UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 7

Trabajo de Diploma para optar por el Título de
Ingeniero en Ciencias Informáticas

Componente web Zoonosis del módulo de Higiene y Epidemiología del Sistema Integral para la Atención Primaria de Salud

Autora:

Anabel Campos De Castro

Tutores:

Lic. Yamilka Gómez León.

Ing. Rudny Rangel Marrero.

La Habana, junio de 2011

"Año 53 de la Revolución"

Datos de Contacto

Nombre: Lic. Yamilka Gómez León - (yamilkagl@uci.cu): Graduado de Licenciatura en Ciencias de la Computación en el año 2004. Actualmente labora en la Universidad de Ciencias Informáticas (UCI), desempeñándose como Asesor Técnico Docente de la asignatura Sistemas de Bases de Datos en el Departamento Docente Central de Ingeniería y Gestión de Software. Además, imparte docencia en la Facultad # 7 y tiene categoría docente de Asistente. Ha cursado diversos diplomados y cursos de superación. Actualmente cursa las asignaturas de la Maestría Informática Aplicada.

Nombre: Ing. Rudny Rangel Marrero - (rrangel@uci.cu): Graduado de Ingeniero en Ciencias Informáticas en el año 2010 con Título de Oro. Actualmente labora en la Universidad de Ciencias Informáticas (UCI), desempeñándose como profesor de la Facultad # 7 vinculado a la producción. Actualmente es parte del proyecto Sistema Integral para la Atención Primaria de Salud (SIAPS) donde se desarrolla como jefe de módulo Higiene y Epidemiología.

Yo, como soy tu entrenador, te miro en la pista y, a veces, hago de liebre, para que me sigas. Luego te dejo sola, para que aprendas a caminar.

Wilfredo de Jesús Campos Cremé

Resumen

En la Atención Primaria de Salud (APS) se gestionan grandes volúmenes de información relacionadas con el registro de zoonosis. Debido a la cantidad de datos que se procesa y la lentitud del flujo de transmisión del mismo por los diferentes niveles, la obtención de las estadísticas es un proceso demorado y resulta engorrosa la generación de los reportes.

El objetivo de este Trabajo de Diploma es desarrollar el Componente web Zoonosis del módulo de Higiene y Epidemiología del Sistema Integral para la Atención Primaria de Salud. (AlasSIAPS), para facilitar la gestión de los datos en el departamento de zoonosis de la APS.

El desarrollo del componente se basa en tecnologías libres, multiplataforma y sobre una arquitectura en capas, utilizando Java como lenguaje de programación e implementando el patrón de arquitectura Modelo-Vista-Controlador. Como Sistema de Gestión de Bases de Datos se utiliza PostgreSQL y como servidor de aplicaciones el JBoss Server. Son utilizadas las librerías de componentes web JBoss UI y RichFaces.

Entre los beneficios que aporta se encuentra proveer un control estricto de los datos de los pacientes, para acceder a la información clínica del paciente de manera ágil, facilitar los procesos de diagnóstico, tratamiento y seguimiento; y el sistema ayudará a la realización de investigaciones médicas mediante la revisión de los diagnósticos, entre otras tareas. El uso de esta aplicación permite la visualización de la información referente al paciente, gestionada por los demás módulos que comprenden AlasSIAPS.

Palabras Claves: Atención Primaria de Salud, AlasSIAPS, zoonosis, Higiene y Epidemiología.

Tabla de Contenidos

Introducción

La epidemiología es una de las ciencias básicas de la salud pública y se enfoca en los problemas de salud de la población. A través de ella se pueden identificar y estudiar las principales causas que provocan, permiten o facilitan la ocurrencia de grandes enfermedades epidémicas o pandemias, así como las enfermedades que afectan a grupos humanos por la carencia de alimentos específicos y, más recientemente, las enfermedades o afecciones relacionadas con el modo, las condiciones y los estilos de vida de la población.

“Desde la más remota antigüedad las sociedades humanas, en correspondencia con su nivel de desarrollo, han prestado atención a los problemas de salud y a la medicina, aplicando variantes que han permitido enfrentar las pandemias que han afectado a grandes grupos poblacionales. Por ejemplo, en la medicina China existe evidencia de un enfoque preventivo en materia de salud con la publicación del *Nei-Ching*, en el que figuran la viruela y los métodos para su prevención. Igualmente, la civilización egipcia incorpora aspectos relacionados con el saneamiento; los hebreos incluyen, en la ley mosaica, el Levítico, primer código de higiene, escrito el año 1500 antes de nuestra era”. [1]

Con el desarrollo de la humanidad y sus progresos se dieron los primeros pasos para el desarrollo de la sanidad y aparece la primera escuela de Higiene Ambiental y Experimental. No es hasta 1779 que el médico alemán Johann Peter Frank menciona la importancia de la organización de los estados y de las medidas sanitarias para la prevención de enfermedades. [2]

Desde tiempos coloniales en Cuba se realizan actividades de Higiene y Epidemiología. Ejemplo de ello es la primera acción de vacunación contra la viruela, liderada por el Dr. Tomás Romay Chacón, en 1802, y la presentación en la Real Academia de Ciencias Médicas, Físicas y Naturales de La Habana, de la doctrina del Dr. Carlos Juan Finlay, quien identificó al mosquito casero *Culex fasciatus*, conocido actualmente con el nombre de *Aedes aegypti*, como el vector transmisor de la fiebre amarilla.

El 17 de enero de 1899 en Cuba se establecieron algunas regulaciones sanitarias, que también fueron orientadas a partir de 1902 por el Dr. Carlos Juan Finlay. Pero no es hasta el año 1924 que se adopta el Código Sanitario Internacional, vigente hasta hoy con algunas modificaciones. A finales de los años 30 se crea la Unidad Sanitaria de Marianao, en La Habana, constituyéndose así el primer centro de atención a los problemas referentes a la Higiene y la Epidemiología de la comunidad.

INTRODUCCIÓN

A partir de los inicios del Gobierno Revolucionario en 1959, comienza un renacimiento del quehacer sanitarista nacional. Tan temprano como 1962 el recién estrenado Sistema Nacional de Salud comienza un estudio preliminar de la situación sanitaria y epidemiológica de la población cubana. Luego de la puesta en función de la "Medicina en la Comunidad" (1974) y de la implementación del modelo "Médico y Enfermera de la Familia", el análisis de la situación higiénico epidemiológica de la población pasó a ser una de las tareas básicas de estos especialistas. Actualmente el desempeño diario de estos profesionales del sector de la Salud, constituye una fortaleza para el Programa de Higiene y Epidemiología; dado que la comunidad está en constante monitoreo y control. [3]

Con la intención de optimizar los servicios médicos, la atención al paciente y acelerar los procesos concernientes con el trabajo diario de los médicos, el país ha perfilado como objetivo la informatización del sector de la salud. Por eso en los últimos años un grupo de instituciones cubanas han desarrollado sistemas para mejorar los servicios de la salud.

Una de las instituciones involucradas en la informatización de la sociedad cubana es la Universidad de las Ciencias Informáticas (UCI), encargada de la creación y distribución de software para el país y para otros países con los que Cuba mantiene relaciones de cooperación, especialmente los países del ALBA. La UCI cuenta con una infraestructura formada por 30 polos productivos que desarrollan proyectos temáticos, ubicados en 10 Facultades; la número 7 es la encargada de producir diferentes tipos de software para los programas del Sistema Nacional de Salud y uno de los proyectos con la que cuenta la facultad, es el APS del Sistema Integral de Atención Primaria de Salud (alas SIAPS).

Su función primordial es la de informatizar los principales procesos del alas SIAPS. Uno de los que actualmente se desarrolla, es el relacionado con las actividades que se realizan de Higiene y Epidemiología.

El Sistema Nacional de Salud (SNS) alcanza tres niveles de atención médica establecidos en: Atención Médica Primaria, Atención Médica Secundaria y Atención Médica Terciaria, además tiene una serie de componentes que lo conforman y una red de unidades que lo sustenta.

El desarrollo alcanzado por la Salud Pública Cubana brindó posibilidades objetivas al país, para continuar perfeccionando la estrategia de enfrentamiento integral a los problemas de salud y a las necesidades de salud de la población.

En este contexto, a partir del primer semestre de 1993, se definió un grupo de estrategias por el Área de Higiene y Epidemiología del Ministerio de Salud Pública, con el objetivo central de dar una respuesta mucho más eficiente a la situación sanitaria del país. [4]

En todo el país funcionan centros provinciales y municipales de Higiene y Epidemiologia que se nutren con las informaciones generadas en las diferentes instituciones de la salud. Tienen como misión mejorar la salud de la población, prevenir las enfermedades e infecciones, pronosticar las tendencias de una enfermedad, determinar si una enfermedad o problema de salud es prevenible o no y mantener un control de las mismas, además de participar en las investigaciones epidemiológicas orientadas al conocimiento de las enfermedades transmisibles y no transmisibles y su modificación. Uno de los departamentos de los centros de Higiene y Epidemiologia atiende la zoonosis, enfermedades que se transmiten de forma natural entre animales vertebrados y el hombre. [5]

En las últimas décadas ha ocurrido un incremento de algunas manifestaciones de zoonosis como resultado de los cambios socioculturales y ambientales que han determinado que los animales compartan, cada día con más frecuencia, su hábitat con las personas.

Para la detección de las zoonosis, el sistema de vigilancia en salud se nutre de fuentes del sector salud provenientes de los sistemas de información estadísticas del MINSAP —sistemas de información estadística de mortalidad, de enfermedades de declaración obligatoria (EDO), de información directa (SID), de información de letalidad y otros—, así como de los resultados de investigaciones realizadas, los informes de los programas de salud, los subsistemas de vigilancia existentes, los informes de países y organismos internacionales, la aplicación de técnicas de búsqueda activa de información para la vigilancia —encuestas, sitios centinelas, técnicas de evaluación rápida— y de la opinión de la población. Asimismo, se nutre de la información proveniente de sectores como Hidroeconomía, Educación, Meteorología y Veterinaria, entre otros, cuya actividad se vincula a la salud de la población. [6]

En el manejo de este complejo flujo de información, estos departamentos presentan serias limitaciones con la recepción de las informaciones que se envían de los diferentes centros de salud a través de mensajeros o por vía telefónica, lo que en ocasiones provoca incongruencias, errores y malentendidos entre los informantes y los informados. Por otra parte, se ha ido acumulando un gran volumen de información, que a pesar de que estar ordenada y clasificada, se dificulta su almacenamiento por falta de espacio. Debido a la cantidad de información que se procesa y la lentitud del flujo de transmisión de la misma por los diferentes niveles, la obtención de las estadísticas es un proceso demorado y resulta

engorrosa la generación de los reportes, lo que retrasa la toma oportuna de decisiones en los diferentes niveles del Programa de Higiene y Epidemiología.

La falta de uniformidad en los documentos y el desorden de los mismos trae consigo que la información no se encuentre cuando se necesita, lo que provoca una baja disponibilidad de la misma en los distintos niveles, dificultando el control de los registros por las instancias superiores. Los informes que se deben hacer de forma semanal utilizan registros de años anteriores, los cuales no se encuentran o son ilegibles por el paso del tiempo.

Teniendo en cuenta esta situación se plantea como **problema a resolver** el siguiente**:** ¿Cómo facilitar el proceso de gestión de la información en los departamentos de zoonosis de los centros de Higiene y Epidemiología de la Atención Primaria de Salud? Teniendo en cuenta el problema a resolver en la presente investigación el **objeto de estudio** se identifica como**:** El proceso de gestión de la información en los centros de Higiene y Epidemiología de la Atención Primaria de Salud.

Para dar solución al problema antes mencionado se define como **objetivo general de la investigación**: Desarrollar el componente informático para el Sistema Integral de Atención Primaria de Salud que facilite el proceso de gestión de la información en los departamentos de zoonosis de los Centros de Higiene y Epidemiología. Dentro del objeto de estudio se delimita como **campo de acción**: El proceso de gestión de la información en los departamentos de zoonosis de los centros de Higiene y Epidemiología de la Atención Primaria de Salud.

Como **idea a defender** en la presente investigación se plantea la siguiente: Con la implantación de un componente informático que facilite la gestión de la información en los departamentos de zoonosis de los centros de Higiene y Epidemiología en la Atención Primaria de Salud, se garantizará una mayor disponibilidad, dominio y mejor aprovechamiento de la información.

Para el desarrollo de la investigación se definieron las siguientes **Tareas de la investigación:**

1. Realizar la fundamentación teórica como resultado de la revisión bibliográfica e investigación del estado actual de la zoonosis en los centros de Higiene y Epidemiología en la APS.
2. Revisar el estado actual en Cuba y las tendencias mundiales vinculadas a la zoonosis.
3. Elaborar los documentos y artefactos correspondientes mediante la metodología definida para las fases de trabajo: Modelado de Negocio, Gestión de Requerimientos, Diseño e

Implementación del proceso de zoonosis.

4. Actualizar el modelo de datos y descripción de las entidades existentes en anteriores levantamientos.
5. Implementar las funcionalidades del proceso de zoonosis.

La metodología de la investigación científica es la reflexión sistemática acerca del método, los procedimientos y las técnicas utilizadas para obtener conocimientos verdaderos y objetivos.

Método de la observación: En la investigación realizada se estudió todo el proceso de gestión de la información del departamento de Zoonosis, con el objetivo de descubrir sus características esenciales.

La entrevista: Mediante la entrevista, se recopilaron los datos necesarios para un mejor entendimiento del negocio a desarrollar y sus principales vulnerabilidades para un manejo adecuado de los datos. Se empleó una entrevista no estructurada, ya que se les realizó a especialistas en el tema.

El documento está estructurado en cuatro capítulos:

CAPÍTULO 1. Fundamentación teórica: Contiene los aspectos esenciales para entender la información referente a la zoonosis. Se describen los conceptos fundamentales asociados al dominio del problema, se hace una valoración crítica de los sistemas informáticos utilizados en la actualidad en la esfera de la salud que se refieren a la zoonosis en la APS y se hace un estudio de los lenguajes, tendencias, herramientas y tecnologías utilizadas para el diseño de aplicaciones web.

CAPÍTULO 2. Características del sistema: Describe los principales procesos vinculados al problema a resolver, definiendo los procesos identificados a través del Modelamiento del Negocio que representa las necesidades del cliente. También se realiza la especificación de los requerimientos funcionales y no funcionales, se identifican los procesos que requieren automatización y se define de forma general una propuesta del sistema a través del modelamiento del sistema.

CAPÍTULO 3. Diseño del sistema: Se describen los aspectos relacionados con el diseño de la solución propuesta y se define la estructura y los elementos del diseño, incluyendo los patrones de diseño a utilizar en el desarrollo de la aplicación.

CAPÍTULO 4. Implementación y prueba: Se realiza una descripción detallada de los métodos más complejos en el desarrollo de la aplicación y los agentes en el proceso de implementación, así como los estándares de diseño, codificación y tratamientos de errores utilizados.

Capítulo 1. Fundamentación teórica

El objetivo fundamental de este capítulo es abordar distintos aspectos que se utilizan como soporte teórico del sistema a desarrollar. Se describen los conceptos asociados al problema y la estructura organizativa del Sistema Nacional de Salud (SNS). También se analizan los conceptos que se deben conocer para poseer un dominio básico de problema y se realiza un análisis detallado del estado del arte en cuanto a sistemas, tecnologías, metodologías y herramientas.

1.1 Sistema Nacional de Salud

El Ministerio de Salud Pública (MINSAP) es el organismo rector del Sistema Nacional de Salud. Es el encargado de dirigir, ejecutar y controlar la aplicación de la política del Estado y del Gobierno en cuanto a la Salud Pública, el desarrollo de las Ciencias Médicas y la Industria Médico Farmacéutica. [7]

Entre sus funciones rectoras está ejercer el control y la vigilancia epidemiológica de las enfermedades y sus factores de riesgo, el control y la vigilancia sanitaria de todos los productos que pueden tener influencia sobre la salud humana, regular y controlar la aprobación, normar las condiciones higiénicas y el saneamiento del medio ambiente en aquellos aspectos que puedan resultar agresivos a la salud humana y controlar su cumplimiento a través de la inspección sanitaria estatal, entre muchas otras.

El SNS se organiza en tres niveles de atención a la población: Atención Primaria, Atención Secundaria y Atención Terciaria.

Atención Primaria: Es la asistencia sanitaria esencial, basada en métodos y tecnologías prácticas, científicamente fundamentadas y socialmente aceptables, puestas al alcance de todos los individuos y familiares de la comunidad, mediante su plena participación y a un costo que la comunidad y el país puedan soportar. [8]

Atención Secundaria: Este nivel se enfoca en la promoción, prevención y diagnóstico de la salud, los cuales brindarán acciones y servicios de atención ambulatoria especializada y de hospitalización a pacientes derivados del primer nivel o de los que se presentan de modo espontáneo con urgencias. [9]

Atención Terciaria: Este nivel se ubica a escala del ámbito nacional y constituye el centro de referencia de mayor complejidad nacional y regional. Aquí laboran especialistas para la atención de problemas patológicos complejos, que necesiten equipos e instalaciones especializadas. [10]

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1 Marco conceptual

Vigilancia epidemiológica: La recolección sistemática, continua, oportuna y confiable de información relevante y necesaria sobre algunas condiciones de salud de la población. El análisis e interpretación de los datos debe proporcionar bases para la toma de decisiones y, al mismo tiempo, ser utilizada para su difusión. [11]

Prevención: La prevención de enfermedades es el conjunto de acciones realizadas para evitar que el daño o la enfermedad aparezcan, se prolonguen, ocasionen daños mayores o generen secuelas evitables. [12]

Promoción de salud: La promoción de salud constituye un proceso político social y una estrategia que puede contribuir efectivamente a la mejora de la calidad de vida y la construcción de una sociedad de bienestar. Su enfoque se basa en la concepción de salud como un proceso social, resultante de las condiciones e interacciones de las personas con su medio local. Desde esta perspectiva, la promoción de la salud considera que la preservación, mantenimiento y mejora de la salud requiere de la participación de la comunidad, del empoderamiento social y del despliegue de esfuerzos intersectoriales. [13]

1.2 Zoonosis

Las zoonosis son enfermedades que se transmiten de forma natural entre animales vertebrados y el hombre, estas pueden clasificarse en sinantrópicas[1] cuando tienen un ciclo urbano o exoantrópicas,[2] cuando el ciclo es selvático. Algunas de estas enfermedades son: el paludismo, el dengue, la leishmaniosis, la rabia, la toxoplasmosis, la triquinosis, la criptococosi, la enfermedad de Creutzfeldt-Jakob, el hanta, entre otros.

Estas enfermedades ganan cada vez mayor importancia debido al aumento de la población y cada vez más la civilización ocupa o comparte, hábitats que antes sólo lo ocupaban los animales vertebrados y los insectos. Además, las características tropicales del país, las extensas áreas agrícolas, y los regímenes lluviosos son factores favorables para el incremento de las mismas en la población.

Por el creciente aumento de estas enfermedades por encima de los índices habituales en Cuba, se crearon departamentos de zoonosis en todos los centros de Higiene y Epidemiología del país, para un mayor control de los casos y darle un tratamiento y seguimiento de acuerdo al grado de las enfermedades

[1] Se refiere a animales que viven en estrecha asociación con los seres humanos.
[2] Se refiere a animales que viven en un ciclo selvático.

existentes. Por otra parte, se incrementan las tareas de capacitación de los recursos humanos y de educación para la salud, a través de la formación de personal profesional y técnico. Juega un papel primordial la difusión en la población, de los riesgos asociados con las zoonosis y sus medidas preventivas, para lograr en la población, una mejora de la calidad de vida.

1.3 Análisis de las soluciones existentes

Antecedentes nacionales

Diseño del Módulo Higiene y Epidemiología del Sistema Integral para la Atención Primaria

El trabajo está conformado por la realización del análisis y diseño del Módulo Higiene y Epidemiología del Sistema Integral para la Atención Primaria de Salud. Fue desarrollado en la Universidad de las Ciencias Informáticas (UCI) en el curso 2009-2010. El mismo aborda el proceso de zoonosis en los centros de Higiene y Epidemiología, a partir de un estudio preliminar de estos procesos, donde se realizó una propuesta de negocio, requerimientos, análisis y diseño.

Una vez estudiado el negocio a profundidad se pudo determinar la existencia de nuevas actividades que son imprescindibles para lograr el desarrollo exitoso del mismo y que no fueron concebidas en esa investigación. A partir de esto se hace ineludible una nueva propuesta, y la implementación de un sistema que integre todas las actividades identificadas para la gestión de la información.

Sitio Web de Higiene y Epidemiología

En la Escuela Latinoamericana de Medicina se cuenta con un sitio web de Higiene y Epidemiología, dirigido a los estudiantes, profesionales y trabajadores de la escuela. Su objetivo principal es contribuir al desarrollo del trabajo educativo y elevar la calidad de vida al brindar conocimientos esenciales, a través de la divulgación de informaciones sobre los diferentes aspectos que abarca la especialidad de Higiene y Epidemiología.

Constituye un espacio de comunicación y reflexión que permite divulgar y contribuir al conocimiento de aspectos tan importantes como las enfermedades trasmisibles, las no trasmisibles, el control de algunos vectores, el control sanitario internacional, la higiene de los alimentos, la higiene escolar, entre otros aspectos y con el que a su vez se contribuye a desarrollar la cultura computacional de los estudiantes y trabajadores.

A pesar del alto nivel promocional que el sitio ofrece, no cuenta con funcionalidades que permitan tabular informaciones o datos estadísticos que pueden resultar de gran ayuda para confeccionar tablas resúmenes de las enfermedades.

Antecedentes Internacionales

Sistema de Alerta Sanitaria Veterinaria

Una red informática española que integra las bases de datos sanitarias, denominada RASVE, contiene la información necesaria para realizar una evaluación epidemiológica de las alertas y la comunicación con otras aplicaciones, pero no abarca toda la información necesaria en el sector de Higiene y Epidemiología, como el seguimiento de los casos y un tratamiento a los pacientes o a los animales afectados.

Sistema Mundial de Información Sanitaria

El Sistema Mundial de Información Sanitaria, más conocido por sus siglas en inglés WAHIS (World Animal Health Information System), es un sistema informático disponible en Internet que permite ingresar los datos relativos a las enfermedades animales para informar después a la comunidad internacional acerca de los acontecimientos epidemiológicos. El acceso a este sitio está restringido a los usuarios autorizados por la Organización Mundial de Sanidad Animal (OIE), por lo que no se puede dar uso a sus datos. Además, el sitio es solamente informativo, no maneja otro tipo de información que no sea la de avisar los brotes epidémicos y realizar reportes semanales y anuales con esta información.

Sistema de Notificación de Enfermedades Animales

Es un sistema informático español de notificación de brotes primarios y secundarios. Tiene carencia en cuanto a la prevención de las enfermedades, seguimiento de casos y atención a las personas involucradas.

1.4 Situación problémica

Los centros de Higiene y Epidemiologia de la APS, y más específicamente los departamentos de Zoonosis en todo el país, presentan una serie de problemas, entre ellos los relacionados con la gestión de la información. Durante cada día a estos departamentos llegan los datos de cada uno de los casos que se detectan en las poblaciones, así como información de otras acciones que se realizan en apoyo a las actividades de estos departamentos.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En la actualidad, el procesamiento de la información se realiza de forma manual, lo cual aumenta la posibilidad de que existan alteraciones como consecuencia de equivocaciones producto al gran cúmulo de datos a recoger. Este procesamiento se debe realizar de una manera dinámica para que no pierda valor la información, y generalmente no se dispone de tiempo suficiente para generar toda la información en un corto plazo, por lo que resulta engorroso para el personal que lo realiza lograr la calidad necesaria.

Toda la información que se procesa es recogida en modelos confeccionados en papel, material que con el paso del tiempo se maltrata y en algunos casos puede ocurrir la pérdida de información importante para los reportes que se realizan de forma semanal y anual.

Por otra parte, existe falta de uniformidad en la información debido a la no utilización de estándares, lo que provoca problemas de comunicación entre los distintos niveles de atención a la salud que recibe la información. Otro problema de gran envergadura se evidencia a la hora de realizar búsquedas de algún dato que se necesite, pues que se torna muy lento el proceso debido primordialmente a que en los lugares donde se encuentran archivados los documentos existe en ocasiones desorden por la falta de espacio para su almacenamiento y ocurren pérdidas de los materiales o deterioro por el transcurso del tiempo. Además, las búsquedas implican la revisión y lectura de cada uno de los documentos, lo que determina una mayor cantidad de tiempo para obtener la información solicitada.

Estas dificultades limitan la capacidad de respuesta de estos centros a la población, trayendo como consecuencia que se vea afectada la calidad de los servicios que se ofrece.

Otro problema frecuente es que el especialista encargado debe dedicar mucho tiempo y esfuerzo al procesamiento de la información y a la obtención de estadísticas debido a que lo hace manualmente a partir de la documentación en copia dura, lo cual conlleva a que los departamentos de estadísticas del país no posean información detallada y a corto plazo. Esto dificulta grandemente la realización de investigaciones y la generación de reportes y consultas que permitan tener una información actualizada en cualquier momento del año y que ayuden a identificar posibles brotes o epidemias de enfermedades que necesitan un seguimiento por su impacto en la comunidad.

El Ministerio de Salud Pública (MINSAP) ha definido la informatización como una de sus prioridades, por tanto, es de una vital importancia crear sistemas informáticos para el control de la información que se maneja en estos departamentos de zoonosis, lo que sin dudas va a tributar a un superior aprovechamiento de la jornada laboral y un mejoramiento de la calidad del trabajo hasta el momento desempeñado.

1.5 Tecnologías y herramientas utilizadas

Las herramientas y tecnologías fueron definidas por el Centro de Informática Médica (CESIM) para el desarrollo de aplicaciones en el departamento de Atención Primaria de Salud. En esta sección se mencionan algunas de estas tecnologías, metodología y herramientas utilizadas en el desarrollo del Componente Web Zoonosis del SIAPS, así como algunas de sus características más significativas.

1.6 Metodologías de desarrollo

En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. Las metodologías son un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software.

La creciente dimensión de los proyectos relacionados con los sistemas de información hace que sea imprescindible acometerlos con la mayor solidez metodológica, por esto la universidad decidió utilizar el proceso de mejora con el objetivo de alcanzar el nivel 2 de Capability Maturity Model Integration (CMMI).

Proceso de mejora

CMMI (Capability Madurity Model Integration), es un modelo de referencia para el crecimiento de capacidades y madurez, que se enfoca tanto en procesos de administración como de Ingeniería de Sistemas y Software. [14]

- CMMI-SE/SW/IPPD/SS
- CMMI-SE/SW/IPPD
- CMMI-SE/SW
- CMMI-SW

Notación Utilizada para Modelar los Procesos del Negocio (BPMN).

La Notación para el Modelado de Procesos de Negocio (BPMN) es un nuevo estándar que permite el modelado de procesos de negocio, en un formato de flujo de trabajo. BPMN proporciona a los negocios la capacidad de entender sus procedimientos internos en una notación gráfica, facilitando a las organizaciones la habilidad para comunicar esos procedimientos de una manera estándar.

Business Process Management (BPM)

Es una metodología empresarial para el modelado, integración, monitoreo y mejora continua de procesos de negocios a través de la gestión sistémica de los mismos.

De manera integral se puede entender BPM como el mejoramiento de la gestión de los procesos de negocio de una organización de principio a fin, a partir de la definición deliberada, colaborativa e incremental de la tecnología, para alcanzar claridad en la dirección estratégica, alineación de los recursos de la empresa y disciplina de mejoramiento continuo, necesarias para cumplir las expectativas de los clientes.

Beneficios que proporciona BPM:

- Visibilidad de los procesos de las empresas.
- Mayor flexibilidad y agilidad para adaptación al cambio.
- Dirigir los esfuerzos de la empresa de una manera planeada y alineada con los objetivos estratégicos.
- Permite la documentación de los procesos de su compañía, permitiendo la identificación de puntos a mejorar y perfeccionar.
- Permite conocer indicadores de rendimientos de los procesos de negocios en tiempo real, posibilitando la toma de decisiones de manera temprana. [15]

Lenguaje Unificado de Modelado (UML).

El Lenguaje de Modelado Unificado (UML: Unified Modeling Language) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80 y principios de los 90.UML es llamado un lenguaje de modelado, no un método. Los métodos consisten de ambos de un lenguaje de modelado y de un proceso. El UML fusiona los conceptos de la orientación a objetos aportados por Booch, OMT y OOSE (Booch, G. et al., 1999). UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. Los autores de UML apuntaron también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios. [16]

1.7 Estilos arquitectónicos

Se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarnos en el desarrollo de software dentro de un sistema informático. Así, los programadores, diseñadores, ingenieros y analistas pueden trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado. [17]

Arquitectura Cliente/Servidor

La modalidad o arquitectura Cliente/Servidor es aquella en la que confluyen una serie de aplicaciones basadas en dos categorías que cumplen funciones diferentes (una requiere servicios y la otra los brinda) pero que a la vez, pueden realizar tanto actividades en forma conjunta como independientemente. Esas dos categorías son justamente cliente y servidor.

En el caso del cliente, es aquel que requiere un servicio del servidor. En esta categoría se realizan funciones de software basándose en el hardware pero en caso de no tener la capacidad de procesar los datos necesarios, recurre al servidor y espera a que este le brinde los servicios solicitados. El cliente es una estación de trabajo o computadora que está conectada a una red a través de la cual puede acceder al servidor.

Por el contrario, el servidor es la máquina desde la que se suministran servicios y que está a la espera del requerimiento del cliente. Una vez hecho, busca la información solicitada y le envía la respuesta al cliente; incluso puede enviar varios servicios a la vez, lo que es posible porque entre ellos están conectados mediante redes LAN o WAN. [18]

Patrón Modelo-Vista-Controlador (MVC)

Se utiliza el patrón de diseño Modelo-Vista-Controlador, para el diseño de aplicaciones con sofisticados interfaces. Se trata de realizar un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos.

Elementos del patrón:

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.

- Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
- Lleva un registro de las vistas y controladores del sistema.

El controlador es responsable de:

- Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas.

Las vistas son responsables de:

- Recibe los datos del modelo y los muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque, además, lo instancia).
- Pueden dar el servicio de "Actualización ()", para que sea invocado por el controlador o por el modelo. [20]

Tecnologías en la capa de Presentación

JavaServer Faces (JSF) 1.2

La tecnología JavaServer Faces es un framework de interfaz de componentes de usuarios del lado del servidor para las aplicaciones web basadas en la tecnología Java. Los principales componentes de la tecnología JSF son:

- Una API para:
  - ✓ Representar componentes de Interfaz de Usuario (UI) y gestionar su estado.
  - ✓ Manejar eventos, validar en el servidor y conversión de datos.
  - ✓ Definir la navegación de páginas.
  - ✓ Soporte de internacionalización y accesibilidad.
- Dos librerías de etiquetas JSP personalizadas para expresar componentes en una página JSP y enlazar los componentes a objetos del servidor. [21]

RichFaces 3.3.1

RichFaces es un framework de código abierto que añade capacidad Ajax dentro de aplicaciones JSF existentes sin recurrir a JavaScript. RichFaces incluye ciclo de vida, validaciones, conversores y la gestión de recursos estáticos y dinámicos. Los componentes de RichFaces están construidos con soporte Ajax y un alto grado de personalización del "look-and-feel" que puede ser fácilmente incorporado dentro de las aplicaciones JSF. [22]

Ajax4JSF

Ajax4jsf es una librería open source que se integra totalmente en la arquitectura de JSF y extiende la funcionalidad de sus etiquetas dotándolas con tecnología Ajax de forma limpia y sin añadir código Javascript. Mediante este framework se puede variar el ciclo de vida de una petición JSF, recargar determinados componentes de la página sin necesidad de recargarla por completo, realizar peticiones al servidor automáticas, control de cualquier evento de usuario, etc. En definitiva Ajax4jsf permite dotar a nuestra aplicación JSF de contenido mucho más profesional con muy poco esfuerzo. [23]

Facelets 1.1

JavaServer Facelets es un framework para plantillas (templates) centrado en la tecnología JSF (JavaServer Faces), por lo cual se integran de manera muy fácil. Este framework incluye muchas características siendo las más importantes:

- Facilidad en la creación del templating para los componentes y páginas.
- Habilidad de separar los UIComponents en diferentes archivos.
- Soporte completo a EL (Expression Language).
- Validación de EL en tiempo de construcción.
- No es necesaria configuración XML.[24]

Java Script

Es un lenguaje de programación que aporta características dinámicas (datos variables en función del tiempo y el modo de acceso, interactividad con el usuario, personalización, etc.) a las páginas Web, escritas en lenguaje HTML. [25]

HTML

El HTML, Hyper Text Markup Language (Lenguaje de marcación de Hipertexto) es el lenguaje de marcas de texto utilizado normalmente en la www (World Wide Web). HTML no es propiamente un lenguaje de programación como C++, Visual Basic, etc., sino un sistema de etiquetas. HTML no presenta ningún compilador, por lo tanto, algún error de sintaxis que se presente, éste no lo detectará y se visualizara en la forma como él entienda. [26]

IReport 3.1.4

La herramienta iReport es un constructor / diseñador de informes visual, poderoso, intuitivo y fácil de usar para JasperReports escrito en Java. Este instrumento permite que los usuarios corrijan visualmente informes complejos con cartas, imágenes, subinformes, etc. iReport está además integrado con JFreeChart, una de la biblioteca gráficas OpenSource más difundida para Java. Los datos para imprimir pueden ser recuperados por varios caminos incluso múltiples uniones JDBC, TableModels, JavaBeans, XML, etc.

Tecnologías en la capa de Negocio

Servidor de Aplicaciones 4.2

JBoss es un proyecto de código abierto, con el que se consigue un servidor de aplicaciones basado en J2EE, e implementado al 100 % en Java. Por lo tanto al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo que lo soporte.

JBoss implementa todo el paquete de servicios de J2EE (EJB, JMS, JTS/JTA, Servlets/JSP, JNDI, etc.) y también ofrece características tales como los clustering, JMX, Web Services y la integración IIOP, y la principal característica que desde que JBoss está licenciado bajo la LGPL, puede libremente usarse sin costo alguno en cualquier aplicación comercial o ser redistribuirlo. [27]

Entre sus principales características se encuentran que es de código abierto, escalable, de alto desempeño, con arquitectura modular, producto de licencia de código abierto sin coste adicional, cumple los estándares, es confiable a nivel de empresa, entre otros.

Enterprise Java Beans (EJB) 3.0

Enterprise Java Beans es una arquitectura que define la forma de construir componentes distribuidos del lado del servidor. Esta tecnología garantiza que los componentes programados sean escalables, eficaces y seguros a pesar de lo sencillo de su desarrollo. Sus principales características son:

- Los EJB son componentes distribuidos, esto quiere decir que se pude hablar de componentes que se ejecutan en diferentes servidores, contra diferentes bases de datos, o no, es una decisión del analista-desarrollador.
- Los EJB se ejecutan dentro de un servidor de aplicaciones, estos servidores deben cumplir el estándar fijado por SUN Microsystem INC., y se encargan de gestionar recursos como red, los pool de conexiones, o la gestión de la seguridad. El desarrollador construye los EJB y los servidores de aplicaciones los gestionan.[28]

Tecnologías en la capa de Acceso a Datos

Hibernate 3.3

Hibernate parte de una filosofía de mapear objetos Java, también conocidos como “POJOs” (Plain Old Java Objects). Con Hibernate no es necesario escribir código específico en nuestros objetos ni hacer que hereden de clases determinadas. En vez de eso se trabaja con ficheros XML y objetos que proporciona la librería. Una de las principales características de Hibernate es su flexibilidad, envolviéndolo todo bajo un marco de trabajo común. [29]

Java Persistence API (JPA) 2.0

El Java Persistence API fue desarrollado por el grupo de expertos de EJB 3.0, aunque su uso no se limita a los componentes software EJB. También puede utilizarse directamente en aplicaciones web y aplicaciones clientes; incluso fuera de la plataforma Java EE, por ejemplo, en aplicaciones Java SE. En su definición se han combinado ideas y conceptos de los principales frameworks de persistencia como Hibernate, Toplink y JDO, y de las versiones anteriores de EJB. Todos estos cuentan actualmente con una implementación JPA.

JBoss Seam 2.1.1

JBoss Seam es un poderoso y moderno framework que integra la capa de presentación (JSF) con la capa de negocios y persistencia (EJB). Una característica importante es que se pueden hacer validaciones en los POJOs (Plain Old Java Objects), mientras que en los frameworks tradicionales todo el estado es administrado básicamente en la sesión HTTP, por su parte Seam provee una mayor granularidad de contextos de estado. [30] Se encuentra en todas las capas de desarrollo.

1.8 Herramientas a utilizar

Visual Paradigm 6.4

El Visual Paradigm es una herramienta CASE que utiliza “UML” como lenguaje de modelado, con el uso del acercamiento orientado al objeto. Esta herramienta apoya los estándares más altos de las notaciones de Java y de UML. Genera productos de calidad, soporta aplicaciones web y es fácil de instalar y actualizar. En sentido general reduce significativamente los esfuerzos en todas las etapas del ciclo de vida de desarrollo de software. [31]

Eclipse 3.5.2

Para el desarrollo del software se utilizará el Eclipse, que es un entorno integrado (IDE) para desarrollo de aplicaciones con java. Está soportado por IBM, es un proyecto open source, multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido". Se está convirtiendo en el estándar de facto de los entornos de desarrollo para Java. Y es que Eclipse no es tan sólo un IDE, se trata de un marco de trabajo modular ampliable mediante complementos (plugins). De hecho, existen complementos que nos permiten usar Eclipse para programar en PHP, Perl, Python, C/C++, etc. [32]

PostgreSQL 8.4

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales.

PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. [33]

Cliente de Base de datos pgAdmin III 1.10.5

Es la herramienta Open Source de administración por excelencia para sus bases de datos PostgreSQL. Algunas de sus características son: el soporte completo para UNICODE, edición rápida de consultas y datos multihilo y soporte para todos los tipos de objetos de PostgreSQL. [34]

Toad Data Modeler 3.3

Toad Data Modeler es una ayuda a la hora de diseñar y crear bases de datos. No solamente nos permite crear con toda libertad toda clase de esquemas, diagramas y diseños, sino que también genera el código SQL para construirlas. Soporta cualquier sistema gestor de bases de datos: Access, Firebird, InterBase, MySQL, Oracle, Paradox, Postgre, Sybase, entre otros

En este capítulo se profundizó en los conocimientos de conceptos necesarios para una correcta comprensión de la investigación. Se mostró el resultado del estudio de los sistemas existentes en el mundo para un manejo de la información relacionada con la zoonosis, destacándose los elementos que impiden su uso en Cuba. Además, se realizó un análisis de las tecnologías, metodologías, lenguajes y herramientas que serán utilizadas a lo largo del desarrollo del sistema propuesto.

Capítulo 2. Características del sistema

En el presente capítulo se describen las características del sistema a construir y se detallan los principales procesos del negocio identificados en el departamento de Zoonosis. De los procesos se obtendrán los documentos que contienen la información gestionada para el correcto funcionamiento de esta área, algunos de ellos son los modelos del negocio y los diagramas correspondientes, describiéndose los actores y trabajadores que intervienen en los mismos. Además, se podrá tener una visión del sistema a partir de las funcionalidades requeridas y las restricciones que se imponen.

2.1 Modelo de negocio

Un proceso de negocio consiste en una colección de actividades que son realizadas coordinadamente en un ambiente técnico y organizacional. La conjunción de estas actividades logra un objetivo del negocio. Los procesos de negocio toman una o más tipos de entradas (precondiciones) y crea una salida (pos condición) que posee un determinado valor para el cliente. [35]

Diagrama de Procesos del Negocio

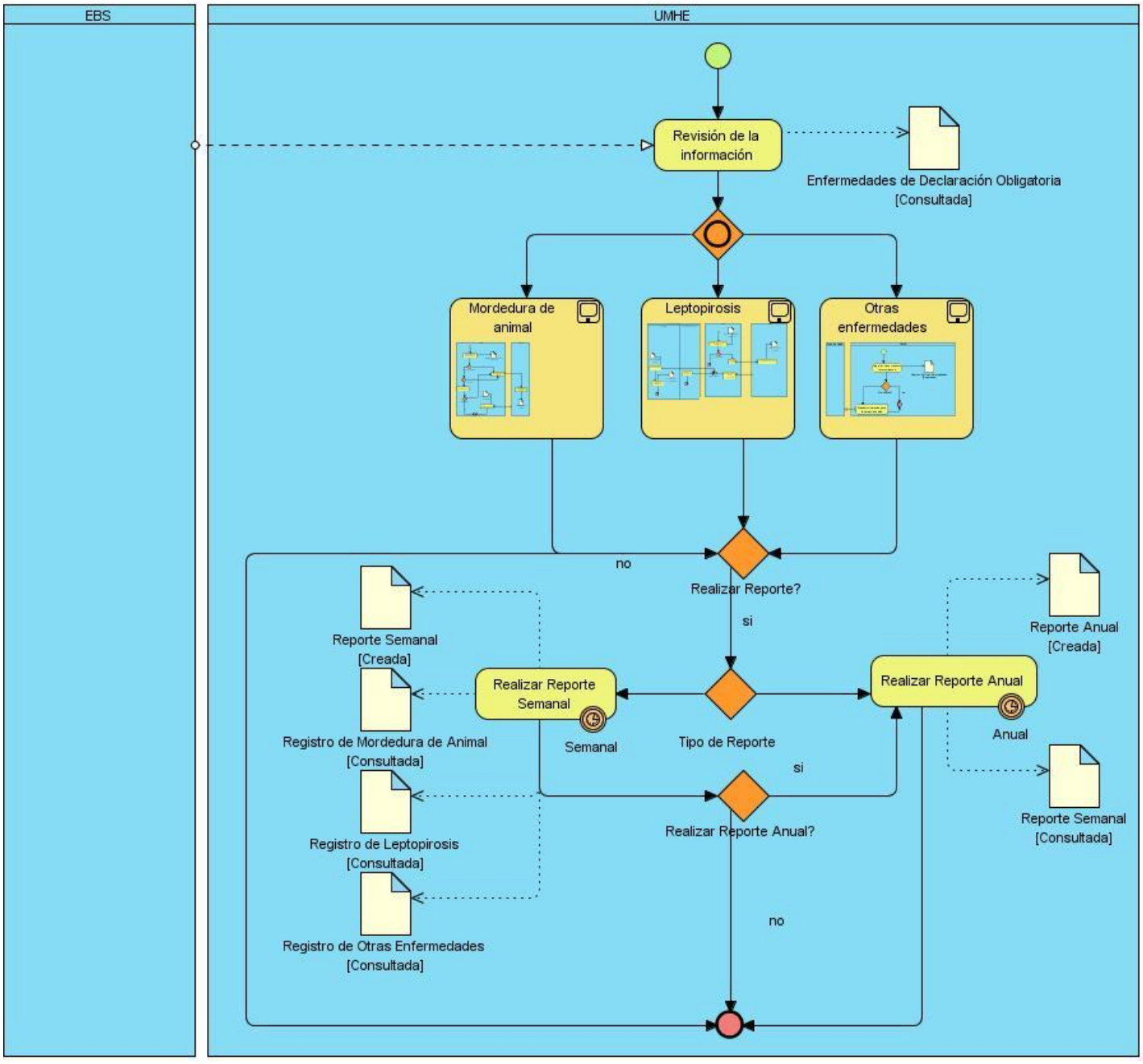


Figura 2 Diagrama de Procesos del Negocio. Proceso zoonosis.

2.1.1 Proceso Zoonosis

Nombre:	P1_ Proceso de zoonosis
Objetivos:	Definir tipo de zoonosis para darle su correcto tratamiento.
Evento(s) que lo generan:	Llegada de casos a los departamentos de Zoonosis.
Precondiciones:	Que exista una infección por un animal.
Pos condiciones:	No procede
Reglas de Negocio:	Regla de negocio (Regla Textual # 21) Ver documento APS_SIAPS_0116_RNeg_HigEpid_W.
Responsables:	Especialista del departamento higiene y epidemiología.
Clientes internos:	CESIM
Clientes externos:	No procede
Entradas:	No procede
Salidas:	Reportes Zoonosis
Actividades	1. Recibir la información. 2. Subproceso de Mordedura animal. 3. Subproceso de Otras enfermedades. 4. Subproceso de Leptopirosis. 5. Realizar Reportes

2.1.1.1 Descripción de actividades del proceso zoonosis

Todo el proceso de zoonosis se inicia una vez que llega la información de un paciente de un área de salud a un centro de Higiene y Epidemiología en la Atención Primaria de Salud y se remiten los datos al departamento de Zoonosis para que sea revisada y valorada por un epidemiólogo, quien identifica la posible enfermedad zoonósica. En dependencia del caso que corresponda se introduce la información del paciente en el registro de la enfermedad y se orienta la realización de exámenes para detectar si es positivo o no, a alguna de las enfermedades zoonósicas, que pueden ser: leptopirosis, mordedura de animal, criptococosis, toxoplasmosis, u otras.

Una vez finalizada la realización de este subproceso o de estos subprocesos (porque se pueden gestionar varios casos de forma paralela), se actualizar la información en el registro de la enfermedad con los resultados de los análisis, para saber si se descarta el caso o fue positivo.

Por otra parte, de forma semanal se realiza un reporte, en el cual se recoge toda la información referente a las enfermedades y las especificidades de los casos que fueron llegando al departamento a lo largo de la semana. También se realizarán reportes anuales que contendrán la información de las enfermedades zoonósicas almacenadas en los reportes semanales de todo el año, y se realizará una comparación con el año anterior en ese mismo período para establecer las diferencias en cuanto al crecimiento o decrecimiento de las enfermedades zoonósicas.

2.2 Propuesta del sistema

Se propone el modelado de un sistema que permita la gestión la información en los departamentos de Zoonosis en la APS. En el flujo de trabajo de Requerimientos se define qué es lo que el sistema debe realizar para lo cual se determinan las funcionalidades requeridas y las restricciones que se imponen. El análisis del problema y la comprensión de las necesidades de los interesados son los principales objetivos de los requisitos durante la fase inicial de un proyecto. Durante las fases de elaboración y de construcción, el énfasis se traslada hacia la definición inicial y el posterior perfeccionamiento de la definición del sistema en términos de los requisitos detallados. La gestión del ámbito del sistema y los cambios continuos en los requisitos se tratan durante todo el proyecto.

2.3 Especificación de los requerimientos de software

La especificación de los requerimientos de software es una descripción completa del comportamiento del sistema que se va a desarrollar. Es decir, especifica qué debe hacer el sistema en sus requerimientos. Un

requerimiento puede definirse como un elemento necesario dentro de un sistema, que puede representar una condición o capacidad, para satisfacer un contrato, estándar, especificación u otro documento formal. Esta actividad debe hacerse con gran precisión, por el papel primordial que juega en la producción de software, enfocada en la definición de lo que se desea producir, mediante una descripción más clara del comportamiento del sistema.

Los requerimientos se dividen en: **Funcionales**: son capacidades o condiciones que el sistema debe cumplir. **No funcionales:** son propiedades o cualidades que el producto debe tener.

Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir.

Después de analizados los procesos del negocio, se han definido los siguientes requisitos funcionales:

Gestionar Animales Transmisores de Enfermedades	
Buscar animales transmisores de enfermedades.	Agregar animales transmisores de enfermedades.
Modificar animales transmisores de enfermedades.	Eliminar animales transmisores de enfermedades.
RF 18- Gestionar Tipos de Enfermedades Zoonósicas	
Buscar tipos de enfermedades zoonósicas.	Agregar tipos de enfermedades zoonósicas.
Modificar tipos de enfermedades zoonósicas.	
Gestionar Estado de Comprobación.	
Buscar estado de comprobación.	Agregar estado de comprobación.
Modificar estado de comprobación.	
Gestionar Registro de Zoonosis.	
Buscar registro de zoonosis.	Agregar registro de zoonosis.

Modificar registro de zoonosis.	Detalles registro de zoonosis.
Exportar registro de zoonosis.	
Generar Reportes de zoonosis.	
Exportar reportes de zoonosis.	

Requerimientos no funcionales

Los requisitos no funcionales se refieren a las propiedades o cualidades del sistema. Estos responden a cualidades que el producto debe tener y las características para que este sea atractivo, confiable, usable y seguro.

Seguridad

- Se mantendrá seguridad y control a nivel de usuario, garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo a la función que realizan. Las contraseñas podrán cambiarse solo por el propio usuario o por el administrador del sistema.
- Se mantendrá un segundo nivel de seguridad a nivel de estaciones de trabajo, garantizando sólo la ejecución de las aplicaciones que hayan sido definidas para la estación en cuestión.
- El sistema implementará un mecanismo de auditoría para el registro de todos los accesos efectuados por los usuarios, proporcionando un registro de actividades (log) de cada usuario en el sistema.
- Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la BD, independientemente de que para el sistema, este elemento ya no exista.
- El sistema permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.
- Las informaciones médicas relacionadas con los pacientes y que vayan a ser intercambiadas con otros policlínicos por la red pública, viajarán cifradas para evitar accesos o modificaciones no autorizadas.

Restricciones de diseño

La capa de presentación contendrá todas las vistas y la lógica de la presentación. El flujo web se manejará de forma declarativa y basándose en definiciones de procesos del negocio. La capa del negocio mantendrá el estado de las conversaciones y procesos del negocio que concurrentemente pueden estar siendo ejecutados por cada usuario. La capa de acceso a datos contendrá las entidades y los objetos de acceso a datos correspondientes a las mismas. El acceso a datos está basado en el estándar JPA y particularmente en la implementación del motor de persistencia Hibernate.

Interfaz de usuario

Las ventanas del sistema contendrán los datos claros y bien estructurados, además de permitir la interpretación correcta de la información. La interfaz contará con teclas de función y menús desplegables que faciliten y aceleren su utilización. La entrada de datos incorrecta será detectada claramente e informada al usuario. Todos los textos y mensajes en pantalla aparecerán en idioma español.

Requerimientos de hardware

Estaciones de trabajo

En la solución se incluyen estaciones de trabajo para las consultas del Sistema para la Atención Primaria de Salud (AlasSIAPS), las que necesitan capacidad de hardware que soporte un sistema operativo que cuente con un navegador actualizado y que siga los estándares web, por lo que se escogieron estaciones de trabajo de 256 Mb de memoria RAM y un microprocesador de 2.0 Hz.

Servidores

La solución estará conformada, fundamentalmente, por servidores de alta capacidad de procesamiento y redundancia, que permitan garantizar movilidad y residencia de la información y las aplicaciones bajo esquemas seguros y confiables. Servidores de Base de datos: Procesador Dual - Core 4GB de memoria y 2x72GB de disco. Servidores de Aplicaciones: Procesador Dual - Core 4GB de memoria y 2x72GB de disco.

Requerimientos de software

CAPÍTULO 2. CARACTERÍSTICAS DEL SISTEMA

El sistema debe correr en sistemas operativos Windows, Unix y Linux, utilizando la plataforma JAVA (Java Virtual Machine, JBoss AS y PostgreSQL). El sistema deberá disponer de un navegador web, estos pueden ser IE 7, Opera 9, Google chrome 1 y Firefox 2 o versiones superiores de estos.

En este capítulo se realizó la descripción y modelación de los procesos del negocio asociados al campo de acción, obteniendo una perspectiva del sistema que se desea desarrollar a partir de los requerimientos funcionales y no funcionales, los cuales definen las capacidades y cualidades que la aplicación debe tener. Con la culminación de este capítulo se sentaron las bases para el desarrollo de las disciplinas de Diseño e Implementación del Componente Web Zoonosis.

Capítulo 3. Diseño del sistema

El propósito del presente capítulo es definir la estructura y elementos del diseño, describir requerimientos funcionales en términos de clases y sus objetos, representándolos gráficamente en Diagramas de Clases, así como fundamentar los patrones de diseño empleados.

3.1 Modelo de diseño

El Modelo de diseño es un modelo de objetos centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además, sirve de abstracción de la implementación del sistema y de ese modo es utilizado como una entrada fundamental de las actividades asociadas al flujo de trabajo de implementación. La utilización de otro subsistema es una forma de organización de este artefacto, en porciones más manejables. Los subsistemas y clases del diseño representan abstracciones del subsistema y componentes de la implementación del sistema. Estas abstracciones son directas y representan una sencilla correspondencia entre el diseño y la implementación. (36)

3.1.1 Definición de elementos de diseño

En el Modelo de diseño se establece la realización de las funcionalidades en clases teniendo en cuenta elementos como el lenguaje de programación, la arquitectura definida, entre otras. Está compuesto por los diagramas de clases del diseño así como los diagramas de interacción que pueden ser de secuencia o colaboración. Se definió la realización de un diagrama de clases por cada funcionalidad y uno por cada gestionar para el caso de los codificadores. El diagrama de clases responderá a la arquitectura definida que tiene como característica principal el uso del patrón Modelo Vista Controlador (MVC).

Todo el sistema está basado en el patrón Model 2, que no es más que el patrón MVC definido para las aplicaciones Java EE y que tiene como característica que usa un único servlet como controlador para toda la aplicación, atendiendo al patrón Front Controller (controlador frontal o fachada). El poder centralizar en un solo punto servicios como la gestión de conexiones a base de datos, comprobaciones de seguridad o gestión de errores favorecen que la aplicación sea mucho más robusta y aísla de todos estos aspectos al resto de los componentes.

Como resultado de la utilización del patrón MVC y debido a que la solución es una aplicación web para la realización de los diferentes diagramas se definió el uso de las facilidades de extensión brindadas por el

UML específicamente: estereotipos web.

Los estereotipos web como se les conocen hoy en día son resultado de la extensión propuesta por Jim Conallen en 1998 al UML para el diseño aplicaciones Web, siendo los elementos más significativos los estereotipos: Server Page, Client Page, y Form. El diseño de la solución será modelado con dichos estereotipos, por lo que a continuación se brinda una explicación breve de cada uno de ellos:

Server Page  **:** Representa la clase que tiene código que se ejecuta en el servidor, la cual se encarga de construir o generar el resultado HTML.

Client Page  **:** Es una página Web con formato XHTML. Son interpretadas por el navegador. Cada página cliente es construida por una sola página de servidor.

HTML Form  **:** Es una colección de elementos de entrada que están contenidos en la página cliente. Sus atributos son los elementos de entrada del formulario (input, textareas, radiobuttons, checkboxes, hiddenfields, entre otros).

Es importante destacar que una página cliente es construida por una sola página servidora; esta a su vez, puede completar su funcionamiento incluyendo algunas de las librerías que componen el framework JSF.

3.2 Patrones de diseño

Un patrón es una solución recurrente para un problema en un contexto o la respuesta al problema dentro de un contexto que ayuda a resolver las dificultades. Los patrones de diseño son directrices y principios estructurados que describen un problema común y entregan una buena solución ya probada a la que se le da un nombre. La utilización de los mismos en la concepción del componente ayudará a diseñar correctamente en menos tiempo, a construir clases reutilizables, facilitará la documentación y conducirá a la definición de una arquitectura pequeña, simple y comprensible.

Para que una solución sea considerada un patrón debe poseer ciertas características:

- ✓ Debe haber comprobado su efectividad al resolver problemas similares en ocasiones anteriores.

- ✓ Debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

Los patrones de diseño tienen como ventajas que:

- ✓ Proponen una forma de reutilizar la experiencia de los desarrolladores.
- ✓ Están basados en la recopilación del conocimiento de los expertos en desarrollo de software.
- ✓ Es una experiencia real, probada y funciona. Es historia y ayuda a no cometer los mismos errores.

Entre los que más se utilizan se encuentran los patrones **GRASP** que son los patrones para la asignación de responsabilidades, donde se destacan por su uso en el diseño:

- ✓ **Experto:** Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.
- ✓ **Alta cohesión:** La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Este patrón expresa que la información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible relacionada con la clase.
- ✓ **Bajo acoplamiento:** Es la idea de tener las clases lo menos ligadas entre sí posible, de tal forma que en caso de producirse una modificación en alguna de ellas, tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre clases.
- ✓ **Creador:** Este patrón como su nombre lo indica es el que crea, el que guía la asignación de responsabilidades relacionadas con la creación de objetos. Se asigna la responsabilidad de que una clase B cree un objeto de la clase A solamente cuando:
    - ✓ B contiene a A.
    - ✓ B es una agregación (o composición) de A.
    - ✓ B almacena a A.
    - ✓ B tiene los datos de inicialización de A (datos que requiere su constructor).
    - ✓ B usa a A.

- ✓ **Controlador:** Asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento.

Patrones Del "Gang of Four"

Patrones GOF como:

- **De creación:** Abstraen el proceso de creación de instancias.
- **Estructurales:** Se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño.
- **De comportamiento:** Atañen a los algoritmos y a la asignación de responsabilidades entre objetos.

En la elaboración del presente modelo de diseño y con el objetivo de perfeccionar la calidad de los diagramas correspondientes a esta disciplina fueron aplicados los patrones mencionados anteriormente.

3.3 Diagramas de clases del diseño

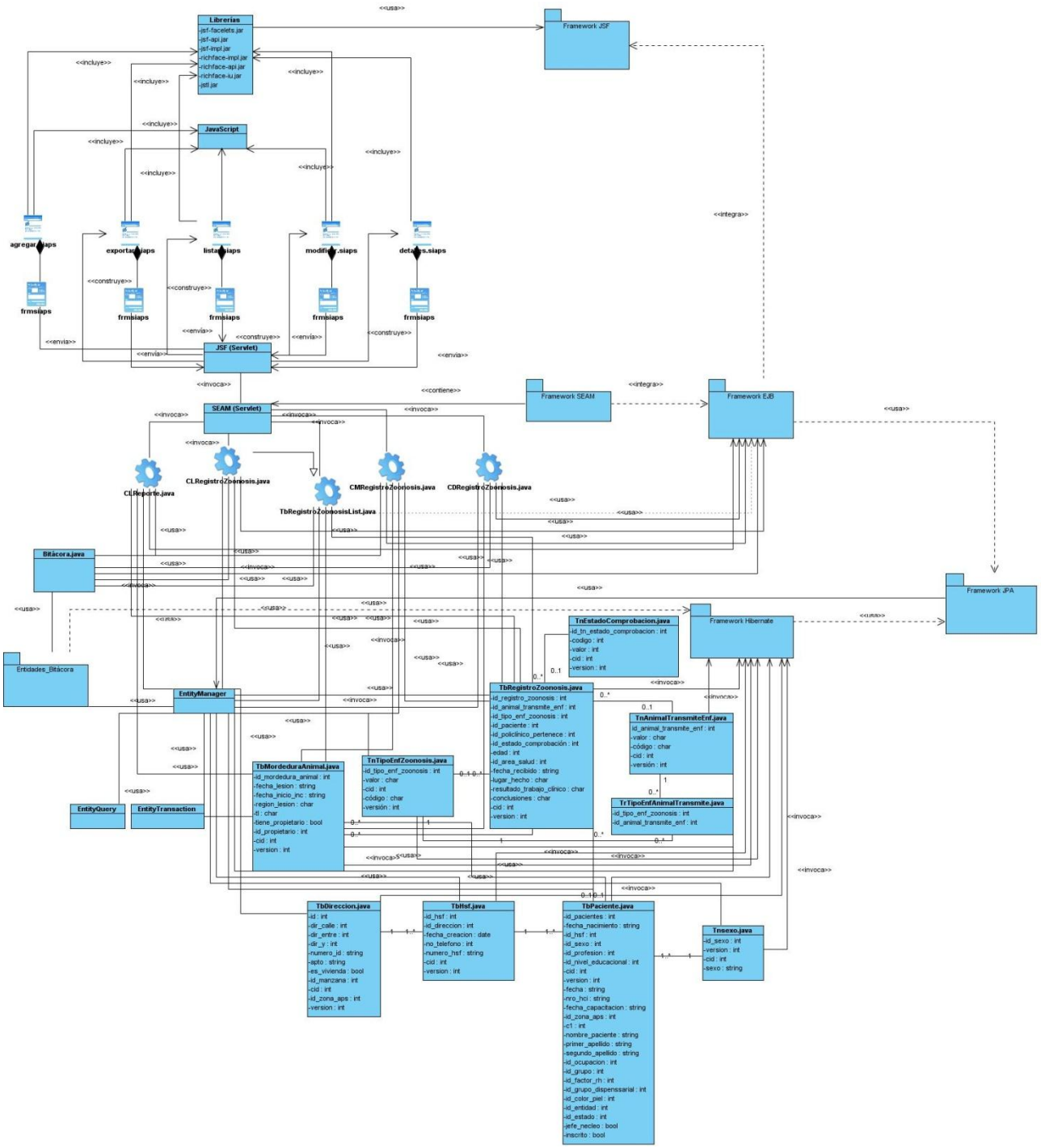


Figura 3. Diagrama de clases del diseño. Gestionar Registro de Zoonosis.

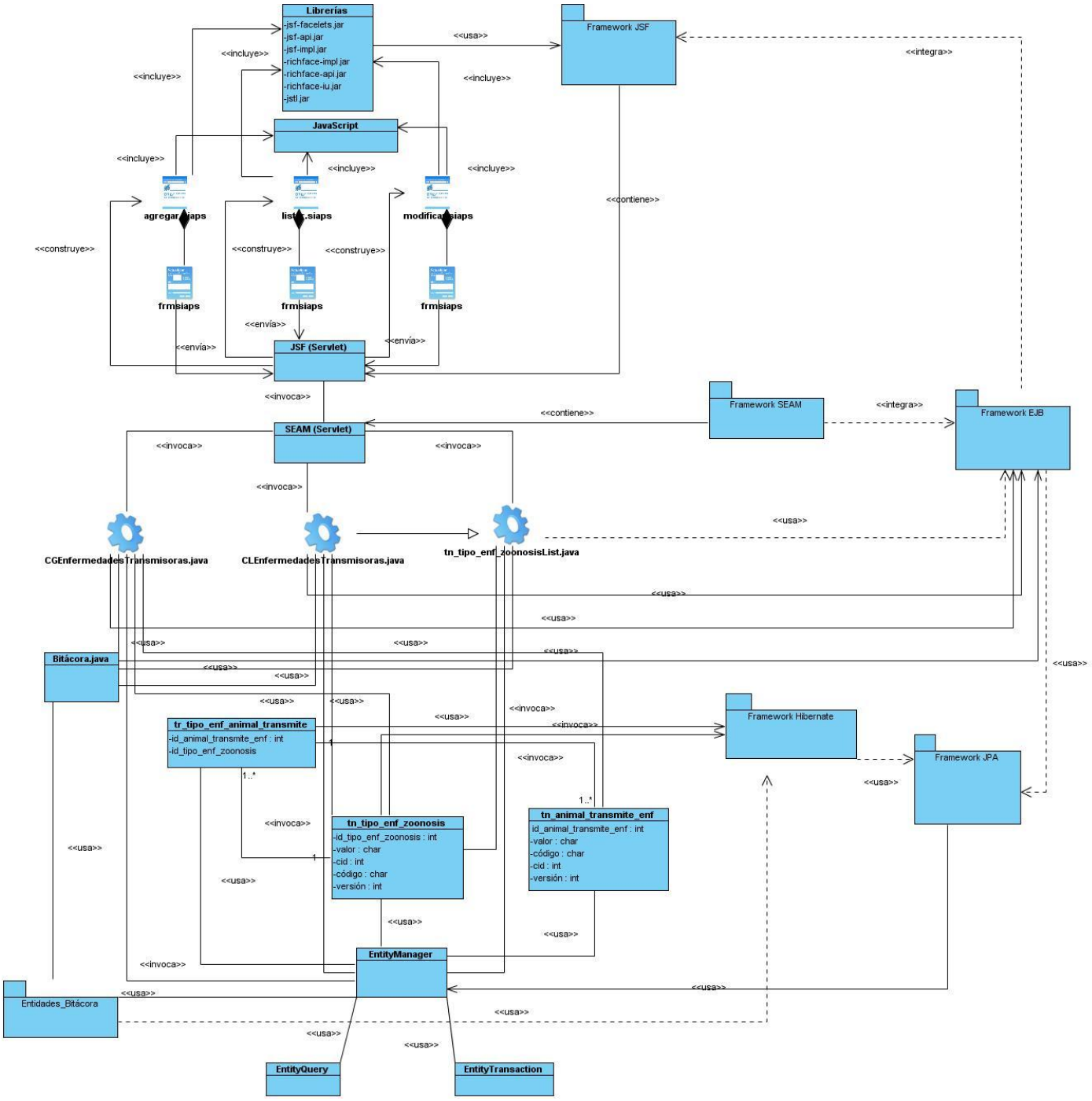


Figura 4. Diagrama de clases del diseño. Gestionar Tipos de Enfermedades Zoonósicas.

3.4 Descripción de las clases del diseño

3.4.1 Descripciones textuales

3.4.1.1 Clases comunes. Descripción

Capa de presentación

Nombre:	**Propósito:**
	Proveer el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas.

Figura 5. Clase JavaScript

Descripción:

Permite realizar las validaciones del lado del cliente y construir páginas más dinámicas integradas a un navegador web

Nombre:	**Propósito:**
	Permite la creación de interfaces.

Figura 6. Clase Librerías

Descripción:

Contiene un conjunto de librerías que permiten construir una interfaz de usuario con componentes reutilizables y extensibles. Reducen significativamente la carga de construir y mantener aplicaciones web con componentes de interfaz del lado del servidor.

Nombre:	**Propósito:**
	Interceptar las peticiones a las páginas JSF.

Figura 7. Clase JSF Servlet

Descripción:

Es el controlador de JSF que intercepta las peticiones de las páginas clientes, asociándoles a estas páginas, clases Java que recogen la información introducida y que disponen de métodos que responden a las acciones del usuario. Además, prepara el contexto JSF antes de enrutar a las páginas correspondientes e interviene en la construcción de la respuesta para generar la vista, luego de ser invocada una petición.

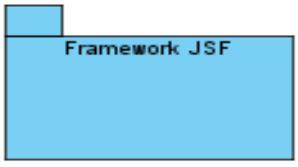
Nombre:	**Propósito:**
	Simplificar el desarrollo de las interfaces de usuario en aplicaciones java basadas en el patrón Modelo-Vista-Controlador.

Figura 8. Paquete Framework JSF

Descripción:

La tecnología Java Server Faces es un framework de los componentes de la interfaz de usuario y es válido para todas aquellas aplicaciones web basadas en la tecnología Java, está basado en el patrón MVC. Forma parte de la especificación JEE 5.

Capa de Negocio

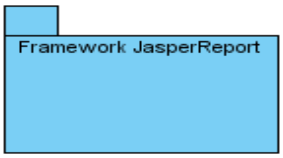
Nombre:	Propósito:
	Añadir características de generación de reportes a aplicaciones Java.

Figura 10. Paquete Framework JasperReports.

Descripción:

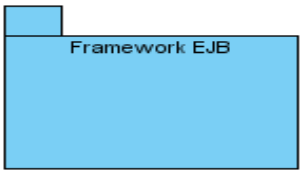
Es una librería de clases de Java de código abierto desarrollada para facilitar el agregar capacidades de reporte a las aplicaciones Java. Permite realizar reportes de código abierto que tiene como función el llevar documentos ricos en contenido a la pantalla, a la impresora, o a archivos PDF, HTML, XLS, CSV y XML.

Nombre:	Propósito:
	Proveer las funcionalidades para realizar las auditorias del sistema.

Figura 11. Clase Bitácora.java

Descripción:

Permite realizar las funciones para almacenar datos como la fecha, hora, usuario, contraseña, entre otros, del usuario que inicia y finaliza la sesión. Además, datos como el modulo y funcionalidad accedidos así como las que acciones ejecutadas.

Nombre:	**Propósito:**
 Figura 12. Paquete Framework EJB	Encapsular la lógica de negocio que cumplimenta el propósito de la aplicación. **Observaciones:** El Framework EJB está incluido en las capas de Negocio y Persistencia.

Descripción:

Es una plataforma para construir aplicaciones de negocio portables, escalables, y reutilizables utilizando el lenguaje de programación Java. El objetivo de Enterprise JavaBeans (EJB) 3.0 es simplificar el desarrollo de aplicaciones Java y estandarizar el API de persistencia para la plataforma Java. Forma parte de la especificación JEE 5.

Capa de Datos

Nombre:	**Propósito:**
 Figura 13. Clase EntityManager	Gestionar las entidades proveyendo servicios de persistencia.

Descripción:

Permite realizar las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) que impliquen entidades.

Nombre:	Propósito:
	Agregar consultas que pueden aplicarse a las entidades del modelo.

Figura 14. Clase EntityManager

Descripción:

Permite encontrar objetos persistentes manejando cierto criterio de búsqueda. Permite realizar peticiones a la base de datos y controla cómo se ejecuta dicha petición. Se utiliza para enlazar los parámetros de la petición, limitar el número de resultados devueltos por la petición y para ejecutar dicha petición.

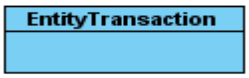
Nombre:	Propósito:
	Agrupar las operaciones sobre datos persistentes en una unidad transaccional.

Figura 15. Clase EntityManager

Descripción:

Permite realizar operaciones sobre datos persistentes de manera que agrupados formen una unidad de trabajo transaccional, en el que todo el grupo sincroniza su estado de persistencia en la base de datos o todos fallan en el intento, en caso de fallo, la base de datos quedará con su estado original. Maneja el concepto de todos o ninguno para mantener la integridad de los datos.

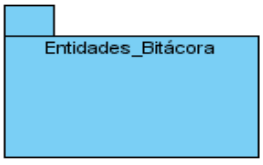
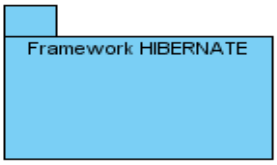
Nombre:	Propósito:
	Agrupar las entidades que contienen la información de las auditorías del sistema.

Figura 16. Clase Entidades Bitácora

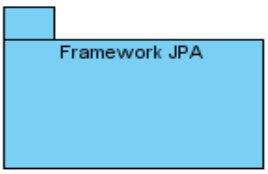
Descripción:

Contiene el conjunto de entidades que poseen la información de los usuarios y sus trazas en cuanto a sesión utilizada, módulos accedidos, funcionalidades permitidas, acciones realizadas y atributos modificados. Son utilizadas por la clase Bitácora.java para realizar las auditorías del sistema.

Nombre:	**Propósito:**
 Figura 17. Paquete Framework HIBERNATE	Proveer el mapeo objeto/relacional con la base de datos.

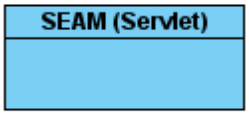
Descripción:

Conjunto de clases agrupadas en componentes que constituyen una herramienta de Mapeo objeto/relacional ó ORM de código abierto (Object Relational Mapping) y un generador de sentencias SQL. Permite diseñar objetos persistentes que podrán incluir polimorfismo, relaciones, colecciones, y un gran número de tipos de datos. De una manera muy rápida y optimizada permite generar Bases de Datos en cualquiera de los entornos soportados: Oracle, PostgresSQL, DB2, MySQL, entre otras.

Nombre:	**Propósito:**
 Figura 18. Paquete Framework JPA	Unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional.

Descripción:

Conjunto de clase agrupadas en componentes que constituyen la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB 3.0 como parte de JSR 220, aunque su uso no se limita a los componentes software EJB. Permite unificar la manera en que funcionan las utilidades que proveen un mapeo objeto-relacional. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos.

Nombre:	**Propósito:**
 Figura 19. Clase SEAM Servlet	Proveer la interacción de la capa de presentación con la de negocio. **Observaciones:** No se encuentra en ninguna capa en específico, sino que se encuentra ubicado entre la capa de Presentación y la de Negocio.

Descripción:

Es el controlador de SEAM que capta las peticiones derivadas de la interacción del usuario después de interactuar con el Servlet de JSF. Enrutar las peticiones hacia los Beans que posibilitarán darle respuesta a la petición solicitada. Interviene en la integración de las capas de presentación y negocio.

3.1.1.1 Gestionar Registro de Zoonosis . Descripción.

Capa de Presentación

Nombre:	**Propósito:**
 Figura 20. Clase Agregar.siaps	Proveer la interacción con el usuario.

Descripción:

La clase Agregar.siaps es una página web que se ejecuta del lado del cliente sobre un navegador. Permitirá insertar todos los datos necesarios para poder llenar el registro de zoonosis y que todo quede correctamente registrado. Posee un conjunto de validaciones en JavaScript que permite no realizar peticiones innecesarias y por lo tanto se incrementa su usabilidad. Utiliza diferentes librerías basadas en el Framework JSF.

Nombre:	**Propósito:**
	Enviar los datos a las páginas servidoras.

Figura 21. frmsiaps

Descripción:

La clase frmsiaps contiene una colección de elementos de entrada que están contenidos en la página cliente para llenar el registro de zoonosis y que todo quede correctamente registrado. Sus atributos son los elementos de entrada del formulario (inputboxes, textareas, checkboxes, entre otros). No tienen operaciones, el método para el paso de los parámetros es $_POST.

Nombre:	**Propósito:**
	Proveer la interacción con el usuario.

Figura 22. Clase Listado.siaps

Descripción:

La clase Listado.siaps es una página web que se ejecuta del lado del cliente sobre un navegador. Permitirá buscar cualquier registro de zoonosis. Posee un conjunto de validaciones en JavaScript que permite no realizar peticiones innecesarias y por lo tanto se incrementa su usabilidad. Utiliza diferentes librerías basadas en el Framework JSF.

Nombre:	Propósito:
	Enviar los datos a las páginas servidoras.

Figura 23. frmsiaps

Descripción:

La clase frmsiaps contiene una colección de elementos de entrada que están contenidos en la página cliente para listar todos los datos necesarios de los registros zoonosis. Sus atributos son los elementos de entrada del formulario (inputboxes, textareas, checkboxes, entre otros). No tienen operaciones, el método para el paso de los parámetros es $_POST.

Nombre:	Propósito:
	Proveer la interacción con el usuario.

Figura 24. Clase Modificar.siaps

Descripción:

La clase Modificar.siaps es una página web que se ejecuta del lado del cliente sobre un navegador. Permitirá acceder a modificar cualquier registro hecho con anterioridad. Posee un conjunto de validaciones en JavaScript que permite no realizar peticiones innecesarias y por lo tanto se incrementa su usabilidad. Utiliza diferentes librerías basadas en el Framework JSF.

Nombre:	Propósito:
	Enviar los datos a las páginas servidoras.

Figura 25. frmsiaps

Descripción:

La clase frmsiaps contiene una colección de elementos de entrada que están contenidos en la página cliente para modificar todos los datos necesarios de cualquier registro hecho con anterioridad. Sus atributos son los elementos de entrada del formulario (inputboxes, textareas, checkboxes, entre otros). No tienen operaciones, el método para el paso de los parámetros es $_POST.

Nombre:	**Propósito:**
	Proveer la interacción con el usuario.

Figura 26. Clase Detalles.siaps

Descripción:

La clase Detalles.siaps es una página web que se ejecuta del lado del cliente sobre un navegador. Permitirá desde la cual podemos ver los detalles de cualquier registro escogido previamente, en la cual se mostrará una serie de datos que no se muestran en el registro. Posee un conjunto de validaciones en JavaScript que permite no realizar peticiones innecesarias y, por lo tanto, se incrementa su usabilidad. Utiliza diferentes librerías basadas en el Framework JSF.

Nombre:	**Propósito:**
	Enviar los datos a las páginas servidoras.

Figura 27. frmsiaps

Descripción:

La clase frmsiaps contiene una colección de elementos de entrada que están contenidos en la página cliente para mostrar todos los datos necesarios de un registro de zoonosis. Sus atributos son los elementos de entrada del formulario (inputboxes, textareas, checkboxes, entre otros). No tienen operaciones, el método para el paso de los parámetros es $_POST.

Nombre:	Propósito:
	Proveer la interacción con el usuario.

Figura 28. Clase Exportar.siaps

Descripción:

La clase Exportar.siaps es una página web que se ejecuta del lado del cliente sobre un navegador. Permitirá exportar los datos del registro de Zoonosis, además te brinda la opción de exportar tanto en formato WORD, como en PDF. Posee un conjunto de validaciones en JavaScript que permite no realizar peticiones innecesarias y, por lo tanto, se incrementa su usabilidad. Utiliza diferentes librerías basadas en el Framework JSF.

Nombre:	Propósito:
	Enviar los datos a las páginas servidoras.

Figura 29. frmsiaps

Descripción:

La clase frmsiaps contiene una colección de elementos de entrada que están contenidos en la página cliente para exportar todos los datos del registro de zoonosis. Sus atributos son los elementos de entrada del formulario (input boxes, text areas, check boxes, entre otros). No tienen operaciones, el método para el paso de los parámetros es $_POST.

Capa de Negocio

Nombre:	**Propósito:**
 Figura 30. Clase CCRegistroZoonosis.java	Proveer una respuesta a las peticiones realizadas en la vista.

Descripción:

La clase CCRegistroZoonosis.java es una clase que se ejecuta del lado del servidor. Permite darle respuesta a las peticiones que se desencadenan en la vista a través de los métodos que contienen. Se encarga de gestionar la acción de insertar los datos sobre la página cliente correspondiente. Hace uno del Framework EJB que encapsula la lógica de negocio, integrándose con la vista a través del Framework SEAM.

Nombre:	**Propósito:**
 Figura 31. Clase TbRegistroZoonosisList.java	Proveer una respuesta a las peticiones realizadas en la vista.

Descripción:

La clase TbRegistroZoonosisList.java es una clase que se genera de forma automática el realizar la ingeniería inversa y que se ejecuta del lado del servidor. Permite darle respuesta a las peticiones que se desencadenan en la vista a través de los métodos que contienen. Se encarga de gestionar la acción de listar los datos solicitados sobre la página cliente correspondiente. Hace uno del Framework EJB que

encapsula la lógica de negocio, integrándose con la vista a través del Framework SEAM.

Nombre:	**Propósito:**
 Figura 32. Clase CLRegistroZoonosis.java	Proveer una respuesta a las peticiones realizadas en la vista.

Descripción:

La clase CLRegistroZoonosis.java es una clase que se ejecuta del lado del servidor. Hereda todas las funcionalidades de la clase autogenerada TbRegistroVacunaList.java y permite guardar los cambios originales de la misma evitando que los mismos se pierdan al realizar la ingeniería inversa. Hace uno del Framework EJB que encapsula la lógica de negocio, integrándose con la vista a través del Framework SEAM.

Nombre:	**Propósito:**
 Figura 33. Clase CMRegistroZoonosis.java	Proveer una respuesta a las peticiones realizadas en la vista.

Descripción:

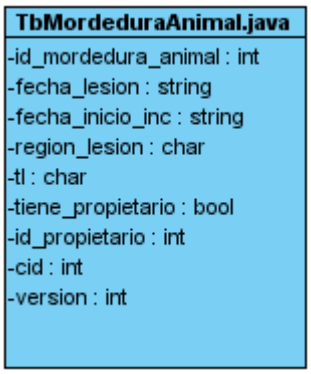
La clase CMRegistroZoonosis.java es una clase que se ejecuta del lado del servidor. Permite darle respuesta a las peticiones que se desencadenan en la vista a través de los métodos que contienen. Se encarga de gestionar la acción de modificar los datos de cualquier registro hecho con anterioridad que se solicite sobre la página cliente correspondiente. Hace uno del Framework EJB que encapsula la lógica de negocio, integrándose con la vista a través del Framework SEAM.

Nombre:	Propósito:
 Figura 35. Clase CDRegistroZoonosis.java	Proveer una respuesta a las peticiones realizadas en la vista.

Descripción:

La clase CDRegistroZoonosis.java es una clase que se ejecuta del lado del servidor. Permite darle respuesta a las peticiones que se desencadenan en la vista a través de los métodos que contienen. Se encarga de gestionar la acción de mostrar los datos de cualquier registro de zoonosis, que es solicitado desde la página cliente correspondiente. Hace uno del Framework EJB que encapsula la lógica de negocio, integrándose con la vista a través del Framework SEAM.

Capa de Datos

Nombre:	Propósito:
 Figura 39. Clase TbMordeduraAnimal.java	Proveer el mapeo con la base de datos.

Descripción:

La clase TbMordeduraAnimal.java es una clase que se ejecuta del lado del servidor. En ella se almacenan los datos de la mordedura de animal. Representa una tabla en el modelo de datos relacional y cada instancia de esta entidad corresponde a un registro en esa tabla. Es persistida por las clases servidoras para darle una respuesta a las páginas clientes. Hace uso del Framework Hibernate y JPA.

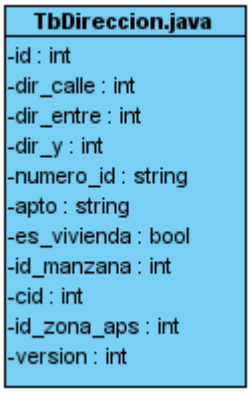
Nombre:	**Propósito:**
	Proveer el mapeo con la base de datos.

Figura 40. Clase TbDireccion.java

Descripción:

La clase TbDireccion.java es una clase que se ejecuta del lado del servidor. En ella se almacenan la dirección del paciente. Representa una tabla en el modelo de datos relacional y cada instancia de esta entidad corresponde a un registro en esa tabla. Es persistida por las clases servidoras para darle una respuesta a las páginas clientes. Hace uso del Framework Hibernate y JPA.

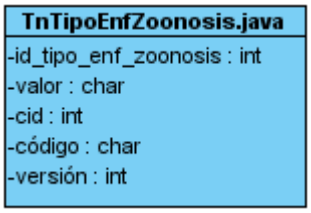
Nombre:	**Propósito:**
	Proveer el mapeo con la base de datos.

Figura 41. Clase TnTipoEnfZoonosis.java

Descripción:

La clase TnTipoEnfZoonosis.java es una clase que se ejecuta del lado del servidor. En ella se almacenan los datos de las distintas enfermedades zoonósicas. Representa una tabla en el modelo de datos relacional y cada instancia de esta entidad corresponde a un registro en esa tabla. Es persistida por las clases servidoras para darle una respuesta a las páginas clientes. Hace uso del Framework Hibernate y JPA.

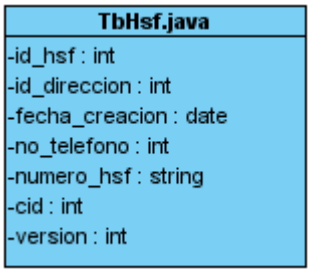
Nombre:	**Propósito:**
	Proveer el mapeo con la base de datos.

Figura 42. Clase TbHsf.java

Descripción:

La clase TbHsf.java es una clase que se ejecuta del lado del servidor. En ella se almacenan los datos de la historia de salud familiar del paciente. Representa una tabla en el modelo de datos relacional y cada

instancia de esta entidad corresponde a un registro en esa tabla. Es persistida por las clases servidoras para darle una respuesta a las páginas clientes. Hace uso del Framework Hibernate y JPA.

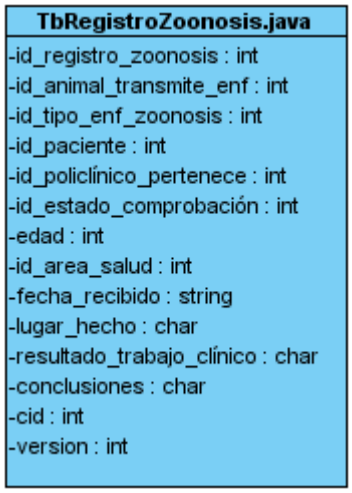
Nombre:	**Propósito:**
	Proveer el mapeo con la base de datos.

Figura 43. Clase TbRegistroZoonsosis.java

Descripción:

La clase TbRegistroZoonsosis.java es una clase que se ejecuta del lado del servidor. En ella se almacenan los datos del registro de zoonosis. Representa una tabla en el modelo de datos relacional y cada instancia de esta entidad corresponde a un registro en esa tabla. Es persistida por las clases servidoras para darle una respuesta a las páginas clientes. Hace uso del Framework Hibernate y JPA.

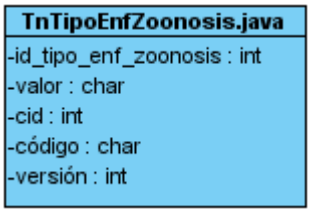
Nombre:	**Propósito:**
	Proveer el mapeo con la base de datos.

Figura 44. Clase TrTipoEnfZoonosis.java

Descripción:

La clase TrTipoEnfZoonosis.java es una clase que se ejecuta del lado del servidor. En ella se almacenan la relación de las enfermedades y los animales que la transmiten. Representa una tabla en el modelo de datos relacional y cada instancia de esta entidad corresponde a un registro en esa tabla. Es persistida por las clases servidoras para darle una respuesta a las páginas clientes. Hace uso del Framework Hibernate y JPA.

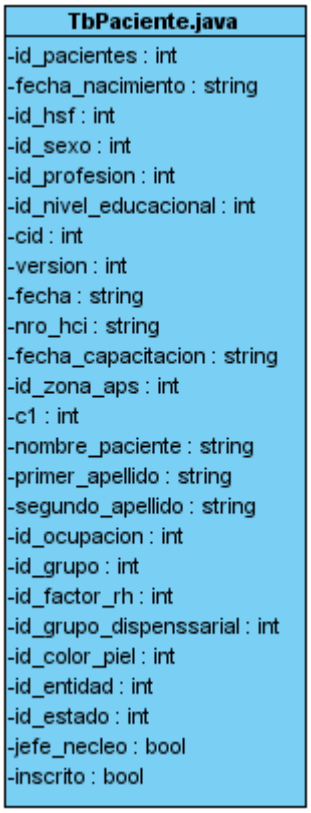
Nombre:	**Propósito:**
	Proveer el mapeo con la base de datos.

Figura 45. Clase TbPaciente.java

Descripción:

La clase TbPaciente.java es una clase que se ejecuta del lado del servidor. En ella se almacenan los datos de los pacientes. Representa una tabla en el modelo de datos relacional y cada instancia de esta entidad corresponde a un registro en esa tabla. Es persistida por las clases servidoras para darle una respuesta a las páginas clientes. Hace uso del Framework Hibernate y JPA.

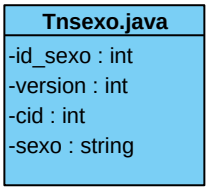
Nombre:	**Propósito:**
	Proveer el mapeo con la base de datos.

Figura 46. Clase TnSexo.java

Descripción:

La clase TrSexo.java es una clase que se ejecuta del lado del servidor. En ella se el sexo del paciente. Representa una tabla en el modelo de datos relacional y cada instancia de esta entidad corresponde a un registro en esa tabla. Es persistida por las clases servidoras para darle una respuesta a las páginas clientes. Hace uso del Framework Hibernate y JPA.

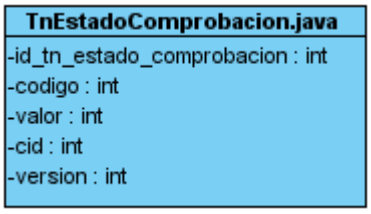
Nombre:	**Propósito:**
	Proveer el mapeo con la base de datos.

Figura 47. Clase TnEstadoComprobacion.java

Descripción:

La clase TnEstadoComprobacion.java es una clase que se ejecuta del lado del servidor. Representa una tabla en el modelo de datos relacional y cada instancia de esta entidad corresponde a un registro en esa tabla. Es persistida por las clases servidoras para darle una respuesta a las páginas clientes. Hace uso del Framework Hibernate y JPA.

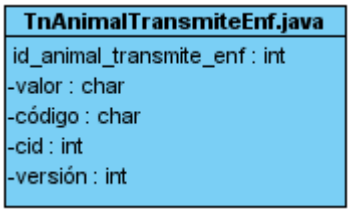
Nombre:	**Propósito:**
	Proveer el mapeo con la base de datos.

Figura 48. Clase TnAnimalTransmiteEnf.java

Descripción:

La clase TnAnimalTransmiteEnf.java es una clase que se ejecuta del lado del servidor. Representa una tabla en el modelo de datos relacional y cada instancia de esta entidad corresponde a un registro en esa tabla. Es persistida por las clases servidoras para darle una respuesta a las páginas clientes. Hace uso del Framework Hibernate y JPA.

3.5 Modelo de datos

El modelo de datos es uno de los artefactos más importante dentro del diseño. Este modelo proporciona una representación visual y física de los datos persistentes del sistema, que en el futuro serán la base de datos. Se obtiene a partir del diagrama de clases persistentes y su forma se expresa mediante un diagrama de UML, siendo sus elementos esenciales las entidades, atributos y relaciones entre entidades. Para ver el Modelo de Datos del sistema remitirse al expediente de proyecto.

3.5.1 Descripción de tablas y atributos

Describir las tablas y atributos que conforman el modelo de datos constituye un elemento esencial. El objetivo principal es hacer comprender con claridad cada uno de los elementos (entidades, atributos y relaciones) que conforman el modelo. En el expediente de proyecto podrá encontrar toda la información bien detallada sobre el modelo de datos, así como las descripciones realizadas.

Al concluir el desarrollo de este capítulo se han obtenido los artefactos del flujo de Análisis y Diseño, necesario para el desarrollo del sistema y que constituyen entradas imprescindibles en el flujo de implementación, como es el caso del modelo de diseño, con los diagramas de clases del diseño, la descripción de los patrones utilizados y el modelo lógico de datos, así como la descripción de cada una de las clases del diseño, tablas y atributos de la base de datos con el objetivo de entender mejor el sistema. A partir de este punto, se puede comenzar a construir la aplicación, teniendo en cuenta todos los requerimientos y funcionalidades derivados de esta etapa.

Capítulo 4. Implementación

La implementación es el centro en la fase de construcción, aunque se lleva a cabo en iteraciones de otras fases, durante la de elaboración crea la línea base de la arquitectura y durante la de transición trata defectos tardíos encontrados en algunas distribuciones del sistema. La implementación actual del sistema se denota a través del modelo de implementación en términos de componentes y subsistemas.

Este capítulo constituye la continuidad al Modelo de diseño, en él se presenta el Diagrama de despliegue de la solución propuesta, se describen los métodos más importantes, los estándares de codificación y el tratamiento de errores en la aplicación.

4.1 Justificación de Integración con otros sistemas

En la actualidad ningún sistema se concibe de forma aislada, por lo que se hace necesaria su integración con otros sistemas en conjunto. Es por esto que el Centro de Informática Médica (CECIM), plantea como solución el sistema AlasSIAPS, como plataforma única para la gestión, procesamiento y transmisión de la información clínica en el SNS, logrando una comunicación a nivel de Bases de Datos.

El componente Web Zoonosis no está exento de esto. El mismo se integra al módulo de Higiene y Epidemiología, quien se nutre de los restantes módulos del sistema: el de Configuración proporciona la seguridad de la aplicación, brinda la información asociada a los departamentos, los servicios médicos y a las entidades de salud. El de Medicina Familiar es el encargado de gestionar los datos del paciente que se utilizan en las funcionalidades involucradas.

4.2 Diagrama de despliegue

El modelo de despliegue describe la distribución física del sistema, muestra cómo están distribuidos los componentes de software entre los distintos nodos de cómputo. Este diagrama está compuesto por protocolos, dispositivos y procesadores, los nodos representan a estos dos últimos, además, poseen relaciones que representan medios de comunicación entre ellos. Permite comprender la correspondencia entre la arquitectura de software y la arquitectura de hardware.

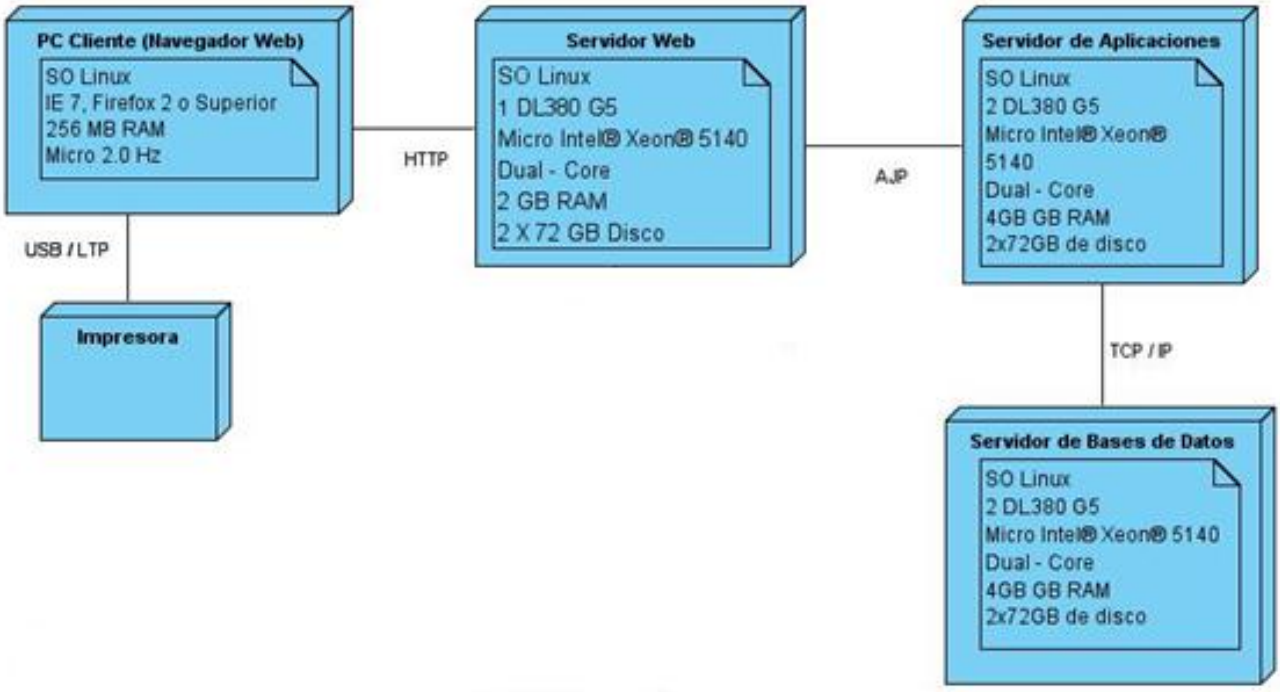


Figura 49. Diagrama de despliegue

4.2.1 Descripción de los elementos

Dispositivos:

Servidor Web: Servidor Web Apache que provee el servicio de interfaz al usuario final mediante un portal convencional y otro WAP para dispositivos móviles, pues es el ordenador que estará como fachada a Internet, al mismo tiempo será el puente o proxy para entrar al clúster de servidor (res) de aplicaciones que proporciona Jboss. En escenarios híbridos brindará la ejecución y actualización de la solución local mediante la tecnología Java Web Start.

Servidor(es) de Aplicaciones: Servidor de Aplicaciones Jboss certificado por SUN para el estándar JEE5, hospedará la solución integrada, proveerá de un clúster para balancear la carga de peticiones hechas por los usuarios garantizando de esta manera disponibilidad de la información mostrada.

Servidor de Bases de Datos: Servidor de Datos PostgreSQL, en el cual residirá toda la información.

Servidor de almacén de datos: Servidor de Datos PostgreSQL, en el cual residirá la información como un histórico manejable desde diferentes aristas como un cubo, de manera que sobre él se pueden realizar estadísticas complejas, predicciones y establecer comportamiento en indicadores.

Servidor de enlace (Bridge): Servidor de Cola de mensajes básicamente ActiveMQ, el cual enrutará mensajes estandarizados, hacia el subsistema local u otro sistema dentro del mismo sector de la salud.

PC cliente: Serán las estaciones de los usuarios, las cuales servirán para acceder al sistema web o integrado mediante el navegador o al sistema local o de escritorio mediante un ícono en el escritorio del ordenador.

4.3 Estrategias de codificación. Estándares y estilos a utilizar

Con el propósito de que exista homogeneidad entre las aplicaciones que se encuentran integradas al Sistema Integral para la Atención Primaria de Salud AlasSIAPS se han definido una serie de estándares tanto para el diseño como para la codificación, además, cómo se le dará el tratamiento de errores al módulo para que los códigos fuentes de las aplicaciones y los mensajes que se emitan mantengan una uniformidad.

4.3.1 Estándares de diseño

Idioma

- Se debe utilizar como idioma el español y las palabras no se acentuarán.

Identación

- Lograr una estructura uniforme para los bloques de código así como para los diferentes niveles de anidamiento.

Inicio y fin de bloque

- Se recomienda dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}. Lo mismo sucede para el caso de las instrucciones if, else, for, while, do while, switch, foreach.

Comentarios, separadores, líneas, espacios en blanco y márgenes

- Objetivo: Establecer un modo común para comentar el código, de forma tal que sea comprensible con sólo leerlo una vez.

Ubicación de comentarios

- Al inicio de cada clase o función y al final de cada bloque de código. Se recomienda comentar al inicio de la clase o función especificando el objetivo de la misma, así como los parámetros que usa (especificar tipos de dato, y objetivo del parámetro) entre otras cosas.

Líneas en blanco

- Se emplean antes y después de métodos, clases y estructuras. Se recomienda dejar una línea en blanco antes y después de la declaración de una clase o de una estructura y de la implementación de una función.

Espacios en blanco

- Entre operadores lógicos y aritméticos. Se recomienda usar espacios en blanco entre estos operadores para lograr una mayor legibilidad en el código.

Ejemplo: producto = nomproducto.

Apariencia de variables

- Las variables tendrán un prefijo para el tipo de datos en minúsculas. El nombre que se le da a las variables debe comenzar con la primera letra en minúscula, la cual identificará el tipo de datos al que se refiere, en caso de que sea un nombre compuesto se empleará notación CamellCasing**.

Ejemplo: sNombrePaciente.

Apariencia de constantes

- Todas sus letras en mayúsculas. Se deben declarar las constantes con todas sus letras en mayúsculas.

Clases y objetos

- Objetivo**:** Nombrar las clases e instancias de forma estándar para todas las aplicaciones.

Apariencia de clases y objetos

- Primera letra en mayúscula. Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúsculas, en caso de que sea un nombre compuesto se empleará

notación PascalCasing*. Ejemplo: MiClase (). Para el caso de las instancias se comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.

Apariencia de atributos

- Primera letra en minúscula. El nombre que se le da a los atributos de las clases debe comenzar con la primera letra en minúscula, la cual estará en correspondencia al tipo de dato al que se refiere, en caso de que sea un nombre compuesto se empleará notación CamellCasing**.

Bases de Datos, Tablas, esquemas y Campos

Apariencia de la Base de Datos

- Las 3 primeras letras corresponden al nombre del departamento, seguido de underscoard y a continuación el prefijo bd, luego el nombre completamente en minúsculas precedido por underscoard; en caso de que sea un nombre compuesto se empleará notación. Los nombres serán cortos y descriptivos.

Ejemplo: APS_bd_medio_diag.

Apariencia de las tablas

- Las 2 primeras letras representan el tipo. Todas las letras en minúsculas. El nombre a emplear para las tablas debe comenzar con el prefijo tb seguido de underscoard y luego debe escribirse todas las letras en minúsculas. En caso de que sea un nombre compuesto se utilizará underscoard para separarlo.

Ejemplo: “tb_inscripcion”.

Tablas que representen relaciones

- Las 2 primeras letras representan el tipo. Todas las letras en minúsculas. El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tr seguido de underscoard y el nombre será la concatenación del nombre de las dos tablas que la generaron separados por uderscoard todo en minúscula. Ejemplo: “tr_examen_grupo_intervalo”.

Tablas que representen nomencladores

- Las 2 primeras letras representan el tipo. Todas las letras en minúscula. El nombre a emplear para estas tablas de relación debe comenzar con el prefijo tn seguido de underscoard. El nombre será corto y descriptivo todo en minúscula. Ejemplo: “tn_muestra”.

Apariencia de los campos

- Todas las letras en minúscula. El nombre a emplear para los campos debe escribirse con todas las letras en minúscula para evitar problemas con el Case Sensitive del gestor. Ejemplo: “estado”.

Nombre de los campos

- En caso de identificadores. Todos los campos identificadores van a comenzar con el identificador id seguido de underscoard y posteriormente el nombre del campo. Ejemplo: id_muestra.

Sentencias SQL

- Todas las letras en mayúsculas. Las palabras correspondientes a las sentencias SQL y sus parámetros deben ir en mayúsculas.

4.4 Tratamiento de excepciones

Las excepciones son situaciones que pueden presentarse y que requieren un tratamiento especial. Dicho evento puede ser desde serios problemas de hardware, hasta los simples errores de programación y pueden ser tratados mediante una estructura de control que poseen los lenguajes de programación de alto nivel, diseñada para manejar condiciones anormales que pueden ser tratadas por el mismo programa que se desarrolla. A esta estructura de control se le conoce como tratamiento de excepciones.

Validar los datos es una tarea común que se produce a través de cualquier aplicación, desde la capa de presentación, a la capa de persistencia. A menudo la misma lógica de validación es implementada en cada capa, resultando lento y propenso a errores. Para evitar la duplicación de estas validaciones en cada capa, los desarrolladores a menudo utilizan los paquetes de validaciones lógicas en el modelo de dominio, llenando las clases de dominio con código de validación que es en realidad metadatos sobre la propia clase.

En el sistema propuesto se utilizan todas las facilidades que brinda la plataforma para el tratamiento de excepciones. Para cada fragmento de código donde se espere una situación anómala, se definen las

excepciones correspondientes para luego ser tratadas, evitando la interrupción del sistema. También se emplean un conjunto de tipos de excepciones predefinidas por los marcos de trabajos que se utilizan en el sistema.

El uso de diferentes tecnologías y la integración que existe entre ellas, permiten capturar y controlar posibles situaciones desde diferentes puntos de la aplicación. En las páginas clientes se cuenta con un conjunto de componentes denominados validadores, que permiten establecer tipos de datos y formatos controlando que el envío de los activos al servidor sean los esperados.

4.5 Seguridad

Para que cualquier sistema informático sea calificado como seguro debe contar con un correcto equilibrio entre las siguientes características:

Integridad

- La información sólo puede ser modificada por quien está autorizado y de manera controlada.

Confidencialidad

- La información sólo debe ser legible para los autorizados.

Disponibilidad

- La información debe estar disponible cuando se necesita.

Irrefutabilidad (No repudio)

- El uso y/o modificación de la información por parte de un usuario debe ser irrefutable, es decir, que el usuario no puede negar dicha acción.

En correspondencia con dichas características se plantean a continuación un conjunto de acciones que, llevadas a cabo, permitirá a los usuarios finales disfrutar de un software seguro:

- Se mantendrá seguridad y control a nivel de usuario (Ver figura 50), garantizando el acceso de los mismos sólo a los niveles establecidos de acuerdo a la función que realizan.
- Las contraseñas podrán cambiarse solo por el propio usuario o por el administrador del sistema.
- Se mantendrá un segundo nivel de seguridad a nivel de estaciones de trabajo, garantizando sólo la ejecución de las aplicaciones que hayan sido definidas para la estación en cuestión.

- Se registrarán todas las acciones que se realizan, llevando el control de las actividades de cada usuario en todo momento.
- Se establecerán mecanismos de control y verificación para los procesos susceptibles de fraude. Los mecanismos serán capaces de informar al personal autorizado sobre posibles irregularidades que den indicios sobre la introducción de información falseada.
- El sistema implementará un mecanismo de auditoría para el registro de todos los accesos efectuados por los usuarios, proporcionando un registro de actividades (log) de cada usuario en el sistema.
- Ninguna información que se haya ingresado en el sistema será eliminada físicamente de la BD, independientemente de que para el subsistema este elemento ya no exista.
- Las informaciones médicas relacionadas con los pacientes y que vayan a ser intercambiadas con otros policlínicos por la red pública, viajarán cifradas para evitar accesos o modificaciones no autorizadas.
- El sistema permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

Figura 50. Acceso al sistema

En el capítulo se ha concluido la fase de implementación. Se desarrolló un componente web con la totalidad de las funcionalidades previstas para el correcto funcionamiento de los procesos de zoonosis del módulo de Higiene y Epidemiología de la Atención Primaria de Salud. Se describen los elementos necesarios para que el sistema sea desplegado, así como los estándares de codificación usados durante la implementación del sistema y el manejo de excepciones en la aplicación.

Conclusiones

La realización del presente trabajo ha posibilitado cumplir con los objetivos propuestos, por lo que se pueden plantear las siguientes conclusiones:

- Los sistemas informáticos existentes en Cuba y el mundo para la gestión de la información relacionada con la zoonosis que fueron analizados, no satisfacen las necesidades actuales detectadas a partir del estudio de los procesos de negocio de la APS en Cuba.
- La adopción de las pautas de diseño de interfaz de usuario permitió que el nuevo proceso sea visualmente homogéneo a los existentes en el módulo dentro del SIAPS, obteniendo un componente web uniforme.
- Como resultado de las fases modelado del negocio, captura de requisitos, diseño e implementación, se obtuvieron los artefactos necesarios para el desarrollo del subsistema Web Zoonosis del SIAPS.
- Se desarrolló un componente web que facilita el proceso de gestión de la información en los departamentos de zoonosis de los Centros de Higiene y Epidemiología.

Recomendaciones

Las recomendaciones de la investigación están dirigidas a sugerir acciones para complementar el producto obtenido. Para un buen desempeño y puesta en marcha de la aplicación se hacen las siguientes recomendaciones:

- Incluir una funcionalidad que alerte al módulo de Enfermería para que realice el proceso de vacunación asociado a un caso de zoonosis.
- Incluir un sistema en las Áreas de Salud que automatice la gestión de la información de las enfermedades de declaración obligatoria (EDO), que llegan diariamente a los centros de Higiene y Epidemiología.
- Conformar y aplicar un plan de adiestramiento para el personal médico que utilizará el sistema desarrollado, con el objetivo de capacitarlo para el correcto uso de cada una de las opciones definidas.

Referencias Bibliográficas

1. RADA, GABRIEL: *Escuela de Medicina Ponticima Universidad Católica de Chile.* [En línea] http://escuela.med.puc.cl/recursos/recepidem/parEpidem1.html
2. MÁS LAGO, PEDRO**:** "La Higiene y Epidemiología en Cuba a los 50 años del triunfo de la Revolución", *Revista Cubana de Higiene y Epidemiología*. [En línea] 21 de mayo de 2009. http://scielo.sld.cu/scielo.php?pid=S1561-30032009000200001&script=sci_arttext
3. ROJAS OCHOA, F. Y MARTÍNEZ CALVO, S: 50 Aniversario de la revolución cubana. Programa de Higiene y Epidemiología. [En línea] http://revolucioncubana.cip.cu/logros/modelo-social-socialista/salud/programa-de-higiene-y-epidemiologia
4. INFOMED. PORTAL DE SALUD DE CUBA: "La práctica de la vigilancia en salud pública. Un nuevo enfoque en la República de Cuba". [En línea] http://www.sld.cu
5. ECURED. Ministerio de Salud Pública*.* [En línea] http://www.ecured.cu/index.php/MINSAP
6. Ídem referencia 4
7. LEMUS, ELIA ROSA, RADAMÉS BORROTO, EUGENIO Y ANEIROS-RIBA, R: *Atención Primaria de Salud, Medicina Familiar y Educación Médica,* La Paz, Biblioteca de Medicina, 1998, Vol. XXXIV.
8. PISCOYA ANGELES, PATRICIA: SlideShare. Niveles de Atención en Sal*ud*, 2008. [En línea]. http://www.slideshare.net/pathyp75/niveles-de-salud-presentation
9. Ídem referencia 8.
10. Ídem referencia 8.
11. BEJARANO CASTRO, MÓNICA Y OTROS: "*Caracterización de los pacientes con lesiones de causa externa mediante un sistema de vigilancia epidemiológica"*, Revista Colombiana de Cirugía. [En línea]. http://www.encolombia.com/medicina/cirugia/Cirug%C3%ADa213-06/Ciru21306caracteriza-ci%C3%B3n.htm
12. MACHADO ALBA, JORGE ENRIQUE**:** *Revista de Ciencias Humanas*, No. 26.
13. MAZZETTI SOLER, PILAR Y OTROS**:** *Marco Conceptual Metodológico para el Abordaje de Promoción de la Salud.* LIMA, Perú, [s.n.], 2005.

REFERENCIAS BIBLIOGRÁFICAS

14. PÉREZ, CARMELO LÓPEZ: *Entorno Visual de Aprendisaje. Modelo de Madurez de la Capacidad del Software1.* 29 de 12 de 2004. [En línea]. http://eva.uci.cu/file.php/438/Clases_2010-2011/Taller1/MaterialesComplementarios/CMMI.pdf

15. GEOTHESIS. *BPM. El futuro de los procesos de negocio.* 01 de Agosto de 2008. [En línea] http://www.geothesis.com/index.php?option=com_content&view=article&id=478:bpm-el-futuro-de-los-procesos-de-negocio&catid=21:artulos&Itemid=100

16. JACOBSON, I. BOOCH Y G. RUMBAUGH: *El Proceso Unificado de Desarrollo de software*, [s.l.], Addison-Wesley, 2000.

17. MARCOS GUGLIELMETTI**:** *Mastermagazine. Definición de Arquitectura Software.* 10 de Febrero de 2005. [En línea]. http://www.mastermagazine.info/termino/3916.php

18. ANALÍA LANZILLOTTA**:** Mastermagazine. *Definición de Cliente / Servidor.* [En línea] 10 de 02 de 2005. http://www.mastermagazine.info/termino/4294.php

19. EASTERN SOFTWARE SYSTEM PVT.LTD**:** *Arquitectura en tres capas.* 06.

20. LAGO, RAMIRO: Proactiva calidad. *Java*, abril de 2007. [En línea]. http://www.proactiva-calidad.com/-java/principal.html

21. JUNTA DE ANDALUCIA: *Introducción a JSF.* [En línea]. http://www.juntadeandalucia.es/xwiki/bin-/view/MADEJA/JSF+Introduccion

22. JUNTA DE ANDALUCIA: Junta de Andalucia. *RichFaces.* [En línea]. http://www.juntadeandalucia.es/-xwiki/bin/view/MADEJA/RichFaces

23. Adictos al trabajo. *Introducción a Ajax4jsf.* [En línea] http://www.adictosaltrabajo.com/tutoriales/-Ajax4Jsf.php

24. Scribd. *Facelets y JSF – Uso de Templates.* [En línea]. http://www.scribd.com/doc/44720276/F-acelets-y-JSF

25. HISPANETWORK PUBLICIDAD Y SERVICIOS, S.L.: Servicio Antivirus. *Definicion JAVASCRIPCT*, 1ro de octubre de 2009. [En línea] http://antivirus.interbusca.com/glosario/JAVASCRIPT.html

26. RAVIOLI, PABLO: Monografias. *Lenguaje de programación para paginas web.* [En línea] http://www.monografias.com/trabajos7/html/html.shtml

REFERENCIAS BIBLIOGRÁFICAS

27. loikjuhny. *jboss.* [En línea] http://www.scribd.com/doc/19026497/JBOSS

28. JAVA HISPANO: *Introduccrión a Enterprise Java Beans,* 01 de 07 de 2002. [En línea] http://www.javahispano.org/contenidos/es/introduccrion_a_enterprise_java_beans/

29. JUNTA DE ANDALUCIA: Junta de Andalucia. *Ficha Hibernate.* [En línea] http://www.juntadeandalucia-.es/xwiki/bin/view/MADEJA/Ficha+hibernate

30. ANDREAS VIKLUND: Web Aplication- Plataforma J2EE. *JBoss Seam Framework,* febrero 20, 2008. [En línea] http://wilmanchamba.wordpress.com/2008/02/20/jboss-seam-framework/

31. Visual Paradigm UML for Enterprise Edition. *Paradigm, Visual.* [En línea]. http://www.visual-paradigm.com/product/vpuml/

32. ATTRIBUTION-NONCOMMERCIAL-SHAREALIKE 2.5: *Guía Ubuntu*. *Eclipse.* [En línea].

33. —. *PosgreSQL-es .* [En línea] http://www.postgresql-es.org/sobre_postgresql

34. RAFAEL MARTINEZ: Guía de la comunidad para las herramientas GUI de PostgreSQL. *Open Source / Software Libre.* [En línea] http://wiki.postgresql.org/wiki/CommunityGuidetoPostgreSQLGUI-Tools/es

35. **(**APS), ÁREA TEMÁTICA ATENCIÓN PRIMARIA PARA LA SALUD: Definición de la metodología de desarrollo enfocada a procesos.

36. BOOCH, J. R. *El proceso unificado de desarrollo de Software,* España, (2000).

Bibliografía

1. **(**APS), ÁREA TEMÁTICA ATENCIÓN PRIMARIA PARA LA SALUD: Definición de la metodología de desarrollo enfocada a procesos.
2. Adictos al trabajo. *Introducción a Ajax4jsf.* [En línea] http://www.adictosaltrabajo.com/-tutoriales/Ajax4Jsf.php
3. ANALÍA LANZILLOTTA: *Mastermagazine*. *Definición de Cliente / Servidor.* [En línea] 10 de 02 de 2005. http://www.mastermagazine.info/termino/4294.php
4. ANDREAS VIKLUND: Web Aplication- Plataforma J2EE. *JBoss Seam Framework.* [En línea] febrero 20, 2008. http://wilmanchamba.wordpress.com/2008/02/20/jboss-seam-framework/
5. ATTRIBUTION-NONCOMMERCIAL-SHAREALIKE 2.5: *Guía Ubuntu*. *Eclipse.* [En línea].
6. BEJARANO CASTRO, MÓNICA Y OTROS: Revista Colombiana de Cirugía. *Caracterización de los pacientes con lesiones de causa externa mediante un sistema de vigilancia epidemiológica.* [En línea] http://www.encolombia.com/medicina/cirugia/Cirug%C3%ADa21306/Ciru21306_caracterizaci%C3%B3n.htm
7. EASTERN SOFTWARE SYSTEM PVT.LTD**:** *Arquitectura en tres capas.* 06.
8. ECURED. Ministerio de Salud Pública*.* [En línea] http://www.ecured.cu/index.php/MINSAP
9. GEOTHESIS. *BPM. El futuro de los procesos de negocio.* 01 de Agosto de 2008. [En línea] http://www.geothesis.com/index.php?option=com_content&view=article&id=478:bpm-el-futuro-de-los-procesos-de-negocio&catid=21:artulos&Itemid=100
10. HISPANETWORK PUBLICIDAD Y SERVICIOS, S.L.: Servicio Antivirus. *Definicion JAVASCRIPCT*, 1ro de octubre de 2009. [En línea] http://antivirus.interbusca.com/glosario/JAVASCRIPT.html
11. INFOMED Portal de salud de cuba. *La práctica de la vigilancia en salud pública. Un nuevo enfoque en la República de Cuba.* [En línea] http://www.sld.cu.
12. JACOBSON, I. BOOCH Y G. RUMBAUGH: *El Proceso Unificado de Desarrollo de software*, [s.l.], Addison-Wesley, 2000.

BIBLIOGRAFÍA

13. JAVA HISPANO: *Introduccrión a Enterprise Java Beans,* 01 de 07 de 2002. [En línea] http://www.javahispano.org/contenidos/es/introduccrion_a_enterprise_java_beans/

14. JUNTA DE ANDALUCIA: *Introducción a JSF.* [En línea]. http://www.juntadeandalucia.es/xwiki/bin-/view/MADEJA/JSF+Introduccion

15. JUNTA DE ANDALUCIA: Junta de Andalucia. *Ficha Hibernate.* [En línea] http://www.juntadeandaluciaes/xwiki/bin/view/MADEJA/Ficha+hibernate

16. JUNTA DE ANDALUCIA: Junta de Andalucia. *RichFaces.* [En línea]. http://www.juntadeandalucia.es/-xwiki/bin/view/MADEJA/RichFaces

17. LAGO, RAMIRO: Proactiva calidad. *Java*, abril de 2007. [En línea]. http://www.proactiva-calidad.com/-java/principal.html

18. LEMUS, ELIA ROSA, RADAMÉS BORROTO, EUGENIO Y ANEIROS-RIBA, R: *Atención Primaria de Salud, Medicina Familiar y Educación Médica,* La Paz, Biblioteca de Medicina, 1998, Vol. XXXIV.

19. loikjuhny. *jboss.* [En línea]. http://www.scribd.com/doc/19026497/JBOSS

20. MACHADO ALBA, JORGE ENRIQUE: *Revista de Ciencias Humanas, No.* 26.

21. MARCOS GUGLIELMETTI: Mastermagazine. *Definición de Arquitectura Software,* 10 de febrero de 2005. [En línea]. http://www.mastermagazine.info/termino/3916.php

22. MÁS LAGO, PEDRO**:** "La Higiene y Epidemiología en Cuba a los 50 años del triunfo de la Revolución", *Revista Cubana de Higiene y Epidemiología*. [En línea] 21 de mayo de 2009. http://scielo.sld.cu/scielo.php?pid=S1561-30032009000200001&script=sci_arttext

23. MAZZETTI SOLER, PILAR Y OTROS: *Marco conceptual metodológico para el abordaje de Promoción de la Salud.* Lima, Perú, 2005.

24. MINHO CONILL, E; Y FAUSTO RODRÍGUEZ Y MARCIA C: *Análisis de la Integración de la Atención Primaria de Salud en la Red de Servicios en Europa y América Latina*, marzo. 2009.

25. PÉREZ, CARMELO LÓPEZ: Entorno Visual de Aprendisaje. *Modelo de Madurez de la Capacidad del Software1.* [En línea] 29 de 12 de 2004. http://eva.uci.cu/file.php/438/Clases_2010-2011/Taller-1/Materiales_Complementarios/CMMI.pdf

BIBLIOGRAFÍA

26. PISCOYA ÁNGELES, PATRICIA: SlideShare. *Niveles de Atención en Salud,* 2008. [En línea]. http://www.slideshare.net/pathyp75/niveles-de-salud-presentation

27. *PosgreSQL-es .* [En línea] http://www.postgresql-es.org/sobre_postgresql

28. RADA, GABRIEL: Escuela de Medicina Ponticima Universidad Católica de Chile. [En línea] http://escuela.med.puc.cl/recursos/recepidem/parEpidem1.html.

29. RAFAEL MARTINEZ: Guía de la Comunidad para las herramientas GUI de PostgreSQL. *Open Source /Software Libre.* [En línea]. http://wiki.postgresql.org/wiki/Community_Guide_-to_Postgre-SQL-GUI_Tools/es

30. RAVIOLI, PABLO: Monografias. *Lenguaje de programación para paginas web.* [En línea]. http://www.monografias.com/trabajos7/html/html.shtml

31. ROJAS OCHOA, F. Y MARTÍNEZ CALVO, S: 50 Aniversario de la revolución cubana. *Programa de Higiene y Epidemiología.* [En línea]. http://revolucioncubana.cip.cu/logros/modelo-social-socialista/salud/programa-de-higiene-y-epidemiologia

32. Scribd. *Facelets y JSF – Uso de Templates.* [En línea]. http://www.scribd.com/doc/44720276/F-acelets-y-JSF

33. STUSSER BELTRANENA, RODOLFO J. Y ALFREDO RODRÍGUEZ DÍAZ**:**. *La informatización de la atención primaria de salud. Revista Cubana de Medicina General Integral*. 2006. [Disponible en: http://bvs.sld.cu/revistas/mgi/vol22_4_06/mgi12406.htm#cargo

34. Visual Paradigm UML for Enterprise Edition. *Paradigm, Visual.* [En línea] http://www.visual-paradigm.com/product/vpuml/

Glosario de Términos

Cliente servidor: Modelo para construir sistemas de información, que se sustenta en la idea de repartir el tratamiento de la información y los datos por todo el sistema informático, permitiendo mejorar el rendimiento del sistema global de información.

Componente: Parte física y reemplazable de un sistema que se ajusta a, y proporciona, la realización de un conjunto de interfaces.

Equipos Básicos de Salud: Binomio conformado por el médico y enfermera de la familia, que atiende una población geográficamente determinada, que puede estar ubicado en la comunidad, centros laborales o educacionales.

Policlínico: Unidad de salud donde se brindan servicios médicos a una población geográficamente determinada, perteneciente al nivel asistencial de Atención Primaria de Salud.

Servicio: Unidad de software que encapsula alguna funcionalidad de negocio y proporciona estas a otros servicios a través de interfaces públicas bien definidas.

Servicio Web: Es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

Software: Conjunto de programas y procedimientos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema.

Software Libre: Es el software que, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

Unidad de Salud: Centro de trabajo que pertenece al Ministerio de Salud Pública (MINSAP).

Zoonosis: Enfermedades que se transmiten de forma natural entre animales vertebrados y el hombre.