Universidad de las Ciencias Informáticas

Facultad 7

Título: Implementación del módulo Higiene y Epidemiología del sistema Control Sanitario Internacional.

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

Autores: Ricardo Collada de la Rosa.

Josué Socorro Guzmán.

Tutores: Ing. José Alejandro Segura Roque.

Ing. Yunaysy Ortiz Batista.

Asesor: Msc. Jorge Castillo Maceo

Ciudad de La Habana – Junio de 2008

"Año 50 de la Revolución"

“Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad.”

Albert Einstein.

Declaración de Autoría

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

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

Ricardo Collada de la Rosa	Josué Socorro Guzmán
_____	_____
Firma del Autor	Firma del Autor
Ing. José Alejandro Segura Roque	Ing. Yunaysy Ortiz Batista
_____	_____
Firma del Tutor	Firma de la Tutora

Datos de contacto

Tutores:

José Alejandro Segura Roque: Ingeniero en Ciencias Informáticas.

Email: jsegura@uci.cu

Yunaysy Ortiz Batista: Ingeniero en Ciencias Informáticas.

Email: yunaysy@uci.cu

Asesor:

Jorge Castillo Maceo: Máster en ciencias militares. Profesor asistente.

Email: castillo@uci.cu

Agradecimientos

Queremos agradecer:
A la Universidad de las Ciencias Informáticas, al eterno Fidel y a la Revolución cubana por esta gran oportunidad.
A los tutores, cuyo apoyo ha sido fundamental para realizar con éxito esta tesis.
A los compañeros del MINSAP por su atención dedicación y apoyo siempre. Muy en especial a Diana Rodríguez.

De Ricardo:

Por sobre todas las cosas agradezco la amistad, el cariño y la comprensión de aquellas personas que me han inspirado siempre a seguir adelante y me han dado la fuerza de llegar hasta aquí.

Gracias a mi familia, que siempre me apoyó y me guió, que ha estado conmigo en cada desventura dándome fuerzas y en cada victoria compartiendo el regocijo y la alegría.

A la familia de Yadira que es como la mía propia, en ellos he encontrado un refugio de paz, comprensión y positividad. Gracias por eso.

A mis amigos de la universidad y a los de toda la vida. Muy en especial a Leslier, Yamel, Vladimir, Yadira, Yans, Raciel y Dayron. Amigos en con los que siempre conté, y que han sabido responderme, soportarme y ayudarme.

A los compañeros del Área Temática Sistemas Especializados. A Daniel Miranda, un as del CodeIgniter que sembró la base de mis conocimientos en este framework.

A mis profesores de siempre. Especialmente a Marta Rosa Abreu, César N. Richard, Julio Lázaro Betancourt y Rafael Trujillo.

De Josué:

A mi madre que siempre estuvo a mi lado, hasta en los momentos en que no me lo merecía, esta tesis es más tuya que mía, por ti y para ti.

A mi novia y compañera por más de una década, por ser tan paciente y cariñosa, ojalá y ese sea su carácter por siempre y ojalá cuando mire esta dedicatoria dentro de 20 años estés todavía a mi lado y no me tenga que arrepentir de haber escrito esto.

A mi familia, mi abuela Guille, mi tía Mary, mi tío Robe (El Gato) y todos sus gaticos, a mi padre Juan Francisco Socorro, a mis hermanos Iván y Frankito y mis respectivas cuñadas, a Ibis y Alfon, los mejores suegros del mundo, a todos en general por el apoyo y los consejos, esto demuestra que al final les hice caso.

A mis sobrinitos, porque sí.

A mi compañero de tesis, Ricardo(el Yeti) y su novia YadiYadi.

A mis amigos, los de siempre y los nuevos, no quiero mencionar nombres, sería muy bochornoso que me fallara ahora la memoria de la que tanto presumo y no sería justo de mi parte omitir a nadie, La Gente del Aula(los que estamos y todos los que quedaron por el camino), El Pikete de los Coleópteros, Los de la Humboldt, Los Biscochos, Los Grillos, Los Menes, todos los de San Antonio y a todos los que hicieron posible que estos 5 años perduren por siempre en mi memoria y en las historias que le contaré a mis nietos.

A todos, muchas gracias.

Dedicatoria

Ricardo.

A mis abuelos Ana Rubio y Eddy de la Rosa.

A mis abuelos Marta Menéndez y Armando Collada.

A mi mamá, Ana Margarita de la Rosa, una amiga invaluable.

A Yadira Marrero (yadi), mi novia.

A mi familia.

A mis amigos.

Josue.

A Magalis la mujer que más quiero en el mundo. La mejor madre.

A mi familia.

Resumen

En los últimos tiempos, con el aumento de las relaciones políticas, culturales y económicas de Cuba, ha aumentado la entrada al país de enfermedades emergentes y reemergentes. Si a esto se suman, las agresiones en el campo biológico del gobierno de los EEUU, se puede entender que la vigilancia epidemiológica se convierte en una misión de primer orden para el país. Un modo de contribuir a la efectividad de esta tarea, es el desarrollo de sistemas informáticos para la automatización del proceso.

Para ello, en este trabajo se propone implementar una aplicación Web que permita viabilizar la gestión de la información relacionada con la vigilancia epidemiológica al viajero y a las enfermedades tropicales en Cuba.

Para el desarrollo de este sistema fueron utilizadas herramientas y tecnologías basadas fundamentalmente en software libre como el framework CodeIgniter, el lenguaje PHP 5, el gestor de bases de datos MySQL 5, entre otras.

Con la instalación y puesta en práctica del módulo Higiene y Epidemiología, se podrá automatizar un proceso vital en el mecanismo de la salud pública cubana. Los datos que este brinda, son de gran importancia en la toma de decisiones y el manejo de recursos por parte del MINSAP, sobre todo, en momentos en que la situación epidemiológica del país sea delicada.

Palabras clave

Implementación, Control Sanitario Internacional, Higiene y Epidemiología, Arquitectura, Base de Datos, CodeIgniter.

Índice

INTRODUCCIÓN

Las nuevas relaciones diplomáticas, culturales y comerciales que ha establecido y fortalecido Cuba en los últimos años, han traído como consecuencia un aumento considerable del tráfico internacional de personal cubano y extranjero, sobre todo hacia y desde regiones endémicas de enfermedades transmisibles y donde estas aparecen con gran frecuencia, lo cual incrementa considerablemente el riesgo de importación de muchas de ellas.

Se suma el hecho de que en Cuba existen vectores que pueden propagarlas de forma vertiginosa y que el ecosistema de la isla brinda las condiciones óptimas para la reproducción y desarrollo de dichos vectores y la propagación de las enfermedades.

Solamente entre enero y septiembre del año 2007 arribaron por el aeropuerto internacional José Martí de Ciudad de la Habana 879 480 viajeros, (34 669 más que en el 2006) y de ellos en vigilancia epidemiológica 78 920 (12 650 más con relación al año anterior), es importante señalar que se importaron 55 casos de dengue en la isla, de ellos 44 procedentes de Venezuela, 3 de Jamaica, 3 de México, 1 de Guyana,1 de Costa Rica,1 de Haití y 1 Honduras, y que 11 arribaron al país entre los días 16 de Septiembre y 30 de Octubre.

No se debe pasar por alto que en ocasiones se han introducido enfermedades y vectores en el país con fines políticos, y como parte de las estrategias genocidas que sigue el gobierno de los Estados Unidos desde hace ya mucho tiempo para, a través de hambre, enfermedades y terror, doblegar al pueblo cubano en su lucha por la igualdad y la justicia social.

En 1981, por ejemplo se presenta en Cuba una de las epidemias de dengue más severas y la primera de dengue hemorrágico en el hemisferio. De forma general se registraron 344 203 casos, de los cuales 10 312 fueron diagnosticados como fiebre hemorrágica del dengue (FHD). Fallecieron 158 personas, de ellos 101 niños. Todas las evidencias y la desclasificación posterior de documentos e informaciones de la C.I.A. demuestran que el dengue hemorrágico fue introducido en Cuba como parte de la Guerra Biológica de los Estados Unidos contra el país. Pero no sólo se introdujo el virus, sino que se negó la posibilidad de adquirir los productos químicos para eliminar el agente transmisor. El país tuvo que desplazar aviones a Europa y Japón para adquirirlos.

En este contexto el tema del control sanitario internacional adquiere gran importancia para la salud pública cubana, la que durante los años de Revolución ha adquirido niveles solo comparables con países desarrollados en los que las enfermedades crónicas tienen un papel relevante y las transmisibles son muy escasas, lo cual es consecuencia fundamentalmente del sistema de salud cubano, el cual tiene, creados y estructurados desde sus propios inicios los sistemas de vigilancia epidemiológica a viajeros, enfermedades transmisibles y la comunidad en general.

El programa nacional de Control Sanitario Internacional (CSI), que no solo abarca lo concerniente a la vigilancia epidemiológica, sino se extiende a temas como el control de vectores y la salud ambiental, ha sido una prioridad fundamental del estado cubano y el MINSAP en especial.

En el marco actual, el control y vigilancia epidemiológicos se lleva a cabo a través de redes de información establecidas por el MINSAP que van desde el Área de Salud hasta el propio Ministerio, permitiendo tener conocimiento de datos significativos concernientes al estado epidemiológico del país a todos los niveles; sin embargo la información se mueve a través de conversaciones telefónicas o correspondencia por correo convencional.

A pesar de estar muy bien organizado, el sistema de vigilancia epidemiológica es susceptible a ciertos errores, entre otros de carácter humano, y puede traer como consecuencia que no siempre se manejan datos totalmente actualizados, incoherencia en la información que se recibe de los distintos niveles y en cierta medida la pérdida de datos valiosos.

La información concerniente a la vigilancia epidemiológica, a los datos históricos de pacientes de enfermedades tropicales y de mayor significación se almacena en libros de Microsoft Office Excel. En los cuales es muy complejo realizar un análisis estadístico o cuantitativo a posteriori, pues no todas las fuentes de información envían los datos en el mismo formato. Esto dificulta muchísimo el trabajo de la dirección municipal, y provincial de Higiene y Epidemiología y más aún del Vice Ministerio, al cual llega información de todas partes del país, haciendo complicado asignar recursos a los lugares que más lo necesitan, presentar informes a instancias superiores y obrar con precisión en casos de epidemias. Siendo esta la **situación problémica**.

De ahí que el **problema científico** sea ¿Cómo viabilizar la gestión de la información referente a la vigilancia epidemiológica al viajero y a las enfermedades tropicales en Cuba? En aras de darle solución a dicha problemática el **objeto de estudio** está enfocado en el proceso de gestión de la información referente al control sanitario internacional cubano, y su **campo de acción,** en el proceso

de gestión de la información referente a la vigilancia epidemiológica al viajero y a las enfermedades tropicales en Cuba.

Para dar solución al problema planteado, se determina como el **objetivo general** de este trabajo implementar una aplicación Web que permita viabilizar la gestión de la información relacionada con la vigilancia epidemiológica al viajero y a las enfermedades tropicales en Cuba. Como **resultado esperado** se debe obtener la implementación de las funcionalidades y diseño de la base de datos (BD) del módulo Higiene y Epidemiología (HE) del sistema Control Sanitario Internacional.

Para poder llevar a la práctica estos posibles resultados y dar cumplimiento a los objetivos trazados, se desarrollarán las siguientes **tareas de la investigación**:

1 Analizar las herramientas y tecnologías a utilizar para el desarrollo de la aplicación.

2 Obtener modelos de implementación y despliegue.

3 Realizar el diseño de la base de datos.

4 Seleccionar estilos y patrones de arquitectura.

5 Realizar la implementación del módulo.

6 Integrar la aplicación con componentes existentes en el Sistema de Información para la salud (SISalud).

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

Capitulo 1: Fundamentación Teórica: Se enmarca en el contexto de las nuevas tecnologías de la informática, las tendencias actuales de herramientas y tecnologías que serán utilizadas para el desarrollo de la aplicación que se pretende desarrollar.

Capítulo 2: Elementos de arquitectura de software: Aborda los elementos fundamentales de la arquitectura sobre la que se desarrollará el sistema.

Capítulo 3: Descripción y análisis de la solución propuesta: Describe la propuesta de solución al problema científico, anteriormente expuesto, se describen las clases que intervienen para formar el software, así como el rol que desempeñan en él, se describirá la estructura de la BD y otros aspectos de interés en materia del desarrollo del software.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

En este capítulo se tratan los temas relacionados con las tendencias, herramientas y técnicas que permiten realizar de forma más segura, cómoda y económica el desarrollo de la solución informática que se pretende llevar a cabo. Aborda cada uno de estos aspectos justificando en cada caso por qué fue escogida la herramienta o tecnología, sin descartar, por supuesto, sus posibles defectos y contrariedades.

§ 1.1 TENDENCIAS ACTUALES.

El desarrollo de software es una actividad muy dinámica y muy valorada por las empresas, las cuales cada día demandan más y mejores aplicaciones para soportar su operación cotidiana. La tendencia actual en el desarrollo de software está fuertemente orientada a la web, dadas las ventajas que este medio presenta, como son la accesibilidad desde cualquier equipo que cuente con conexión y un navegador, lo que se compara muy favorablemente con las aplicaciones tradicionales que requieren una instalación especial en cada uno de los equipos cliente.

La web actual no se entiende como páginas que contienen información y que son consultadas por una masa más o menos grande de personas. La web actual es participativa, la gente no solo es usuaria de las páginas sino que contribuye a enriquecerlas con sus conocimientos, opiniones o simplemente clasificándolas en tags (etiquetas). Ya no es el monopolio de unas cuantas empresas sino que es de todos.

Este cambio en la forma de entender la web ha llevado a nuevas maneras de hacer negocio en Internet y también a nuevas maneras de diseñar y definir la arquitectura de las aplicaciones que serán ofertadas en ella. Todas estas son posibles gracias a una serie de tecnologías nuevas y no tan nuevas, las cuales permiten llevar a la web, aplicaciones que antes solamente podían ser vistas en entornos de escritorio. Esta nueva forma de entenderla permite a las empresas replantear sus desarrollos viendo las aplicaciones web como la solución definitiva a sus necesidades.

En la actualidad existen varias tendencias en el desarrollo de aplicaciones Web que son utilizadas a nivel mundial, entre ellas las más populares son:

1. PHP (§1.5): Principalmente en Servidores UNIX, Linux y puede usarse en Windows aunque no es lo más común.
2. ASP.NET: Tecnología de Microsoft orientada a servidores Windows.
3. JSP: Tecnología multiplataforma también que puede usar UNIX, Linux o Windows.

La Universidad de las Ciencias Informáticas (UCI), como productora de software tiene en consideración estas tecnologías, las cuales están distribuidas en los diversos proyectos de acuerdo con las necesidades de los mismos, de los clientes o de la propia universidad, ya que ninguna de ellas debe escogerse sobre las otras a la ligera; hay que hacer un análisis valorativo de las ventajas y desventajas de cada una y seleccionar la que satisfaga las necesidades específicas de un software determinado.

El Área Temática Sistemas Especializados de la UCI, inmersa en el desarrollo de varios software para la informatización de la Salud Publica en Cuba, ha apostado por el uso de PHP en su versión 5, debido sobre todo a que los software existentes en el país en esta rama de las Ciencias Informáticas, incluyendo el SISalud, y otros de gran importancia, están desarrollados bajo la arquitectura LAMP (§1.2); lo que facilitaría la intercomunicación con los otros sistemas y la ventaja de "hostear" las aplicaciones en los propios servidores del MINSAP.

Para el desarrollo en PHP5 se utilizará un framework que sigue la filosofía del patrón arquitectónico Modelo Vista Controlador (MVC), uno de los más completos y eficaces en la actualidad, se está hablando de CodeIgniter (§1.6), que si bien no es de los más populares entre un gran número de frameworks MVC para PHP que existen en la actualidad, es de muy sencillo aprendizaje, y brinda toda una serie de facilidades a los desarrolladores que serán abordadas más adelante en este capítulo.

Para la programación del lado del cliente, es necesario el uso de HTML, y como es ya muy común, también se trabaja javascript (§1.7), lenguaje muy utilizado por un gran número de aplicaciones. Para llevar a cabo el desarrollo del software con las tecnologías anteriormente expuestas, como entorno de desarrollo (IDE) se escogió la herramienta Zend Studio (§1.8).

Con la finalidad de llevar a la práctica el software HE del proyecto Control Sanitario Internacional, enmarcado en el Área Temática Sistemas Especializados de la UCI, el cual es el eje central de este trabajo de diploma, se utilizan las herramientas y tecnologías mencionadas, en las que se profundiza a continuación.

§ 1.2 PLATAFORMA LAMP (LINUX-APACHE-MYSQL-PHP).

Una plataforma LAMP es un conjunto de herramientas *opensource* (de código abierto, modificable) que trabajando juntas permiten disponer de un servidor web. Las herramientas que aglutina el término LAMP son Linux, Apache, MySQL y PHP, Perl, o Python. Así se consigue agrupar todo lo que debe tener una plataforma web: un sistema operativo, un servidor web, una base de datos, y un lenguaje de programación.

Está considerada como una de las mejores herramientas disponibles para que cualquier organización o individuo pueda emplear un servidor web versátil y potente. Aunque creados por separado, cada una de las tecnologías que lo forman dispone de una serie de características comunes. Especialmente interesante es el hecho que estos cuatro productos pueden funcionar en una amplia gama de hardware, con requerimientos relativamente pequeños sin perder estabilidad. Esto ha convertido a LAMP en la alternativa más adecuada para pequeñas y medianas empresas.

Algunas de las ventajas que se obtienen de utilizar LAMP son:

- Soporte a gran cantidad de arquitecturas, como son Intel y compatibles, SPARC, Mips y PPC (Macintosh).
- Código relativamente sencillo y con pocos cambios de una plataforma a otra.
- Parches generados en poco tiempo después de encontrarse un agujero de seguridad.
- Actualizaciones del software vía Internet.
- Posibilidad de incrementar los servicios y funciones desde el código fuente.
- Es totalmente gratuita y de código abierto, sujeta a licencias GNU/GPL y Apache.

Sin embargo, se tiene también una serie de desventajas que deben considerarse:

- Es muy distinto de Windows, lo que dificulta el trabajo a quienes estén acostumbrados a él.
- Las actualizaciones requieren en ocasiones tener conocimientos profundos del sistema.
- Configurar algunos servicios de red requiere de más tiempo que en Windows.
- Mayor coste en materia de recursos humanos.

Todos los elementos que forman LAMP son software libre, de modo que disfrutan de las siguientes ventajas propias del mismo:

- **Libertad de copia y distribución.** Se puede conseguir gratuitamente en Internet. Hay muchísimas fuentes donde conseguir cualquiera de las distribuciones. Si no se tiene una conexión rápida, también regalan Linux en los CD-ROM de muchas revistas especializadas.
- **Libertad de modificación.** Junto a los programas ejecutables, se puede obtener su código fuente. Esto, si se tienen los conocimientos necesarios, permite verificar la seguridad y eficiencia de los mismos, además de modificar y/o añadir las características y comportamientos que se desee. (1)

§ 1.3 SERVIDOR DE APLICACIONES APACHE.

Apache es el servidor web por excelencia, con algo más de un 60% de los servidores de Internet confiando en él (2). Entre sus características más sobresalientes están:

- **Fiabilidad:** Alrededor del 90% de los servidores con más alta disponibilidad funcionan con Apache (3).
- **Gratuidad:** Apache es totalmente gratuito, y se distribuye bajo la licencia Apache Software License (4), que permite la modificación del código.
- **Extensibilidad:** Se pueden añadir módulos para ampliar las ya de por sí amplias capacidades de Apache. Hay una gran variedad de módulos, que permiten desde generar contenido dinámico (con PHP, Java, Perl, Pyton,...), monitorizar el rendimiento del servidor, atender peticiones encriptadas por SSL, hasta crear servidores virtuales por IP o por nombre (varias direcciones web son manejadas en un mismo servidor) y limitar el ancho de banda para cada uno de ellos. Dichos módulos incluso pueden ser creados por cualquier persona con conocimientos de programación

Este potente y famoso servidor se basa en el pionero NCSA server, y surgió a partir de diferentes ampliaciones y parches para el mismo (de ahí su nombre, derivación de 'A patchy server'), cuyo desarrollo se estancó a mediados de 1994. Un grupo de administradores web pusieron en marcha una lista de correo y fundaron el Apache Group. Al año, Apache era el numero 1 en la lista de Netcraft.

En la figura 1 se muestra la lista de Netcraft de la comparativa de servidores web hasta mayo de 2008.

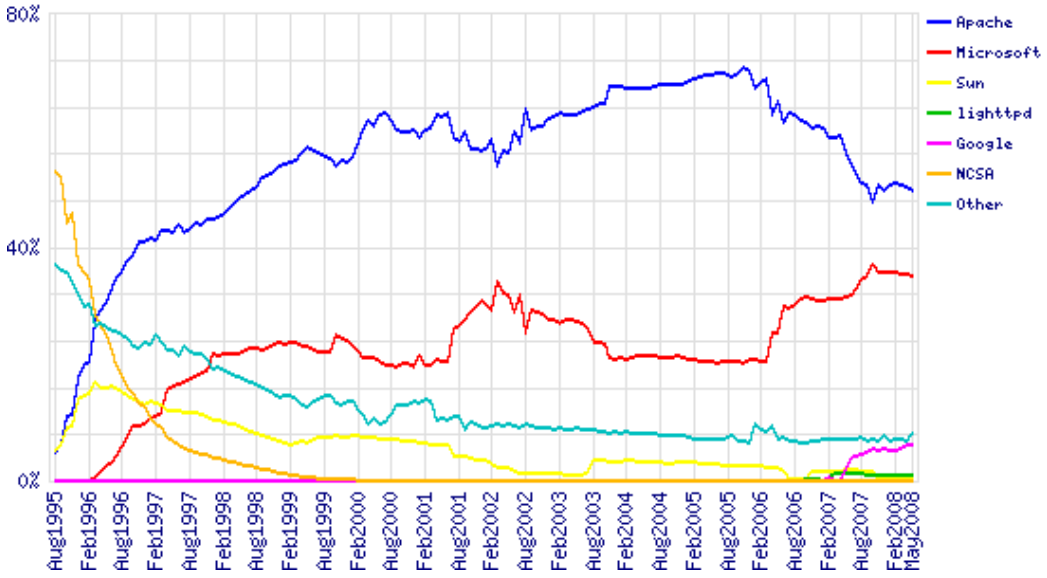

FIGURA 1. COMPARATIVA ENTRE SERVIDORES WEB HASTA MAYO DE 2008. (2)

§ 1.4 Servidor de Bases de Datos MySQL.

La administración y gestión de la información es uno de los puntos clave del éxito en cualquier entidad empresarial. La informática aporta la tecnología que permite satisfacer la necesidad de control de esta información, pero las empresas no se conforman trabajando con aplicaciones o programas que amontonen la información de forma caótica. Los datos deben organizarse de acuerdo a un proceso previo que comprende el análisis y diseño del modelo de datos, así como la elección y posterior configuración del sistema que soportará la BD.

Existen diferentes arquitecturas para los sistemas de gestión de bases de datos, pero la más extendida, y la que más éxito ha tenido, es la arquitectura relacional. MySQL es un servidor de bases de datos relacionales muy rápido y robusto. Es software libre, publicado bajo la licencia GPL (GNU Public License).

Este gestor se creó con la rapidez en mente, de modo que no tiene muchas de las características de los gestores comerciales más importantes, como Oracle, Sybase o SQL Server. No obstante, eso no ha impedido que sea el más indicado para aplicaciones que requieren muchas lecturas y pocas escrituras y no necesiten de características muy avanzadas, como es el caso de las aplicaciones web. MySQL está disponible para un enorme número de sistemas operativos: AIX, BSDi, DEC Unix,

FreeBSD, HP-UX, Linux, Mac OS X, NetBSD, OpenBSD, OS/2 Warp, SGI Irix, Solaris, SCO OpenServer, SCO UnixWare, Tru64 Unix, Windows 95, 98, NT, 2000 y XP.

MySQL fue comprado en febrero de 2008 por la compañía estadounidense con sede en Santa Clara (Silycon Valley, California) Sun Microsystems, cuando esta no se encontraba en su mejor momento económicamente hablado. A pesar de las amplias especulaciones que esto generó sobre si Sun lo convertiría en software propietario o no. El interés de la compañía en este gestor de datos era poder beneficiarse a través de soporte y mantenimiento, además de aumentar las ventas de servidores y almacenamiento a los usuarios de la plataforma.

Sun señala que se han vendido en total 80.000 servidores durante el tercer trimestre del mismo año, lo que significa un 6 % más que el mismo trimestre del año anterior. (5)

A continuación se muestran algunas de las características fundamentales que hacen de MySQL uno de los más potentes gestores de datos en la actualidad. (6)

§ 1.4.1 Interioridades y portabilidad.

- Escrito en C y en C++.
- Probado con un amplio rango de compiladores diferentes.
- Funciona en diferentes plataformas.
- Usa GNU Automake, Autoconf, y Libtool para portabilidad.
- APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
- Uso completo de multi-threaded mediante threads del kernel. Pueden usarse fácilmente múltiples CPUs si están disponibles.
- Proporciona sistemas de almacenamiento transaccional y no transaccional.
- Usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice.
- Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.
- Un sistema de reserva de memoria muy rápido basado en threads.
- Joins muy rápidos usando un multi-join de un paso optimizado.
- Tablas hash en memoria, que son usadas como tablas temporales.

- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.
- El código MySQL se prueba con Purify (un detector de memoria perdida comercial) así como con Valgrind, una herramienta GPL.
- El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado (linkado) en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.

§ 1.4.2 Seguridad.

- Un sistema de privilegios y contraseñas, muy flexible y seguro, que permite verificación basada en el host. Las contraseñas son seguras porque todo el tráfico de ellas está encriptado cuando se conecta con un servidor.

§ 1.4.3 Escalabilidad y límites.

- Soporte a grandes bases de datos. Se usa MySQL Server con bases de datos que contienen 50 millones de registros. También se conocen usuarios que usan MySQL Server con 60.000 tablas y cerca de 5.000.000.000.000 de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).Un índice puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB, o TEXT.

§ 1.4.4 Conectividad.

- Los clientes pueden conectar con el servidor MySQL usando sockets TCP/IP en cualquier plataforma. En sistemas Windows de la familia NT (NT, 2000, XP, o 2003), los clientes pueden usar named pipes para la conexión. En sistemas Unix, los clientes pueden conectar usando ficheros socket Unix.
- En MySQL 5.0, los servidores Windows soportan conexiones con memoria compartida si se inicializan con la opción --shared-memory. Los clientes pueden conectar a través de memoria compartida usando la opción --protocol=memory.

- La interfaz para el conector ODBC (MyODBC) proporciona a MySQL soporte para programas clientes que usen conexiones ODBC (Open Database Connectivity). Por ejemplo, puede usar MS Access para conectar al servidor MySQL. Los clientes pueden ejecutarse en Windows o Unix. El código fuente de MyODBC está disponible. Todas las funciones para ODBC 2.5 están soportadas, así como muchas otras.
- La interfaz para el conector J MySQL proporciona soporte para clientes Java que usen conexiones JDBC. Estos clientes pueden ejecutarse en Windows o Unix.

§ 1.4.5 Localización.

- El servidor puede proporcionar mensajes de error a los clientes en muchos idiomas.
- Soporte completo para distintos conjuntos de caracteres, incluyendo latin1 (ISO-8859-1), german, big5, ujis, y más. Por ejemplo, los caracteres escandinavos 'â', 'ä' y 'ö' están permitidos en nombres de tablas y columnas. El soporte para Unicode está disponible
- Todos los datos se guardan en el conjunto de caracteres elegido. Todas las comparaciones para columnas normales de cadenas de caracteres son case-insensitive.
- La ordenación se realiza acorde al conjunto de caracteres elegido (usando colación Sueca por defecto). Es posible cambiarla cuando arranca el servidor MySQL. Soporta diferentes conjuntos de caracteres que deben ser especificados en tiempo de compilación y de ejecución.

§ 1.4.6 Clientes y herramientas.

- MySQL server tiene soporte para comandos SQL para chequear, optimizar, y reparar tablas. Estos comandos están disponibles a través de la línea de comandos y el cliente **mysqlcheck**. MySQL también incluye **myisamchk**, una utilidad de línea de comandos muy rápida para efectuar estas operaciones en tablas MyISAM.
- Todos los programas MySQL pueden invocarse con las opciones --help o -? para obtener asistencia en línea.

§ 1.5 PHP Y PHP5.

Entre las muchas cosas que distinguen la web de los restantes medios de comunicación, está la capacidad de interacción. En este ámbito, las capacidades del HTML, Javascript y demás tecnologías de cliente son bastante reducidas. Una página realmente profesional no puede limitarse a mostrar información y disponer de formularios para conectarse con los usuarios. Esta necesidad se comprendió muy pronto y provocó el nacimiento del protocolo CGI que permite a los navegadores comunicarse con programas alojados en el servidor.

Con los años, no obstante, se comenzaron a percibir diversos problemas con respecto a los CGIs, entre los cuales el menor no era su complejidad. La popularidad de Javascript o Perl llevó a muchas cabezas pensantes a considerar el uso de los lenguajes de script para ejecutar tareas en el servidor. Así nacieron tecnologías como ASP, PHP, JSP o ColdFusion. Se pueden apreciar cuales son las diferencias de este lenguaje con respecto a las demás alternativas:

1. Es software libre, lo que implica menores costes y servidores más baratos que otras alternativas, a la vez que el tiempo entre el hallazgo de un fallo y su resolución es más corto. Además, el volumen de código PHP libre es mucho mayor que en otras tecnologías, siendo superado por Perl, que es más antiguo. Esto permite construir sitios realmente interesantes con sólo instalar scripts libres como PHP Nuke (weblog, comunidad o bitácora), osCommerce (comercio electrónico con capacidad multilingüe), eZ publish (sistema de gestión de contenidos), phpBB (foros de discusión) o phpMyAdmin (administración de base de datos MySQL).
2. Es muy rápido. Su integración con la base de datos MySQL, también veloz, le permite constituirse como una de las alternativas más atractivas para sitios de tamaño medio-bajo.
3. Su sintaxis está inspirada en C, ligeramente modificada para adaptarlo al entorno en el que trabaja, de modo que si se está familiarizado con esa sintaxis, PHP o JSP son las opciones más atractivas.
4. Su librería estándar es realmente amplia, lo que permite reducir los llamados 'costes ocultos', uno de los principales defectos de ASP.
5. Es relativamente multiplataforma. Funciona en toda máquina que sea capaz de compilar su código, entre ellas diversos sistemas operativos para PC y diversos Unix. El código escrito en PHP en cualquier plataforma funciona exactamente igual en otra.

6. El acceso a las bases de datos de PHP es muy heterogéneo, pues dispone de un juego de funciones distinto por cada gestor.
7. Es suficientemente versátil y potente como para hacer tanto aplicaciones grandes que necesiten acceder a recursos a bajo nivel del sistema como pequeños scripts que envíen por correo electrónico un formulario rellenado por el usuario.

Es una tecnología con mucho futuro, con cada vez más presencia en Internet. Existen muchísimas páginas a lo largo y ancho del mundo que lo utilizan, como SourceForge (sistema de albergue de proyectos de software libre), Granma (edición digital de un periódico en papel), Yahoo (Portal de noticias, servicio de correos entre otros, de los mas populares que existen) o Sport Area (tienda virtual). Por supuesto hay muchos más; en cuanto se navega un poco la extensión .php suena a conocida.

A continuación se enumeran algunas características que hacen de PHP un lenguaje tan popular y especial. (7)

La comunidad PHP.

PHP tiene una comunidad muy grande de desarrolladores, existen miles de lugares donde se pueden encontrar: documentación, tutoriales, ejemplos de código, foros. Si se tiene un problema con PHP se puede encontrar la respuesta en muchos sitios en donde los usuarios comparten el conocimiento adquirido en el proceso de desarrollo.

Aprender PHP es fácil.

PHP es fácil de aprender comparado con otros lenguajes de programación. El lenguaje es semejante a C y Java pues la sintaxis primaria está basada en Perl. Además si se conoce *Javascript* o *ActionScript* hay cierta semejanza entre estos lenguajes por ejemplo en sus estructuras de control. Otro punto es que PHP tiene librerías especializadas en determinados trabajos por lo cual solo se necesita conocer la sintaxis, aplicarla logrando grandes resultados.

Rendimiento.

El rendimiento de PHP es muy bueno y verdaderamente eficiente, utilizando un servidor modesto se pueden atender millones de peticiones al día. Además de ello si se necesita mejorar este rendimiento Zend Technologies ha desarrollado versiones especiales para incrementarlo.

Bajo Costo.

El precio para utilizar PHP es cero, PHP es gratuito y se puede descargar desde www.php.net.

Es Open Source, modificable.

Es *Open Source,* es decir, que se tiene acceso al código fuente. Si se desea agregar o modificar algo para obtener un funcionamiento de acuerdo a ciertas necesidades puede hacerse con total libertad. Esto a diferencia de las aplicaciones comerciales, en las cuales solo queda esperar versiones mejoradas de la empresa desarrolladora. Este punto es importante también pues teniendo acceso al código miles de desarrolladores detectan *bugs* y van corrigiendo y mejorando PHP, logrando tener una aplicación muy segura y constantemente mejorada.

Librerías Incluidas.

PHP fue diseñada para trabajar sobre la web por ello trae un conjunto muy amplio de funciones para ser utilizadas en diferentes tareas relacionadas con la web. Se puede conectar con bases de datos, conectar a *web services*, parsear XML, enviar email, generar PDFs, generar imágenes, etc. Basadas en estas librerías existes clases implementadas para facilitar el trabajo de los desarrolladores. Otro punto es que hay desarrolladores que agregan librerías especializadas para extender las funcionalidades de PHP.

Portabilidad.

PHP está disponible para la mayoría de sistemas operativos existentes. Desde Unix, Linux, Microsoft Windows, MAC, entre otros. Una vez desarrollada, la aplicación PHP puede funcionar en cualquiera de estos sistemas operativos sin necesidad de modificar el código.

Soporte para Programación Orientada a Objetos (POO).

La versión 5 de PHP está diseñada para soporte de características de POO. Características como herencia, métodos y atributos públicos o privados, clases y métodos abstractos, constructores, interfaces y destructores. Con conocimientos de *C++*, *Java o C#* estas características serán muy familiares y con una sintaxis muy similar.

Soporte para gran variedad de Bases de Datos.

PHP tiene soporte para conectarse a una gran variedad de base de datos como: MySQL, PostgreSQL, mSQL, Oracle, dbm, FilePro, HyperWave, Informix, InterBase, Sybase entre otras. Las base de datos hacen que una aplicación sea más robusta y con este soporte la aplicación puede conectarse con facilidad a cualquier base de datos existente.

Soporte.

Zend Technologies la empresa que patrocina PHP ofrece versiones comerciales con todo el soporte que se puede necesitar.

§ 1.6 CODEIGNITER Y LOS FRAMEWORKS MVC.

El MVC (Modelo Vista Controlador) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web y es muy popular sobre todo en *frameworks* para este tipo de aplicaciones como Maverik y Spring para Java o Maverik.NET para la plataforma de Microsoft. Sobre este patrón se abordará más adelante en el capítulo 2.

Existen varios *frameworks* MVC para PHP como Cake, Symfony, WACT, Seagul, Kenda, CodeIgniter, Zoop, entre otros, todos ellos brindan una gama considerable de herramientas que indudablemente enriquecen y simplifican el trabajo de los desarrolladores.

El CodeIgniter es utilizado por una gran comunidad de usuarios, que crece a un ritmo muy prometedor. En la actualidad se pueden encontrar múltiples foros donde este *framework* crece y se enriquece cada vez más. Existe también una cooperación significativa entre los miembros de la comunidad CodeIgniter, se publican preguntas, y respuestas casi instantáneas aparecen, lo cual representa una gran ventaja. A pesar de tener sus propias características es relativamente sencillo de aprender y utilizar, sobre todo porque cuenta con un manual de usuario muy completo y comprensible aún para programadores no muy experimentados en PHP. (8)

Fue construido para codificadores PHP que necesitan una herramienta de desarrollo fácil para crear aplicaciones web simples y elegantes. Entre sus características se encunetran su compatibilidad con

PHP 4 y PHP 5, soporte para múltiples bases de datos, plantillas, validaciones, no requiere instalación, se puede encontrar una librería con un gran número de clases.

Muchos de los *frameworks* para PHP ofrecen MVC, pero ninguno es tan ligero y flexible como CodeIgniter. Lo mejor de todo es que este puede ser tan sólo VC (Vista Controlador), es decir no fuerza al usuario a utilizar una BD para un desarrollo. Además, su separación de código PHP y código HTML no está basada en un template (plantilla) en particular, puede usarse incluso sin ellos.

Cabe resaltar que su soporte de AJAX (Asynchronous JavaScript And XML), aunque mínimo, es lo suficiente para desarrollar aplicaciones vistosas y ágiles.

Sin lugar a dudas CodeIgniter brinda una muy buena implementación del patrón Active Record, independizando al usuario o programador del sistema en el que esté la base de datos. Además brinda una amplia gama de funciones que abstraen de sentencias SQL explícitas y evitan las famosas SQL-Inyections.

§ 1.7 JAVASCRIPT.

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.

La característica principal de Java script, de hecho, es la de ser un lenguaje de scripting, pero, sobre todo, la de ser el lenguaje de scripting por excelencia y, sin lugar a dudas, el más usado. Esta particularidad conlleva una notable serie de ventajas y desventajas según el uso que se le deba dar y teniendo en cuenta la relación que se establece entre el mecanismo cliente-servidor.

Para explicar con pocas palabras dicha relación, podría decirse que el servidor envía los datos al cliente y estos datos pueden llegar en dos formatos diversos: en formato texto (o ASCII) o en formato binario o código máquina: el cliente sabe cómo comprender sólo el formato binario (es decir, la secuencia de 1 y 0), por lo que si los datos llegan en este formato son inmediatamente ejecutables (y, por desgracia, no dejan abierta la posibilidad de efectuar controles), mientras que si el formato es

diverso tienen que ser interpretados y traducidos al formato binario y, por tanto, el cliente necesitará un filtro, o mejor dicho, un intérprete que sepa leer estos datos y los pueda traducir al binario.

Los datos en formato texto son visibles al usuario como simples combinaciones de caracteres y de palabras y son, por tanto, fáciles de manipular, si bien requieran más tiempo para su interpretación a causa de los distintos pasos y de las transformaciones a las que deben someterse para que el cliente pueda comprenderlos: Los datos en formato binario, sin embargo, son difíciles de comprender por el usuario, pero inmediatamente ejecutables por el cliente ya que no requieren fases intermedias.

¿Cuáles son las ventajas y cuáles las desventajas respectivas de los lenguajes de scripting? (9)

El lenguaje de scripting es seguro y fiable porque está en claro y hay que interpretarlo, por lo que puede ser filtrado; para el mismo Java script, la seguridad es casi total y sólo en su primera versión el CIAC (Computer Incident Advisory Committee) señaló problemas de leve entidad, entre ellos la lectura de la caché y de los sitios visitados, de la dirección e-mail y de los files presentes en el disco. Sin embargo, estos fallos se corrigieron ya en las versiones de Netscape sucesivas a la 2.0.

Los script tienen capacidades limitadas, por razones de seguridad, por lo cual no es posible hacer todo con Java script, sino que es necesario usarlo conjuntamente con otros lenguajes evolucionados, posiblemente más seguros, como Java. Dicha limitación es aún más evidente si se pretende operar en el hardware del ordenador, como, por ejemplo, la fijación en automático de la resolución vídeo o la impresión de un documento.

Un problema importante es que el código es visible y puede ser leído por cualquiera, incluso si está protegido con las leyes del copyright.

El código Java script se ejecuta en el cliente por lo que el servidor no es solicitado más de lo debido; un script ejecutado en el servidor, sin embargo, sometería a éste a dura prueba y los servidores de capacidades más limitadas podrían resentir de una continua solicitud por un mayor número de usuarios.

El código del script debe descargarse completamente antes de poderse ejecutar y ésta es la otra cara de la moneda de lo se ha dicho anteriormente: si los datos que un script utiliza son muchos (por ejemplo, una recopilación de citas que se mostrara de manera casual), el tiempo que tardará en descargarse será muy largo, mientras que la interrogación de la misma base de datos en el servidor sería más rápida. (9)

A pesar de no ser perfecto, javascript se ha convertido en el lenguaje de scripting de lado del cliente sin dudas más popular en la actualidad, es muy difícil encontrar algún sitio o portal que no implemente alguna cosa con javascript, que van desde los menús más simples hasta una tecnología, que aunque nueva, ha abierto un cúmulo de posibilidades en el mundo del desarrollo web, la cual está basada en este lenguaje, se está hablando de AJAX.

§ 1.8 ZEND STUDIO

Se trata de un programa de la empresa Zend, impulsores de la tecnología de servidor PHP, orientada a desarrollar aplicaciones web en este lenguaje.

El programa, además de servir de editor de texto para páginas PHP, proporciona una serie de ayudas que pasan desde la creación y gestión de proyectos hasta la depuración de código. El programa entero está escrito en Java, lo que a veces supone que no funcione tan rápido como otras aplicaciones de uso diario. Sin embargo, esto ha permitido a Zend lanzar con relativa facilidad y rapidez versiones del producto para Windows, Linux y MacOS.

Zend Studio consta de dos partes en las que se dividen las funcionalidades de parte del cliente y las del servidor. Las dos partes se instalan por separado, la del cliente contiene el interfaz de edición y la ayuda. Permite además hacer depuraciones simples de scripts, aunque para disfrutar de toda la potencia de la herramienta de depuración habrá que disponer de la parte del servidor. (10)

Lo más destacable es que contiene una ayuda contextual con todas las librerías de funciones del lenguaje que asiste en todo momento ofreciendo nombres de las funciones y parámetros que deben recibir. Aunque esta ayuda contextual no solo se queda en las funciones definidas en el lenguaje, sino que también reporta ayudas con las funciones que se vayan creando, incluso en páginas que se tengan incluidas con la función include().

Otras ayudas que ofrece a la hora de escribir son las típicas en editores avanzados, como permitir editar varios archivos, y moverse fácilmente entre ellos, marcar a qué elementos corresponden los inicios y cierres de las etiquetas, paréntesis o llaves, moverse al principio o al final de una función, identificación automática del código, etc.

Zend Studio implementa además unas interesantes opciones para trabajar en grupo, al integrar el sistema de trabajo conocido como CVS.

Sin duda, más de una vez los programadores de PHP se han visto en un problema por no encontrar un error en algún script que está devolviendo resultados inesperados. En estos casos lo que se suele hacer es escribir el contenido de diversas variables en la página web y esperar que den algún indicio del lugar donde está el error.

En estas tesituras Zend Studio dispone de una herramienta muy interesante de *debug* o depuración. Gracias a ella se pueden ejecutar páginas y conocer en todo momento el contenido de las variables de la aplicación y las variables del entorno como las cookies, las recibidas por formulario o en la sesión. Permite colocar puntos de parada de los scripts y realizar las acciones típicas de depuración.

Además de la ventana para visualizar el contenido de las variables, dispone de otras donde muestra la salida script según se va generando, y otra donde se pueden ver las alertas y errores. Las posibilidades se completan con distintos tipos de depuración, en local, en remoto a partir de una URL. (11)

Conclusiones

En aras de desarrollar un sistema que pretende insertarse en el amplio contexto de los software para la salud y aún más, integrarse a toda una infraestructura que se ha venido desarrollando con la finalidad de constituir la base informatizada de la salud pública cubana se hace imprescindible realizar un análisis de qué herramientas, técnicas y tecnologías se deben poner en práctica para llevar a cabo el software HE. Para ello se escogió la plataforma LAMP formada por la agrupación de Linux, Apache, MySQL y PHP, además de JavaScript que brindarán el soporte tecnológico del software, el cual será desarrollado a su vez con el uso de la herramienta Zend Studio.

Como parte del propio proceso de informatización de la sociedad cubana y como consecuencia del bloqueo económico establecido por Estados Unidos a la isla; las herramientas y tecnologías escogidas para el desarrollo de la aplicación tienen en común que en su mayoría son gratuitas y bajo licencias de software libre, lo cual no las hace para nada menospreciables; tienen una gran calidad, además de toda una comunidad que las utiliza y las respalda, muchas de ellas como Apache y PHP se han extendido a lo largo y ancho de la web y son utilizadas con resultados sorprendentes.

CAPÍTULO 2 ELEMENTOS DE ARQUITECTURA DE SOFTWARE

El presente capítulo está dirigido a proporcionar un contexto general de la arquitectura que sigue el desarrollo del software HE. Se establece un marco teórico que soporta las decisiones y el diseño arquitectónico que constituye la base del software. Asimismo se muestran los recursos que en materia de arquitectura han sido utilizados para diseñar e implementar las funcionalidades del módulo HE.

§2.1 ARQUITECTURA DE SOFTWARE(AS) EN POCAS PALABRAS.

La arquitectura del software proporciona una visión global del sistema a construir. Describe la estructura y la organización de los componentes del software, sus propiedades y las conexiones entre ellos. Los componentes del software incluyen módulos de programas y varias representaciones de datos que son manipulados por el programa. Además, el diseño de datos es una parte integral para la derivación de la AS. La arquitectura marca decisiones de diseño tempranas y proporciona el mecanismo para evaluar los beneficios de las estructuras de sistema alternativas.

De las variadas definiciones que actualmente existen sobre AS, se puede observar que la mayoría se circunscriben en los siguientes conceptos o visiones generales: uno enfocado en el trabajo dinámico de la AS dentro del proceso de ingeniería o el diseño (su lugar en el ciclo de vida de un producto software), otro estático dirigida a la configuración o topología de un sistema contemplado desde un alto nivel de abstracción, y por último el que intenta describir la disciplina que trata los dos primeros. En otras palabras, de una forma u otra la mayoría de las definiciones actuales versan en su esencia en los mismos conceptos.

La definición que se ha acordado como oficial es la provista por el documento de lEEE STD 1471-2000, que expresa:

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

Teniendo en cuenta que la AS se inscribe en el campo de la Ingeniería de Software, vale la pena contrastar sus definiciones, entonces conforme al estándar IEEE 610.12.1990, se entiende como Ingeniería de Software a:

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

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

Ninguna definición de la AS es respaldada unánimemente por la totalidad de los arquitectos; independientemente de las discrepancias entre las diversas definiciones, es común entre todos los autores el concepto de la arquitectura como un punto de vista que concierne a un alto nivel de abstracción. El concepto de abstracción es tomado en el sentido de extraer las propiedades esenciales o examinar selectivamente ciertos aspectos de un problema, posponiendo o ignorando los detalles menos sustanciales o irrelevantes.

La arquitectura no es el software operacional. Es la representación que capacita al ingeniero del software para: I) analizar la efectividad del diseño para la consecución de los requisitos fijados, II) considerar las alternativas arquitectónicas en una etapa en la cual hacer cambios en el diseño es relativamente fácil, y III) reducir los riesgos asociados a la construcción del software.

§2.2 Requisitos no funcionales (RNF).

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.

En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales, es decir una vez se conozca lo que el sistema debe hacer, se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

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. (13)

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.

Una buena definición de requisito no funcional es la dada por Thayer: "(…) es un requisito software que describe no lo que el software hará, sino cómo lo hará, como por ejemplo, requisitos de rendimiento. Los requisitos no funcionales son difíciles de verificar/testear, y por ello son evaluados subjetivamente." (14)

§ 2.2.1 Requerimientos de apariencia o interfaz externa.

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

§ 2.2.2 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.

§ 2.2.3 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.

§ 2.2.4 Requerimientos de portabilidad.

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

§ 2.2.5 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.

§ 2.2.6 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.

§ 2.2.7 Requerimientos de hardware.

En el cliente:

- Procesador Intel Peintium III o superior.
- 2 Gb de memoria RAM.
- Monitor tipo VGA o superior.
- Tarjeta de Red.

En el servidor:

- Procesador Intel Pentium IV o superior.

- 512 Mb de memoria RAM.
- Disco duro de 80 Gb o más.
- Monitor tipo VGA o superior.
- Tarjeta de Red.

§ 2.2.8 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 2.0.
- PHP 5.
- Framework CodeIgniter.
- Servidor de Base de Datos MySQL 5.1.

§ 2.2.9 Requerimientos de diseño e implementación.

- Utilizar los patrones de diseño GRASP.
- Para el análisis y el diseño del sistema debe ser utilizada la metodología RUP, usando el lenguaje de modelación UML y como herramienta para llevarlo a cabo el Visual Paradigm.
- Implementado con el lenguaje de programación php 5
- Desarrollado en Zend Studio.

§ 2.2.10 Requerimientos de ayuda y documentación en línea.

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

§ 2.3 PATRONES Y ESTILOS ARQUITECTÓNICOS.

En la bibliografía referida al estudio de los patrones arquitectónicos se suele llamar de la misma manera a los patrones y estilos, pero la mayoría de los autores los ven de manera separada. Ambos se refieren a formas de estructurar los sistemas y como relacionar los componentes de estos, la diferencia radica en el nivel de abstracción en que se aplican. Los estilos están en un plano de abstracción más alto, definiendo cómo configurar la arquitectura, mientras los patrones están más cercanos al diseño, incluso podría decirse que más cercano a algo físico, pues estos pueden representarse mediante código en un lenguaje de programación determinado.

Vale la pena aclarar la relación entre estilos, patrones de diseño y patrones de arquitectura. Los patrones de diseño de software buscan codificar y hacer reutilizables un conjunto de principios a fin de diseñar aplicaciones de alta calidad. Se aplican en principio solo en la fase de diseño, aunque la comunidad ha comenzado a definir y aplicar patrones a otras etapas del proceso de desarrollo. Los estilos se han aplicado en la fase de análisis arquitectónico en términos de patrones de arquitectura. Sin bien los patrones de arquitectura son similares a los de diseño, los primeros se concentran en la estructura de alto nivel del sistema. Algunos autores sostienen que los patrones arquitectónicos son virtualmente lo mismo que los estilos, pero si bien existen convergencias entre ambos conceptos, los patrones se refieren más a prácticas de reutilización y los estilos conciernen a teorías sobre la estructura de los sistemas.

Algunos patrones coinciden con los estilos hasta en el nombre conque se les designa. Los elementos por los que difieren son mostrados en la siguiente tabla. (15)

Elemento a comparar.	**Estilo Arquitectónico.**	**Patrón Arquitectónico.**
¿Que son?	Son una categorización de sistemas.	Son soluciones generales a problemas comunes.
¿Qué describen?	Sólo describe el esqueleto estructural general para aplicaciones.	Existen en varios rangos de escala, comenzando con patrones que definen la estructura básica de una aplicación.
Dependencia de contexto.	Son independientes del contexto al que puedan ser	Partiendo de la definición de *patrón*, requieren de la especificación de un contexto del

	aplicados.	problema.
Dependencia entre ellos.	Cada estilo es independiente de los otros.	Depende de patrones más pequeños que contiene, patrones con los que interactúa, o de patrones que lo contengan.
Solución.	Expresan técnicas de diseño desde una perspectiva que es independiente de la situación actual de diseño.	Expresa un problema recurrente de diseño muy específico, y presenta una solución para él, desde el punto de vista del contexto en el que se presenta.

TABLA 1. DIFERENCIAS ENTRE ESTILOS Y PATRONES ARQUITECTÓNICOS.

§ 2.3.1 Estilos.

Los estilos arquitectónicos de software son arquitecturas de software comunes, marcos de referencias arquitectónicas, formas comunes o clases de sistemas.

Se entiende por estilos a las entidades que ocurren en un nivel sumamente abstracto, puramente arquitectónico, que no coincide ni con la fase de análisis propuesta por la temprana metodología de modelado orientada a objetos (aunque sí un poco con la de diseño), ni con lo que más tarde se definirían como paradigmas de arquitectura, ni con los patrones arquitectónicos. (16)

A la hora de definir un estilo arquitectónico es necesario tener en cuenta el tipo de aplicación ya que puede imponer restricciones que acotan la interacción de los componentes, además se tiene en cuenta el patrón de organización general.

Algunos de los principales estilos arquitectónicos que se usan en la actualidad están divididos por Clases de Estilos. (17)

Para desarrollar el software HE se decidió utilizar una familia de estilos muy conocida y bien abordada en la bibliografía existente, los estilos de Llamada y Retorno, y de basar toda la arquitectura del software en cuatro estilos pertenecientes a dicho grupo, cuyas características fundamentales se abordan a continuación.

Estilos de llamada y retorno: Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala. El sistema se constituye de un programa principal el que controla del sistema y varios subprogramas que se comunican con éste mediante el uso de llamadas.

Modelo Vista Controlador (MVC): Reconocido como estilo arquitectónico por Taylor y Medvidovic (18), muy rara vez mencionado en los surveys estilísticos usuales, considerado una micro-arquitectura por Robert Allen y David Garlan (19).

El patrón conocido como Modelo-Vista-Controlador (MVC) separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes (20).

Modelo. El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista. Maneja la visualización de la información.

Controlador. Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

En la figura 2 se muestra la interpretación grafica del MVC según (21).

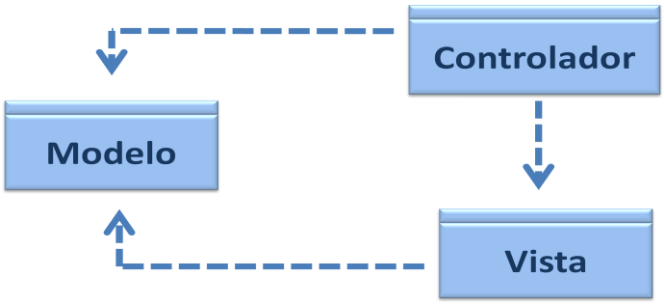

FIGURA 2. EL ESTILO MODELO-VISTA-CONTROLADOR

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases.

Esta separación permite construir y probar el modelo independientemente de la representación visual. La separación entre vista y controlador puede ser secundaria en aplicaciones de clientes ricos y, de hecho, muchos frameworks de interfaz implementan ambos roles en un solo objeto.

En aplicaciones de Web, por otra parte, la separación entre la vista (el browser) y el controlador (los componentes del lado del servidor que manejan los requerimientos de HTTP) está mucho más taxativamente definida.

Ventajas:

- Soporte de vistas múltiples. Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación de Web pueden utilizar el mismo modelo de objetos, mostrado de maneras diferentes.

- Adaptación al cambio. Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo. Este patrón sentó las bases para especializaciones ulteriores, tales como Page Controller y Front Controller.

  Desventajas:

  - Complejidad. El patrón introduce nuevos niveles de indirección y por lo tanto aumenta ligeramente la complejidad de la solución. También se profundiza la orientación a eventos del código de la interfaz de usuario, que puede llegar a ser difícil de depurar. En rigor, la configuración basada en eventos de dicha interfaz corresponde a un estilo particular (arquitectura basada en eventos) que aquí se examina por separado.

  - Costo de actualizaciones frecuentes. Desacoplar el modelo de la vista no significa que los desarrolladores del modelo puedan ignorar la naturaleza de las vistas.

  - Si el modelo experimenta cambios frecuentes, por ejemplo, podría desbordar las vistas con una lluvia de requerimientos de actualización. Hace pocos años sucedía que algunas vistas,

tales como las pantallas gráficas, involucraban más tiempo para plasmar el dibujo que el que demandaban los nuevos requerimientos de actualización.

En el proceso de concepción y diseño del sistema HE apareció MVC como una alternativa arquitectónica muy atractiva y se decidió realizar toda la infraestructura del software sobre este estilo arquitectónico.

El sistema fue diseñado para tener un alto rendimiento y se concibió con la idea de separar la lógica del negocio y el acceso a datos de las interfaces de usuario, evitando que cuando uno de estos se viera afectado, influyera sobre los otros. Se quería tener además las clases agrupadas y clasificadas según sus funcionalidades esto llevó indefectiblemente a MVC; de hecho se podría considerar que la arquitectura del software es básicamente MVC con rasgos en mayor o menor medida de las demás.

Arquitecturas basadas en componentes: Los sistemas de software basados en componentes se basan en principios definidos por una ingeniería de software específica. Los componentes son las unidades de modelado, diseño e implementación. Las interfaces están separadas de las implementaciones, y las interfaces y sus interacciones son el centro de incumbencias en el diseño arquitectónico. Los componentes soportan algún régimen de introspección, de modo que su funcionalidad y propiedades puedan ser descubiertas y utilizadas en tiempo de ejecución.

Ventajas:

- Permite alcanzar un mayor nivel de reutilización de software.
- Permite 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.
- 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.

El software HE trabajará como un componente del SISalud, y para lograr un máximo de eficiencia deberá integrarse a otros software que a su vez son componentes dentro de este sistema.

Arquitecturas orientadas a objetos: Los componentes del estilo se basan en principios orientados a objetos como encapsulamiento, herencia y polimorfismo. Entre las cualidades de este estilo, la más básica concierne a que se puede modificar la implementación de un objeto sin afectar a sus clientes.

Ventaja:

- Permite la estandarización de interfaces de componentes.
- Se puede modificar la implementación de un objeto sin afectar a sus clientes.
- Es posible descomponer problemas en colecciones de agentes en interacción.
- Un objeto es ante todo una entidad reutilizable en el entorno de desarrollo.

Desventaja:

- Los componentes se encuentran fuertemente acoplados. Sólo soportan interacciones atómicas.
- Para poder interactuar con otro objeto a través de una invocación de procedimiento, se debe conocer su identidad.

Arquitectura en capas: Los sistemas o arquitecturas en capas constituyen uno de los estilos que aparecen con mayor frecuencia mencionados como categorías mayores del catálogo, o, por el contrario, como una de las posibles encarnaciones de algún estilo más envolvente. En (22) Garlan y Shaw definen el estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Instrumentan así una vieja idea de organización estratigráfica que se remonta a las concepciones formuladas por el patriarca Edsger Dijkstra en la década de 1960, largamente explotada en los años subsiguientes.

En la práctica, las capas suelen ser entidades complejas, compuestas de varios paquetes o subsistemas. El uso de arquitecturas en capas, explícitas o implícitas, es frecuentísimo; solamente en Pattern Almanac 2000 hay cerca de cien patrones que son variantes del patrón básico de capas. Patrones de uso común relativos al estilo son Façade, Adapter, Bridge y Strategy (23).

En un estilo en capas, los conectores se definen mediante los protocolos que determinan las formas de la interacción. Los diagramas de sistemas clásicos en capas dibujaban las capas en adyacencia, sin conectores, flechas ni interfaces; en algunos casos se suele representar la naturaleza jerárquica del sistema en forma de círculos concéntricos (24).

Las restricciones topológicas del estilo pueden incluir una limitación, más o menos rigurosa, que exige a cada capa operar sólo con capas adyacentes, y a los elementos de una capa entenderse sólo con otros elementos de la misma; se supone que si esta exigencia se relaja, el estilo deja de ser puro y pierde algo de su capacidad heurística; también se pierde, naturalmente, la posibilidad de reemplazar de cuajo una capa sin afectar a las restantes, disminuye la flexibilidad del conjunto y se complica su mantenimiento.

Las formas más rígidas no admiten ni siquiera passthrough: cada capa debe hacer algo, siempre. En la literatura especializada hay multitud de argumentos a favor y en contra del rigor de esta clase de prescripciones. A veces se argumenta que el cruce superfluo de muchos niveles involucra eventuales degradaciones de performance; pero muchas más veces se sacrifica la pureza de la arquitectura en capas precisamente para mejorarla: colocando, por ejemplo, reglas de negocios en los procedimientos almacenados de las bases de datos, o articulando instrucciones de consulta en la capa de la interface del usuario.

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 (25).

En la figura 3 se muestra como se estructuró el software HE siguiendo el estilo de arquitectura en capas, colocando los elementos además, de modo tal que se respete el diseño arquitectónico MVC.

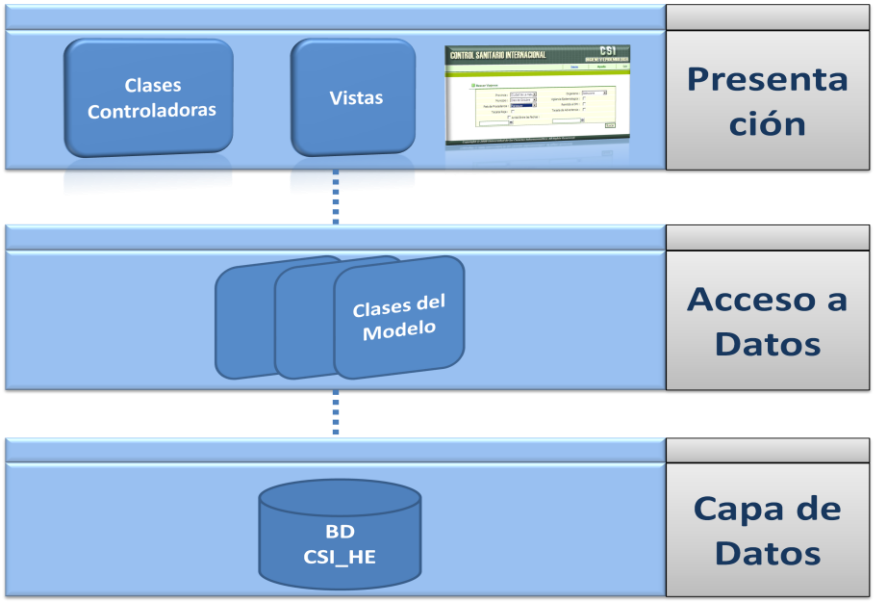

FIGURA 3. LAS CAPAS LÓGICAS EN EL SISTEMA HE.

§ 2.3.2 PATRONES ARQUITECTÓNICOS.

Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos. Los patrones de arquitectura representan el nivel más alto en el sistema de patrones. Ayudan a especificar la estructura fundamental de una aplicación. Cada actividad de desarrollo es gobernada por esta estructura; por ejemplo, el diseño detallado de los subsistemas, la comunicación y colaboración entre diferentes partes del sistema. Cada patrón de arquitectura ayuda a conseguir una propiedad específica en el sistema global. (26)

Entre las ventajas del uso de patrones, se pueden encontrar:

- Permiten la reutilización de soluciones arquitectónicas de calidad.
- Son de gran ayuda para controlar la complejidad de un diseño.
- Facilitan la documentación de diseños arquitectónicos.
- Proporcionan un vocabulario común que mejora la comunicación entre diseñadores.

En la aplicación HE se hace uso de un solo patrón arquitectónico: Modelo Vista Controlador, a pesar de que este fue mencionado como estilo anteriormente, se consideró además como tal, pues define la organización física del sistema y rige prácticas específicas dentro del desarrollo del software. A modo de ilustración se podría decir que las únicas clases que se crearán serán de los tipos controladoras, modelos o vistas.

El utilizar MVC como patrón arquitectónico llevó a escoger CodeIgniter (§1.6) para desarrollar la aplicación debido no solo a las bondades del framework sino que está basado en este.

§ 2.4 MODELO DE DESPLIEGUE.

La vista de despliegue define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos hardware sobre los cuales pueden ejecutarse los elementos software.

En la figura 3 se muestra el diagrama de despliegue para el software HE.

En el servidor de aplicaciones estará desplegado el código fuente del software, es decir todos los ficheros extensión .php que darán lugar al sistema HE. En el servidor del SISalud se encontrarán los componentes o módulos pertenecientes a este sistema informático, los cuales serán utilizados por el sistema HE, 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 entablará una comunicación vía TCP/IP. Los clientes podrán conectarse desde cualquier parte mediante el protocolo HTTP al sistema, permitiendo las interacciones con la información.

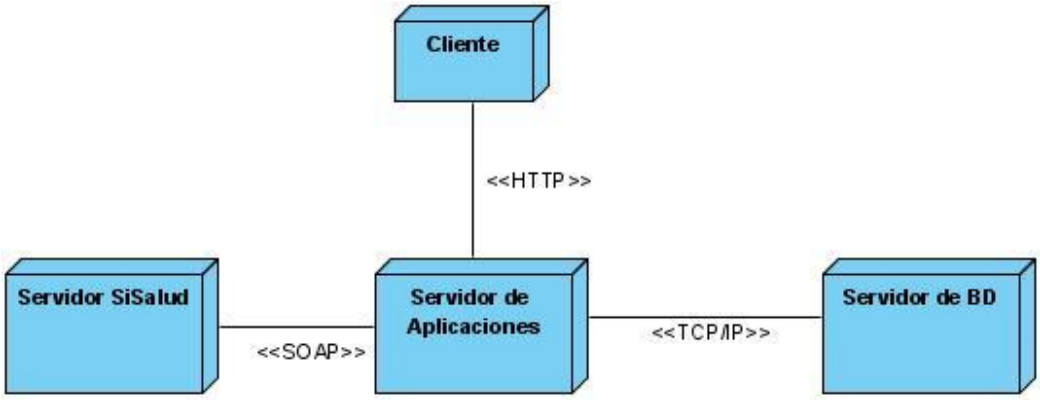

Figura 4. Diagrama de Despliegue.

§ 2.5 MODELO DE IMPLEMENTACIÓN.

La vista de implementación muestra el software como los elementos físicos que lo integran: componentes, ficheros, librerías. Como se muestra en la figura 5, el sistema se estructuró en subsistemas de implementación. Para mayor comprensión de la estructura del modelo de implementación sería bueno analizar dos conceptos, el de subsistema de implementación y el de componentes.

Subsistema de Implementación: Una colección de componentes y otros subsistemas de implementación usados para estructurar el modelo de implementación y dividirlos en pequeñas partes.

Componente: Es la parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces y proporciona la realización de los mismos. Un componente típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios).

El software cuenta con tres subsistemas de implementación primarios encapsulados en el subsistema que constituye el módulo HE, otro externo que contiene los módulos del SISalud. Los que serán reutilizados por la aplicación y una base de datos que brinda el soporte en materia de almacenamiento y manipulación de la información. Los subsistemas de implementación Vistas, Modelo y Controladoras están estructurados de modo tal que agrupan clases cuyo factor común es el rol que desempeñan dentro del patrón arquitectónico MVC (tratado anteriormente en este capítulo).

De este modo en el subsistema Vistas se encuentran encapsulados los componentes que permiten la interacción directa con el usuario final del sistema, que muestran y recogen información a estos usuarios. El subsistema Controladoras contiene todas las clases que manipulan los eventos del usuario y realizan peticiones al modelo para mostrarlas en las vistas. El subsistema Modelo, agrupa las clases que interactúan con la base de datos y velan por el cumplimento de las reglas del negocio. En el caso del subsistema SISalud, este contiene módulos con los que se comunican con el subsistema Controladoras como los registros de ciudadanos, ubicaciones o unidades de salud, (§3.1) que son reutilizados por el sistema HE.

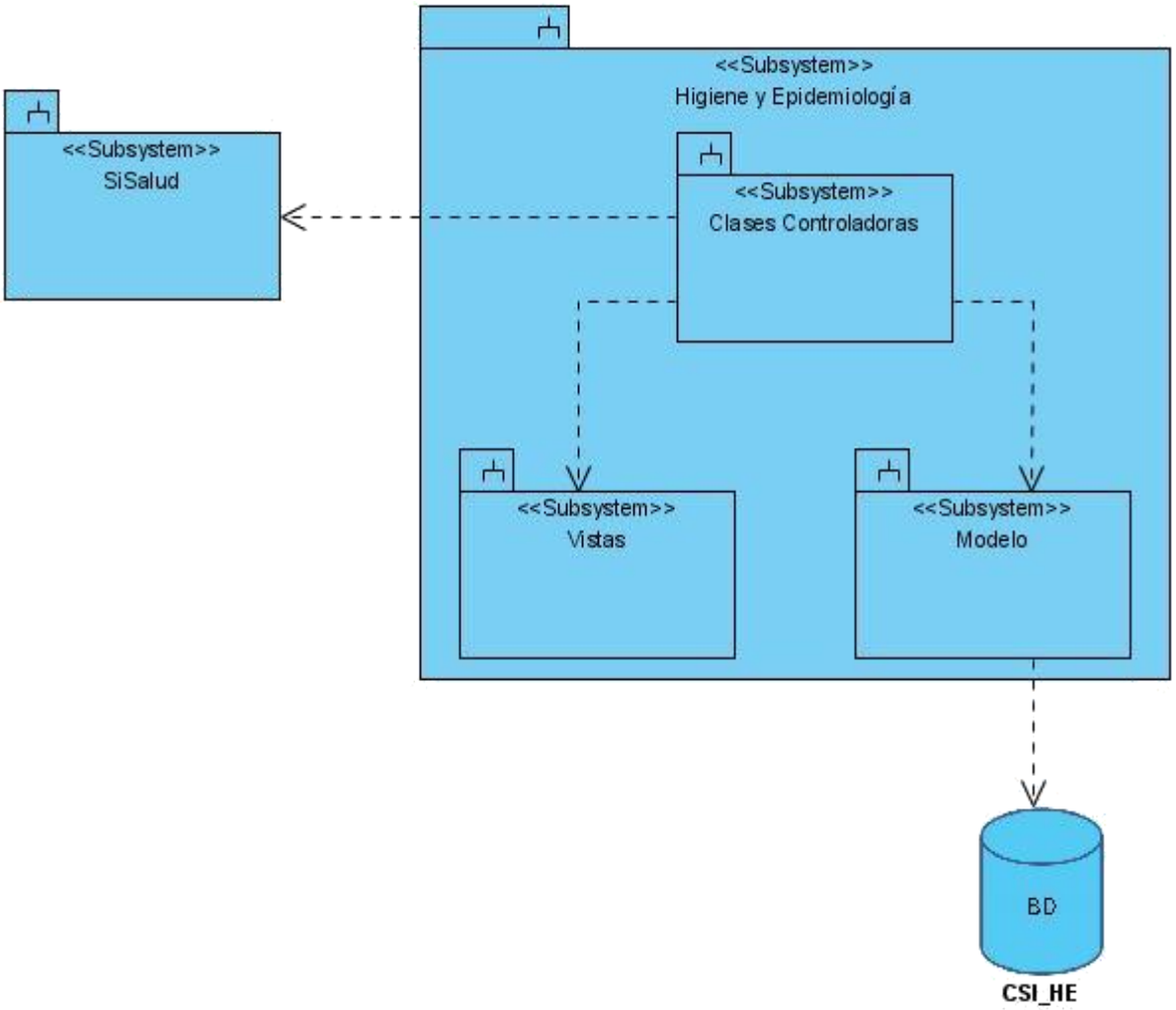

FIGURA 5. DIAGRAMA DE COMPONENTES.

A continuación se muestran los subsistemas de implementación de un modo más detallado. El subsistema SISalud es un ente externo que es reutilizado por la aplicación HE, está dividido a su vez en varios subsistemas de los cuales el software solo interactúa con cinco (§3.1).

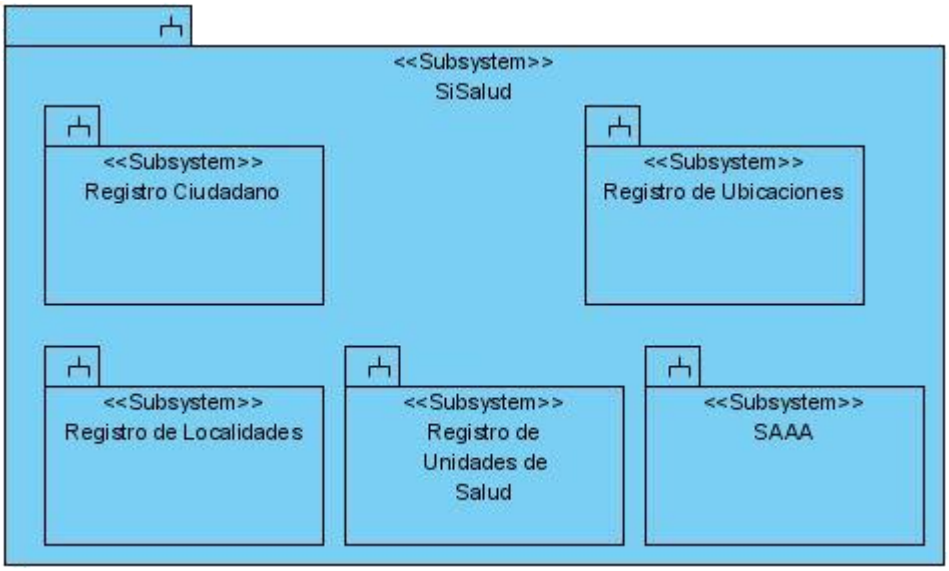

FIGURA 6. SUBSISTEMA SISALUD.

El subsistema Controladoras, es el rector de las actividades de la aplicación, este contiene cinco ficheros que representan componentes, los cuales interactúan con los demás subsistemas coordinando las acciones del software.

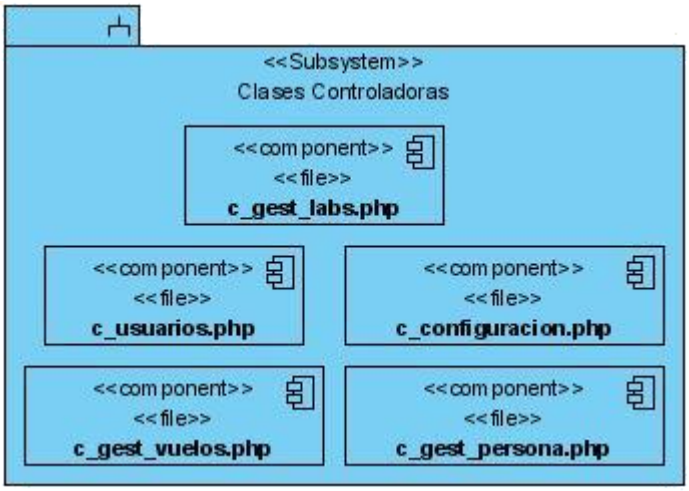

FIGURA 7. SUBSISTEMA CLASES CONTROLADORAS.

El subsistema modelos es el encargado de interactuar con la base de datos, este está dividido en cinco componentes uno por cada componente del subsistema controladoras.

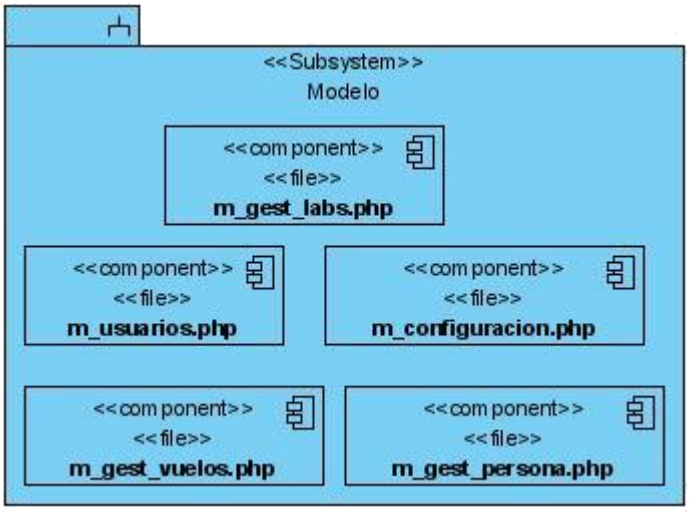

FIGURA 8. SUBSISTEMA MODELO.

El subsistema de vistas se divide a su vez en cinco subsistemas que responden a grupos de componentes con funcionalidades comunes dentro del software HE. Aquí se agrupan los componentes que interactúan con el usuario, estos son manejados por los del subsistema Controladoras.

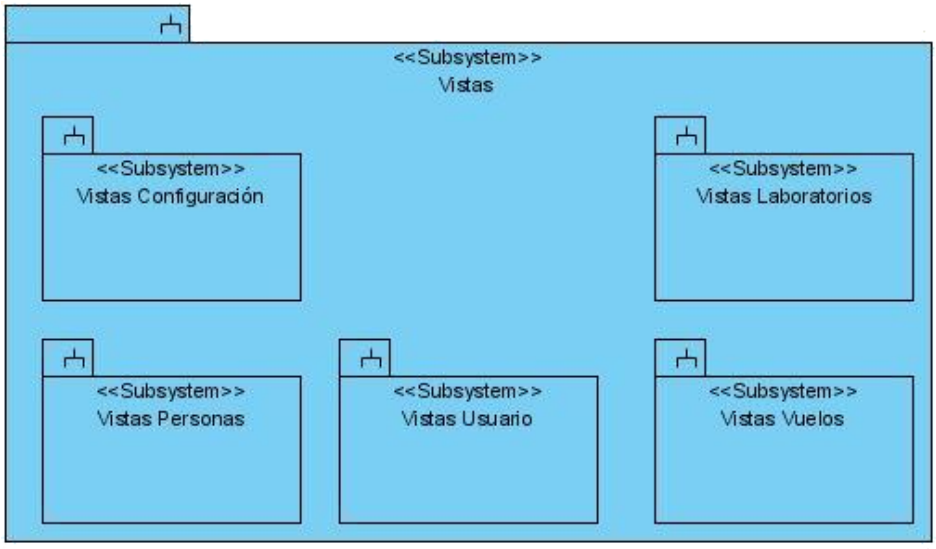

FIGURA 9. SUBSISTEMA VISTAS.

A continuación se describen detalladamente cada uno de estos subsistemas que componen las vistas del software.

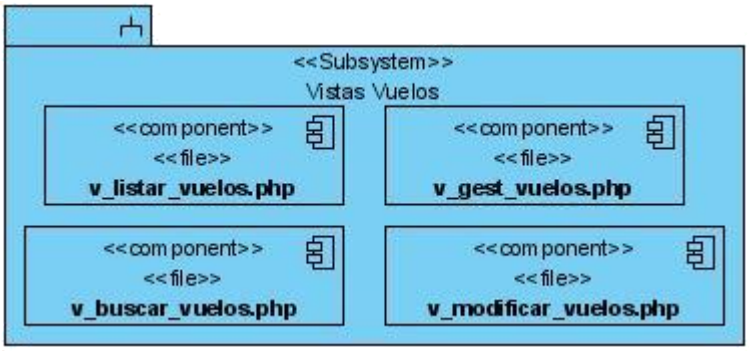

FIGURA 10. SUBSISTEMA VISTAS DE VUELOS.

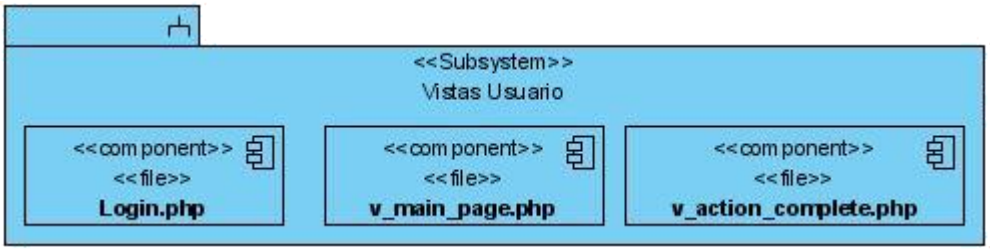

FIGURA 11. SUBSISTEMA VSITAS DE USUARIO.

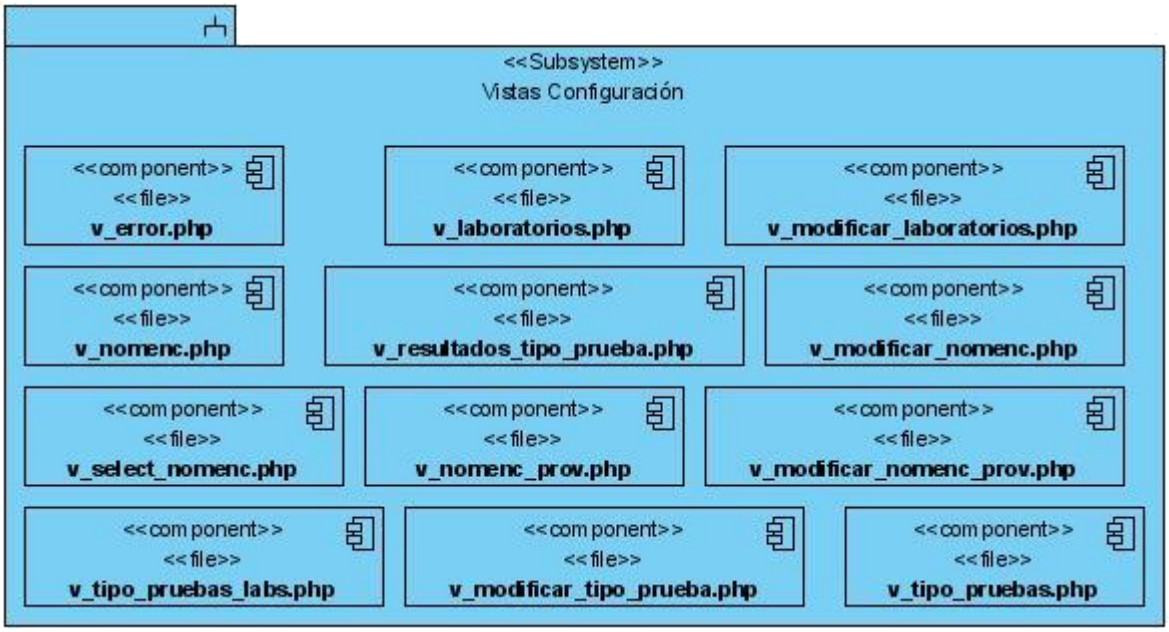

FIGURA 12. SUBSISTEMA VSITAS DE CONFIGURACIÓN.

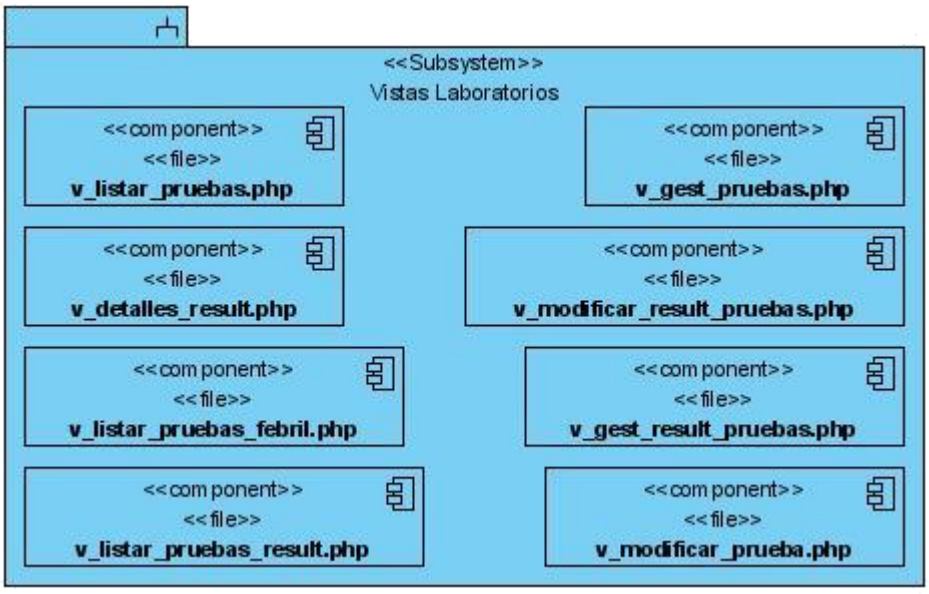

FIGURA 13. SUBSISTEMA VSITAS DE LABORATORIOS.

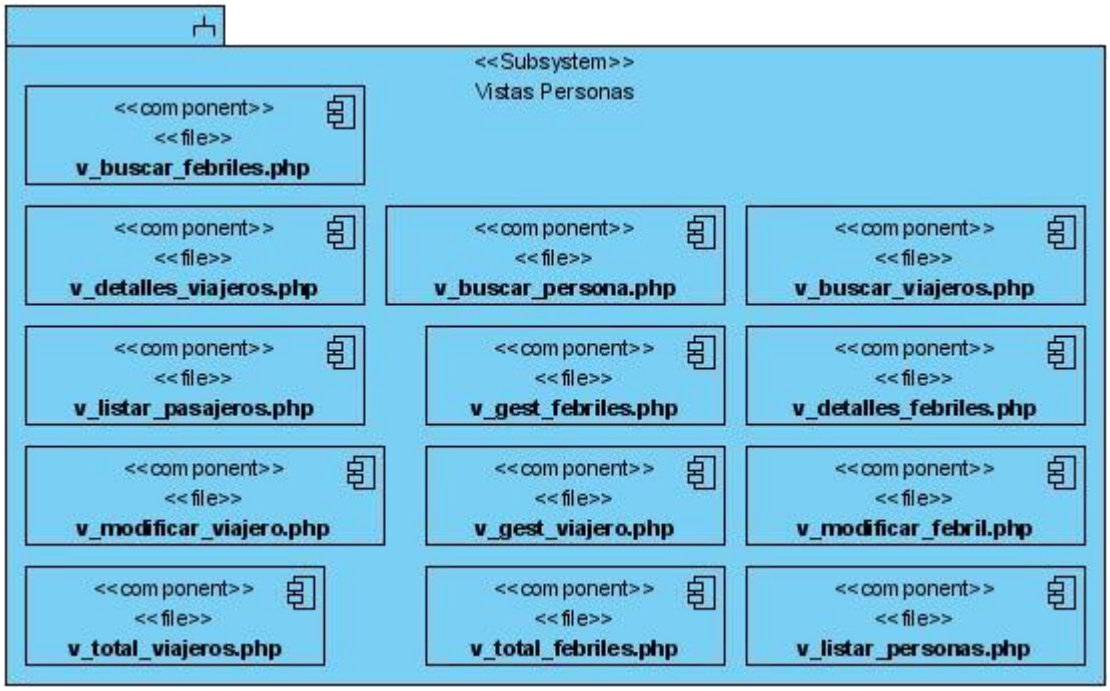

FIGURA 14. SUBSISTEMA VSITAS DE PERSONAS.

Es crucial tener bien definido el modo en que los componentes interactúan entre sí. En el caso del software HE; escrito en el lenguaje de programación PHP, todos los componentes son ficheros ya que este lenguaje no es compilado sino interpretado. Las relaciones que se establecen entre estos ficheros son dependencias. En el diagrama de la figura 15 se ilustran las relaciones entre los componentes a modo general, aquí mismo se puede apreciar de forma práctica como se implementa el patrón arquitectónico MVC (tratado en §2.3).

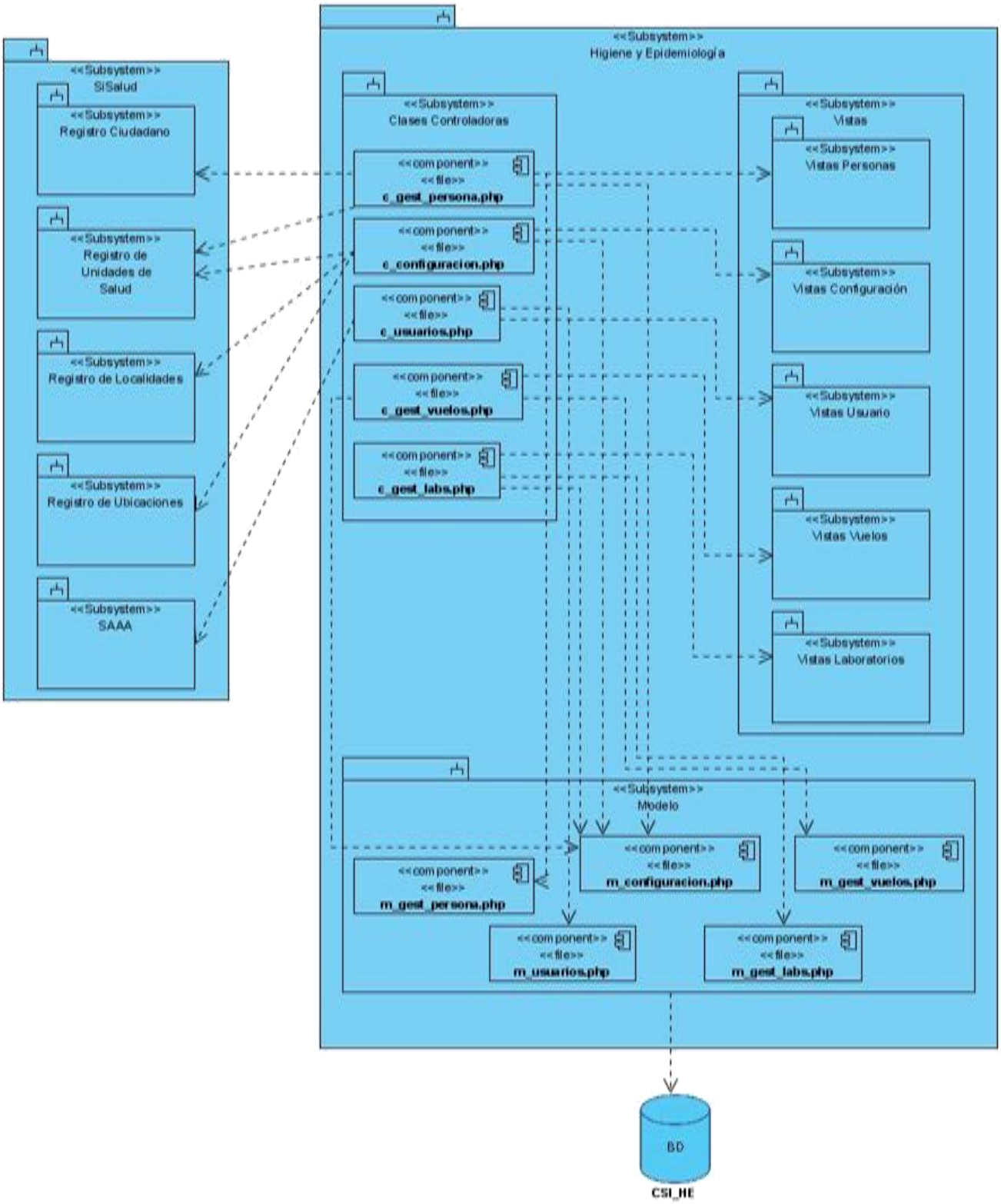

FIGURA 15. RELACIONES ENTRE LOS COMPONENTES.

Distribución física de los componentes.

En la figura 16 se muestra la distribución de los componentes en los nodos físicos.

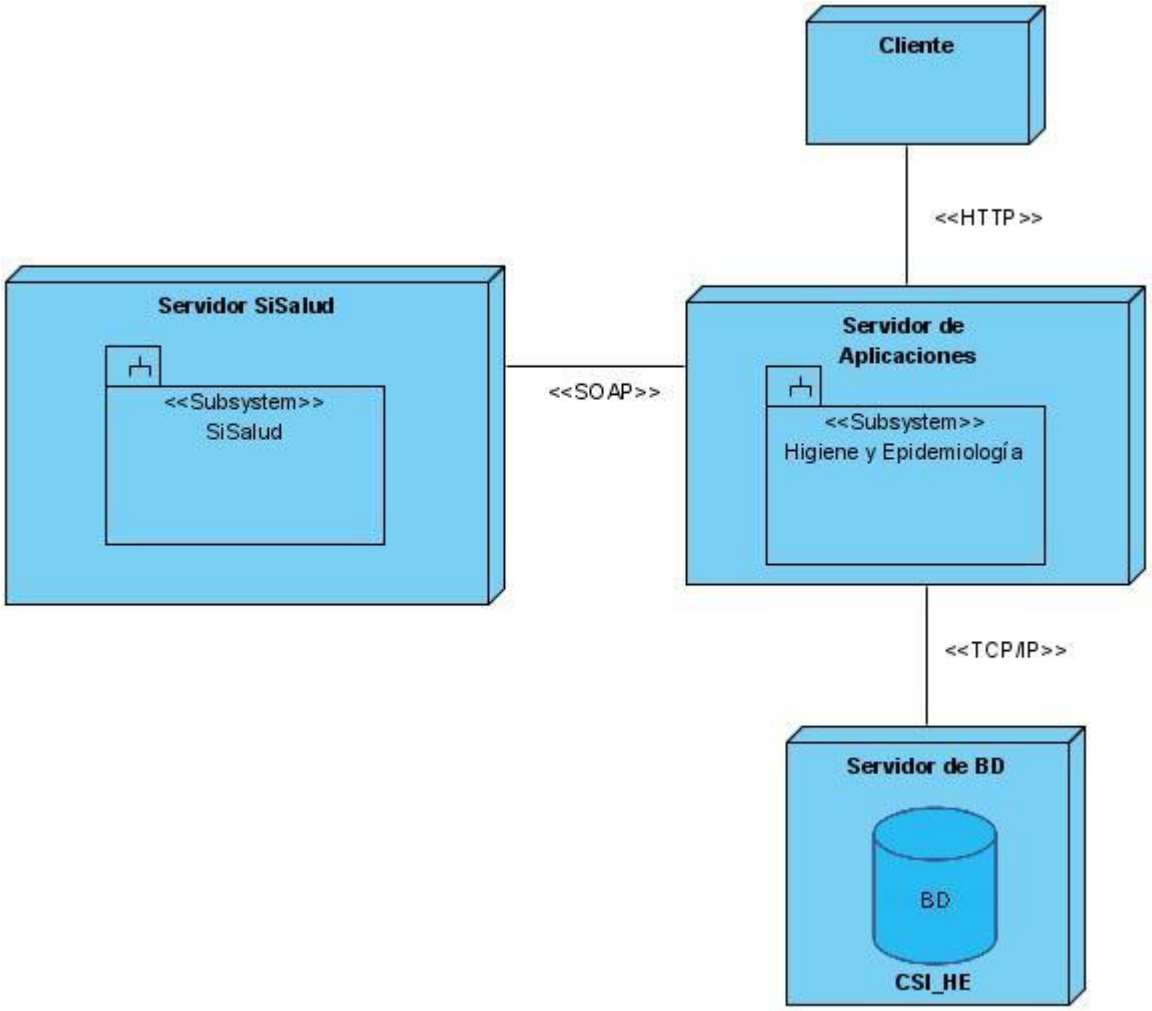

FIGURA 16. VISTA DE DESPLIEGUE REFINADA.

En esta vista del diagrama de despliegue se realiza un refinamiento, ubicando los componentes del software en los nodos donde estarán desplegados. Se puede apreciar que el subsistema primario de implementación Higiene y Epidemiología estará desplegado en el servidor de aplicaciones y se comunicara vía SOAP con el servidor del SISalud. Los clientes no tendrán código de la aplicación.

Conclusiones

En este capítulo, se realizó un análisis arquitectónico devenido en diseño estructural del sistema HE. Se especificaron aspectos teóricos necesarios para realizar dicho diseño, como la elección de patrones y estilos, basados en ellos se dividió el sistema en subsistemas y se especificó cómo estarían ubicados en la práctica a la hora de desplegarlo. Todo esto, trae como consecuencia una mayor comprensión de lo que se quiere y se debe hacer para llevar a la práctica el sistema HE, de modo tal que se satisfagan los requisitos que son el primer escalón en todo este proceso.

CAPÍTULO 3 DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA.

Para llevar a cabo la implementación del sistema HE, se deben tener en cuenta varios aspectos entre los cuales se encuentran el diseño realizado por los analistas, las características de la plataforma de desarrollo sobre la cual se llevará a cabo el sistema, así como, las características de lenguaje de programación y de la BD.

En el presente capítulo se realizará una valoración de los componentes ya existentes en el SISalud que podrían ser reutilizados por la aplicación, además de un recorrido por las clases que soportan las funcionalidades del sistema, los principales métodos de estas y cómo interactúan entre ellas. También se hace un acercamiento al modelo de la BD y sus principales características, se describen las entidades y sus atributos y se muestra el modelo físico.

§3.1 INTEGRACIÓN CON OTROS MÓDULOS O SISTEMAS.

La integración con otros sistemas o módulos de la misma aplicación permite la reutilización de funcionalidades. Como parte de la misma estrategia de utilizar una arquitectura basada en componentes, los sistemas con los que se integra el software HE son tratados como tal, además juegan un papel fundamental en los procesos de negocio que se llevan a cabo.

El software fue concebido para integrarse a toda una solución que está brindando la Universidad de las Ciencias Informáticas en cooperación con la empresa Softel, el SISalud, el cual maneja toda una serie de información que al ser reutilizada por parte del sistema HE evita no solo la redundancia de datos sino también que se pongan en práctica procesos de llenado, edición y mantenimiento de estos por parte de los clientes finales; los cuales solo consumirían esta información, de la que se encargaría totalmente el SISalud. Como es natural se ahorra además tiempo de desarrollo, al no tener los programadores que concentrarse en implementar funcionalidades, sino consumirlas lo cual es mucho más rápido y simple y es la base de la arquitectura basada en componentes.

Los componentes o módulos del SISalud que fueron identificados como necesarios para cumplimentar las funcionalidades que el sistema brinda, así como qué información se necesita de cada uno de estos módulos, se pueden observar en la tabla 2 que se muestra a continuación.

Componente	***Servicios Consumidos***	***Datos Utilizados***	***Justificación***
Sistema de Seguridad (SAAA)	Autenticar	• Idusaurio • Idusuarioexterno • Nombre • Certificado • Nivel • Idnivel	Con el objetivo de conocer el usuario que está autenticado en el sistema, a qué nivel pertenece y qué tipo de usuario es.
Registro de Unidades de Salud (RUS).	BuscarTotal	• Id_unidad • Nombre_unidad • Id_subordinacion • Subordinación • Id_municipio	Para conocer cuál es el nombre de la US en la que se encuentra el usuario y para saber cuál es el nivel al que se subordina.
Registro de Ubicaciones (RU).	ListarProvincias	• Provincias o Id o descri	Para listar todas las provincias.
	ListarMunicipio	• Idmunicipio • Nombre • Provincia o Idprovincia o nombre	Para listar los municipios.
Registro de Ciudadano (RC)	Buscar_Ciudadano	• Buscar_ Ciudadano o Idciudadano o Nombre o Ap1 o Ap2 o Carnetidentidad o Sexo	Para saber los datos de todos los pacientes.
	BuscarTotal_ Ciudadano	• BusquedaTotal_ Ciudadano o Idciudadano	Para saber los datos de un paciente determinado.

		○ Nombre ○ Ap1 ○ Ap2 ○ Carnetidentidad ○ Calle ○ Numero ○ Entre ○ Apto ○ Idmunicipio ○ Sexo	
Registro de Colaboración.	Buscar_Paises	• Buscar_ Pais ○ IdPais ○ NombrePais ○ Region	Para saber los datos de los países.

TABLA 2. COMPONENTES DEL SISALUD PARA LA INTEGRACIÓN

§3.2 DESCRIPCIÓN DE LAS CLASES Y OPERACIONES NECESARIAS.

§3.2.1 CONSIDERACIONES PREVIAS

Las clases que se diseñaron por parte de los analistas y fueron implementadas para dar lugar al sistema HE, así como su estructura y las funcionalidades contenidas en ellas, se deben sobre todo a que el diseño se realizó, no solo para el lenguaje de programación PHP (§1.4), sino y en especial para el framework (o plataforma de desarrollo) en la que se puso en práctica la solución informática, el CodeIgniter (§1.6).

Aunque ya ha sido analizado este framework (§1.6). Es importante, antes de profundizar en las clases del sistema, especificar algunos elementos respecto al mismo.

CodeIgniter, a pesar de ser una plataforma potente y con múltiples funcionalidades fue escrito en PHP4 y aunque soporta la orientación a objetos (OO) y todas las ventajas y novedades del PHP5, es

muy pobre en lo que programación orientada a objetos (POO) se refiere. Además está el hecho de que PHP como lenguaje de scripting no maneja tipos de datos; característica muy significativa de la POO.

Es muy común entonces encontrar en cualquier aplicación desarrollada en CodeIgniter clases que contienen muchos métodos, los cuales por lo general están altamente relacionados. Por ejemplo las relaciones de composición o débiles y las relaciones de herencia se manejan generalmente por una sola clase. A modo de ilustración si se tiene en el diseño una clase entidad *Persona* y otra *Dirección*, y en el software la dirección es una entidad débil de la persona, o sea, no existe en un contexto fuera de esta; en CodeIgniter se haría solo una clase *CPersona* que maneje también la dirección, aún más, si existiera una clase Trabajador que herede de Persona, la misma CPersona sería la encargada de manipular la información de un trabajador.

Esto sucede porque el framework utiliza las clases como encapsuladoras de funcionalidades, y solo se instancian con el objetivo de realizar llamadas a funciones; las clases nunca se utilizan como contenedoras de información o con el objetivo de transferir o manipular objetos.

Tampoco es muy común ver en sistemas desarrollados en CodeIgniter clases entidad, dado que en PHP todas las clases heredan de stdClass normalmente se evita crear un código adicional declarando clases entidades, que eventualmente se podrían tratar del mismo modo que objetos stdClass y se crean objetos de este tipo, lo que ayuda a la transparencia del código y aminora el tiempo de desarrollo.

En este aspecto se podría concluir que la programación en el framework CodeIgniter no es puramente OO, aunque este paradigma se utiliza, permanecen rasgos de programación estructurada, además por las propias características del lenguaje de scripting que es el eje central del framework. No obstante a esto, el sistema fue implementado respetando los patrones GRASP propuestos por los analistas.

§3.2.2 Descripción de las clases.

Para cumplimentar todas las funcionalidades que el sistema requiere y en correspondencia con el modo que el framework estipula, se implementaron cinco clases controladoras y cinco modelos de las cuales se detallan a continuación aquellas que se consideran críticas pues gestionan o representan los procesos fundamentales e imprescindibles del negocio, las clases de configuración y seguridad así como una librería muy significativa para el sistema se muestran en los ***anexos*** Anexo 1. Clases de configuración y seguridad. ***y Anexo*** 2. La librería MY_Controller)***.***

La clase *C_Gest_Vuelos* usa como modelo a *M_GestVuelos* y ambas se encargan de gestionar información referente a los vuelos que arriban a los aeropuertos.

Nombre: C_Gest_Vuelos	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	C_Gest_Vuelos()
Descripción:	Constructor de la clase, por defecto.
Nombre:	Visualizar($data = array())
Descripción:	Método para mostrar la pantalla de gestión de vuelos.
Nombre:	InsertarVuelo()
Descripción:	Método para insertar un vuelo.
Nombre:	ModificarVuelo()
Descripción:	Método para modificar los datos de los vuelos.
Nombre:	EliminarVuelo()
Descripción:	Método para eliminar los datos de un vuelo.
Nombre:	PaisesNoEscalas($vuelo)
Descripción:	Método para saber donde no realizó escala el vuelo.
Nombre:	CargarBuscarVuelos()
Descripción:	Muesta la vista del buscador de vuelos.
Nombre:	CriteriosVuelo()
Descripción:	Establece los criterios de búsqueda de los vuelos en una sesión.
Nombre:	BuscarVuelos()
Descripción:	Método para buscar un vuelo.
Nombre:	DetallesVuelo()
Descripción:	Método para ver los datos de los vuelos.

TABLA 3. CLASE C_GEST VUELOS

Nombre: M_Gest_Vuelos	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Gest_Vuelos()
Descripción:	Constructor de la clase, por defecto.
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 4. CLASE M_GEST_VUELOS

Unas clases que juegan un papel fundamental son las de gestión de personas C_Gest_Persona y M_Gest_Persona, las cuales manipulan toda la información referente a febriles y viajeros, las que son clases hijas de persona.

Nombre: C_Gest_Persona	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	C_Gest_Persona()
Descripción:	Constructor de la clase, por defecto.
Nombre:	BuscarFebril():
Descripción:	Método para buscar los datos de una persona para insertarlo como febril en la BD.
Nombre:	__ConstruirFebril()
Descripción:	Método que recoge los datos que vienen del formulario y devuelve un objeto de tipo febril.
Nombre:	AdicionarFebril():
Descripción:	Método que recibe diferentes parámetros del método InsertarFebril () y devuelve un objeto al método correspondiente en la controladora.
Nombre:	ModificarFebril()
Descripción:	Método para modificar los datos de los febriles.
Nombre:	CambiarFebril()
Descripción:	Método para cambiar al febril por otro que se encuentre en el SISALUD.

Nombre:	EliminarFebril()
Descripción:	Método para eliminar un febril.
Nombre:	CargarBuscarFebriles()
Descripción:	Método que muestra la vista del buscador de febriles
Nombre:	CriteriosFebriles()
Descripción:	Método que establece una sesión para los criterios de búsqueda de forma tal que no se pierdan cuando se refresca la página o cuando se cambia de índice en el paginado.
Nombre:	BuscarFebrilesPor()
Descripción:	Método para buscar febriles según los criterios que hay en la sesión (método CriteriosFebriles) y muestra los resultados de la búsqueda.
Nombre:	DetallesFebril($per,$fech)
Descripción:	Método que busca y muestra todos los datos de un febril dado su id de persona y la fecha de inicio de los síntomas.
Nombre:	CambiarEstadoFebril()
Descripción:	Método que muestra las vistas para cambiar los estados.
Nombre:	NuevoEstadoFebril()
Descripción:	Método que permite cambiar los estados de los febriles.
Nombre:	BuscarViajero()
Descripción:	Método que se encarga de buscar un viajero.
Nombre:	CargarBuscarViajeros()
Descripción:	Método que carga la vista del buscador que le sirve la función BuscarViajerosPor ().
Nombre:	CriteriosViajeros()
Descripción:	Método que establece una sesión para los criterios de búsqueda de forma tal que no se pierdan cuando se refresca la página o cuando se cambia de índice en el paginado.
Nombre:	BuscarViajerosPor()
Descripción:	Método que permite buscar los viajeros por algún criterio de búsqueda para generar los reportes.
Nombre:	AdicionarViajero()
Descripción:	Método que adiciona viajeros a un vuelo.
Nombre:	EliminarViajero()
Descripción:	Método que elimina viajeros de un vuelo.
Nombre:	ModificarViajero()
Descripción:	Método para modificar los datos los viajeros.
Nombre:	CambiarViajero()
Descripción:	Método que se utiliza para cambiar los datos de una persona del registro de ciudadanos por otra en calidad de viajero.
Nombre:	DetallesViajero($id_persona,$id_viaje)
Descripción:	Método que busca y muestra todos los datos de un viajero dado su id de persona y el id

	del viaje.
Nombre:	VistaViajerosVuelo()
Descripción:	Método que muestra la lista de viajeros de un vuelo almacenado en la sesión junto a los datos del vuelo.
Nombre:	CriteriosPersona()
Descripción:	Método que se conecta al SISalud para obtener los datos de las personas.
Nombre:	BuscarPersona()
Descripción:	Método que permite buscar personas en el SISalud.
Nombre:	BuscarPersonaSisSalud($CriteriosBusqueda)
Descripción:	Método que permite buscar personas en el SISalud según criterios de búsqueda.
Nombre:	DetallesPersona($id)
Descripción:	Método que permite ver los datos de las personas.

TABLA 5. LA CLASE C_GEST_PERSONA

Nombre: M_Gest_Persona	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Gest_Persona()
Descripción:	Constructor de la clase, por defecto.
Nombre:	InsertarPersona($persona)
Descripción:	Método para buscar los datos de una persona para insertarlo como febril en la BD.
Nombre:	InsertarViajero($viajero)
Descripción:	Método para insertar viajeros en la BD.
Nombre:	ModificarViajero($viajero)
Descripción:	Método para modificar viajeros en la BD.
Nombre:	CambiarViajero($old_viajero,$new_viajero)
Descripción:	Método para cambiar los datos viejos de un viajero por los nuevos en la BD.
Nombre:	InsertarFebril($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:	ModificarControlFoco($control_foco)
Descripción:	Método para cambiar los datos viejos del control de foco por los nuevos en la BD.
Nombre:	CambiarFebril($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:	InsertarCentroTrab($centro)
Descripción:	Método busca e inserta los centros ocupacionales en la BD.
Nombre:	ModificarCentroTrab($centro)
Descripción:	Método que modifica los centros de trabajo de la BD.
Nombre:	InsertarDireccion($direccion)
Descripción:	Método que permite insertar la dirección en la BD.
Nombre:	ModificarDireccion($direccion)
Descripción:	Método que se encarga de modificar la dirección en la BD.
Nombre:	BuscarDireccion($direccion)
Descripción:	Método para buscar la dirección de las personas.
Nombre:	BuscarIdSexo($sexo):
Descripción:	Método para buscar el sexo de las personas.
Nombre:	BuscarCentroTrab($centro)
Descripción:	Método que permite buscar el centro de trabajo de una persona.
Nombre:	BuscarDirPersona($id_persona)
Descripción:	Método que permite buscar la dirección de una persona dado el id y retorna $dir tipo stdclass (clase dirección) la persona no tiene dirección retorna 0.
Nombre:	BuscarProvPersona($id_persona)
Descripción:	Método que dado el id de una persona devuelve un objeto de tipo provincia.
Nombre:	Dir2Str($direccion)
Descripción:	Dado un objeto de tipo *Dirección* lo devuelve en forma de cadena en un formato dado.
Nombre:	ExistePersona($id_persona)
Descripción:	Método para saber si existe una persona dada.
Nombre:	ExisteViajeroEnViaje($id_viajero,$id_viaje):
Descripción:	Método que verifica si existe un viajero en un viaje dado su id de viajero y el id del viaje.
Nombre:	ExisteFebril($id_persona,$fecha_inicio_sintomas)
Descripción:	Método que verifica si existe un febril dado su id de persona y la fecha de inicio de los síntomas.
Nombre:	GetViajeros($criterios = array(),$per_page = '',$offset = ''):
Descripción:	Método que recibe un arreglo de los criterios de búsqueda y devuelve los datos básicos de los viajeros que los cumplen.
Nombre:	GetTotalViajeros($criterios = array(),$per_page = '', $offset = ''):
Descripción:	Método que muestra los datos de los viajeros.
Nombre:	EliminarViajero($viajero)
Descripción:	Método que se utiliza para eliminar los datos de un viajero dado en la BD.
Nombre:	CambiarEstadoFebril($febril)
Descripción:	Método que se utiliza para cambiar el estado de un paciente por otro en la BD.
Nombre:	GetTotalFebriles($criterios = array(),$per_page = '', $offset = '')

Descripción:	Método que muestra los datos de los febriles.
Nombre:	GetControlFoco($febril)
Descripción:	Método que muestra los datos del control de foco.
Nombre:	GetCtroTrabFebril($febril)
Descripción:	Método que muestra los datos del centro de trabajo.

TABLA 6. CLASE M_GEST_PERSONA

C_Gest_Labs y *M_Gest_Labs* son las clases designadas para el control de pruebas que se realizan en los laboratorios, gestionar los resultados de estas, así como asociarlas a un paciente determinado.

Nombre: C_Gest_Labs	
Tipo de clase: Controlador	
Para cada responsabilidad:	
Nombre:	C_Gest_Labs()
Descripción:	Constructor de la clase, por defecto.
Nombre:	OrientarPruebas()
Descripción:	Muestra una pantalla para orientarle pruebas a un febril.
Nombre:	__CrearPrueba()
Descripción:	Devuelve un objeto tipo *Prueba* creado con los datos procedentes de un formulario.
Nombre:	AdicionarPrueba()
Descripción:	Envía el objeto tipo *Prueba* a la modelo para ser insertado en la BD.
Nombre:	PruebaEnviada($lab = '')
Descripción:	Muestra una pantalla confirmando que la prueba fue enviada y dice a qué laboratorio.
Nombre:	ModificarPrueba()
Descripción:	Envía una Prueba a la modelo para que esta actualice sus datos en la BD.
Nombre:	EliminarPrueba()
Descripción:	Manda a eliminar una prueba a la modelo.
Nombre:	__GetCurrentLab()

Descripción:	Devuelve los datos del laboratorio al que pertenece el usuario que está logueado.
Nombre:	CriteriosPruebasPendientes()
Descripción:	Establece criterios de búsqueda de pruebas pendientes en la sesión.
Nombre:	PruebasPendientesLab()
Descripción:	Retorna una lista de las pruebas pendientes de un laboratorio.
Nombre:	CriteriosPruebasRealizadas()
Descripción:	Establece criterios de búsqueda de pruebas realizadas en la sesión
Nombre:	PruebasRealizadasLab()
Descripción:	Retorna una lista de las pruebas realizadas de un laboratorio.
Nombre:	__CB_Result_Cuali($id_tipo_prueba,$id_result = -1)
Descripción:	Devuelve los posibles resultados de una prueba en formato *formdropdown*.
Nombre:	__CrearResultado()
Descripción:	Crea un objeto tipo Resultado de prueba con datos de un formulario.
Nombre:	InsertarResultados()
Descripción:	Manda a la modelo un objeto Resultado para insertarlo en la BD.
Nombre:	ResultadoRegistrado()
Descripción:	Muestra una pantalla confirmando que el resultado fue guardado.
Nombre:	ModificarResultados()
Descripción:	Manda a la modelo un objeto Resultado para actualizarlo en la BD.
Nombre:	ElimiarResultado()
Descripción:	Manda a la modelo un objeto Resultado para eliminarlo de la BD.
Nombre:	CriteriosPruebasXPaciente()
Descripción:	Establece criterios de búsqueda de pruebas asociadas a un paciente en la sesión.
Nombre:	PruebasXPaciente()
Descripción:	Muestra las pruebas asociadas a un Paciente.
Nombre:	ResultadosPrueba()

Descripción:	Muestra los resultados de una prueba.

TABLA 7. CLASE C_GEST_LABS

Nombre: M_Gest_Labs	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Gest_Labs()
Descripción:	Constructor de la clase, por defecto.
Nombre:	InsertarPrueba($prueba,$lab)
Descripción:	Inserta una prueba en la BD.
Nombre:	ModificarPrueba ($prueba,$lab,$tipo_prueba_old)
Descripción:	Modifica los datos de una prueba en la BD.
Nombre:	EliminarPrueba ($prueba)
Descripción:	Elimina una prueba de la BD.
Nombre:	EliminarResultado ($prueba)
Descripción:	Elimina un resultado de la BD.
Nombre:	PuedeRealizarsePrueba ($prueba)
Descripción:	Determina si los datos de una prueba cumplen con las reglas del negocio.
Nombre:	PuedeRealizarsePruebaEnLab ($prueba,$lab)
Descripción:	Determina si una prueba se puede realizar en un laboratorio.
Nombre:	GetPruebas ($criterios = array(), $per_page = '',$offset = '')
Descripción:	Devuelve una lista con todas las pruebas según algún criterio de búsqueda.
Nombre:	InsertarResultados ($resultado)
Descripción:	Inserta un resultado en la BD.
Nombre:	ModificarResultado ($resultado)
Descripción:	Modifica los datos de un resultado en la BD.

Nombre:	GetResultados ($criterios = array(), $per_page = '',$offset = '')
Descripción:	Devuelve una lista con todos los resultados de las pruebas según algún criterio de búsqueda.
Nombre:	TipoPruebaResultCuali($id_tipo_prueba)
Descripción:	Dado un tipo de prueba devuelve los posibles resultados cualitativos.

TABLA 8. CLASE M_GEST_LABS

§3.3 DISEÑO DE LA BASE DE DATOS.

§3.3.1 DIAGRAMA ENTIDAD RELACIÓN.

Para llevar a la práctica (y sobre todo, con éxito), un sistema software, una de las cosas más importantes y que más tiempo toma, es el diseño de las bases de datos. De aquí parte prácticamente la construcción del sistema y los errores en el modelo de datos definitivamente influirán en la constitución y rendimiento del mismo. En el caso especial de los software de gestión, como es el sistema HE, la manipulación correcta y oportuna de la información contenida en la base de datos, la cual, por lo general es bastante voluminosa, es crucial.

Dadas las características del sistema, el modelo de datos o entidad relación puede fue dividido en submodelos que ayudan a simplificar la comprensión de la estrategia trazada para el almacenamiento y manipulación de los datos, estos son: uno para el control de viajeros, otro para el control de enfermedades y otro para la seguridad.

§3.3.1.1 MODELO LÓGICO.

El modelo lógico es prácticamente un diagrama de clases persistentes, expresa las clases o entidades que intervienen el la futura creación de la base de datos así como las relaciones entre ellas, aunque lo hace de forma tal que quede claro cuáles atributos de estas entidades serán llaves primarias de las tablas y cuáles foráneas, lo cual es más práctico.

El modelo lógico de la base de datos para el sistema HE cuenta con 28 entidades 128 atributos 28 llaves y 36 relaciones entre las entidades.

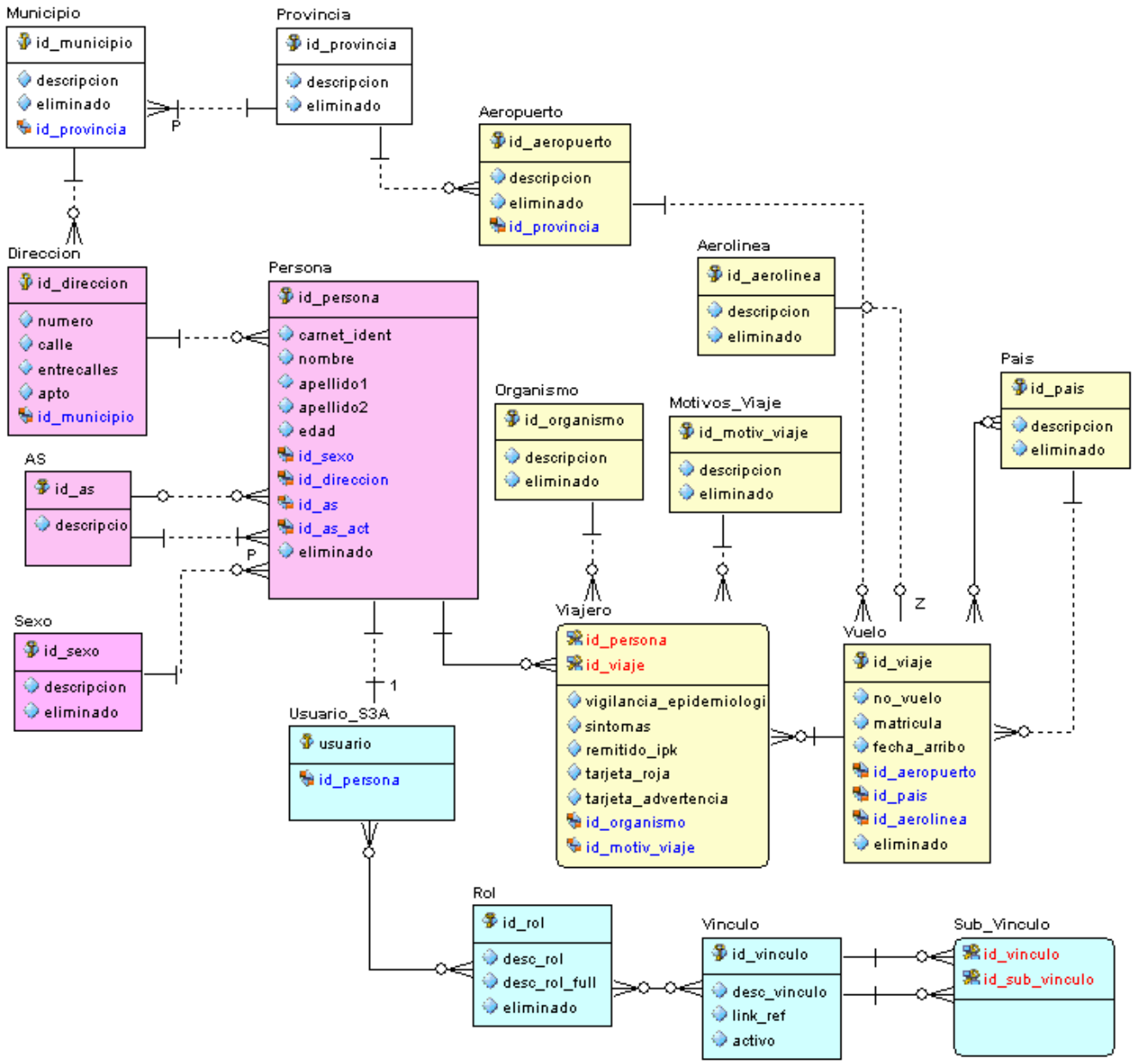

FIGURA 17. SUBMODELO CONTROL DE VIAJEROS Y SUBMODELO SEGURIDAD(A PARTIR DE LA ENTIDAD USUARIO_S3A)

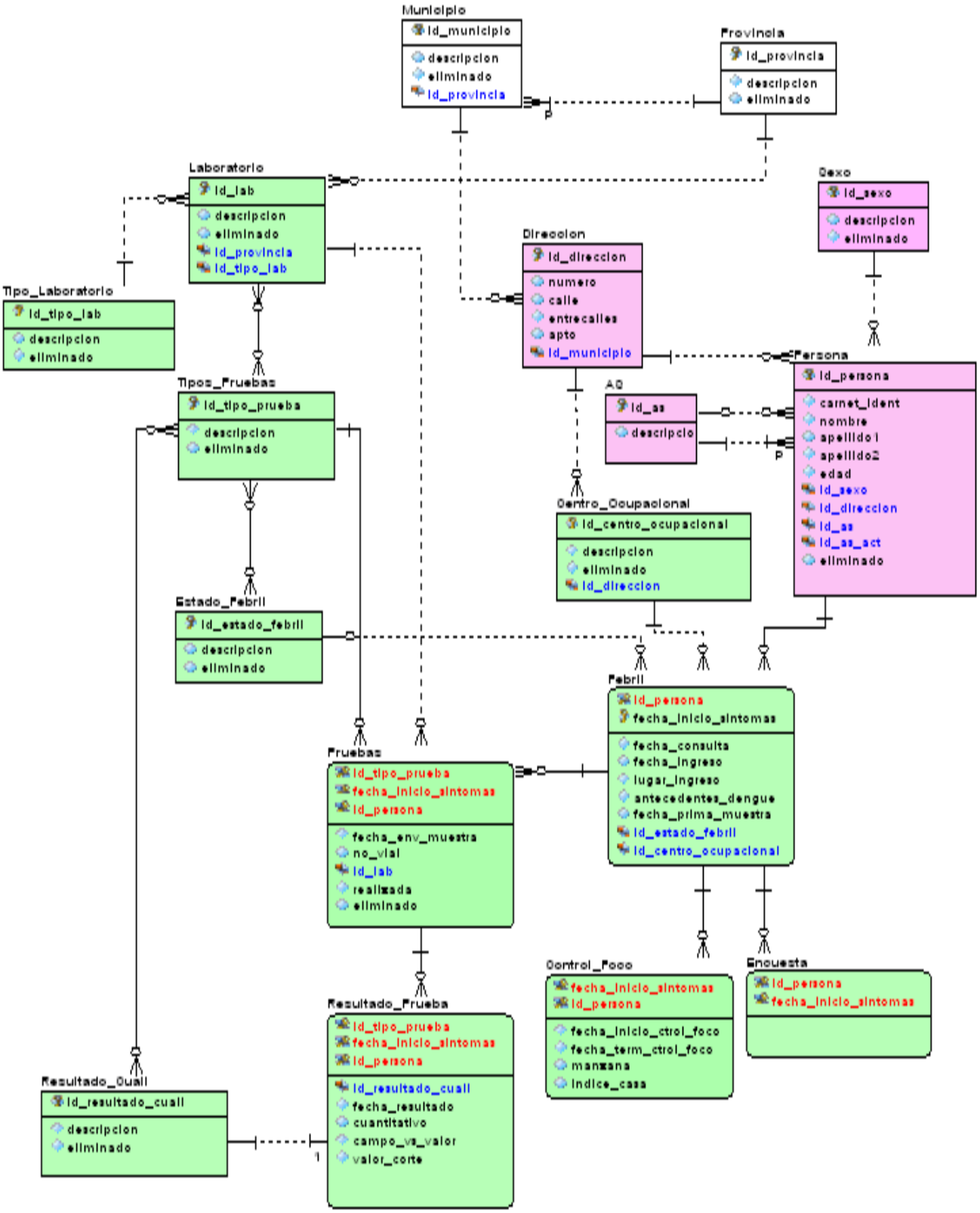

FIGURA 18. SUBMODELO CONTROL DE DENGUE

§3.3.1.2 MODELO FÍSICO.

El modelo físico expresa como quedará la base de datos ya desplegada físicamente en el servidor, es más específico que el modelo lógico pues está ligado a un sistema gestor de bases de datos en específico (en este caso MySQL). Aquí surgen nuevas tablas producto de relaciones que se establecen entre las entidades del modelo lógico.

El modelo físico de HE consta de 34 tablas, 140 columnas, 76 índices y 42 llaves foráneas.

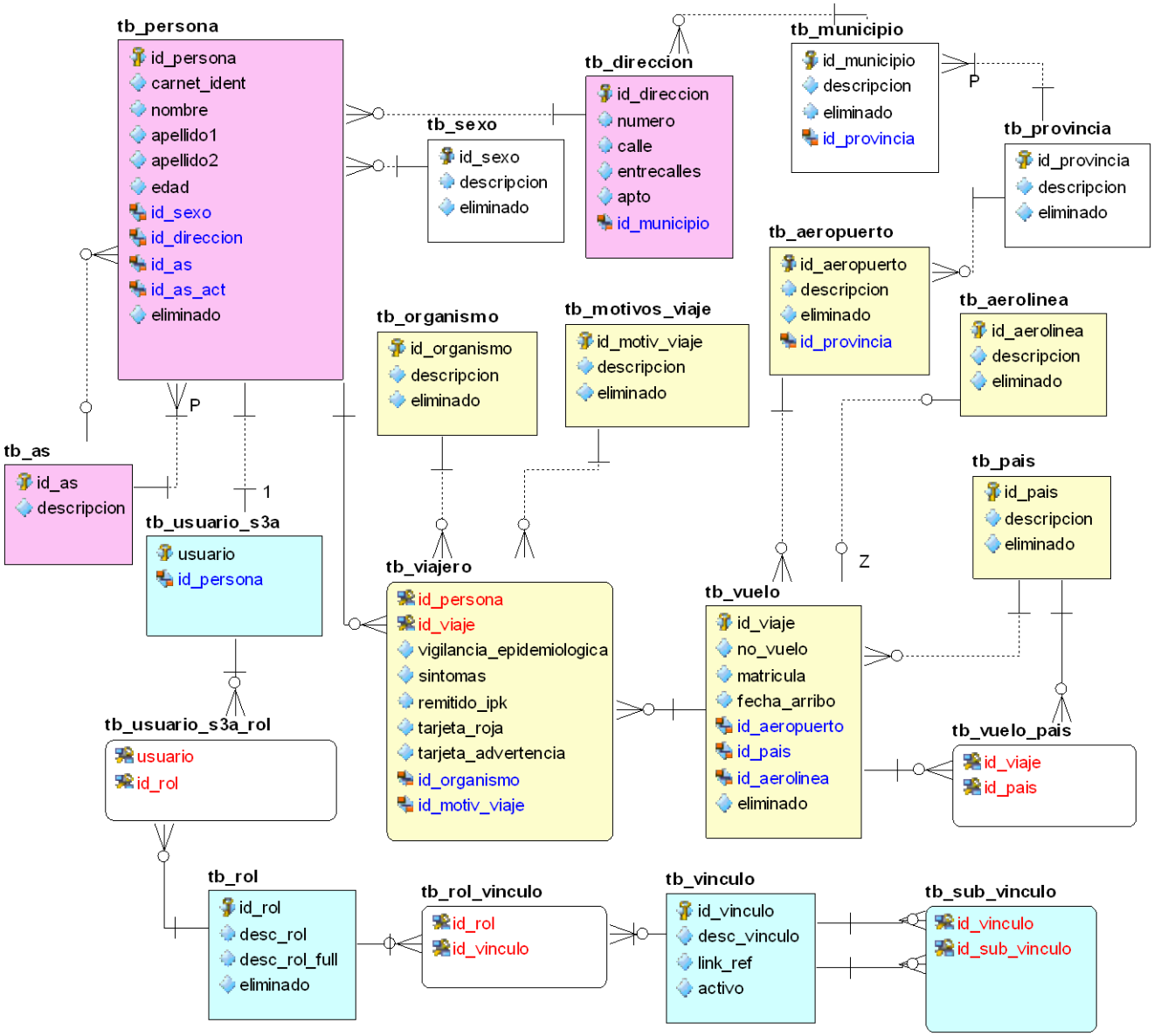

FIGURA 19. MODELO FÍSICO (1/2)

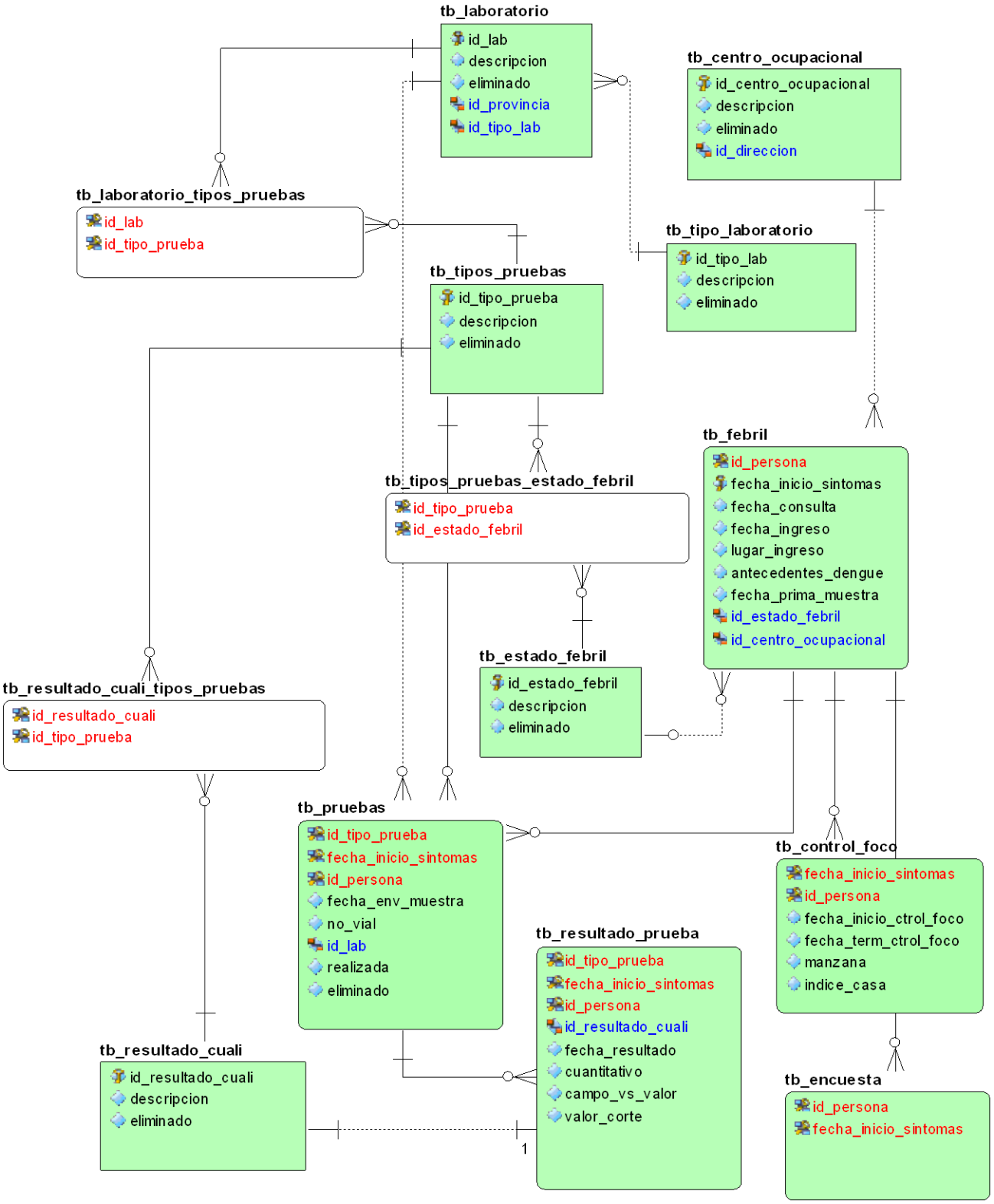

FIGURA 20. MODELO FÍSICO (2/2)

§3.3.2 DESCRIPCIÓN DE LAS TABLAS.

A continuación se relacionan las tablas que describen entidades significativas en el negocio del sistema HE, estas son la base de la manejabilidad e integridad de los datos que gestiona el software. Las tablas relacionadas con la configuración y seguridad (en su mayoría nomencladores) se describen en el ***Anexo 3.*** Otras tablas de la base de datos.

Nombre: tb_persona		
Descripción: En esta tabla se almacenan los datos de las personas.		
Atributo	**Tipo**	**Descripción**
id_paciente	VARCHAR (50)	Llave primaria (PK) de la tabla.
carnet_ident	CHAR(11)	
nombres	VARCHAR (50)	Guarda el nombre de la persona.
apellido1	VARCHAR (50)	Guarda el primer apellido de la persona.
apellido2	VARCHAR (50)	Guarda el segundo apellido de la persona
edad	INT	Guardar la edad de la persona.
id_sexo	INT	Código que identifica el sexo de cada persona.
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 9. LA TABLA PERSONAS

Nombre: tb_direccion		
Descripción: En esta tabla se almacenan los datos de la dirección.		
Atributo	**Tipo**	**Descripción**
id_direccion	INT	Llave primaria (PK) de la tabla.
numero	VARCHAR (10)	Número de la casa.
calle	VARCHAR (50)	Nombre de la calle.
entrecalles	VARCHAR (100)	Nombres de las entrecalles.
apto	VARCHAR (10)	Número del apartamento.
id_municipio	INT	Código que identifica el municipio al que pertenece.

TABLA 10. LA TABLA DIRECCIÓN

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 11. 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.

TABLA 12. LA TABLA PRUEBAS

Nombre: tb_vuelo		
Descripción: Almacena datos sobre el vuelo.		
Atributo	**Tipo**	**Descripción**
id_viaje	INT	Llave primaria (PK) de la tabla.
no_vuelo	VARCHAR(20)	Número del vuelo.
matricula	VARCHAR(20)	Matricula 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 13. 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 14. 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 que estado se encuentra el febril.
id_centro_ocupacional	INT	Identifica el centro ocupacional.

TABLA 15. LA TABLA FEBRIL.

Nombre: Centro_Ocupacional		
Descripción: En esta tabla se almacenan los datos del centro ocupacional.		
Atributo	**Tipo**	**Descripción**
id_centro_ocupacional	INT	Llave primaria (PK) de la tabla.
descripcion	VARCHAR (50)	Descripción del centro ocupacional.
eliminado	TINYINT	Campo que se refiere a si fue eliminado o no.
id_direccion	INT	Identifica la dirección del centro ocupacional.

TABLA 16. LA TABLA CENTRO OCUPACIONAL.

Nombre: Control_Foco		
Descripción En esta tabla se almacenan los datos del control de foco.		
Atributo	**Tipo**	**Descripción**
fecha_inicio_sintomas	DATE TIME	Llave primaria (PK) de la tabla.
id_persona	VARCHAR (50)	Llave primaria (PK) de la tabla.
fecha_inicio_ctrol_foco	DATE TIME	Fecha en la que se inició el control de foco.
fecha_term_ctrol_foco	DATE TIME	Fecha en la que se terminó el control de foco.
manzana	VARCHAR (10)	Identifica la manzana a la que se le realiza el control de foco.
indice_casa	VARCHAR (10)	Se refiere al índice de vectores en la casa.

TABLA 17. LA TABLA CONTROL DE FOCO.

§3.4 MECANISMO DE SEGURIDAD.

La seguridad es un factor de peso en la implementación de cualquier aplicación, esta verifica que la información no pueda ser visualizada por personal no autorizado, o utilizada con fines impropios o contrarios a las políticas del cliente. En la actualidad no se concibe un software de ningún tipo sin pensar en un mecanismo de seguridad que lo respalde, o al menos en algún medio de proteger la información, lo cual deviene de todos modos y por muy simple que sea este medio en una práctica de seguridad informática.

El módulo HE trabaja con información bastante sensible en lo referente a la salud pública cubana, por tanto desde que se concibió la aplicación, se tuvo presente el hecho de que había que diseñar y poner en práctica algún mecanismo que evitara a usuarios no autorizados acceder a los datos que el sistema maneja. Para lograr esta finalidad se ideó e implementó un mecanismo muy simple de seguridad que se divide en dos partes fundamentales, *Autenticación* y *Autorización*.

Aunque el SAAA (Sistema de Autenticación, Autorización y Auditoría) se encarga de estas labores para la plataforma PLASER (Plataforma de Servicios) y se piensa extender a aplicaciones en otras plataformas, este no satisface las necesidades en materia de autorización que el sistema requiere.

§3.4.1 AUTENTICACIÓN.

Como parte de las estrategia de integración que se describió anteriormente en §3.1, el sistema utiliza para garantizar la autenticación el módulo SAAA, mediante el cual se le permite a los usuarios autenticarse y determinar si tienen permiso de entrar al sistema HE, además el SAAA brinda una clave o token a los usuarios registrados que les permite realizar consultas a otros registros del SISalud. Sin embargo la aplicación está preparada para manejar la autenticación por sí misma, aunque para ello habría que realizar algunos cambios mínimos en el código fuente, en cualquier caso podría decirse que en un entorno sin SAAA el sistema continuaría funcionando con un número mínimo de operaciones afectadas.

§3.4.2 AUTORIZACIÓN.

Donde realmente se realizan aportes al mecanismo de seguridad por parte del equipo de desarrollo es en cuanto a la autorización. Lo primero que se tiene en cuenta es realizar un sistema basado en roles,

donde un usuario pertenece a uno o más roles y un rol agrupa un conjunto de usuarios con privilegios comunes en el sistema.

La pregunta entonces sería ¿Qué tipos de privilegios tiene un rol en el sistema? En muchos mecanismos de seguridad un usuario o rol tiene permisos para ejecutar métodos de una determinada clase, en el SAAA por ejemplo un usuario tiene una lista de módulos a los que puede acceder, sin embargo, al ser el HE un módulo, el SAAA no resuelve problemas puntuales en cuanto a la autorización dentro del mismo.

Los privilegios que se manejan en el software HE son a nivel de método o función, es decir, un rol tiene acceso a determinadas funciones en la aplicación, estas están contenidas en clases, por lo tanto un rol tiene acceso a una clase y a un método de la misma.

Para comprender cómo funciona este mecanismo hace falta comprender cómo se hacen las llamadas a funciones en el CodeIgniter.

URLs en CodeIgniter.

Para llamar una función en el framework CodeIgniter, basta con escribir en la barra de direcciones del navegador la dirección URL del sitio y seguido el nombre de la clase controladora y el método que se pretende invocar:

http://host_dir/site_name/index.php/controlador1/metodoX.

De este modo se invoca al método X() de la clase Controlador1.

Asimismo CodeIgniter divide las direcciones URL en un arreglo llamado uri_segments, el cual tiene en la primera posición tiene una cadena de caracteres que contiene la URL hasta index.php, en la segunda posición se encuentra en nombre de la clase y en la tercera la función. En la figura 3.5 se observa como CodeIgniter realiza las llamadas a funciones a través de la dirección URL.

En la figura 21 se muestra como se realiza el proceso de llamadas a funciones en el framework.

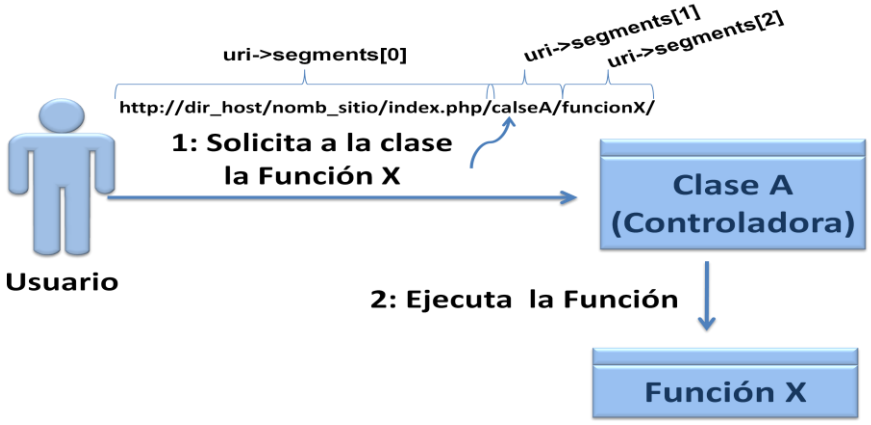

FIGURA 21. LLAMDAS A FUNCIONES EN CODEIGNITER

Teniendo en cuenta que todas las clases controladoras son hijas de MY_Controller *(ver anexo 2)* y que cada vez que se va a ejecutar un método se carga la clase y por ende siempre se hace una llamada al constructor de la misma, el que a su vez invoca al constructor del padre, la idea de la autorización apareció casi de inmediato. Se instaló un filtro en el constructor de MY_Controller el cual verifica si el o los roles asignados al usuario tiene permisos para invocar al método contenido en determinada clase (esta y el método se obtienen vía uri_segments, los permisos de la BD), de ser así entonces la clase ejecuta normalmente el método solicitado, en caso contrario la misma redirecciona al usuario a la clase *C_Usuarios* la cual tiene un método llamado *NoAcces* que se encarga de comunicarle el error de autorización.

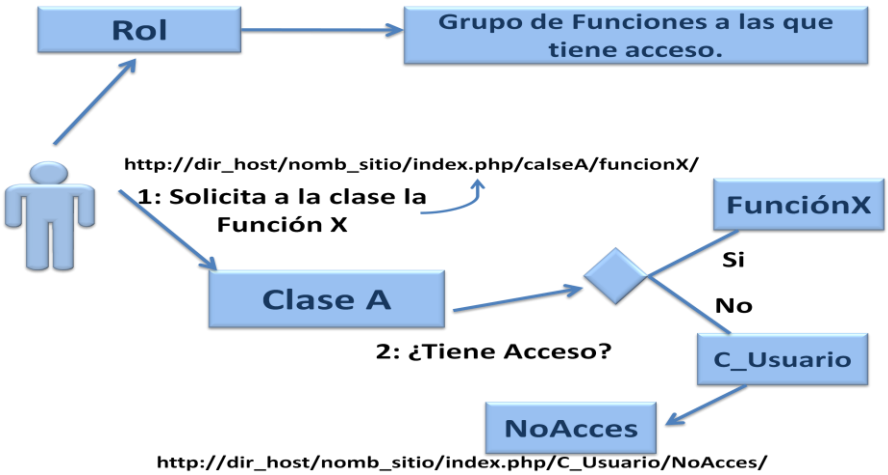

FIGURA 22. LLAMADAS A FUNCIONES APLICANDO MECANISMO DE SEGURIDAD

Conclusiones

En el presente capítulo se han ofrecido los elementos necesarios para la implementación del software HE. Se describieron las clases fundamentales del sistema, y sus funcionalidades, las cuales responden a las necesidades del cliente identificadas por los analistas durante en flujo de trabajo levantamiento de requisitos. Además, se realizó un análisis de la base de datos que permite comprender los mecanismos fundamentales de almacenamiento de la información. Con todos estos datos se valoran los componentes externos al sistema con los cuales se debe integrar el software para complementar sus funcionalidades. Además en el capítulo se describió un mecanismo de seguridad diseñado para el framework CodeIgniter, para garantizar de manera satisfactoria la autorización y la autenticación en el sistema HE.

CONCLUSIONES GENERALES

Para dar cumplimiento a los objetivos de este trabajo de diploma y de ese modo resolver la situación problémica planteada:

- Se mostraron las ventajas y desventajas de las herramientas y tecnologías utilizadas en el desarrollo del software, estas son: L.A.M.P, Apache 2, PHP 5, MySQL 5, CodeIgniter y ZendStudio.
- Se realizaron los modelos de implementación, de despliegue del sistema y el diseño de la base de datos.
- Se seleccionó Modelo Vista Controlador (MVC) como estilo arquitectónico principal en el diseño de la arquitectura del sistema. Además de los estilos: Capas, Basada en Componentes y Orientada a Objetos. Como patrón el propio MVC.
- Se implementó el módulo Higiene y Epidemiología del sistema Control Sanitario Internacional.
- Se realizó la integración del sistema con los componentes del SISalud identificados como necesarios.

RECOMENDACIONES

Los autores de este trabajo de diploma, recomiendan:

- Utilizar la tecnología AJAX para mejorar la interacción con el usuario, en especial la herramienta YUI (Yahoo User Interface).
- Desarrollar las funcionalidades de seguimiento a viajeros y febriles en las Aéreas de Salud.
- Implementar el caso de uso Gestionar Encuesta Epidemiológica, una vez que el MINSAP culmine la elaboración de este documento.
- Extender el software al control de otras enfermedades tropicales como chagas y paludismo entre otras, no solamente el dengue.
- Integrar el sistema a un módulo de lectura de pasaportes para facilitar la búsqueda de viajeros en el aeropuerto.

REFERENCIAS BIBLIOGRÁFICAS

1. **Ciberaula.** Master en LAMP. *Ciberaula.* [En línea] 2006. [Citado el: 13 de Febrero de 2008.] http://www.ciberaula.com/curso/lamp/que_es/ .

2. **Netcraft Ltd.** Market Share for Top Servers Across All Domains August 1995 - May 2008. *Netcraft.* [En línea] Mayo de 2008. [Citado el: 1 de junio de 2008.] http://news.netcraft.com/archives/2008/05/index.html.

3. **Netcratf Ltd.** Sites with longest running systems by average uptime in the last 7 days. *Netcraft.* [En línea] 8 de Junio de 2008. [Citado el: 8 de Junio de 2008.] http://uptime.netcraft.com/up/today/top.avg.html.

4. **Apache Group.** Licencias Apache. *Apache.* [En línea] [Citado el: 10 de Enero de 2008.] http://www.apache.org/licenses/.

5. **Equipo Telecommunity.** Sun Microsystems en pérdidas. *Telecommunity.* [En línea] 9 de Junio de 2008. [Citado el: 10 de Junio de 2008.] http://www.telcommunity.com/visor.php?id_noticia=22972.

6. **MySQL AB.** Las principales características de MySQL. *MySQL.* [En línea] [Citado el: 10 de Enero de 2008.] http://dev.mysql.com/doc/refman/5.0/es/features.html.

7. **PHP.NET.** Manual de PHP. *PHP.NET.* [Online] 2008. [Cited: Febrero 12, 2008.] http://www.php.net/manual/es/index.php.

8. **Ellis Lab.** Code Igniter User Guide. *Code Igniter.* [En línea] 2008. [Citado el: 10 de Enero de 2008.] http://codeigniter.com/user_guide/index.html.

9. **Valdelli, Ilario.** Curso Javascript. *HTMLPOINT.* [En línea] 2006. [Citado el: 10 de Enero de 2008.] http://www.htmlpoint.com/javascript/corso/js_02.htm.

10. **Zend.** Zend Studio Features. *Zend.* [En línea] 2008. [Citado el: 5 de Marzo de 2008.] http://www.zend.com/en/products/studio/features.

11. **Alvarez, Miguel Angel.** Evaluando Zend Studio. *Maestros del Web.* [En línea] 3 de Novienmbre de 2003. [Citado el: 8 de Enero de 2008.] http://www.maestrosdelweb.com/editorial/zendstudio/.

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

13. *ISW1 Conf 3 Flujo de trabajo de requerimientos.* **Dpto ISW UCI.** Ciudad Habana : s.n., 2007.

14. **Doffman, M. and Thayer, R.** *Standards, Guidelines and Examples on System and Software.* s.l. : IEEE Computer Society Press, 1990.

15. **CAMACHO, E., CARDESO, F. y NUÑEZ, G.** Arquitecturas de Software. [En línea] 2004. http://prof.usb.ve/lmendoza/Documentos/PS-6116/Guia%20Arquitectura%20v.2.pdf.

16. **Kruchten, P.** Architectural Blueprints—The "4+1" View Model of Software Architecture. . [Online] 1995. http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf..

17. *IDEM a 15.*

18. **Taylor, Richard, et al.** *A Component- and Message-Based Architectural Style for GUI Softwar.* Seattle : ACM Press, 1995.

19. **Allen, Robert y Garlan, David.** *The Wright Architectural Description Language.* Carnegie Mellon : s.n., 1996.

20. **Burbeck, Steve.** *Application programming in Smalltalk-80: How to use Model-View-Controller (MVC).* University of Illinois : s.n., 1992.

21. **Microsoft.** Model View Controller. *MSDN.* [En línea] 2008. [Citado el: 2 de Febrero de 2008.] http://msdn.microsoft.com/en-us/library/ms978748.aspx.

22. **Garlan, David and Shaw, Mary.** *An introduction to software architecture.* CMU Software Engineering Institute : s.n., 1994.

23. *IDEM a 12.*

24. *IDEM a 21.*

25. *IDEM a 22.*

26. **Welicki, León.** Patrones y Antipatrones: una Introducción. [Online] 2007. http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3317/default.aspx#M15.

BIBLIOGRAFÍAS

- **Banco Mundial.** Informe sobre el Desarrollo Mundial. Invertir en salud. Washington D.C. : s.n., 1993.
- **Bravo, Vivian Noriega.** Un nuevo reto en la formación del especialista en Higiene y Epidemiología. [En línea] 18 de Diciembre de 2007. http://bvs.sld.cu/revistas/hie/vol46_1_08/hie05108.htm.
- **Cerami, Ethan.** *Web Services Essentials.* s.l. : O'Reilly, 2003.
- *Enfermedades transmisibles impacto e historia.* **Escamilla-Guerrero, Guillermo.** 3, Ciudad México : Gac Méd Méx, 2004, Vol. 140.
- *El componente ambiental de la vigilancia integrada para el control y la prevención del dengue.* **GARCIA MELIAN, Maricel, MARINE ALONSO, María de los Ángeles y DIAZ PANTOJA, Cristina.** 1, Ciudad Habana : Revista Cubana de Higiene-Epidemiología, Enero-Abril 2007, Vol. 45. 1561-3003.
- *Enfermedades transmisibles, genero y equidad en la salud.* **Hartigan, Pamela.** 7, Hardvard Center for Population and Development Studies : s.n., 2001.
- **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Ciudad Habana : Félix Varela, 2004.
- **Larman, Craig.** *Introducción al análisis y diseño orientado a objetos.* Ciudad Habana : Félix Varela, 2004.
- **Pressman, Roger.** *Ingeniería del Software. Un enfoque práctico.* Ciudad Habana : Félix Varela, 2005.
- **Lenn, Bass, C., P. and Kazman, Rick.** *Software Architecture in Practice.* s.l. : Addison-Wesley Professional, 2003.
- *Comparación de datos de la vigilancia ambiental y de grupos vecinales para prevenir el dengue.* **MARINE ALONSO, María de los Ángeles, GARCIA MELIAN, Maricel and TORRES ROJO, Yisel.** 1, Ciudad Habana : Rev Cubana Hig Epidemiol, 2007, Vol. 45. ISSN 1561-3003.
- *Las grandes epidemias y la gripe aviar.* **Suárez-Ognio, Luis.** 1, Perú : Acta Medíca Perú, 2006, Vol. 23.
- *Aproximación al análisis del dominio Higiene y Epidemiología en Cuba a través de la producción científica de una revista especializada.* **Tarrago, Nancy Sánchez.** 1, Ciudad Habana : Revista Cubana Higiene-Epidemioloía, Enero-Abril 2007, Vol. 45. 1561-3003.
- **Ricardo, Febe Ángel Ciudad.** Utilización del patrón modelo – vista – controlador (mvc) en el diseño de software educativos. [En línea] 2007. http://www.informaticahabana.com/evento_virtual/?q=node/223&ev=3er%20Congreso%20Internacional%20de%20Tecnolog%C3%ADas,%20Contenidos%20Multimedia.

ANEXOS

ANEXO 1. CLASES DE CONFIGURACIÓN Y SEGURIDAD.

Aunque no implementan la lógica del negocio ni representan un peso sustancial en el éxito de la aplicación, las clases de configuración y seguridad, aportan en materia de manejabilidad y operatividad un valor extraordinario al sistema, haciéndolo en todo caso mucho más flexible, configurable y seguro.

La clase *C_Configuracion* se encarga de manejar la información referente a los nomencladores y la configuración del sistema. Dada la estructura de los mismos la clase los agrupa en un grupo general, otros que tienen id de provincia y otros que no entran en estas clasificaciones como son laboratorios y tipos de pruebas, por lo tanto el código de la clase es relativamente pequeño, debido, por ejemplo, a que los nomencladores que tienen la estructura (*id_nomenclador, descripción, eliminado)* son manejados por los mismos métodos.
Esta clase utiliza solamente a la modelo *M_Configuracion*, la cual presenta características similares.

Nombre: C_Configuracion	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	C_Configuracion()
Descripción:	Constructor de la clase, por defecto.
Nombre:	Index()
Descripción:	Función inicial de la clase, determina con cual nomenclador se está trabajando, de no haber ninguno muestra una vista para seleccionar alguno.
Nombre:	GetMcpios()
Descripción:	Toma los municipios de la clase modelo y los devuelve en formato XML.
Nombre:	VerTiposPruebasLabs($id_lab)
Descripción:	Recibe un id de laboratorio y devuelve el lisado de tipos de pruebas que se realizan en él.
Nombre:	VerResultadosTipoPrueba($id_tipo_prueba)
Descripción:	Recibe un id de tipo de prueba y devuelve los posibles resultados cualitativos para este tipo de prueba.
Nombre:	Insertar()
Descripción:	Se encarga de que la clase modelo inserte un nomenclador cuya estructura sea (id_x, descripción, eliminado) en la base de datos.

Nombre:	InsertarConProv()
Descripción:	Se encarga de que la clase modelo inserte un nomenclador que además tenga un id provincia en la base de datos.
Nombre:	InsertarLab()
Descripción:	Se encarga de que la clase modelo inserte un nomenclador de tipo laboratorio en la base de datos.
Nombre:	InsertarTipoPrueba()
Descripción:	Se encarga de que la clase modelo inserte un nomenclador tipo de prueba en la base de datos.
Nombre:	Eliminar
Descripción:	Método que indica a la clase modelo que marque un nomenclador determinado como eliminado (la eliminación de los nomencladores es lógica no física).
Nombre:	Modificar()
Descripción:	Se encarga de que la clase modelo modifique los datos de un nomenclador cuya estructura sea (id_x, descripción, eliminado).
Nombre:	ModificarConProv()
Descripción:	Se encarga de que la clase modelo modifique los datos de un nomenclador que contenga id de provincia.
Nombre:	ModificarLab()
Descripción:	Se encarga de que la clase modelo modifique los datos de un nomenclador de tipo laboratorio.
Nombre:	ModificarTipoPrueba()
Descripción:	Se encarga de que la clase modelo modifique los datos de un nomenclador tipo de prueba.
Nombre:	GestionarTabla()
Descripción:	Envía a la vista los datos para mostrar las paginas de listado de cualquier nomenclador, incluye el algoritmo de paginado.
Nombre:	GetDatosTabla($offset)
Descripción:	Función utilizada por GestionarTabla para obtener los datos del nomenclador sea cual fuere a partir de la posición offset.

TABLA 18. LA CLASE C_CONFIGURACION

M_Configuracion es probablemente la clase más utilizada en el software, no solo interviene en configurar el sistema como modelo de *C_Configuracion*, sino que brinda la información de los nomencladores a las demás clases que los usan, además tiene métodos para devolver los datos en diferentes formatos como tipo *fields* (o campos de la base de datos, formato muy utilizado para los algoritmos de paginado), también en formato de *lista, tabla* o *dropdown* (el cual es muy utilizado para

las listas desplegables también conocidas como *ComboBox* o *DropdownLits*). Además de métodos internos que verifican que los nomencladores cumplan ciertas reglas de negocio establecidas para la configuración del sistema. Es la clase más extensa de la aplicación.

Nombre: M_Configuracion	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Configuracion()
Descripción:	Constructor de la clase, por defecto.
Nombre:	Insertar ($nomenclador)
Descripción:	Inserta un nomenclador de cualquier tipo que tenga estructura (id, descripción, eliminado) en la base de datos.
Nombre:	InsertarConProv ($tabla,$descripcion,$id_prov)
Descripción:	Inserta un nomenclador que además tenga un id de provincia
Nombre:	InsertarMcpio ($list_mcpio)
Descripción:	Inserta los municipios que vienen de registro de ubicaciones del SISalud.
Nombre:	InsertarProvincia ($list_prov)
Descripción:	Inserta las provincias que vienen de registro de ubicaciones del SISalud.
Nombre:	InsertarLab($lab)
Descripción:	Inserta un objeto tipo laboratorio en la base de datos.
Nombre:	InsertarTipoPrueba($tp)
Descripción:	Inserta el objeto tipo de prueba que recibe como parámetro en la BD.
Nombre:	Eliminar($tabla,$id)
Descripción:	Elimina un nomenclador del tipo $tabla cuyo id coincida con $id.

Nombre:	GetNomenc ($nomenc,$where='',$per_page='',$offset='')
Descripción:	Devuelve los datos primarios de un nomenclador tipo $nomenc, según los criterios $wehre
Nombre:	GetNomencID ($nomenc,$id)
Descripción:	Devuelve un nomenclador tipo $nomenc cuyo id es $id
Nombre:	Estados2Str ($list_estados)
Descripción:	Recibe una lista de objetos tipo estado febril y los devuelve en un formato determinado de cadena de caracteres.
Nombre:	Id2Str($nombre_tb)
Descripción:	Dado el nombre de una tabla devuelve el nombre del campo id en forma de string.
Nombre:	GetConProv($nomenc,$per_page='',$offset='')
Descripción:	Retorna los datos primarios de los nomencladores que tienen provincia.
Nombre:	GetAeroProv($prov)
Descripción:	Dado un id de provincia devuelve el aeropuerto internacional de la misma.
Nombre:	GetTiposPruebas($per_page = '',$offset = '')
Descripción:	Devuelve todos los tipos de prueba.
Nombre:	GetTipoPruebaFull($id_tipo_prueba)
Descripción:	Dado un id de tipo de prueba devuelve todo lo relacionado con ella.
Nombre:	GetTipoPruebaEstadoFebril($id_tipo_prueba = -1, $id_estado_febril = -1)
Descripción:	Devuelve todos los tipos de prueba y en que estado febril se realizan.
Nombre:	GetLabs($per_page = '',$offset = '')
Descripción:	Devuelde la lista de todos los laboratorios.
Nombre:	GetLabsFull($id_lab)
Descripción:	Devuelve todos los datos relacionados con el laboratorio cuyo id se pasa por

	parámetro.
Nombre:	GetLabs4Prov($prov = '',$tipo_lab = '')
Descripción:	Dado un id de provincia y un tipo de laboratorio devuelve los datos del que coincide con esos parámetros.
Nombre:	GetFields($nomenc)
Descripción:	Devuelve todos los datos de un nomenclador dado en formato de field o campo de la base de datos.
Nombre:	GetMcpios($id_prov)
Descripción:	Devuelve los municipios que pertenecen a una provincia dada.
Nombre:	GetProv4Mcpio($id_mcpio)
Descripción:	Devuelve la provincia a la cual pertenece un municipio dado.
Nombre:	ContarNomenc($nomenc)
Descripción:	Devuelve la cantidad de un nomenclador dado que existe en la base de datos sin ser eliminado.
Nombre:	VerTiposPruebasLabs($id_lab)
Descripción:	Devuelve los tipos de prueba que se realizan en un laboratorio dado.
Nombre:	VerResultadosTipoPrueba($id_tipo_prueba)
Descripción:	Devuelve que resultados cualitativos puede tener un tipo de prueba específico.
Nombre:	ListarNomenc($nomenc)
Descripción:	Devuelve un nomenclador dado en formato de lista
Nombre:	ModificarN($nomenc,$value)
Descripción:	Modifica los nomencladores tipo $nomenc con el valor del objeto $value.
Nombre:	ModificarConProv($nomenc,$value)
Descripción:	Modifica los nomencladores tipo $nomenc con el valor del objeto $value.

Nombre:	ModificarTipoPrueba($tipo_prueba)
Descripción:	Modifica los nomencladores tipo de prueba asignándoles el objeto recibido por parámetros.
Nombre:	ModificarLab($lab)
Descripción:	Modifica los nomencladores laboratorio asignándoles el objeto recibido por parámetros.
Nombre:	CB_Provs($id_prov_sel = -1,$params = '')
Descripción:	Devuelve un listado de provincias en formato de dropdownlist.
Nombre:	CB_Mcpios($id_mcpio = -1,$id_prov = -1)
Descripción:	Devuelve un listado de municipios en formato de dropdownlist.
Nombre:	CB_MotivosV($id_motivosV = -1)
Descripción:	Devuelve un listado de motivos de viaje en formato de dropdownlist.
Nombre:	CB_Organismo($id_org = -1)
Descripción:	Devuelve un listado de organismos del estado en formato de dropdownlist.
Nombre:	CB_Sexo()
Descripción:	Devuelve un listado de sexos en formato de dropdownlist.
Nombre:	CB_Tipos_Lab($id_tipo_lab = -1)
Descripción:	Devuelve un listado de tipos de laboratorio en formato de dropdownlist.
Nombre:	CB_Aeropuertos($id_aero_sel = -1,$adds = '')
Descripción:	Devuelve un listado de aeropuertos en formato de dropdownlist.
Nombre:	CB_Aerolineas($id_aerolinea_sel = -1,$adds = '')
Descripción:	Devuelve un listado de aerolíneas en formato de dropdownlist.
Nombre:	CB_Paises($id_pais_sel = -1,$adds = '')

Descripción:	Devuelve un listado países en formato de dropdownlist.
Nombre:	CB_Estado_Febril($id_estado = -1, $adds = '')
Descripción:	Devuelve un listado de estados febriles en formato de dropdownlist.
Nombre:	CB_Tipos_Pruebas($id_tipo_prueba = -1,$id_estado_feb = -1)
Descripción:	Devuelve un listado tipos de prueba que se pueden realizar en un estado febril dado en formato de dropdownlist.
Nombre:	__PruebaXEstado($id_tipo_prueba,$id_estado_feb)
Descripción:	Devuelve la lista de pruebas que se pueden realizar en un estado dado.
Nombre:	CB_Labs($id_lab = -1)
Descripción:	Devuelve un listado de estados febriles en formato de dropdownlist.
Nombre:	CB_Resultado_Cuali ($id_result_cuali = -1)
Descripción:	Devuelve un listado de resultados cualitativos en formato de dropdownlist.

TABLA 19. LA CLASE M_CONFIGURACION

C_Usuario es una clase que se complementa con las funcionalidades de *M_Usuario* para brindar al sistema un exquisito mecanismo de seguridad basado en roles que se enfrasca en el hecho de que cada usuario debe acceder solo a las funcionalidades que le están permitidas, por lo cual se encargan también de gestionar el menú para la aplicación.

Nombre: C_Usuario	
Tipo de clase: Controlador	
Para cada responsabilidad:	
Nombre:	C_Usuario()
Descripción:	Constructor de la clase, por defecto.
Nombre:	Index()

Descripción:	Si el usuario no está logeado lo manda a autenticarse.
Nombre:	NoAcces()
Descripción:	Muestra una pantalla cuando alguien está tratando de entrar sin autorización al sistema
Nombre:	Login()
Descripción:	Se encarga del logueo por parte de la aplicación.
Nombre:	AutenticateSAAA()
Descripción:	Loguea a los usuarios en el SAAA.
Nombre:	ShowMainPage()
Descripción:	Muestra una página de bienvenida.
Nombre:	Salir()
Descripción:	Saca a los usuarios del sistema.

TABLA 20. LA CLASE C_USUARIO

Nombre: M_Usuario	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	M_Usuario()
Descripción:	Constructor de la clase, por defecto.
Nombre:	GetRoles4User ($username)
Descripción:	Devuelve una lista de roles a los que pertenece el usuario.
Nombre:	IsUserInRol ($username,$rol)
Descripción:	Dice si un usuario pertenece a un rol.
Nombre:	GetUser ($username)
Descripción:	Devuelve los datos de un usuario.
Nombre:	GetAllUsers()
Descripción:	Devuelve una lista con todos los usuarios.

Nombre:	GetLinks4User ($username)
Descripción:	Devuelve una lista con todos los vínculos a los que puede acceder un usuario.
Nombre:	HaveAutorization($username ,$link)
Descripción:	Dice si un usuario puede acceder a un vínculo.

TABLA 21. LA CLASE M_USUARIO

ANEXO 2. LA LIBRERÍA MY_CONTROLLER

MY_Controller es una librería que se implementó por parte del equipo de desarrollo del módulo HE y sin la cual definitivamente el software no hubiese llegado del diseño de los analistas a la realidad.

Esta librería encapsula funcionalidades que pueden ser invocadas en cualquier momento por las clase controladora que se encuentre activa, pues todas las clases de este tipo heredan de *MY_Controller*, la que a su vez es hija de *Controller*, como establece CodeIgniter que deben hacer todas las clases controladoras.

Las funcionalidades contenidas en la clase *MY_Controller* simplifican varios procesos que son comunes en prácticamente todas las clases controladoras, se encarga además de un mecanismo para mostrar las interfaces de la aplicación (o interfaces de usuario), basado en *templates* que aminora a sobremanera el código en las vistas. Además tiene un peso significativo en la seguridad del sistema; cada vez que se invoca una función perteneciente a cualquier clase por parte del usuario se ejecuta el constructor de *My_Controller* el cual verifica que ese usuario tenga permisos para ejecutar el método solicitado, además crea el menú en dependencia de los privilegios del usuario autenticado.

Nombre: MY_Controller	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	MY_Controller()
Descripción:	Constructor de la clase, por defecto.
Nombre:	render()
Descripción:	Visualiza las vistas aplicándoles una plantilla o template.

Nombre:	CrearMenu()
Descripción:	Crea el menú del sitio en dependencia del usuario que este autenticado.
Nombre:	TieneAcceso()
Descripción:	Dice si un usuario tiene acceso a una funcionalidad determinada.
Nombre:	IsUserInRole($username,$rol)
Descripción:	Dice si un usuario dado pertenece a determinado rol de la aplicación.
Nombre:	CargarModulo($Modulo, $ModoSeguro = true)
Descripción:	Carga un servicio web y devuelve un objeto de este servicio para llamar a sus métodos.
Nombre:	ActionComplete($msj = '')
Descripción:	Muestra un mensaje determinado si una acción se realizo satisfactoriamente.

TABLA 22. LA LIBRERÍA MY_CONTROLLER

ANEXO 3. OTRAS TABLAS DE LA BASE DE DATOS.

Las tablas que se relacionan a continuación no representan entidades significativas en el contexto del negocio para el sistema HE, sin embargo agilizan y facilitan la configuración y prestaciones del software.

Nombre: tb_aeropuerto		
Descripción En esta tabla se almacenan los datos de los aeropuertos.		
Atributo	**Tipo**	**Descripción**
id_aeropuerto	INT	Llave primaria (PK) de la tabla.
descripcion	VARCHAR (50)	Descripción del aeropuerto.
eliminado	TINYINT	Campo que se refiere a si fue eliminado o no.

id_provincia	INT	Identifica la provincia a la que pertenece.

TABLA 23. TABLA AEROPUERTO

Nombre: tb_motivos_viaje		
Descripción: Almacena datos sobre motivos del viaje.		
Atributo	**Tipo**	**Descripción**
id_motiv_viaje	INT	Llave primaria (PK) de la tabla.
descripcion	VARCHAR(50)	Descripción del motivo del viaje.
eliminado	TINYINT	Campo que se refiere a si es eliminado o no.

TABLA 24. TABLA MOTIVOS DE VIAJE

Nombre: tb_organismo		
Descripción: En esta tabla se almacena los datos de los diferentes organismos.		
Atributo	**Tipo**	**Descripción**
id_organismo	INT	Llave primaria (PK) de la tabla.
descripcion	VARCHAR(50)	Descripción del organismo.
eliminado	TINYINT	Campo que se refiere a si es eliminado o no.

TABLA 25. TABLA ORGANISMOS

Nombre: tb_provincia		
Descripción: En esta tabla se almacenan los datos de la provincia.		
Atributo	**Tipo**	**Descripción**
id_provincia	INT	Llave primaria (PK) de la tabla..
descripcion	VARCHAR (50)	Descripción de la provincia.
eliminado	TINYINT	Campo que se refiere a si fue eliminado o no.

TABLA 26. TABLA PROVINCIA

Nombre: tb_municipio		
Descripción: En esta tabla se almacenan los datos del municipio		
Atributo	**Tipo**	**Descripción**
id_municipio	INT	Llave primaria (PK) de la tabla.
descripcion	VARCHAR (50)	Descripción del municipio.
eliminado	TINYINT	Campo que se refiere a si fue eliminado o no.
id_provincia	INT	Código que identifica la provincia al que pertenece.

TABLA 27. TABLA MUNICIPIO

Nombre: tb_as		
Descripción: En esta tabla se almacenan los datos del área de salud.		
Atributo	**Tipo**	**Descripción**
id_as	INT	Llave primaria (PK) de la tabla.

desc_as	VARCHAR (50)	Este campo es una descripción del área de salud

TABLA 28. TABLA ÁREA DE SALUD

Nombre: tb_sexo		
Descripción: Esta tabla contiene lo referente al sexo.		
Atributo	**Tipo**	**Descripción**
id_sexo	INT	Llave primaria (PK) de la tabla.
descripcrion	VARCHAR (50)	Valores que puede tomar el sexo.
eliminado	TINYINT	Se refiere a si fue eliminado o no.

TABLA 29. TABLA SEXO

Nombre: Laboratorio		
Descripción: Esta tabla contiene lo referente al laboratorio.		
Atributo	**Tipo**	**Descripción**
id_lab	INT	Llave primaria (PK) de la tabla.
descripcion	VARCHAR (50)	Descripción del laboratorio.
eliminado	TINYINT	Se refiere a si fue eliminado o no.
id_provincia	INT	Se refiere al id de la provincia a que pertenece.
Id_tipo_lab	INT	Se refiere al id del tipo de laboratorio a que pertenece.

TABLA 30. TABLA LABORATORIOS

Nombre: Tipo_Laboratorio		
Descripción: Esta tabla contiene lo referente al tipo de laboratorio.		
Atributo	**Tipo**	**Descripción**
id_tipo_lab	INT	Llave primaria (PK) de la tabla.
descripcion	VARCHAR (50)	Descripción del tipo de laboratorio.
eliminado	TINYINT	Se refiere a si fue eliminado o no.

TABLA 31. TABLA TIPO DE LABORATORIO

Nombre: Tipos_Pruebas		
Descripción: Esta tabla contiene lo referente al tipo de laboratorio.		
Atributo	**Tipo**	**Descripción**
id_tipo_prueba	INT	Llave primaria (PK) de la tabla.
descripcion	VARCHAR (50)	Descripción de los tipos de pruebas.
eliminado	TINYINT	Se refiere a si fue eliminado o no.

TABLA 32. TABLA TIPOS DE PRUEBAS

Nombre: Estado_Febril		
Descripción: Esta tabla contiene lo referente al tipo de laboratorio.		
Atributo	**Tipo**	**Descripción**
id_estado_febril	INT	Llave primaria (PK) de la tabla.
descripcion	VARCHAR (50)	Descripción del estado febril.
eliminado	TINYINT	Se refiere a si fue eliminado o no.

TABLA 33. TABLA ESTADO FEBRIL

Nombre: Resultado_Cuali		
Descripción: Esta tabla contiene lo referente al tipo de laboratorio.		
Atributo	**Tipo**	**Descripción**
id_resultado_cuali	INT	Llave primaria (PK) de la tabla.
descripcion	VARCHAR (50)	Descripción de los resultados cualitativos.
eliminado	TINYINT	Se refiere a si fue eliminado o no.

TABLA 34. TABLA RESULTADO CUALITATIVO

Nombre: Usuario_S3A		
Descripción: En esta tabla se almacenan los datos del usuario del S3A.		
Atributo	**Tipo**	**Descripción**
usuario	VARCHAR (20)	Llave primaria (PK) de la tabla.
id_persona	VARCHAR (50)	Identifica la persona que se autentica.

TABLA 35. TABLA USUARIO DEL SAAA

Nombre: Rol		
Descripción: En esta tabla se almacenan los datos del rol.		
Atributo	**Tipo**	**Descripción**
id_rol	INT	Llave primaria (PK) de la tabla.
desc_rol	VARCHAR (50)	Breve descripción del rol.
desc_rol_full	CAHR(10)	Descripción para mostrar al usuario.
eliminado	TINYINT	Se refiere a si fue eliminado o no.

TABLA 36. TABLA ROL

Nombre: Vinculo		
Descripción: En esta tabla se almacenan los datos del usuario del S3A.		
Atributo	**Tipo**	**Descripción**
id_vinculo	INT	Llave primaria (PK) de la tabla.
desc_vinculo	VARCHAR (50)	Breve descripción del vínculo.
link_ref	VARCHAR (200)	Este es el vínculo.
activo	TINYINT	Se refiere a si esta activo o no.

TABLA 37. TABLA VINCULO.

GLOSARIO DE TÉRMINOS

API: (Del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

ASP.NET: Es un framework para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores para construir sitios web dinámicos, aplicaciones web y servicios web XML.

C#: Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA e ISO.

C: Lenguaje de programación creado en 1972 por Ken Thompson y Dennis M. Ritchie en los Laboratorios Bell como evolución del anterior lenguaje B, a su vez basado en BCPL.

C++: Lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C. Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

Framework: Es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado.

GNU/GPL: La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente su acrónimo del inglés GNU GPL, es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

GRASP: Son patrones generales de software para asignación de responsabilidades, es el acrónimo de "General Responsibility Assignment Software Patterns".

Hostear: Término informático procedente del inglés Host (anfitrión). Se refiere al acto de colocar o albergar aplicaciones en servidores.

Java: Es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90.

JavaServer Pages (JSP): Es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

Linux: Es un sistema operativo tipo Unix (también conocido como GNU/Linux) que se distribuye bajo la GNU GPL, es decir que es software libre. Su nombre proviene del Núcleo de Linux, desarrollado en 1991 por Linus Torvalds. Es usado ampliamente en servidores y super-computadores,

Mac OS X: Es el actual sistema operativo de la familia de ordenadores Macintosh.

Macintosh: (Abreviado Mac) es el nombre con el que actualmente se refiere a cualquier computadora personal diseñada, desarrollada, construída y comercializada por Apple Inc.

ODBC: (Open Database Connectivity) es un estándar de acceso a Bases de Datos desarrollado por Microsoft Corporation, el objetivo de ODBC es hacer posible el acceder a cualquier dato desde cualquier aplicación, sin importar qué Sistema Gestor de Bases de Datos almacene los datos,

Opensource: Es el término con el que se conoce al software distribuido y desarrollado libremente.

Oracle: Sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), fabricado por Oracle Corporation. Considerado de los mas completos que existen.

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.

Solaris: Sistema operativo desarrollado por Sun Microsystems.

SQL Server: Sistema de gestión de bases de datos relacionales (SGBD) basado en el lenguaje Transact-SQL, y específicamente en Sybase IQ, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea. Así de tener unas ventajas que a continuación se pueden describir.

SSL: El protocolo SSL es un sistema diseñado y propuesto por Netscape Communications Corporation. Proporciona sus servicios de seguridad cifrando los datos intercambiados entre el servidor y el cliente con un algoritmo de cifrado simétrico, típicamente el RC4 o IDEA.

Sybase IQ: Es un motor de bases de datos altamente optimizado para inteligencia empresarial, desarrollado por la empresa Sybase.

TCP/IP: Conjunto de protocolos de red en la que se basa Internet y que permiten la transmisión de datos entre redes de computadoras.

UNIX: (Registrado oficialmente como UNIX®) es un sistema operativo portátil, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

Video Graphics Array (VGA): Se refiere tanto a una pantalla de computadora analógica estándar; conector VGA de 15 clavijas D subminiatura que se comercializó por primera vez en 1988 por IBM; o la resolución 640 × 480.

Windows: Sistema Operativo de Microsoft.

XML: Sigla en inglés de Extensible Markup Language («lenguaje de marcas extensible»), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).