Universidad de las Ciencias Informáticas

Facultad 7

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Implementación del Sistema Integral de Vigilancia Epidemiológica. Versión 1.0.1

Autores: Mairelys Sánchez Boudet

Amaury Zada Rivas

Tutores: Ing. Ricardo Collada de la Rosa

Ing. Débora González Tolmo

Ciudad de La Habana, Julio de 2010

"Año 52 de la Revolución"

DATOS DE CONTACTO

Ing. Ricardo Collada de la Rosa. Graduado de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el 2008. Instructor recién graduado en adiestramiento. Durante su trabajo como profesor ha impartido la asignatura de Segundo Perfil, la cual imparte actualmente.

En la vinculación con la producción pertenece al Departamento de Sistemas Especializados en Medicina del Centro Especializado en Soluciones de Informática Médica y específicamente se desempeña como Arquitecto Principal del Centro Especializado en Soluciones de Informática Médica.

Correo electrónico: rcollada@uci.cu

Ing. Débora González Tolmo. Graduada de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el 2009. Instructor recién graduado en adiestramiento. Actualmente imparte PP2 – PP5 en el Departamento Sistemas Especializados en Medicina.

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 Teleconsulta donde se desempeña como analista principal.

Correo electrónico: dtolmo@uci.cu

RESUMEN

Cuba cuenta con amplias relaciones internacionales, además una de las actividades más importantes de su economía es el turismo. Es por ello que día a día entran en la isla un número variado de personas de varios países. Por este motivo es de gran importancia un buen control epidemiológico en el país, con el objetivo de que enfermedades transmisibles no se propaguen en el territorio. Además el clima cubano posee diversas condiciones que favorecen el desarrollo y evolución de enfermedades tropicales capaces de dañar en gran medida la salud de la población.

Por estas causas el Estado Cubano de conjunto con el Ministerio de Salud Pública (MINSAP) crearon el programa de Control Sanitario Internacional. Este proyecto tiene como objetivo evitar la propagación de enfermedades e impedir afectaciones a la población en caso de existir alguna situación de riesgo epidemiológico.

El presente trabajo tiene como objetivo realizar la implementación de una nueva versión del sistema Higiene y Epidemiología que cumpla con todas las funcionalidades identificadas por el cliente y esté en correspondencia con la arquitectura definida por el grupo de arquitectura del Centro de Informática Médica (CESIM) y la Facultad 7 para el Departamento de Sistemas Especializados. Para desarrollar el sistema se seleccionó como metodología el Proceso Unificado de Desarrollo (RUP), que hace uso del Lenguaje Unificado de Modelado (UML). Se utilizó la herramienta Visual Paradigm 6.4 para el modelado del sistema y se trabajó sobre el framework Symfony 1.4.3 para la implementación de la aplicación.

PALABRAS CLAVE

Control Sanitario Internacional, Higiene y Epidemiologia, Arquitectura.

TABLA DE CONTENIDOS

INTRODUCCIÓN

En Cuba después del triunfo revolucionario, el 1ro de enero de 1959, se estrecharon lazos diplomáticos, culturales, de amistad y colaboración económica y social con disimiles países de América Latina y el resto del mundo. Esto produjo un aumento considerable de la entrada al país de personal proveniente de regiones endémicas de enfermedades transmisibles, fundamentalmente de países subdesarrollados.

En estos países las condiciones económicas y sociales son críticas debido a la crisis económica y social existente, lo que provoca que sus sistemas de salud y control sanitario sean poco eficientes y deteriorados. Todo ello unido al hecho de que el clima cálido de la isla posee condiciones favorables para que estas enfermedades se desarrollen y reproduzcan con facilidad, provocó un incremento del riesgo de transmisión y propagación de enfermedades tropicales en el territorio.

Desde el año 1899 ya habían sido establecidas en Cuba las primeras regulaciones sanitarias de fronteras, teniendo a partir de 1902 las orientaciones del Dr. Carlos J. Finlay. En 1980 se pone en vigor el primer Programa Nacional de Control Sanitario Internacional (CSI) que preveía acciones de control a realizar por una parte de la red de servicios de salud. En los años siguientes se trabajó fuertemente en pos de refinar la estructura y funcionamiento de este programa con el objetivo de establecer una vigilancia epidemiológica óptima, capaz de detectar cualquier síntoma de enfermedad.

El programa de CSI actualmente está conformado por 3 subprogramas bien estructurados: Salud Ambiental, Control de Vectores y Vigilancia Epidemiológica. Este último es el objeto central de esta investigación debido a que constituyen una prioridad fundamental del Ministerio de Salud Pública (MINSAP). Es importante aclarar que estos programas están aún en desarrollo, y que parte de dicho desarrollo está llevándose a cabo en la Universidad de las Ciencias Informáticas (UCI).

Un suceso importante es que los procesos de control y vigilancia epidemiológica actualmente se realizan a través de redes de información establecidas por el MINSAP que van desde el Área de Salud hasta el propio Ministerio. Estas redes posibilitan el conocimiento de información significativa referente al estado epidemiológico del país en todos los niveles.

La información se gestiona mediante conversaciones telefónicas y correo electrónico, lo que conlleva a que el sistema esté expuesto a riesgos a causa de errores humanos. Esta situación puede tener como consecuencia que no siempre se manejan datos totalmente actualizados y por ende se pierda información valiosa. Es también relevante el hecho de que para almacenar datos históricos de pacientes con enfermedades tropicales y de gran significación se utilicen libros Microsoft Office Excel.

En el curso 2007-2008 en la Universidad de las Ciencias Informáticas se desarrolló una primera versión de un sistema para la vigilancia epidemiológica. El mismo permitía viabilizar la gestión de la información relacionada con la vigilancia epidemiológica al viajero y las enfermedades tropicales en Cuba (Sistema Higiene y Epidemiología v 1.0).

El sistema se desarrolló sobre el framework CodeIgniter, que a pesar de ser un marco de trabajo sencillo de utilizar, carece de un conjunto de funcionalidades que pudieran limitar el desarrollo óptimo de nuevas versiones del sistema actual y presenta una interfaz gráfica poco amigable y segura.

Además esta aplicación no está en correspondencia con el lineamiento de la arquitectura propuesta por el grupo de arquitectura del Centro de Informática Médica (CESIM) y la Facultad 7. Por esta razón en el curso 2008-2009 se realizó un nuevo análisis y diseño de este software, el cual no se implementó. De lo antes expuesto se infiere la **situación problémica.**

A raíz de la situación anterior el **problema científico** es: ¿Cómo obtener un sistema funcional a partir del diseño realizado para viabilizar el proceso de gestión de la información relacionada con la vigilancia epidemiológica al viajero y las enfermedades tropicales en Cuba?

Por tanto el **objeto de estudio** se enmarca en el proceso de gestión de la información referente a la vigilancia epidemiológica, delimitando como **campo de acción** el proceso de gestión de la información referente a la vigilancia epidemiológica al viajero y al control de las enfermedades tropicales en Cuba.

Para dar solución al problema planteado, se determinó como **objetivo general** implementar la versión 1.0.1 del Sistema Integral de Vigilancia Epidemiológica.

Con el fin de llevar a cabo estos posibles resultados y lograr cumplir el objetivo trazado, se realizarán las siguientes **tareas de la investigación:**

1. Realizar un análisis acerca de los procesos de negocio asociados a la vigilancia epidemiológica al viajero y a las enfermedades tropicales en Cuba para el Sistema Integral de Vigilancia Epidemiológica.
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 a la vigilancia epidemiológica al viajero y a las enfermedades tropicales en Cuba para el Sistema Integral de Vigilancia Epidemiológica.
4. Realizar la actualización del modelo de datos.
5. Realizar la implementación de los procesos de negocio asociados a la vigilancia epidemiológica al viajero y a las enfermedades tropicales en Cuba para el Sistema Integral de Vigilancia Epidemiológica utilizando los patrones establecidos en el Análisis y Diseño.

Para dar cumplimiento a estas tareas de manera satisfactoria el presente trabajo de diploma estará estructurado en tres capítulos:

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, por lo que se exponen los requisitos no funcionales del mismo, se efectúa 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 explicación de las técnicas de validación y por último una descripción de la vista de implementación.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA.

En este capítulo se realizará un análisis y descripción de las herramientas definidas para la implementación del sistema propuesto, las cuales fueron previamente definidas en el documento de arquitectura de la facultad 7 de la Universidad de la Ciencias Informáticas. Además se llevará a cabo una valoración de las tecnologías y metodologías a emplear para el desarrollo del mismo.

1.1 Sistema Higiene y Epidemiología [Versión 1.0]

En el curso 2007-2008 el grupo de Sistemas Especializados en Medicina de la Universidad de las Ciencias Informáticas, implementó el sistema Control Sanitario Internacional (CSI). Entre sus módulos contiene el subsistema de Higiene y Epidemiología, que tiene dos partes fundamentales: el control de enfermedades tropicales, esta primera versión solo se limitó al control de Dengue mediante la vigilancia a pacientes con síntomas febriles inespecíficos, y el seguimiento a viajeros con el objetivo de evitar la introducción y propagación de estas enfermedades en la región.

Además, recoge una serie de datos necesarios y permite al usuario estar actualizado en cada momento de la situación epidemiológica en el país. El software gestiona la información en la base: aeropuertos, hospitales y unidades de salud, los que muestran la información más actualizada a las autoridades sanitarias en materia de control epidemiológico en los niveles municipal, provincial y nacional, según los permisos dentro del sistema.

1.2 Procesos de negocio.

Dentro de los procesos más significativos que se detectaron en el negocio estuvieron los siguientes:

- **Detectar febriles inespecíficos:**

Todo el proceso de vigilancia epidemiológica inicia una vez que al paciente se le detecta en el Área de Salud síntomas febriles inespecíficos, es decir, que no se haya detectado la causa que provoca la fiebre.

Luego se le orienta realizarse los exámenes correspondientes para detectar si tiene alguna de las enfermedades que se encuentran sujetas al Control Sanitario Internacional.

Cuando un paciente acude a consulta con síntomas febriles inespecíficos se le orienta que se realice un examen de sangre al sexto día de presentar los síntomas en su Área de Salud. De la misma se envía una muestra al Laboratorio Provincial de Higiene y Epidemiología (LPHE) y se utiliza dicha muestra para realizar la prueba de IgmSuma. Si el resultado de esta es positivo o borderline (en el límite) el paciente pasa a ser un caso sospechoso de Dengue.

- **Realizar pruebas:**

Una muestra de la sangre extraída al paciente es enviada al laboratorio del Instituto de Medicina Tropical Pedro Kourí (IPK) para realizar la prueba de Elisa. Una vez que están los resultados de la misma, si es positivo o borderline, el paciente pasa a ser un caso probable de Dengue y se le orienta la prueba de IGG. A los 21 días se le toma al paciente una segunda muestra de sangre que es enviada al LPHE donde se prepara y se envía al IPK para analizarla, si el resultado de esta prueba es positivo o indeterminado, el paciente pasa a ser un caso confirmado de Dengue y se toman las medidas pertinentes para su tratamiento.

Otra de las pruebas que se realiza es el PCR/Aislamiento que se les puede hacer a los pacientes sin necesidad de que pasen por el proceso anterior o en el transcurso del mismo, si el resultado de la misma es positivo, el paciente pasa directamente a ser un caso confirmado de Dengue.

- **Recopilación de datos del viajero:**

Un aspecto fundamental a tener en cuenta es el arribo de viajeros cubanos, estudiantes extranjeros residentes en Cuba o personal diplomático residente en el país, que llegan a Cuba procedentes de zonas endémicas de dengue, los mismos son controlados como parte de la vigilancia epidemiológica del Programa de Control Sanitario Internacional, una vez que estos arriban al aeropuerto son atendidos por el personal de salud y si algunos son detectados con síntomas febriles inespecíficos se trasladan al IPK, donde pasan por el mismo proceso que los pacientes detectados en el Área de Salud.

En caso de que en el aeropuerto no se le detecten síntomas febriles inespecíficos deben ser visitados por el médico de la familia en las primeras 72 horas después de haber llegado al Área de Salud. El médico de la familia debe mantener una vigilancia estricta de los mismos por la posibilidad de haber estado en

contacto con la enfermedad y ser portadores de la misma, por lo que se convierte su presencia en un factor de riesgo local que incrementa la probabilidad de existencia de un brote de Dengue en la zona.

1.3 Metodología de Desarrollo.

En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. Las metodologías de desarrollo son un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Además son importantes para el desarrollo del software por que ofrecen documentación formal. Son útiles para garantizar la eficacia según los requisitos iniciales y la eficiencia al minimizar las pérdidas de tiempo en el proceso de generación de software.

1.3.1 Proceso Unificado de Desarrollo (RUP).

El Proceso Unificado de Desarrollo es un proceso de ingeniería del software. Proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo. Su propósito es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costos y calendario predecibles. (1)

Es una metodología basada en un pequeño grupo de principios claves: el equipo de un proyecto de software debe planificar el desarrollo; debe conocer hacia donde se dirige; debe documentar el proyecto de una manera perdurable y extensible. Constituye una infraestructura flexible de desarrollo de software que proporciona prácticas recomendadas, probadas y una arquitectura configurable. (2)

Además incorpora el concepto de "mejores prácticas" para la ingeniería de software, definido por tres características fundamentales: (3)

1. Dirigido por Casos de Uso: el desarrollo está dirigido a satisfacer las necesidades de los usuarios del sistema expresadas en Casos de Uso.

2. Centrado en la arquitectura: el desarrollo se centra en una arquitectura bien definida, con relaciones claras entre sus distintos componentes.

3. Iterativo e Incremental: el problema y la solución se organizan en pequeñas piezas, de manera que cada iteración se dirige específicamente al desarrollo de un conjunto de ellas. Cada iteración se construye sobre la base creada por las iteraciones anteriores, agregándole capacidades al sistema.

La metodología RUP divide en 4 fases el desarrollo del software:

- **Inicio**: el objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración**: en esta etapa el objetivo es determinar la arquitectura óptima.
- **Construcción**: en esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.
- **Transición**: el objetivo es llegar a obtener una versión del proyecto lista para instalarse.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. (4)

1.4 Lenguaje Unificado de Modelado (UML 2.0).

UML 2.0 es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema orientado a objetos.

Además ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado de Desarrollo) pero no especifica en sí mismo qué metodología o proceso usar. (5)

Su utilización es independiente del lenguaje de programación y de las características de los proyectos. Se diseñó por la necesidad de modelar cualquier tipo de proyecto, tanto informático como de arquitectura, o

de cualquier otra rama. Posibilita la modificación de todos sus miembros mediante estereotipos y restricciones.

Características:

- Es una especificación basada en Booch, OMT y OOSE, de allí sus principios.
- Divide cada proyecto en un número de diagramas que representan las distintas vistas del proyecto y juntos representan la arquitectura del mismo.
- Permite describir un sistema en diferentes niveles de abstracción.

1.5 Herramienta CASE.

Las herramientas CASE (**C**omputer **A**ided **S**oftware **E**ngineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras (6).

1.5.1 Visual Paradigm 6.4

Herramienta **CASE** que utiliza “UML” como lenguaje de modelado, con el uso del acercamiento orientado al objeto. Esta herramienta 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. El análisis de las transiciones de diseño y, a continuación, a la aplicación se encuentran perfectamente integradas en la herramienta CASE, reduciendo así significativamente los esfuerzos en todas las etapas del ciclo de vida de desarrollo de software. (7)

Este software de modelado ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, generar código inverso, código desde diagramas y documentación.

Características del Visual Paradigm 6.4:

- Entorno de creación de diagramas para UML 2.0.
- Diseño centrado en Casos de Uso y enfocado al negocio que genera un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de integrarse en los principales IDEs. (Integrated Development Environment)
- Disponibilidad en múltiples plataformas.
- Exporta código de aproximadamente 10 lenguajes de programación incluyendo el PHP.

1.6 Lenguajes de programación

Idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.

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 PHP5

Lenguaje de programación de estilo clásico, es decir, es un lenguaje de programación con variables, sentencias condicionales, bucles y funciones. Es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado al usar especialmente para desarrollo web y puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida.

Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. Al ser ejecutado en el servidor, nos permite acceder a los recursos que tenga el servidor como por ejemplo podría ser una base de datos. El programa PHP es ejecutado en el servidor y el resultado enviado al navegador. El resultado es normalmente una página HTML pero igualmente podría ser una página WML.

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador, pero sin embargo para que sus páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP. Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

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 muy fácil de aprender para quien ya conoce lenguajes similares como el C++ o Java, pero, dada su simplicidad sintáctica y su manejabilidad, no es tampoco difícil para quien se acerca por primera vez a este lenguaje. Sin embargo, esto puede ser un arma de doble filo porque la simplicidad se basa en una disponibilidad de objetos limitada, por lo que algunos procedimientos, aparentemente muy sencillos, requieren script bastante complejos.

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. (8)

Este lenguaje 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 y cargas de páginas, entre otros. (9)

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.

Los objetivos principales que persigue un Framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones.

Un framework Web, por tanto, podemos definirlo como un conjunto de componentes que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web. (10)

1.7.1 Symfony 1.4.3

Framework PHP que facilita el desarrollo de las aplicaciones web. Separa la lógica de negocio, del 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.

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, Linux, etc.) como en plataformas Windows. (11)

Características generales de Symfony:

- Fácil de instalar y configurar en sistemas Windows, Mac y Linux.
- Funciona con todas las bases de datos comunes (MySQL, PostgreSQL, SQLite, Oracle, MS SQL Server).
- Basado en la premisa de "convenir en vez de configurar"**,** en la que el desarrollador solo debe configurar aquello que no es convencional.
- Preparado para aplicaciones empresariales, ya que se puede adaptar con facilidad a las políticas y arquitecturas propias de cada empresa u organización.
- Flexible hasta cualquier límite y extensible mediante un completo mecanismo de plugins.
- 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.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.

1.7.2 Librería YUI 2.6

La librería **Yahoo! User Interface** es un conjunto de utilidades y controles escritos en JavaScript que se utilizan para crear aplicaciones web dinámicas complejas. Además, la librería YUI incluye varias utilidades relacionadas con CSS, por lo que también se considera una librería CSS. (12)

Provee una gran cantidad controles de interfaz de usuario lo que permite la homogenización en las interfaces visuales de los productos de software. Es un extenso catálogo con cerca de 300 códigos

JavaScript y CSS de uso libremente para proyectos web, ya sean personales o comerciales. Tiene ejemplos, código fuente y documentación relacionada para poder utilizar y personalizar todos los scripts.

Los scripts están divididos en categorías según su utilidad: animación, autocompletar, cuentagotas, treeview, DOM, CSS, entre otros. (13) En total, se encuentran hasta 40 categorías distintas con varios scripts, cada una permite utilizarlas de una forma básica o más avanzada. YUI se ejecuta correctamente en todos los navegadores modernos e incluso en algún navegador obsoleto.

1.8 Entornos Integrados de Desarrollo (IDEs)

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

1.8.1 NetBeans 6.8

NetBeans es una herramienta de código abierto escrito completamente en Java para programadores. Permite escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Es un producto libre y gratuito sin restricciones de uso.

Soporta el desarrollo de todos los tipos de aplicación Java. Todas las funciones de NetBeans son provistas por módulos. Cada módulo provee una función bien definida, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones. Además contiene todos los módulos necesarios para el desarrollo de aplicaciones Java en una sola descarga, permitiéndole al usuario comenzar a trabajar inmediatamente.

1.8.2 Adobe Dreamweaver 8.0

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. Presenta validaciones

dinámicas en distintos navegadores, es decir que las etiquetas HTML y las reglas CSS, siempre estarán legibles, siendo éstas compatibles con cualquier navegador.

Es el programa de este tipo más utilizado en el sector del diseño y la programación web, por sus funcionalidades. 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 (como la de insertar un hipervínculo, una imagen o añadir un comportamiento) 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, vayas dónde vayas, siempre estarán legibles, siendo éstas compatibles con cualquier navegador del mercado.

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

Programa que está diseñado para transferir hipertextos, páginas web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

Los servidores web permiten alojar sitios web y brindar esta información de forma pública o restringida a los clientes. Estas computadoras deben de contar con altos requerimientos de velocidad, memoria y espacio en disco duro para ser capaces de soportar todas las peticiones que se les hace simultáneamente y no colapsar por necesidad de recursos.

1.9.1 Servidor Web Apache 2.2

Apache es un software libre, servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras. Presenta, entre otras características, mensajes de error altamente

configurables, bases de datos de autenticación y negociado de contenido. Ha sido desde Abril de 1996 el servidor HTTP más usado.

Características de Apache

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita y de código fuente abierto.
- Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a éste, y están ahí para instalarlos cuando se necesite.
- Trabaja con gran cantidad de Perl, PHP y otros lenguajes de script. También trabaja con Java y páginas javascrip. Teniendo todo el soporte que se necesita para tener páginas dinámicas.
- 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. Apache permite la creación de ficheros de log a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en el servidor. (14)

1.10 Sistema Gestor de Base de Datos

Los sistemas de gestión de bases de datos o SGBD son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. 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. (15)

1.10.1 PostgreSQL 8.3

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD. Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es

manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo.

Es un potente motor de bases de datos, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. Es más completo que MySQL ya que permite métodos almacenados, restricciones de integridad y vistas.

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 como:

Aproximación de los datos a un modelo objeto-relacional, y 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.

Este motor tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido; además de poseer Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), tecnología usada para evitar bloqueos innecesarios. (16)

Con en desarrollo de este capítulo realizó un análisis de las herramientas a utilizar en la implementación de la 2da versión del sistema de Vigilancia Epidemiológica. Se demostró que con el uso de la metodología y tecnologías propuestas se realizará un mejor diseño del sistema Higiene y Epidemiología dado el avance y desarrollo internacional que adquieren diariamente, además de estar en correspondencia con la arquitectura definida. Se analizó como con el uso de Symfony y YUI, se logrará un manejo de datos más eficiente, rápido, elegante, seguro, configurable, dadas las facilidades de este framework y librerías respectivamente. Se logró ajustar el desarrollo del sistema planteado a las técnicas propuestas en la arquitectura definida por la Facultad 7 para el Área Temática Sistemas Especializados.

CAPÍTULO 2. ELEMENTOS DE ARQUITECTURA

En este capítulo se realizará una descripción general de la arquitectura que sigue el desarrollo del software. Además se mostrarán los recursos utilizados para diseñar e implementar las funcionalidades del sistema propuesto así como las estrategias de integración y los estándares de codificación que se utilizarán para el desarrollo del mismo.

2.1 Requisitos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable, por ejemplo, pudiera desearse que el sistema responda dentro de un intervalo de tiempo especificado o que obtenga los resultados de los cálculos con un nivel de precisión dado.

Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con la toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. (17)

Los requisitos no funcionales deben establecer restricciones en el producto que está siendo desarrollado, en el proceso de desarrollo y en restricciones específicas que el producto pueda tener.

2.1.1 Requerimientos de apariencia o interfaz externa.

- Se podrán distinguir colores atractivos y acordes con los recomendados para los programas 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 perfectamente encuadrado para resoluciones de 1024 x 768, pero preparado para verse en otras resoluciones.

2.1.2 Requerimientos de usabilidad.

- La aplicación debe ser flexible y de fácil aprendizaje.
- El usuario debe tener múltiples vías por las cuales podrá realizar una misma tarea.
- La aplicación podrá ser usada por cualquier persona que posea conocimientos básicos en computación y en ambientes Web.
- Debe brindar la posibilidad de diálogos, con el objetivo de mantener todo el tiempo orientado al usuario.

2.1.3 Requerimientos de soporte.

- El sistema estará bien documentado para garantizar futuros mantenimientos.
- Se le debe dar mantenimiento periódico a los servidores de bases de datos controlando la integridad de la información.

2.1.4 Requerimientos de portabilidad.

- El producto podrá ser usado bajo cualquier sistema operativo ya sea Unix, Linux o Windows.

2.1.5 Requerimientos de seguridad.

- Se debe restringir las funcionalidades mediante roles de usuarios garantizando que la información sea accesible al usuario autorizado.

- La información deberá ser consultada en dependencia del nivel que ocupe el usuario en la escala administrativa del MINSAP, desde Unidad de Salud (US), pasando por los niveles Municipal, Provincial y Nacional.
- La autenticación de los usuarios en el sistema, será garantizada por el Sistema de Autentificación, Autorización y Auditoría (SAAA) al cual estará conectada la aplicación.
- El sistema deberá estar protegido contra accesos no autorizados y las modificaciones de información.

2.1.6 Requerimientos de confiabilidad.

- El sistema estará disponible las 24 horas del día, tanto para el trabajo de los usuarios como para las acciones de mantenimiento.
- Deberá prevenir los posibles fallos y/o errores que pudieran presentarse y posibilitar una rápida recuperación en dichos casos.

2.1.7 Requerimientos de hardware.

En el cliente:

- Procesador Intel Pentium III o superior.
- 256 MB de memoria RAM.
- Monitor tipo VGA o superior.
- Tarjeta de Red.

En el servidor:

- Procesador Intel Pentium IV o superior.
- 2 GB de memoria RAM.
- Disco duro de 80 Gb o más.

- Monitor tipo VGA o superior.
- Tarjeta de Red.

2.1.8 Requerimientos de software.

En el cliente:

- El cliente debe contar con un navegador web. Se recomiendan Internet Explorer 1.5 o superior y Mozilla Firefox 2.0 o superior.
- Sistema operativo Linux o Windows 98 ó Superior.

En el servidor:

- Sistema operativo Linux.
- Servidor Web Apache 2.2.
- PHP 5.
- Framework Symfony 1.4.3.
- Servidor de Base de Datos PostgreSQL 8.3.

2.1.9 Requerimientos de diseño e implementación.

- Utilizar los patrones de diseño GRASP.
- Implementado con el lenguaje de programación PHP 5

2.1.10 Requerimientos de ayuda y documentación en línea.

- Contará con un manual de usuario para que se pueda explotar al máximo.
- Tendrá una ayuda digital, a la cual se podrá acceder desde cualquier parte de la aplicación.

2.2 Descripción de la arquitectura

"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." Es esta la definición que se ha acordado como oficial, provista por el documento de lEEE STD 1471-2000.

La arquitectura del software proporciona una visión global del sistema a construir. Además marca decisiones de diseño tempranas y proporciona el mecanismo para evaluar los beneficios de las estructuras de sistema alternativas.

La arquitectura es la representación que capacita al ingeniero del software para: analizar la efectividad del diseño para la consecución de los requisitos fijados, considerar las alternativas arquitectónicas en una etapa en la cual hacer cambios en el diseño es relativamente fácil, y reducir los riesgos asociados a la construcción del software.

La arquitectura a utilizar fue definida por el grupo de arquitectura del Centro de Informática Médica (CESIM) y la Facultad 7 para todos los programas implementados que se despliegan en INFOMED utilizando los patrones arquitectónicos. Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. A continuación se muestran los patrones seleccionados con algunas de las características particulares.

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: (18)

- **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.

Ventajas:

- Soporta un diseño basado en niveles de abstracción crecientes, lo cual a su vez permite a los implementadores la partición de un problema complejo en una secuencia de pasos incrementales.
- Admite muy naturalmente optimizaciones y refinamientos.
- Proporciona amplia reutilización. Al igual que los tipos de datos abstractos, se pueden utilizar diferentes implementaciones o versiones de una misma capa en la medida que soporten las mismas interfaces de cara a las capas adyacentes. Esto conduce a la posibilidad de definir interfaces de capa estándar, a partir de las cuales se pueden construir extensiones o prestaciones específicas.

Desventajas:

- Muchos problemas no admiten un buen mapeo en una estructura jerárquica. Incluso cuando un sistema se puede establecer lógicamente en capas, consideraciones de *performance* (rendimiento) pueden requerir acoplamientos específicos entre capas de alto y bajo nivel.
- A veces es también extremadamente difícil encontrar el nivel de abstracción correcto; por ejemplo, la comunidad de comunicación ha encontrado complejo mapear los protocolos existentes en el framework ISO, de modo que muchos protocolos agrupan diversas capas, ocasionando que en el mercado proliferen los drivers o los servicios monolíticos.
- Los cambios en las capas de bajo nivel tienden a filtrarse hacia las de alto nivel, en especial si se utiliza una modalidad relajada.

- También se admite que la arquitectura en capas ayuda a controlar y encapsular aplicaciones complejas, pero complica no siempre razonablemente las aplicaciones simples. (19)

La utilización de esta arquitectura en el caso del módulo Epidemiología 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.

La arquitectura de software de una aplicación basada en componentes consiste en uno o un número pequeño de componentes específicos de la aplicación (que se diseñan específicamente para ella), que hacen uso de otros muchos componentes prefabricados que se ensamblan entre sí para proporcionar los servicios que se necesitan en la aplicación. (20)

En la tecnología de componentes la interfaz constituye el elemento básico de interconectividad. 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: (21)

- **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.

Desventajas:

- Las actualizaciones de los componentes adquiridos no están en manos de los desarrolladores del sistema.

- Si no existen los componentes, toca desarrollarlos y se puede perder mucho tiempo, así como que estos componentes pueden tener conflictos si de estos sale una nueva versión y no está estandarizado con lo que se ha desarrollado en la aplicación ensamblada.

Se debe tener en cuenta que este tipo de arquitecturas 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 el Sistema de Información para la Salud (SISalud) del que consumirá algunos servicios, como son:

- Sistema de Autenticación, Autorización y Auditoría. (SAAA).
- Registro de Unidades de Salud (RUS).
- Registro de Ubicación (RU).
- Registro de Ciudadanos (RC).

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. (22)

Los elementos de este patrón son: (23)

- **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.

Entre las ventajas del uso del patrón Modelo-Vista-Controlador están las siguientes: (24)

- 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.

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

2.3 Estrategias de Integración

El software Higiene y Epidemiología (HE) consume varios servicios desarrollados anteriormente en el SISalud como son:

- El SAAA (Sistema de Autenticación, Autorización y Auditoría) 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. Brinda una clave o token a los usuarios registrados que les permite realizar

consultas a otros registros del SISalud. También se usa para darle permisos de acceso a los usuarios a las acciones determinadas (autorización).

- El RUS (Registro de Unidades de Salud) para conocer cuál es el nombre de la Unidad de Salud (US) en la que se encuentra el usuario.
- El RC (Registro de Ciudadanos) se utiliza para buscar y obtener los datos de las personas que intervienen en el negocio, ya sean febriles, viajeros o personal de la salud.
- El RU (Registro de Ubicaciones) el sistema se nutre de este registro en materia de información geopolítica, sobre todo con la finalidad de conocer las direcciones de donde proceden o hacia donde se dirigen las personas que gestiona el software, también brinda información sobre los países, entre otras.

2.4 Vista de despliegue.

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Esta vista permite determinar las consecuencias de la distribución y la asignación de recursos. (25)

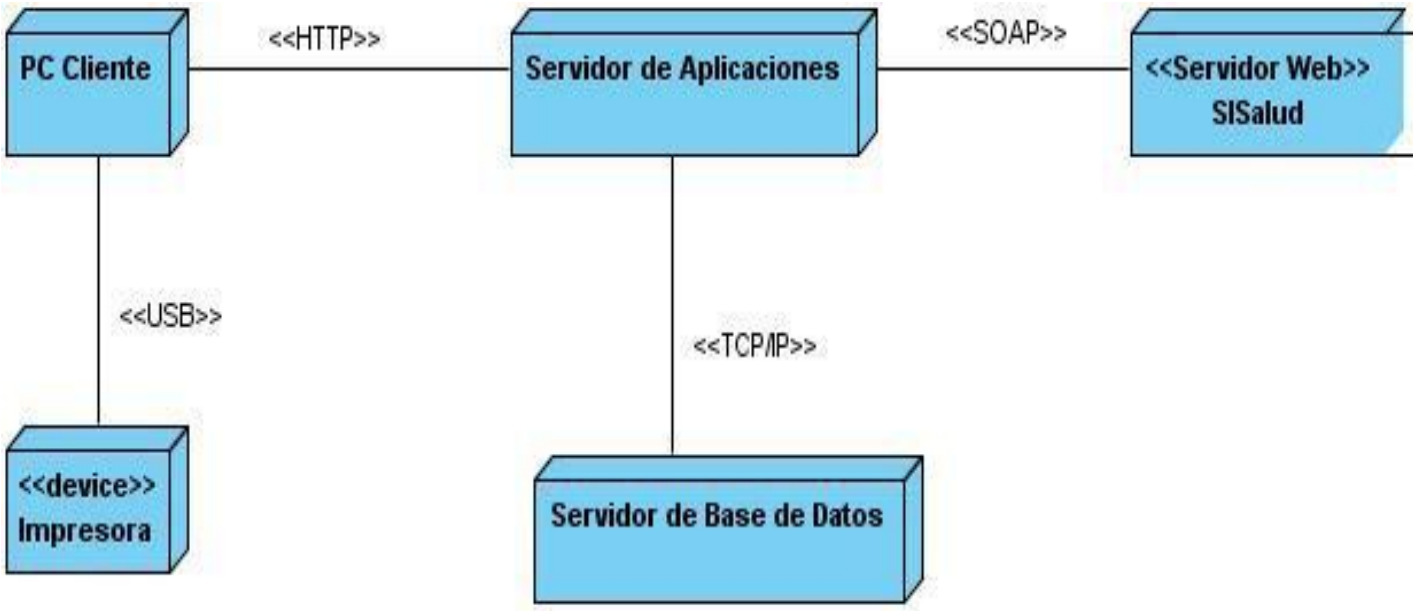


Fig.1: Diagrama de despliegue

En el servidor de aplicaciones estará desplegado el código fuente del software y se encontrarán los componentes o módulos correspondientes a este sistema informático, los cuales serán utilizados por el sistema, como parte de su propia arquitectura basada en componentes. Estas aplicaciones se comunicarán haciendo uso de servicios web XML y mediante el protocolo SOAP, establecido para el intercambio de información en este tipo de servicios.

La aplicación contará además con un servidor de bases de datos con el que se establecerá una comunicación vía TCP/IP. Los clientes podrán conectarse desde cualquier parte mediante el protocolo HTTP al sistema, permitiendo las interacciones con la información. Además, utilizando una impresora podrán obtener cualquier información en formato duro.

2.5 Seguridad.

La seguridad informática consiste en asegurar que los recursos del sistema de información de una organización 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 a estas acciones seguras. (26)

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. (27)

Las tareas de autenticación serán confiadas al SAAA, componente de seguridad del Sistema de Información para la Salud (SISalud). Este módulo permite la autenticación de los usuarios así como determinar si tienen permiso para acceder al sistema de Vigilancia Epidemiológica.

Para las tareas de autorización se utilizará un sistema de roles y permisos previamente definidos en la base de datos de la aplicación que verificará los privilegios de todos los usuarios. Cada usuario tendrá un rol específico y por tanto un nivel de permisos determinados. Dichos permisos le posibilitarán realizar acciones en el sistema.

2.6 Estándares de codificación.

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 niveles de anidamiento.		
Inicio y fin de bloque		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.
Aspectos Generales	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.	
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.		
Ubicación de comentarios	Al inicio de cada clase o función y al final de cada bloque de código.	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.
Líneas en blanco	Se emplean antes y	Se recomienda dejar una línea en blanco antes y

	después de métodos, clases y estructuras.	después de la declaración de una clase o de una estructura y de la implementación de una función
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 arr 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 comenzara con un prefijo que identificara 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: function BuscarUnidad(). 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 funciones	Agrupados por tipos Poner los string 1 numéricos 2, además, agrupar según valores por defecto.	Los parámetros que se le pasan a las funciones se recomienda sean declarados de forma tal que 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	Las 2 primeras letras	Los nombres de las Bases de Datos deben comenzar con el prefijo bd a continuación

de Datos.	representan el tipo.	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: create view '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 utilizara 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 de 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 de 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 utilizara 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	El nombre que se le da a los controles deben

controles.	un prefijo para el tipo de datos en minúscula.	comenzar con las primeras letras 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 detallaron los requisitos no funcionales con los que contará el sistema de Higiene y Epidemiologia, obteniéndose una serie de restricciones importantes para la satisfacción del usuario, entre ellas las de usabilidad, software, hardware, entre otras.

Se realizó una descripción de la arquitectura que presentará el sistema, definiéndose los patrones arquitectónico a utilizar en su desarrollo y se obtuvieron las estrategias de integración que se empleará para facilitar el trabajo de los desarrolladores y evitar la creación de código ya existente.

Se definió que la autenticación del sistema estará delegada al componente SAAA, del Sistema de Información para la Salud, pero los permisos requeridos y los accesos a los usuarios autorizados estarán ya definidos en la base de datos de la aplicación. Además se realizó el diagrama de despliegue para organizar de manera adecuada de las relaciones entre los componentes hardware y software del sistema.

CAPÍTULO 3. DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

Con el objetivo de realizar la implementación del sistema de Higiene y Epidemiología en el presente capítulo se llevará a cabo una valoración de los componentes ya existentes en el SISalud que podrían ser reutilizados por la aplicación. Además se hará un recorrido por las diferentes clases que soportan las funcionalidades del sistema, los principales métodos y cómo interactúan entre ellas. También se hace un acercamiento al modelo de la base de datos y sus principales características, se describen las entidades, sus atributos y se muestra el modelo físico.

3.1 Valoración crítica del diseño propuesto por el analista

El diseño realizado por el analista fue en base al trabajo sobre el framework Symfony y para utilizar el lenguaje de programación PHP5 y aunque se utilizó PostgreSQL como gestor de Base de Datos, la base de datos sufrió cambios en vista a facilitar la implementación del sistema.

El cambio está fundamentado en la incorporación de algunas tablas en la base de datos ya que el sistema consume algunos servicios del SISalud y fue necesario hacer corresponder los atributos de las tablas con los elementos de entrada provistos por el mismo. Por este motivo algunas tablas desaparecen de la BD debido a que la información presente en ellas ya no fue necesaria o aparece vinculada a otros elementos.

Otro de los cambios básicos se realizó en el prototipo no funcional de la aplicación, ya que el diseño del mismo debía ajustarse al diseño del Sistema de Información Hospitalaria (alasHIS).

3.2 Descripción de las nuevas clases u operaciones necesarias.

Unas clases que juegan un papel fundamental son las de gestión de febriles las cuales manipulan toda la información referente al estado febril de las personas así como su evolución.

Nombre: Gestionar datos de febriles inespecíficos: gestionarfebrilActions.php
Tipo de clase : Controladora
Para cada responsabilidad:

Nombre:	executeIndex()
Descripción:	Ejecuta la vista **indexSuccess.php**, que contiene todos los formularios para gestionar febriles inespecíficos (Adicionar, Modificar, Buscar, Eliminar, Visualizar, Registrar y Cambiar) y serán cargados usando librerías YUI.
Nombre:	Adicionarfebril()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para adicionar una persona determinada como febril, en la base de datos del sistema.
Nombre:	Modificarfebril()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para modificar donde aparecen los datos del febril, que se cambian y envían actualizados a la base de datos del sistema.
Nombre:	Buscarfebril()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para buscar donde se brindan algunos criterios para la realización de la búsqueda, que se efectúa por dicha vía o de no seleccionar ningún criterio se realiza una búsqueda de todos los campos.
Nombre:	Eliminarfebril()
Descripción:	Selecciona al febril y elimina, mostrando como resultado un mensaje de información al usuario.
Nombre:	Visualizarfebril()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para visualizar febriles.
Nombre:	Registrarfebril()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para registrar nuevos febriles al Sistema.
Nombre:	Cambiarfebril()

Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para cambiar el estado de un febril.

Tabla: Descripción de la clase controladora Gestionar datos de febriles inespecíficos.

Nombre: Gestionar datos de febriles inespecíficos :TbFebril.class	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	InsertarPersona($persona)
Descripción:	Método para buscar los datos de una persona para insertarlo como febril en la BD.
Nombre:	InsertarFebril($febril)
Descripción:	Método para insertar al febril en la BD.
Nombre:	ModificarFebril($new_febril,$old_febril)
Descripción:	Método para cambiar los datos viejos de un febril por los nuevos en la BD.
Nombre:	CambiarFebril($old_febril,$new_febril)
Descripción:	Método que se utiliza para cambiar los datos de un febril por otro en la BD.
Nombre:	EliminarFebril($febril)
Descripción:	Método para eliminar febriles de la BD.
Nombre:	CambiarEstadoFebril($febril)
Descripción:	Método que se utiliza para cambiar el estado de un paciente por otro en la BD.

Tabla: Descripción de la clase modelo Gestionar datos de febriles inespecíficos.

La clase Gestionar Viajeros es la encargada de manipular toda la información sobre los viajeros así como su condición de salud.

Nombre: Gestionar Viajeros: gestionarviajerosActions.php	
Tipo de clase : Controladora	
Para cada responsabilidad:	
Nombre:	executeIndex()
Descripción:	Ejecuta la vista **indexSuccess.php**, que contiene todos los formularios para gestionar los viajeros (Registrar, Modificar, Visualizar, Eliminar, Cambiar, Buscar) y serán cargados usando librerías YUI.
Nombre:	Registrardatosviajero()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para registrar los datos de un viajero.
Nombre:	Modificardatosviajero()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para modificar donde aparecen los datos de un viajero determinado, permitiéndole al usuario cambiar la información y actualizarlos.
Nombre:	Visualizardatosviajero()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para visualizar los viajeros y los muestra en una lista.
Nombre:	Eliminardatosviajero()
Descripción:	Selecciona un viajero de una lista y lo elimina, mostrando como resultado un mensaje de información al usuario.
Nombre:	Cambiardatosviajero()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para cambiar un

	determinado viajero.
Nombre:	Buscardatosviajero()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para buscar donde se brindan algunos criterios para la realización de la búsqueda, que se efectúa por dicha vía o de no seleccionar ningún criterio se realiza una búsqueda de todos los campos.

Tabla: Descripción de la clase controladora Gestionar Viajeros.

Nombre: Gestionar Viajeros: TbViajero.class	
Tipo de clase : Modelo	
Para cada responsabilidad:	
Nombre:	Registrardatosviajero()
Descripción:	Método que inserta un viajero a la BD.
Nombre:	Modificardatosviajero()
Descripción:	Método para cambiar los datos viejos de un viajero por los nuevos en la BD.
Nombre:	Eliminardatosviajero()
Descripción:	Selecciona un viajero de una lista y lo elimina, mostrando como resultado un mensaje de información al usuario.
Nombre:	Cambiardatosviajero()
Descripción:	Método que se utiliza para cambiar los datos de un viajero por otro en la BD.
Nombre:	Buscardatosviajero()
Descripción:	Método que se utiliza para buscar los datos de un viajero específico en la BD.

Tabla: Descripción de la clase modelo Gestionar Viajeros.

La clase Gestionar Vuelos se encargan de gestionar información referente a los vuelos que arriban a los aeropuertos.

Nombre: Gestionar Vuelo: gestionarvuelosActions.php	
Tipo de clase : Controladora	
Para cada responsabilidad:	
Nombre:	executeIndex()
Descripción:	Ejecuta la vista **indexSuccess.php**, que contiene todos los formularios para gestionar los vuelos (Registrar, Modificar, Visualizar, Eliminar, Buscar) y serán cargados usando librerías YUI.
Nombre:	Registrardatosvuelo()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para registrar los datos de un vuelo.
Nombre:	Modificardatosvuelo()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para modificar donde aparecen los datos de un vuelo determinado, permitiéndole al usuario cambiar la información y actualizarlos.
Nombre:	Visualizardatosvuelo()
Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para visualizar los vuelos y los muestra en una lista.
Nombre:	Eliminardatosvuelo()
Descripción:	Selecciona un vuelo de una lista y lo elimina, mostrando como resultado un mensaje de información al usuario.
Nombre:	Buscardatosvuelo()

Descripción:	Solicita a la vista **indexSuccess.php** mostrar el formulario para buscar donde se brindan algunos criterios para la realización de la búsqueda, que se efectúa por dicha vía o de no seleccionar ningún criterio se realiza una búsqueda de todos los campos.

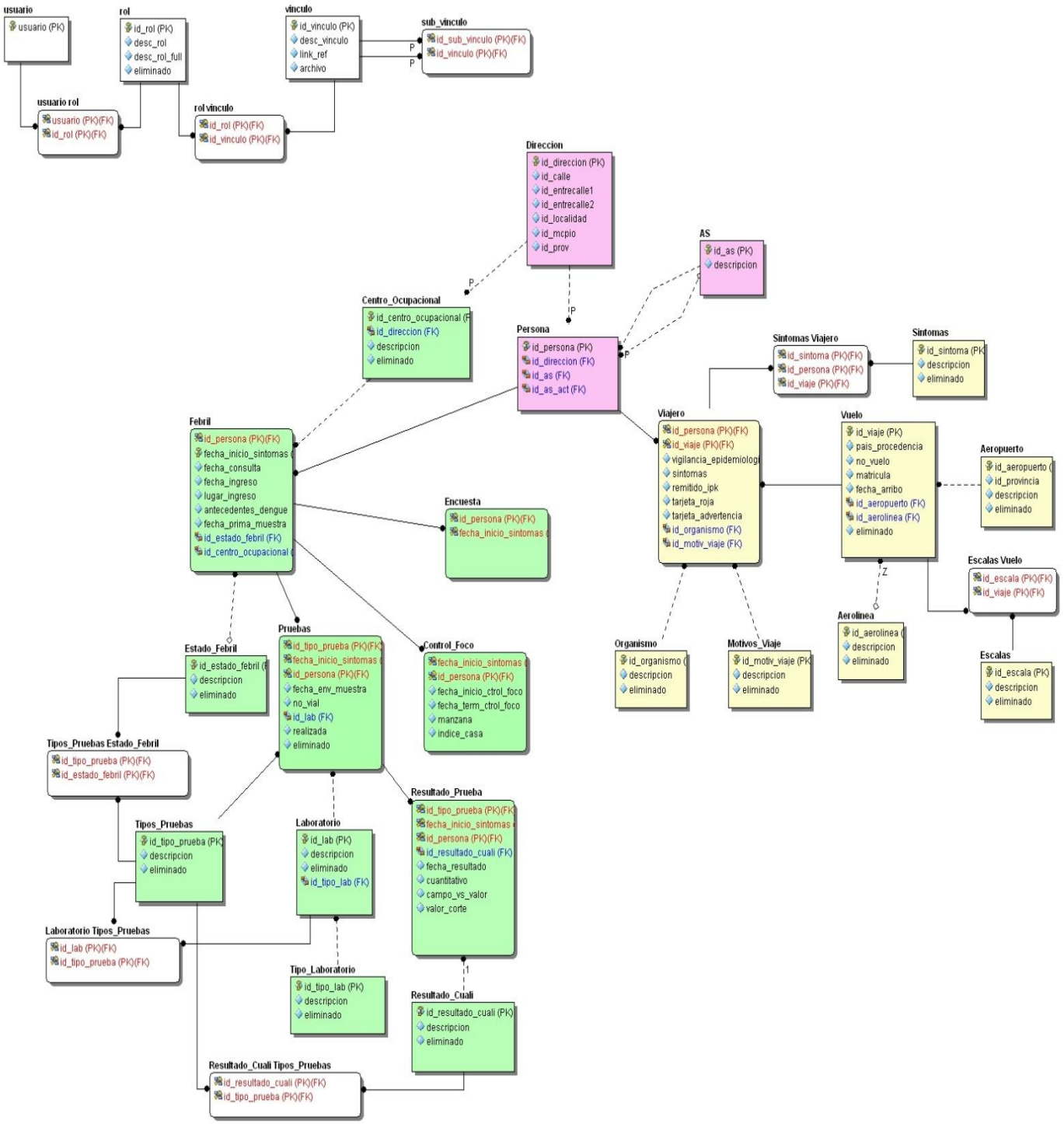
Tabla: Descripción de la clase Controladora Gestionar Vuelos.

Nombre: Gestionar Vuelo: TbVuelo.class	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	InsertarVuelo($vuelo)
Descripción:	Método para insertar el vuelo que se pasa por parámetro
Nombre:	ModificarVuelo($vuelo)
Descripción:	Método para modificar los datos del vuelo que se pasa por parámetro.
Nombre:	EliminarVuelo($id_viaje)
Descripción:	Método para eliminar un vuelo.
Nombre:	PaisesNoEscala($vuelo)
Descripción:	Devuelve una lista de países donde el vuelo no hizo escala.
Nombre:	BuscarVueloID($id_viaje)
Descripción:	Método para buscar un vuelo.
Nombre:	BuscarEscalasVuelo($id_viaje)
Descripción:	Método para buscar dónde realizó escalas el vuelo.

Nombre:	BuscarTotalVuelos($criterioVuelo)
Descripción:	Método para buscar todos los datos relacionados a un vuelo según un criterio de búsqueda.

Tabla: Descripción de la clase Modelo Gestionar Vuelos.

3.3 Modelo de datos



3.4 Breve valoración de las Técnicas de validación

3.4.1 Descripción de las tablas más significativas de la base de datos.

Nombre: tb_persona		
Descripción: En esta tabla se almacenan los datos de las personas.		
Atributo	**Tipo**	**Descripción**
id_persona	VARCHAR (50)	Llave primaria (PK) de la tabla.
id_direccion	INT	Código que identifica la dirección de cada persona.
id_as	INT	Código que identifica el área de salud que le corresponde de cada persona.
is_as_act	INT	Código que identifica el área de salud actual de cada persona.

Tabla 1: PERSONAS.

Nombre: tb_viajero		
Descripción: Almacena datos sobre el viajero.		
Atributo	**Tipo**	**Descripción**
id_paciente	VARCHAR(50)	Llave primaria (PK) de la tabla. (importada de la tabla tb_persona)
id_viaje	INT	Llave primaria (PK) de la tabla. (importada de la tabla tb_vuelo)
vigilancia_ epidemiologica	TINYINT	Si está en vigilancia epidemiológica o no.

sintomas	VARCHAR(100)	Descripción de los síntomas que presenta el viajero.
remitido_ipk	TINYINT	Se refiere a si fue remitido al IPK o no.
tarjeta_roja	TINYINT	Se refiere a si fue sometido a tarjeta roja.
tarjeta_advertencia	TINYINT	Se refiere a si fue sometido a tarjeta de advertencia.
id_organismo	INT	Identifica al organismo al que pertenece.
id_motiv_viaje	INT	Motivo del viaje del viajero.

Tabla 3: LA TABLA VIAJEROS.

Nombre: Pruebas		
Descripción: Almacena datos sobre las pruebas.		
Atributo	**Tipo**	**Descripción**
id_tipo_prueba	INT	Llave primaria (PK) de la tabla.
fecha_inicio_sintomas	DATE TIME	Llave primaria (PK) de la tabla.
id_persona	VARCHAR(50)	Llave primaria (PK) de la tabla.
fecha_env_muestra	DATETIME	Fecha de arribo del vuelo.
no_vial	VARCHAR(10)	Identifica el número del vial.
id_lab	INT	Identifica al laboratorio que envía las pruebas.
realizada	TINYINT	Identifica si fue realizada la prueba.
eliminado	TINYINT	Campo que se refiere a si fue eliminado o no.

Tabla 4: LA TABLA PRUEBAS.

Nombre: tb_vuelo		
Descripción: Almacena datos sobre el vuelo.		
Atributo	**Tipo**	**Descripción**
id_viaje	INT	Llave primaria (PK) de la tabla.
país_procedencia	INT	País de donde viene el vuelo.
no_vuelo	VARCHAR(20)	Número del vuelo.
matricula	VARCHAR(20)	Matrícula del avión.
fecha_arribo	DATETIME	Fecha de arribo del vuelo.
id_aeropuerto	INT	Identifica al aeropuerto al que pertenece.
id_pais	INT	Identifica al país de procedencia.
id_aerolinea	INT	Identifica a la aerolínea al que pertenece.

Tabla 5: LA TABLA VUELOS.

Nombre: Resultado_Prueba		
Descripción: Almacena datos sobre el resultado de las pruebas.		
Atributo	**Tipo**	**Descripción**
id_tipo_prueba	INT	Llave primaria (PK) de la tabla.
fecha_inicio_sintomas	DATE TIME	Llave primaria (PK) de la tabla.
id_persona	VARCHAR(50)	Llave primaria (PK) de la tabla.
id_resultado_cuali	INT	Identifica el resultado cualitativo.

fecha_resultado	DATE TIME	Fecha del resultado.
cuantitativo	VARCHAR(10)	Se refiere al resultado cuantitativo.
campo_vs_valor	VARCHAR(20)	Codificador para establecer parámetros.
valor_corte	VARCHAR(10)	Se refiere al valor establecido para declarar si es positivo o negativo.

Tabla 6: LA TABLA RESULTADOS DE PRUEBAS.

Nombre: Febril		
Descripción: Almacena datos sobre los febriles.		
Atributo	**Tipo**	**Descripción**
id_persona	VARCHAR(50)	Llave primaria (PK) de la tabla.
fecha_inicio_sintomas	DATE TIME	Llave primaria (PK) de la tabla.
fecha_consulta	DATE TIME	Fecha de la consulta.
fecha_ingreso	DATE TIME	Fecha del ingreso.
lugar_ingreso	VARCHAR(100)	Se refiere al lugar donde ingresa el febril.
antecedentes_dengue	TINYINT	Se refiere a si tiene antecedentes de dengue o no.
fecha_prima_muestra	DATE TIME	Fecha en que se toma la primera muestra.
id_estado_febril	INT	Identifica en qué estado se encuentra el febril.
id_centro_ocupacional	INT	Identifica el centro ocupacional.

Tabla 7: LA TABLA FEBRIL.

3.5 Vista de Implementación. (Diagramas de Componentes)

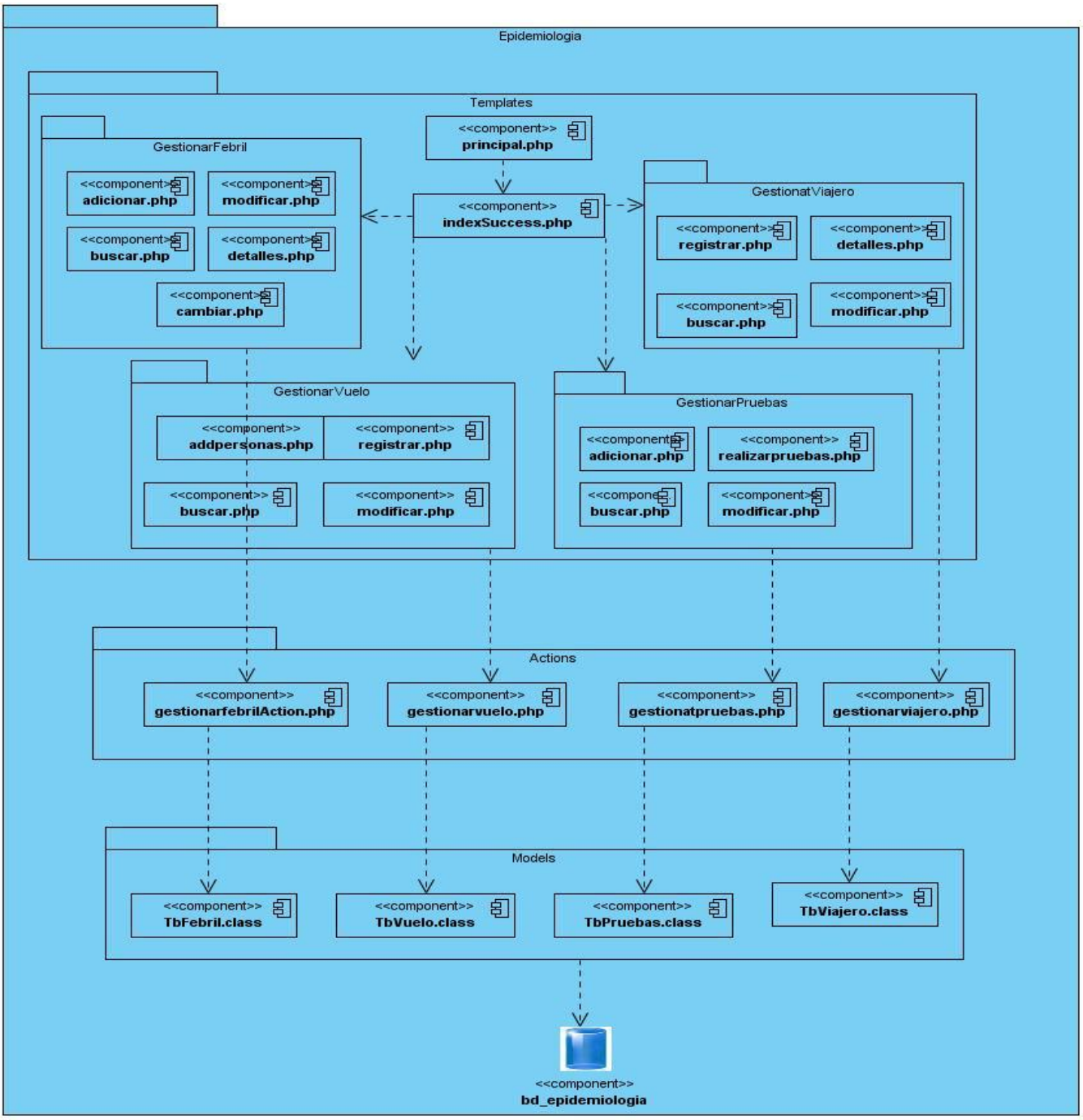


Fig 11: Diagrama de componentes.

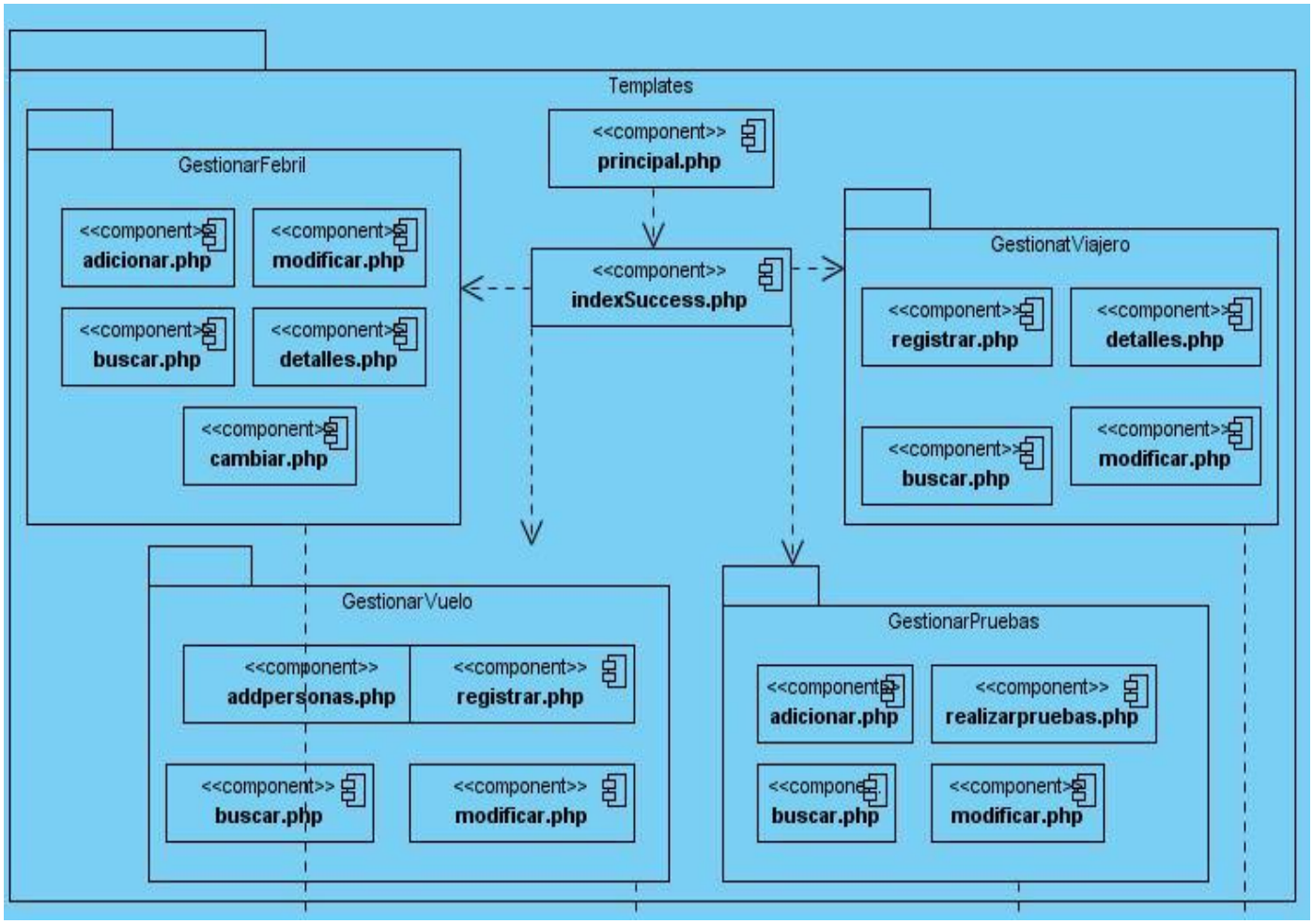


Fig12: Diagrama para las Clases Vistas de Implementación

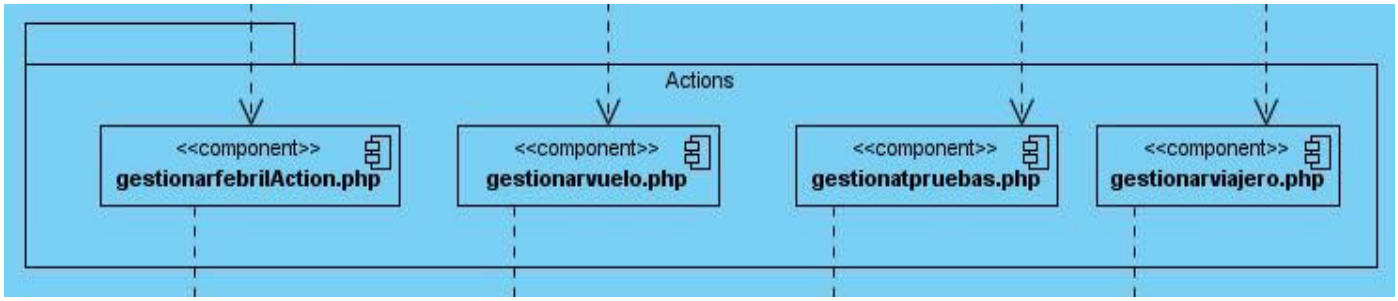


Fig13: Diagrama para las Clases Controladoras de Implementación

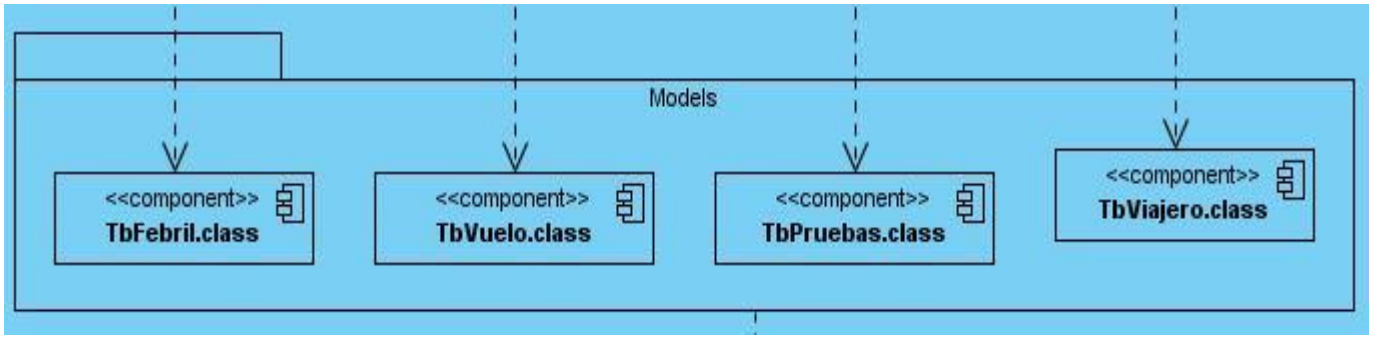


Fig14: Diagrama para las Clases Modelos de Implementación

Con la realización del presente capítulo se logró un análisis del diseño propuesto por los analistas concluyendo en algunas mejoras que benefician al sistema como son: la introducción de nuevas tablas a la base de datos. Además se describieron las clases fundamentales del sistema y sus funcionalidades.

Se detallaron los elementos de la base de datos con el objetivo de especificar cada atributo con los que cuenta la misma y proveer un mejor entendimiento y manipulación de los mismos. También se obtuvo la vista de implementación del sistema con la cual se puede entender de forma explícita la forma en la que se distribuyen los componentes de la aplicación.

CONCLUSIONES

El desarrollo de las tareas de la investigación posibilitó dar cumplimiento al objetivo general de la misma y arribar a las siguientes conclusiones:

- Luego del análisis de los procesos relacionados con la vigilancia epidemiológica al viajero y las enfermedades tropicales en Cuba, así como del diseño del "Sistema Integral de Vigilancia Epidemiológica 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, RC 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 a la independencia tecnológica de 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. Fue necesario eliminar algunas tablas de la BD para evitar redundancia de información y se agregó la tabla: síntomas, para nomenclar esta variable.
- La implementación del Sistema Integral de Vigilancia Epidemiológica V 1.0.1 le permitirá al país la obtención de una herramienta para la vigilancia epidemiológica al viajero y el control de las enfermedades tropicales, lo cual reducirá el riesgo de propagación de estas enfermedades en la isla.

RECOMENDACIONES

Teniendo como base los resultados de esta investigación y la experiencia adquirida durante el desarrollo de la misma, se recomienda:

- Realizar las pruebas de calidad a las funcionalidades para comprobar su total funcionamiento.

REFERENCIAS BIBLIOGRÁFICAS

(1)- RUP [En línea] [Citado el: 9 de febrero de 2010.] http://www.rational.com.ar/herramientas/rup.html.

(2)- Ídem a la referencia 1

(3)- Kruchten, Philippe. The Rational Unified Process An Introduction. s.l.: Addison Wesley, 2001.

(4)- Mendoza Sanchez, María A. informatizate. Metodologías De Desarrollo De Software. [Online] Junio 2004, 7. [Cited: diciembre 16, 2009.] http://www.informatizate.net/articulos/metodologias_de_desar.

(5)- Rumbaugh, James, Jacobson, Ivar y Booch, Grady. El Lenguaje Unificado de Modelado. Manual de Referencia. s.l: Addison Wesley, 1999.

(6)- CABALLERO, ISMAEL y VISCAINO, AURORA. Una Herramienta CASE para ADOO: Visual Paradigm. . Ciudad Real: s.n., 2003.

(7)- Paradigm, Visual. Visual Paradigm UML for Enterprise Edition. [En línea] [Citado el: 05 de 12 de 2008.] http://www.visual-paradigm.com/product/vpuml/.

(8)- Cosas sencillas. [Online] [Cited: dicimbre 18, 2009.]

http://cosassencillas.wordpress.com/category/javascript/.

(9)- Ídem a la referencia 8

(10)- Gutiérrez., Javier J. ¿Qué es un framework web?

(11)-Libros Web. [Online] [Cited: enero 13, 2010.]

http://www.librosweb.es/symfony_1_1/capitulo1/symfony_en_pocas_palabras.html.

(12)- Symfony [Online] [Cited: febrero 9, 2010.] http://www.symfony.es.

(13)-Libros web. [Online] [Cited: febrero 9, 2010.]

http://librosweb.es/css_avanzado/capitulo5/el_framework_yui.html.

(14)- YUI: Yahoo User Interface. [En línea] http://www.n4gash.com/2008/yahoo-user-interface-library-yui-ejemplos-codigos-javascript/

(15)- Slideshare. [Online] [Cited: enero 14, 2010.] http://www.slideshare.net/jeissonlarry/sistema-gestin-de-bases-de-datos-2657624.

(16)- eaprende. [Online] [Cited: enero 14, 2010.] http://www.eaprende.com/gestor-de-basededatos-mysql-postresql-sqlite.html.

(17)-Introducción a Apache. [En línea] [Citado el: 9 de febrero de 2010]

http://linux.ciberaula.com/articulo/linux_apache_intro/

(18)- ISW1 Conf 3 Flujo de trabajo de requerimientos. Dpto ISW UCI. Ciudad Habana: s.n., 2007

(19)- 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/.

(20)-Libros Web. [Online] [Cited: 1 de diciembre, 2010.]

http://www.librosweb.es/symfony_1_0/capitulo2/el_patron_mvc.html

(21)- Arquitectura Basada en Componente. [En línea] [Citado el: 05 de 12 de 2008.]

http://msguayaquil.com/blogs/julioc/archive/2006/05/08/Desarrollo-de-Software-Basado-en-Componentes.aspx

(22)- 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.

(23)- Ingeniería de Software II. “Arquitectura y Patrones de diseño”. Modelo –Vista-Controlador. Conferencia # 2. s.l. : UCI, 2008_2009.

(24)- 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.

(25)- 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

(26)- Marca Huallpara, H. M., & Quisbert Limachi, N. S. (n.d.). Analisis y Diseño de Sistema II. Retrieved enero 12, 2010.

(27)-Libros web. [En línea] [Citado el: 25 de 3 de 2010.] http://www.librosweb.es/symfony/capitulo6/seguridad_de_la_accion.html

BIBLIOGRAFÍA

1. Arquitectura Basada en Componentes [En línea]. - 05 de 02 de 2010. – http://msguayaquil.com/blogs/julioc/archive/2006/05/08/Desarrollo-de-Software-Basado-en-Componentes.aspx
2. 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/.
3. 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.
4. 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.
5. CABALLERO, ISMAEL y VISCAINO, AURORA. Una Herramienta CASE para ADOO: Visual Paradigm. .Ciudad Real: s.n., 2003.
6. Eaprende. [Online] [Cited: enero 14, 2010.] http://www.eaprende.com/gestor-de-basededatos-mysql-postresql-sqlite.html.
7. Garlan, David and Shaw, Mary. *An introduction to software architecture.* CMU Software Engineering Institute : s.n., 1994.
8. Grupo Soluciones Innova. [En línea] [Citado el: 9 de febrero de 2010.] http://www.rational.com.ar/herramientas/rup.html.
9. Ingeniería de Software II. “Arquitectura y Patrones de diseño”. Modelo –Vista-Controlador. Conferencia # 2. s.l. : UCI, 2008_2009.
10. Introducción a Apache. [En línea] [Citado el: 9 de febrero de 2010]
11. Introducción a Apache. [En línea] [Citado el: 9 de febrero de 2010] http://linux.ciberaula.com/articulo/linux_apache_intro/
12. ISW1 Conf 3 Flujo de trabajo de requerimientos. Dpto ISW UCI. Ciudad Habana: s.n., 2007
13. Libros Web. [Online] [Cited: 1 12, 2010.] http://www.librosweb.es/symfony_1_0/capitulo2/el_patron_mvc.html.

14. Libros Web. [Online] [Cited: febrero 9, 2010.] http://librosweb.es/css_avanzado/capitulo5/el_framework_yui.html.

15. Marca Huallpara, H. M., & Quisbert Limachi, N. S. (n.d.). Analisis y Diseño de Sistema II. Retrieved enero 12, 2010.

16. Modelo-Vista-Controlador. [En línea] [Citado el: 25 de 11 de 2008.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

17. Paradigm, Visual. Visual Paradigm UML for Enterprise Edition. [En línea] [Citado el: 05 de 12 de 2008.] http://www.visual-paradigm.com/product/vpuml/.

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

19. Rumbaugh, James, Jacobson, Ivar y Booch, Grady. El Lenguaje Unificado de Modelado. Manual de Referencia. s.l: Addison Wesley, 1999.

20. RUP [En línea] [Citado el: 9 de febrero de 2010.] http://www.rational.com.ar/herramientas/rup.html.

21. Slideshare. [Online] [Cited: enero 14, 2010.] http://www.slideshare.net/jeissonlarry/sistema-gestin-de-bases-de-datos-2657624.

22. Symfony [Online] [Cited: febrero 9, 2010.] http://www.symfony.es.

23. 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.

24. YUI: Yahoo User Interface. [En línea] http://www.n4gash.com/2008/yahoo-user-interface-library-yui-ejemplos-codigos-javascript/

ANEXOS

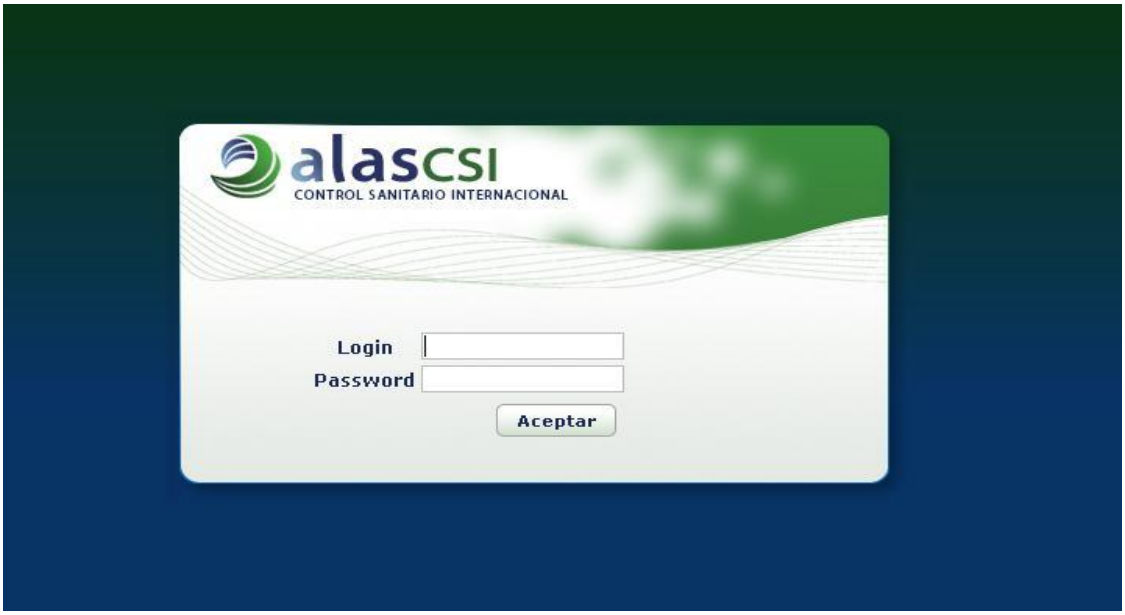


Fig1: Login de la aplicación

Fig2: Página principal del sistema

Fig3: Gestionar Febril

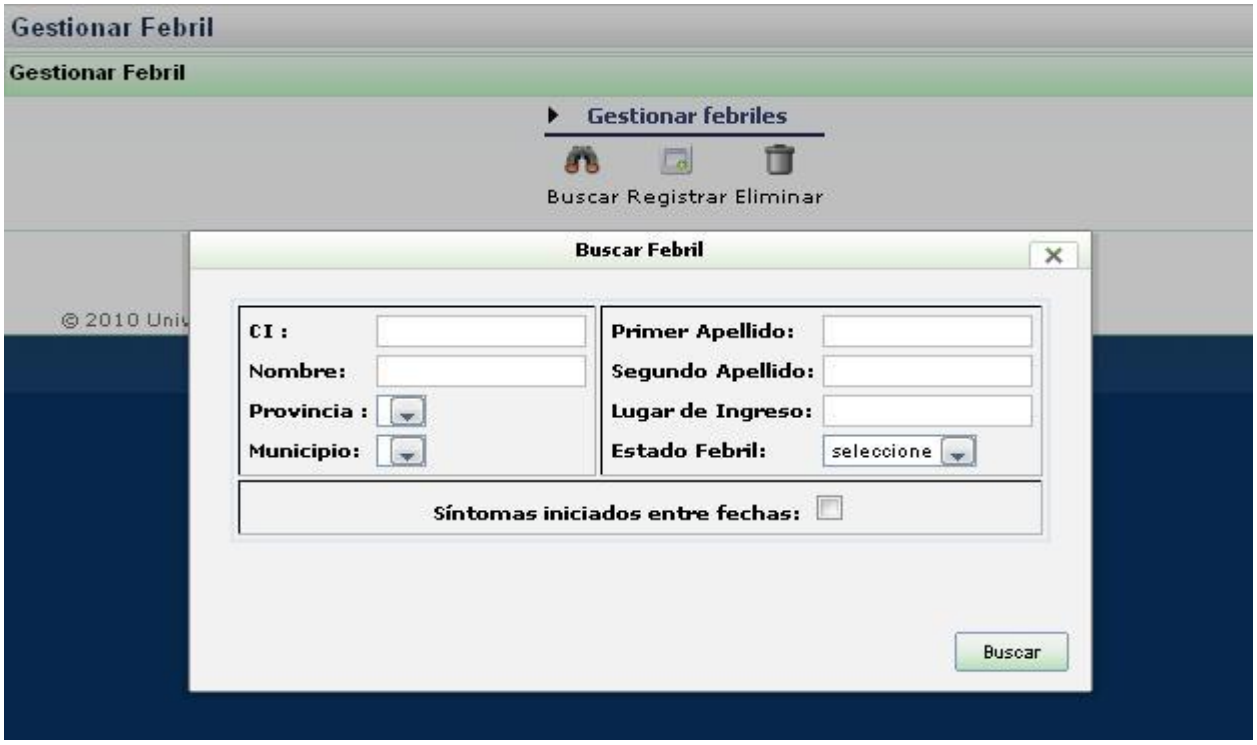


Fig4: Buscar Febril

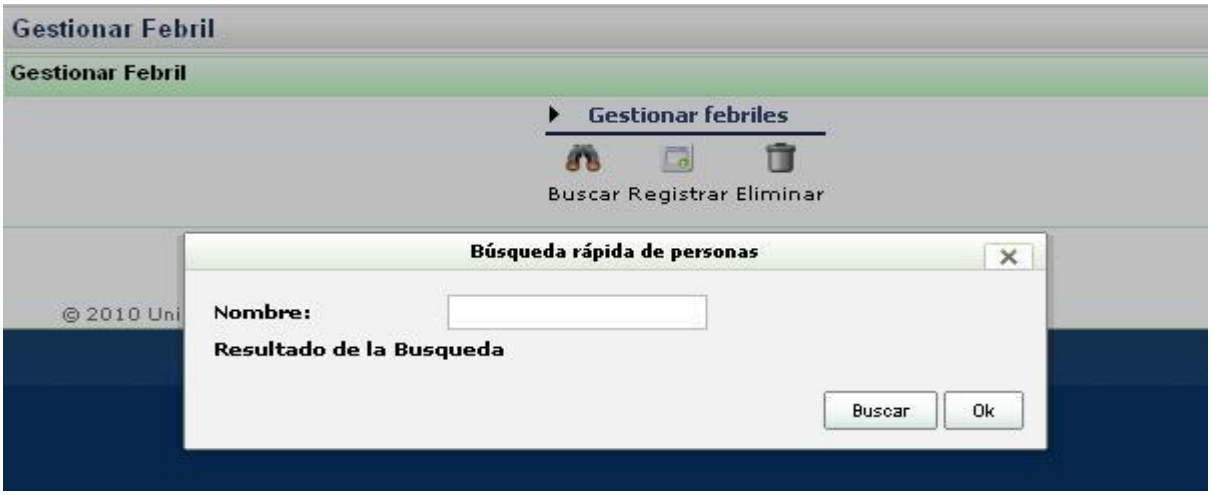


Fig5: Buscar persona

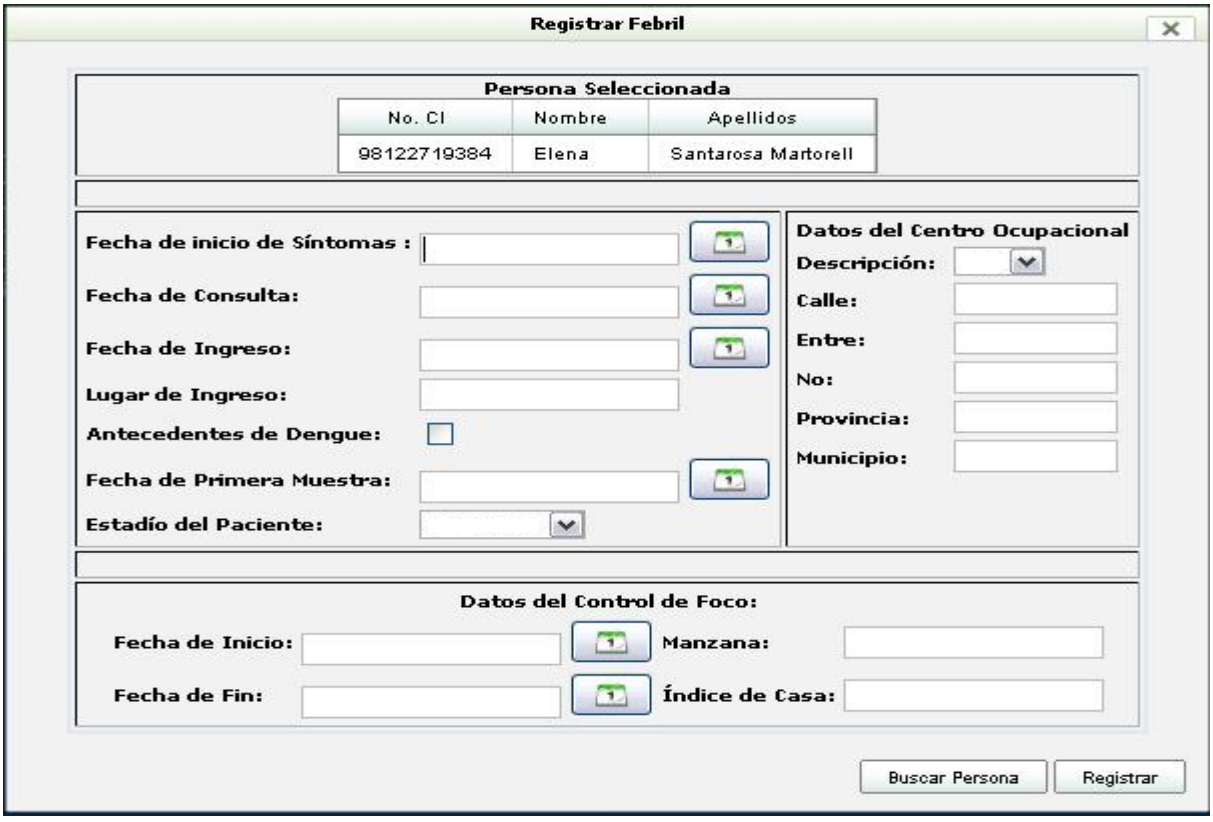


Fig6: Registrar febril

Fig9: Gestionar viajero

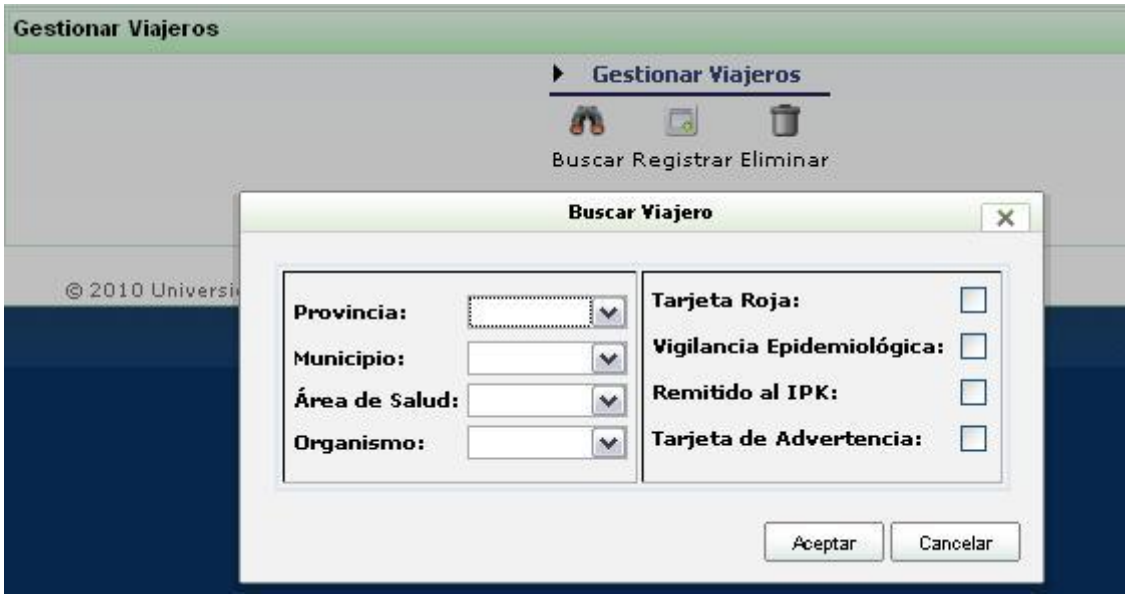


Fig10: Buscar Viajero

Fig11: Registrar viajero

Fig13: Gestionar vuelo

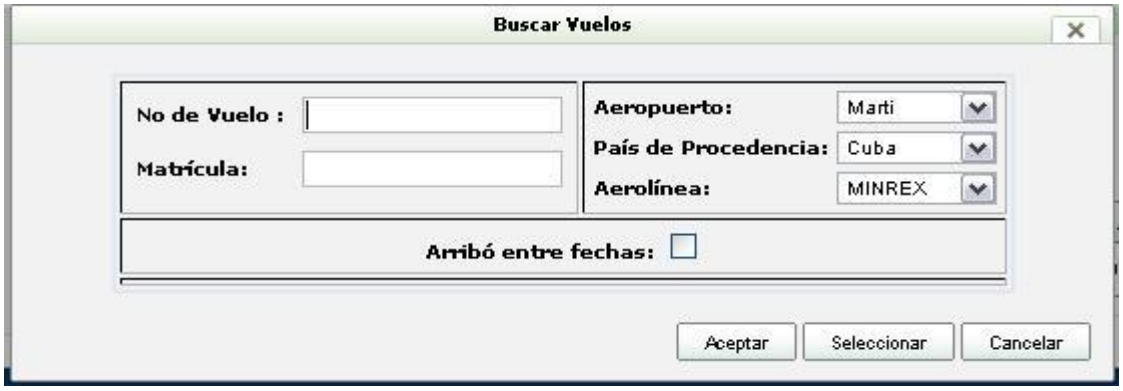


Fig14: Buscar vuelo

Fig15: Registrar vuelo

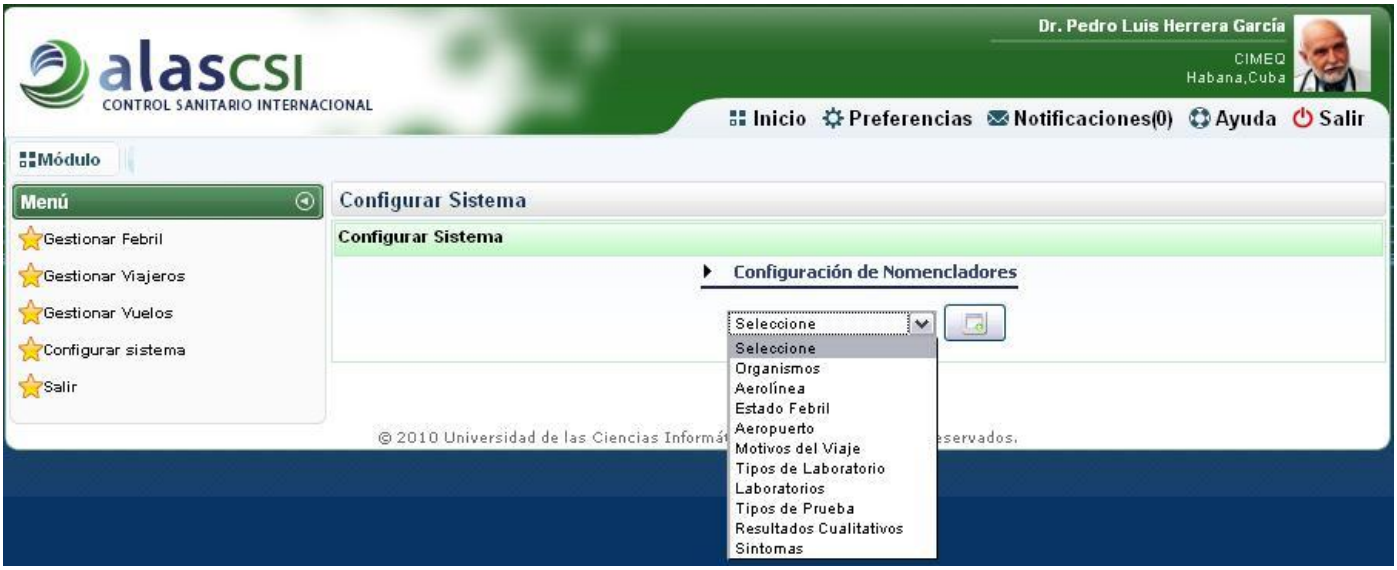


Fig18: Configurar nomencladores

GLOSARIO

CASE: Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

Caso de Uso: Especificación de las secuencias de acciones, incluyendo variaciones que un sistema lleva a cabo y que conduce a un resultado observable de interés para un actor determinado.

GRASP: Patrones Generales de Software para Asignación de Responsabilidades, es el acrónimo de " General Responsibility Assignment Software Patterns".

Herramienta Case: Ingeniería de sistemas asistida por ordenador (Computer-Aided System Engineering - CASE) es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo de sistemas. Su objetivo es automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas.

HTTP: Hypertext Transfer Protocol (en español protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web (www).

IDE: (Integrated Development Environment) Un entorno de desarrollo integrado es un programa compuesto por un conjunto de herramientas para un programador.

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. INFOMED 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.

Metodologías: Se encargan de elaborar estrategias de desarrollo de software que promuevan prácticas adaptativas en vez de predictivas; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente.

MINSAP: El Ministerio de Salud Pública 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.

Patrones: Describe un problema que ocurre una y otra vez en su entorno y describe también el núcleo de la solución al problema, de forma que puede reutilizarse continuamente.

RUP: Proceso Unificado de Software (Rational Unified Process).

Software: Conjunto de instrucciones escritas en un determinado lenguaje, que dirigen a un ordenador para la ejecución de una serie de operaciones, con el objetivo de resolver un problema que se ha definido previamente.

SOAP: (Siglas de Simple Object Access Protocol) es un protocolo estándar creado por Microsoft, IBM y otros, está actualmente bajo el auspicio de la W3C que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. SOAP es uno de los protocolos utilizados en los servicios Web.

TIC: Las Tecnologías de la Información y la Comunicación son un conjunto de servicios, redes, software, aparatos que tienen como fin el mejoramiento de la calidad de vida de las personas dentro de un entorno, y que se integran a un sistema de información interconectado y complementario.

TCP/IP: conjunto de protocolos: Protocolo de Control de Transmisión (TCP) y Protocolo de Internet (IP). Permiten la transmisión de datos entre redes de computadoras.

UCI: Universidad de Ciencias Informáticas.

UML: El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

Vigilancia Epidemiológica: Término inicialmente aplicado a un conjunto de medidas inherentes a la observación de la evolución de enfermos infecciosos o sospechosos y sus contactos, es decir, que no era más que la aplicación del método epidemiológico para el control individual de casos. En su desarrollo, la vigilancia se extendió y su marco de trabajo fue incluyendo todos los aspectos y mecanismos que intervienen en la aparición de enfermedades transmisibles o no, agudas o crónicas, así como desviaciones de la salud, tanto en el orden individual como de toda la población.