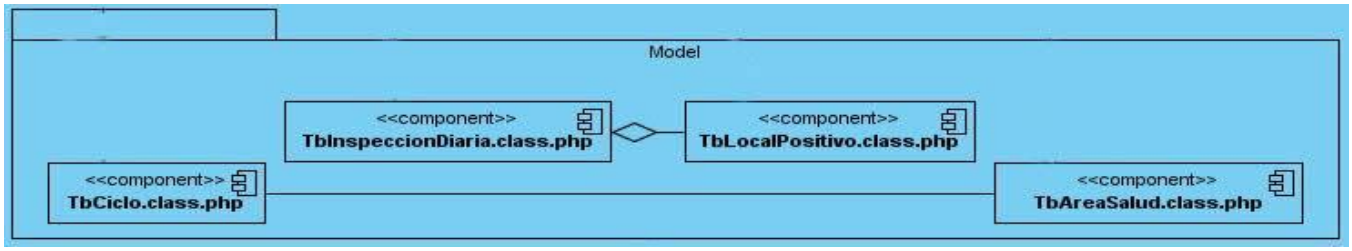

Figura 8: Diagrama para las Clases Modelos de Implementación

Capítulo 3. Descripción y análisis de la solución propuesta

En el presente capítulo se ofrecieron los elementos necesarios para la implementación del sistema de Vectores, así como una valoración crítica del diseño propuesto por el analista, determinándose la utilización de un diseño estándar definido por el Centro Especializado en Soluciones de Informática Médica (CESIM) para todas sus aplicaciones.

Se describieron las clases fundamentales del sistema, y sus funcionalidades, además, se realizó un análisis de la base de datos que permite comprender los mecanismos fundamentales de almacenamiento de la información.

CONCLUSIONES

El desarrollo de las tareas de la investigación posibilitó cumplir con el objetivo general de la misma y arribar a las siguientes conclusiones:

- Luego del análisis de los procesos relacionados con la detección y control de mosquitos Aedes Aegypti, así como del diseño del "Sistema de Detección y Control de Vectores V 1.0.1" se logró el entendimiento de las funcionalidades a implementar y la definición de una estrategia de integración con los módulos: SAAA, RU, RL y RUS del SISalud.
- Durante el proceso de asimilación de la arquitectura propuesta por el Departamento de Sistemas Especializados en Medicina se definió la línea base del desarrollo en torno a los patrones arquitectónicos: MVC, arquitectura en capas y arquitectura basada en componentes, los cuales garantizan la calidad, confiabilidad y reusabilidad del software desarrollado.
- Las herramientas y tecnologías definidas para el desarrollo del sistema están bajo licencia de software libre, lo cual aporta independencia tecnológica a la solución.
- Durante la realización del presente trabajo se hizo necesario, modificar el diseño de la BD propuesto por los analistas; en pos de optimizar todo el mecanismo de almacenamiento y recuperación de información.
- La implementación del Sistema de Detección y Control de Vectores. Versión 1.0.1 le permitirá al país la obtención de una herramienta para la detección y control de los focos de mosquitos Aedes Aegypti, lo que reducirá el riesgo de propagación de enfermedades transmitidas por este vector.

RECOMENDACIONES

Debido a que el alcance de este trabajo comprende solamente las visualizaciones a nivel de Área de Salud los autores recomiendan la implementación de las visualizaciones a los restantes niveles: Municipio, Provincia y Nación.

REFERENCIAS BIBLIOGRÁFICAS

1. Infomed. *Red Telemática de Salud en Cuba.* [En línea] 1999-2000. [Citado el: 9 de Diciembre de 2009.] http://www.sld.cu/sistema_de_salud/ssalud.html.

2. **Martínez Alejandro, Martínez Raúl.** Escuela Politécnica Superior de Albacete. *Guía a Rational Unified Process.* [En línea] 2009. [Citado el: 9 de Febrero de 2010.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf.

3. *Symfony en pocas palabras.* [En línea] [Citado el: 10 de Febrero de 2010.] http://www.librosweb.es/symfony_1_1/capitulo1/symfony_en_pocas_palabras.html.

4. Ídem a la 3. [En línea]

5. Ídem a la 3. [En línea]

6. *Introducción al software libre.* [En línea] [Citado el: 13 de Febrero de 2010.] http://www.atenas.cult.cu/rl/informatica/manuales/sl/introduccion_al_SL/sec-ide.html.

7. **Vegas, Jesús.***El Servidor Web.* [En línea] 21 de Marzo de 2002. http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node20.html.

8. Ciberaula. *Una Introducción a APACHE.* [En línea] 2006. [Citado el: 13 de Febrero de 2010.] http://linux.ciberaula.com/articulo/linux_apache_intro.

9. Ídem a la 8. [En línea]

10. Ídem a la 8. [En línea]

11. **Jacobson, I. y Booch, G. y Rumbaugh, J.***"El Proceso Unificado de Desarrollo de software".* 2000.

12. **Reynoso, Carlos Billy.***Introducción a la Arqutectura de Software.* Buenos Aires : s.n., 2004.

13. Departamento de Lenguajes y Sistemas Informáticos. *Tema 1: Patrones Arquitectónicos.* [En línea] 2009. [Citado el: 9 de Febrero de 2010.] http://www.lsi.us.es/docencia/get.php?id=1891.

14. Arquitectura de programación en 3 capas. [En línea] 2009. [Citado el: 9 de Febrero de 2010.] http://www.elcodigok.com.ar/2007/09/arquitectura-de-programacion-en-3-capas/.

15. **Terreros, Julio Casal.** Desarrollo de Software basado en Componentes. [En línea] [Citado el: 16 de Marzo de 2010.] http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3985/default.aspx.

16. *Ingeniería de Software II. "Arquitectura y Patrones de diseño". Modelo –Vista-Controlador. Conferencia # 2.* s.l. : UCI, 2008_2009.

17. Arquitectura Modelo/Vista/Controlador. Definición de las partes. [En línea] 2009. [Citado el: 9 de Febrero de 2010.] http://www.cica.es/formacion/JavaTut/Apendice/arq_mvc.html.

18. Arquitectura de Software. Capítulo II. Ventajas y desventajas de MVC. [En línea] [Citado el: 9 de Febrero de 2010.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rivera_l_a/capitulo2.pdf.

19. Librosweb. [En línea] [Citado el: 25 de Marzo de 2010.] http://www.librosweb.es/symfony/capitulo6/seguridad_de_la_accion.html.

20. *Fundamentos de Ingeniería de Software.* [En línea] [Citado el: 18 de Marzo de 2010.] http://www.inf.utfsm.cl/~visconti/ili236/.../15-Implementacion.pdf.

BIBLIOGRAFÍA

1. Arquitectura de programación en 3 capas. [En línea] 2009. [Citado el: 9 de Febrero de 2010.] http://www.elcodigok.com.ar/2007/09/arquitectura-de-programacion-en-3-capas/.

2. Arquitectura de Software. Capítulo II. Ventajas y desventajas de MVC. [En línea] [Citado el: 9 de Febrero de 2010.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rivera_l_a/capitulo2.pdf.

3. Arquitectura Modelo/Vista/Controlador. Definición de las partes. [En línea] 2009. [Citado el: 9 de Febrero de 2010.] http://www.cica.es/formacion/JavaTut/Apendice/arq_mvc.html.

4. Atención Primaria de Salud. *Actualización del Programa Nacional de Control Sanitario Internacional.* [En línea] 1999-2002. [Citado el: 2 de Diciembre de 2009.] http://aps.sld.cu/bvs/materiales/programa/procsi.html.

5. **Avila Cruz, Victor Rolando.** Encuentro Virtual de Neurocirugía. *Medicina y Computación.* [En línea] 1999. [Citado el: 10 de Diciembre de 2009.] http://neuroc99.sld.cu/text/medicinacomputacion.html.

6. Ciberaula. *Una Introducción a APACHE.* [En línea] 2006. [Citado el: 13 de Febrero de 2010.] http://linux.ciberaula.com/articulo/linux_apache_intro.

7. Departamento de Lenguajes y Sistemas Informáticos. *Tema 1: Patrones Arquitectónicos.* [En línea] 2009. [Citado el: 9 de Febrero de 2010.] http://www.lsi.us.es/docencia/get.php?id=1891.

8. *Fundamentos de Ingeniería de Software.* [En línea] [Citado el: 18 de Marzo de 2010.] http://www.inf.utfsm.cl/~visconti/ili236/.../15-Implementacion.pdf.

9. **Hernández Hernández, Daybert y Guzmán Pérez, Malena.***Implementación del Módulo Vectores del Sistema Control Sanitario Internacional.* Ciudad de la Habana : s.n., 200-2008.

10. Infomed. *Red Telemática de Salud en Cuba.* [En línea] 1999-2000. [Citado el: 9 de Diciembre de 2009.] http://www.sld.cu/sistema_de_salud/ssalud.htlm.

11. *Ingeniería de Software II. "Arquitectura y Patrones de diseño". Modelo –Vista-Controlador. Conferencia # 2.* s.l. : UCI, 2008_2009.

12. *Introducción al software libre.* [En línea] [Citado el: 13 de Febrero de 2010.] http://www.atenas.cult.cu/rl/informatica/manuales/sl/introduccion_al_SL/sec-ide.html.

13. **Jacobson, I. y Booch, G. y Rumbaugh, J.***"El Proceso Unificado de Desarrollo de software".* 2000.

14. Librosweb. [En línea] [Citado el: 25 de Marzo de 2010.] http://www.librosweb.es/symfony/capitulo6/seguridad_de_la_accion.html.

15. Maestros del Web. *¿Qué es Javascript?* [En línea] 3 de Julio de 2007. [Citado el: 10 de Febrero de 2010.] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-Javascript/.

16. Maestros del Web. *¿Qué son las bases de datos?* [En línea] [Citado el: 20 de Abril de 2010.] http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/.

17. **Mañas, José A.** Laboratorio de Programación. *Prueba de Programas.* [En línea] 16 de Marzo de 1994. [Citado el: 14 de Mayo de 2010.] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s7.

18. **Martínez Alejandro, Martínez Raúl.** Escuela Politécnica Superior de Albacete. *Guía a Rational Unified Process.* [En línea] 2009. [Citado el: 9 de Febrero de 2010.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf.

19. Master Magazine. *Definición de Base de datos.* [En línea] 2004. [Citado el: 20 de Abril de 2010.] http://www.mastermagazine.info/termino/4012.php.

20. Ministerio de la Informática y las Comunicaciones de Cuba. [En línea] 2002. [Citado el: 9 de Diciembre de 2009.] http://www.mic.gov.cu.

21. **Reynoso, Carlos Billy.***Introducción a la Arqutectura de Software.* Buenos Aires : s.n., 2004.

22. **Rosario, Jimmy.** Observatorio para la CiberSociedad. *La Tecnología de la Información y la Comunicación (TIC). Su uso como Herramienta para el Fortalecimiento y el Desarrollo de la Educación*

Virtual. [En línea] 2005. [Citado el: 9 de Diciembre de 2009.] http://www.cibersociedad.net/archivo/articulo.php?art=218.

23. *Sistema de Gestión de Bases de Datos.* [En línea] Agosto de 2004. [Citado el: 9 de Febrero de 2010.] http://www.igac.gov.co:8080/igac_web/UserFiles/File/ciaf/TutorialSIG_2005_26_02/paginas/ctr_sistemasdegestiondebasededatos.html.

24. *Symfony en pocas palabras.* [En línea] [Citado el: 10 de Febrero de 2010.] http://www.librosweb.es/symfony_1_1/capitulo1/symfony_en_pocas_palabras.html.

25. **Terreros, Julio Casal.** Desarrollo de Software basado en Componentes. [En línea] [Citado el: 16 de Marzo de 2010.] http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3985/default.aspx.

26. **Vegas, Jesús.***El Servidor Web.* [En línea] 21 de Marzo de 2002. http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node20.html.

ANEXOS

Anexo 1 Pantalla de Autenticación del Sistema de Detección y Control de Vectores.

Figura 9: Prototipo de Interfaz de Usuario: Autenticación

Anexo 2 Pantalla de entrada al sistema

Figura 10: Prototipo de Interfaz de Usuario: Administrador Nacional

Anexo 3 Pantalla que muestra la funcionalidad Gestionar Ciclo

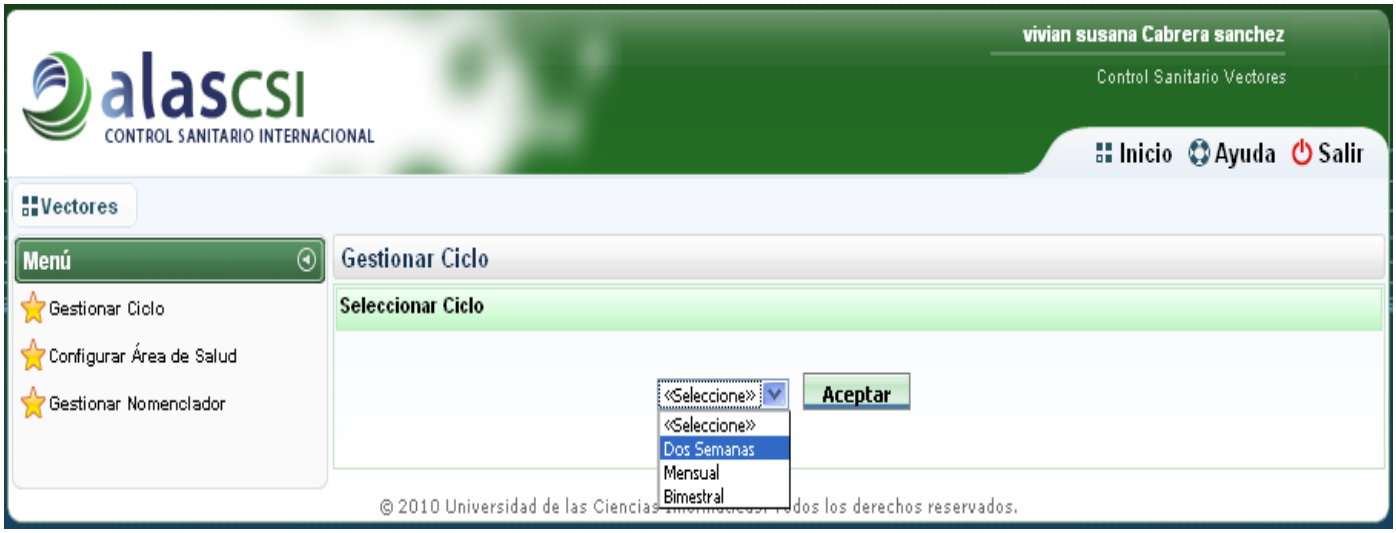

Figura 11: Prototipo de Interfaz de Usuario: Seleccionar Ciclo

Anexo 4 Pantalla que muestra la funcionalidad Listar Ciclos

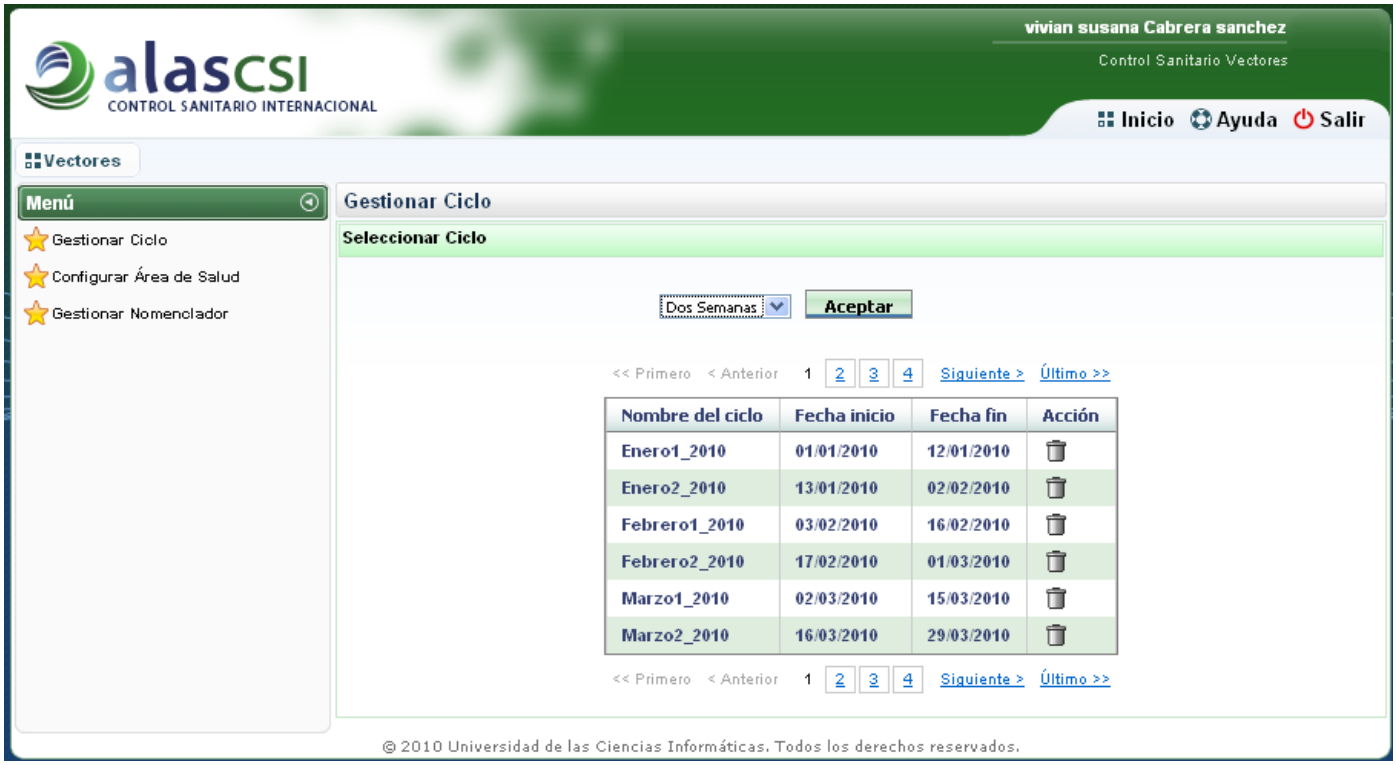

Figura 12: Prototipo de Interfaz de Usuario: Listado de Ciclos (Dos semanas)

Anexo 5 Pantalla que muestra la funcionalidad Insertar Ciclo

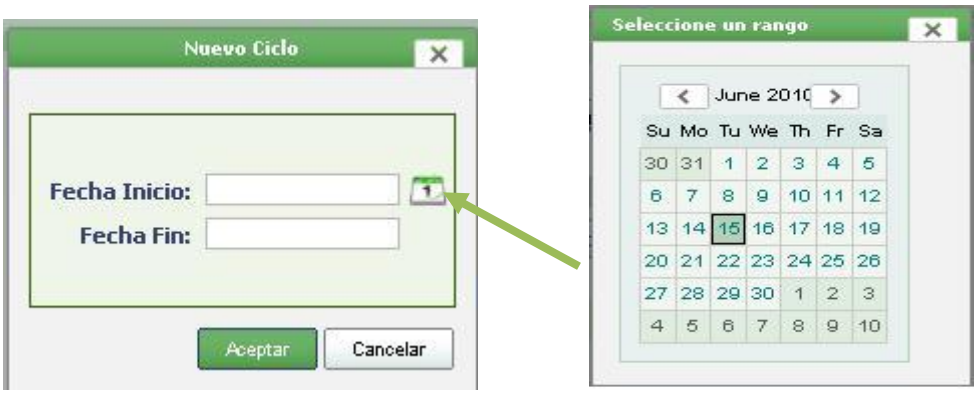

Figura 13: Prototipo de Interfaz de Usuario: Insertar Ciclo

Anexo 6 Pantalla que muestra la funcionalidad Eliminar Ciclo

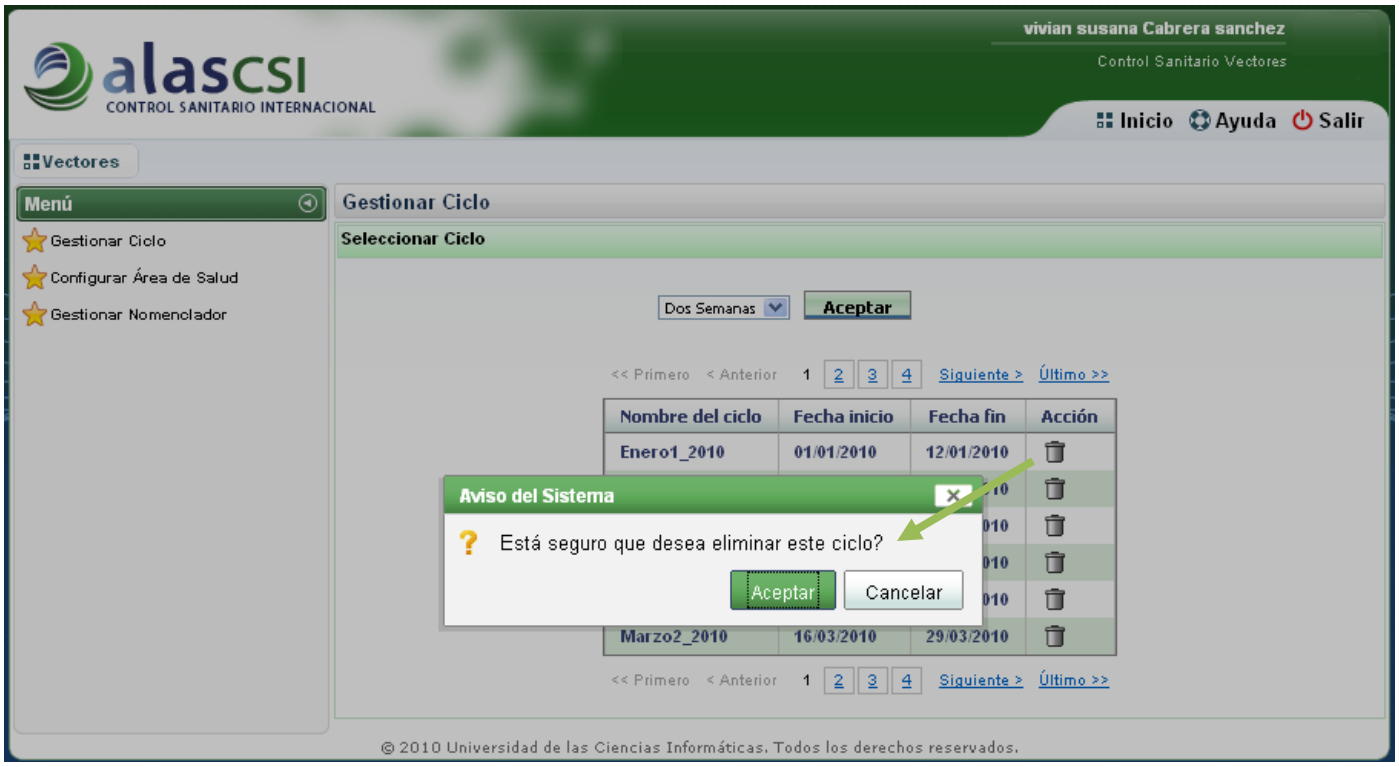

Figura 14: Prototipo de Interfaz de Usuario: Eliminar Ciclo

GLOSARIO

Aedes Aegypti: Aedes Aegypti Linnaeus, 1762, es un mosquito cuyo origen se ubica geográficamente en la Región Etiópica (África), que nuclea la mayor cantidad de especies del Subgénero StegomyiaTheobald, 1901, al cual este culícido pertenece.

Aplicación: Programa informático que proporciona servicios de alto nivel al usuario, generalmente utilizando otros programas más básicos que se sitúan por debajo.

Base de datos: Conjunto de datos organizados entre los cuales existe una correlación y que están almacenados con criterios independientes de los programas que los utilizan. La filosofía de las bases de datos es la de almacenar grandes cantidades de datos de una manera no redundante y que permita las posibles consultas de acuerdo a los derechos de acceso.

Dengue: Enfermedad epidémica caracterizada por fiebre, dolores en los miembros y un exantema seguido de descamación.

Diagrama: Presentación gráfica de un conjunto de elementos y sus relaciones.

Framework: Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Hospederos: Personas que portan el germen de una enfermedad y son transmisoras de ella.

IDE (Integrated Development Environment: Entorno de desarrollo integrado): Editor de código que sirve para depurar y facilitar las diferentes tareas necesarias en el desarrollo de cualquier tipo de aplicación.

INFOMED: Nombre que identifica a la primera red electrónica cubana de información para la salud y surgió como parte de un proyecto del Centro Nacional de Información de Ciencias Médicas de Cuba. Es el Portal de Salud Cubano y la red de personas e instituciones que comparten el propósito de facilitar el acceso a la información de salud en Cuba.

Informatización: Aplicación de sistemas y equipos informáticos al tratamiento de información.

Librerías: Constituyen el conjunto de procedimientos y funciones agrupados en un archivo con el fin de que sean utilizadas por otros programas.

Tecnología de la información y la comunicación (TIC): Conjunto de avances tecnológicos que proporcionan la informática, las telecomunicaciones y las tecnologías audiovisuales, que comprenden los desarrollos relacionados con los ordenadores, Internet, la telefonía, las aplicaciones multimedia y la realidad virtual. Estas tecnologías básicamente proporcionan información, herramientas para su proceso y canales de comunicación.

Vectores: Especie portadora o huésped intermedio de un parásito o virus que transmite el germen de una enfermedad a otro huésped. Los vectores por excelencia son las cucarachas, ratones, mosquitos y moscas.

WEB (WWW): Red de documentos HTML intercomunicados y distribuidos entre servidores del mundo entero.