Universidad de las Ciencias Informáticas

Facultad 7

Título: Diseño y Servicios Web para el Registro de Enfermedades de Declaración Obligatoria del Sistema de Información para la Salud.

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

Autores: Leinier Lubián Lorenzo.
Liusvani Suárez Bárzaga.

Tutores: Ing. Mirna Cabrera Hernández
Ing. David Barreto Medina

Consultante: Ing. Lucía E. Domínguez Abreu

Junio, 2007
Ciudad de La Habana

Declaración de Autoría

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

Para que así conste firmo la presente a los 5 días del mes de Julio del año 2007.

Leinier Lubián Lorenzo	**Liusvani Suárez Barzaga**
Autor	**Autor**
Ing. Mirna Cabrera Hernández	**Ing. David Barreto Medina**
Tutora	**Tutor**

Datos de Contacto

Tutores

Mirna Cabrera Hernández-mirna@softel.cu: Graduado de Ing. Sistema Automatizado de Dirección Técnico Económico (SAD) en el año 1986 en el ISPJAE. Posee categoría docente de Profesor Auxiliar y cursa la maestría de Gestión de Proyectos Informáticos. Ha impartido la asignatura Gestión de Software en la Facultad 7 desde el curso 2005-2006. Ha presentado ponencias en eventos científicos nacionales e internacionales. Se desempeña como Líder del Proyecto APS en la Empresa Softel.

Ing. David Barreto Medina-dbarreto@uci.cu: Profesor graduado de Ing. Industrial en el año 2004. Ha impartido asignaturas como Administración de Empresa, Contabilidad y Comercio Electrónico. Se desempeña como líder del proyecto APS y actualmente está cursando el postgrado de Costo de Proyectos.

Oponente

Caridad Guzmán Vitón-cary@softel.cu: Especialista A en Sistemas Organizativos e Informativos de la empresa Softel. Graduada en Cibernética Matemática en el año 1986 en la Universidad de La Habana. Posee 19 años de experiencia en el desarrollo de software desempeñando diferentes roles. Ha trabajado en las empresas Textiles Rubén Martínez Villena (1987 – 1993) y Softel (1993 hasta la actualidad). Trabajó en la empresa mixta BC BIOCON Internacional S.A. España (1996-2001). Ha pasado varios cursos de superación y postgrado. Ha sido tutora de tesis.

Asesor

Alejandro Martinez Castellini-alexmc@uci.cu: Graduado de licenciatura en Educación especialidad Matemática en el ISPEJV en el año 1993. Posee categoría de Instructor y cursa la maestría en Enseñanza de la Matemática en la Educación Superior, en la Universidad de la Habana. Es plantilla del departamento de Ingeniería Industrial en el ISPJAE y se encuentra en prestación de servicios en la UCI desde hace tres cursos impartiendo en todo momento las asignaturas M1 y M2.

Consultante

Lucia Esperanza Domínguez Abreu-lucyd@softel.cu: Especialista A en Sistemas Organizativos e Informativos de la empresa Softel. Graduada de Ing. en Informática. Posee 15 años de experiencia en el desarrollo de software desempeñando diferentes roles. Trabajó en la empresa mixta BC BIOCON Internacional S.A. España (1995). Ha pasado varios cursos de superación y postgrado. Ha sido tutora de tesis.

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

Aristóteles

Se tiene el talento para honrarse con él,
no para deshonrar a los demás.

José Martí

Agradecimientos

Gracias a todos aquellos que colaboraron de una forma u otra en este trabajo:

A nuestros padres por darnos el apoyo y la confianza durante toda nuestra vida.

A todos los compañeros de trabajo y amigos que tanto nos enseñaron y ayudaron,

A los líderes de proyecto de la empresa Softel.

A Dios, por darnos la fuerza para no rendirnos y la perseverancia para terminar nuestra tesis.

A fin de cuentas, los resultados obtenidos son también suyos.

Dedicatoria

Dedico este trabajo:

A mi madre y a mi abuela, que en paz descanse.

Ambas han hecho que continuara mi camino hasta

este punto y me convirtieron en el hombre que soy.

Leinier Lubián

Este trabajo de diploma se lo dedico:

A toda mi familia y en especial a mi madre,

por ser mi ejemplo de lucha en la vida y por confiar en mí.

A mi tutora Mirna Cabrera, por ayudarme cuando más lo necesitaba.

Liusvani Suárez

Resumen

El trabajo que se presenta tiene como objetivo diseñar e implementar un sistema informático para la gestión de la información estadística de las Enfermedades de Declaración Obligatoria en el Sistema Nacional de Salud. Resultando de vital importancia para que se tenga un control epidemiológico, de manera que se evite la propagación de las enfermedades transmisibles, lográndose así, un mejor control sanitario de la población.

Con el desarrollo del trabajo se obtuvo una aplicación Web. Se modelaron los flujos de diseño e implementación utilizando la metodología Proceso Unificado de Rational (RUP). Se utilizo MySql Server para elaborar la base de datos, como lenguajes de programación Web PHP, XSLT, JavaScript. Además, la arquitectura propuestas por el MINSAP, utilizando la librería de clases PLASER, que posee una arquitectura de tres capas, Basada en Componentes y Orientada a Servicios, usando el paradigma de XML Web Services, específicamente SOAP.

Con el Registro de Enfermedades de Declaración Obligatoria se tendrá un rápido acceso a la información relacionada con las EDO desde cualquier punto con conectividad. La información relacionada con las enfermedades, quien la padece o quien la diagnostica, estará disponible en cualquier momento que se solicite. Con la aplicación se logrará un mejor control y vigilancia epidemiológica en todos los niveles del Sistema Nacional de Salud.

Palabras Clave: Enfermedades de Declaración Obligatoria (EDO), Vigilancia epidemiológica, Sistema Nacional de Salud, tecnologías, arquitectura, Proceso Unificado de Rational (RUP).

Tabla de contenidos

Introducción

El registro de enfermedades y fallecidos por su causa de muerte en Cuba data de finales del siglo XIX. Existen documentos que hacen evidente la notificación de Enfermedades Transmisibles desde la época de la colonia, así consta en las Actas de Capitulación del Ayuntamiento de La Habana.

Cuba se convierte en el primer país del mundo en elevar la sanidad al rango de Ministerio cuando en 1909 se crea la Secretaría de Sanidad y Beneficencia. Es en este año también donde se comienza a editar el *Boletín Oficial de la Secretaría de Sanidad y Beneficencia*, el que se mantiene en publicación hasta 1960. Estas revistas se consideran como la fuente de datos estadísticos más valiosa durante la época de la Neocolonia, ya que incluían, en general, todos los datos relacionados con la salud, como por ejemplo mortalidad, morbilidad, nacimientos, matrimonios y enfermedades infectocontagiosas.

En el período republicano, no existía personal calificado que se dedicara a las Estadísticas de Salud y el personal dedicado a estas funciones estaba obligado a informar a los locales de Sanidad. Esta notificación de Enfermedades Transmisibles, se llevaba en registros primarios de cada localidad, los cuales eran enviados por telegrama a la Secretaría de Sanidad y Beneficencia. Era entonces que se emitía el Boletín Oficial, con periodicidad mensual, que contenía los nuevos casos reportados, los fallecidos y los pacientes que se mantenían con enfermedades de notificación.

Con el Triunfo de la Revolución comenzó una arrolladora transformación del país en todos los ámbitos de la Salud Pública, incluido el de la vigilancia epidemiológica. En 1961, se creó el Sistema de Vigilancia de la Enfermedades de Declaración Obligatoria (EDO), en 1979 se estableció un Sistema de Información Directa (SID), con la finalidad de eliminar algunas deficiencias del EDO y lograr el conocimiento inmediato acerca de situaciones de alarma

epidemiológica y para enfermedades sometidas a vigilancia nacional e internacional, problemas sanitarios de alimentos, problemas de higiene escolar y problemas de higiene ambiental.

Durante el año 1991 el Ministerio de Salud Pública elaboró un documento programático: "Objetivos, Propósitos y Directrices para incrementar la salud de la población cubana 1992-2000" en el que se definieron metas y objetivos de salud hasta el año 2000 y se delineó la estrategia para lograrlo. Esto implicó la necesidad de monitorear determinados indicadores y de evaluar sistemáticamente el estado de salud de la población, a mediano y largo plazo, y a la vez establecer sus tendencias así como evaluar el impacto de los programas de intervención.

El perfeccionamiento de la vigilancia trajo como consecuencia una mayor integración y un mayor nivel de análisis y de utilización de la información que ésta generaba; además, obligó al Ministerio de Salud Pública a reformular algunos aspectos conceptuales y estructurales de la vigilancia epidemiológica en consonancia con los conocimientos existentes sobre la misma en el ámbito internacional y en aras de lograr un mayor beneficio para la toma de decisiones; sobre una base más científica en los diferentes niveles y de acuerdo con la política de descentralización de las instancias de gobierno.

Entonces, se tomó la decisión de ampliar el enfoque hacia la Vigilancia en Salud Pública y crear las Unidades de Análisis y Tendencias en Salud (UATS), como instancias integradoras de toda la información de la vigilancia en el nivel central y en todas las provincias y municipios del país, durante el período 1993-1995.

Debe destacarse, como aporte esencial en estos años, el desarrollo del Componente Alerta-Acción del Sistema de Vigilancia en Salud, que mediante los Reportes diarios de incidencias relevantes de salud del país, los Reportes semanales de incidencias, la Evaluación semanal del comportamiento esperado de eventos de salud seleccionados, la Evaluación semanal del comportamiento de la infestación por Aedes Aegypti, los Reportes técnicos de vigilancia en una primera etapa con evaluaciones epidemiológicas rápidas (cualitativas o cuantitativas) y la Vigilancia Epidemiológica internacional sistemática y oportuna, le han conferido una connotación esencial y relevante en el proceso de alerta temprana y respuesta oportuna ante el escenario sanitario nacional e internacional.

Evidentemente el proceso de notificación de las enfermedades transmisibles es de una importancia vital para el control higiénico – epidemiológico de la población, manteniéndose una comunicación fluida entre las diferentes dependencias de salud pública del país que permitiese tomar medidas de control epidemiológico de tal manera que se evitara la propagación de las enfermedades transmisibles, lográndose así, un mejor control sanitario de la población.

Es indiscutible el auge que está teniendo el desarrollo de las tecnologías informáticas a nivel mundial. Como parte de la Batalla de Ideas, la informatización de los procesos que se desarrollan en el Ministerio de Salud Pública (MINSAP), es una tarea de primordial importancia en la búsqueda de la optimización de los servicios de salud que se brindan a la población, para llegar a tener servicios de salud de excelencia y de este modo estar al nivel de la infraestructura global de información existente.

En las líneas generales del Desarrollo Informático en la Salud se encuentran: la Atención Primaria, Secundaria y Terciaria, el Sistema Integrado de Urgencia Médica, Vigilancia de Salud, Telemedicina, Medicamentos y Fármacos, Epidemiología, Docencia Médica y Biblioteca y Universidad Virtual.

Situación Problémica

En particular, en esta nueva etapa de fortalecimiento del Sistema Nacional de Salud (SNS), la Atención Primaria de Salud (APS) es el eje fundamental de estas transformaciones, se tienen como objetivos fundamentales convertir los Policlínicos en centros de excelencia, cada vez más accesibles a la población y consolidar el Sistema Municipal de Salud, para de esta manera poder dar cumplimiento al principio de la descentralización en las soluciones a los problemas de salud de la comunidad, entre otros.

Actualmente los reportes de Enfermedades de Declaración Obligatoria (EDO) se reciben a nivel nacional pasadas varias semanas después de ocurrido el diagnóstico a un paciente en la unidad de salud que la detectó, lo que implica la pérdida de un tiempo que podría traer como consecuencia el desarrollo de un foco epidemiológico, trayendo consigo un desfasaje entre la notificación y la planificación de las medidas de control. Además, dentro del consolidado EDO,

el cual se ocupa de hacer las captaciones de las EDO con tarjeta, o sin esta, la información que se envía es insuficiente, ya que sólo se envía los nombres del paciente y de la enfermedad.

Desde el año 1991 ha estado en uso un sistema automatizado programado en MSDOS que necesita ser modificado ya que no satisface los requerimientos actuales y no existen programadores que trabajen esa plataforma que ya es obsoleta. Otro sistema, implementado en FoxPro resuelve el problema de hacer cálculos y tablas manualmente, pero sólo a nivel provincial y nacional. La gestión de las tablas de salida es muy rápida con ese sistema, sin embargo después de 14 años de explotación hay herramientas en la computación que pueden facilitar aún más el trabajo. El sistema en FoxPro tiene dos versiones: provincial y nacional. En la versión provincial se introducen semanalmente los datos por municipios, agrupados según sexo y grupos de edad de cada enfermedad. Se envía un fichero a la nación que se imprime y se vuelven a captar estos datos de los municipios, esta vez por provincia.

Las dos versiones tienen tablas de salida similares (una con los municipios de cada provincia, la otra con las provincias) que cruzan las variables contenidas (según sexo, grupos de edad, meses y semanas estadísticas para una determinada enfermedad, para un grupo o para todas). Otro elemento importante es la codificación para las EDO. Hoy las estadísticas de las EDO son construidas en base a un codificador interno de Cuba, lo que implica que no exista una coincidencia con los códigos que poseen los grupos, subgrupos o enfermedades en la Clasificación Internacional de Enfermedades, 10ma edición (CIE–10). Esto trae consigo que a la hora de realizar informes estadísticos del país para ser enviados a los organismos internacionales de salud, como por ejemplo, la Organización Panamericana de Salud (OPS), tarda un tiempo considerable por no codificar inicialmente las EDO teniendo en cuenta la codificación del CIE-10.

Además, el sistema de información estadístico de las EDO es uno de los más complejos, voluminosos y periódicos del Sistema Nacional de Salud (SNS), debido al conjunto de reportes que se deben emitir.

El actual proyecto de informatización de la APS tiene como misión implementar un Programa General de Informatización del Sistema Nacional de Salud teniendo como centro del mismo al

Policlínico, que apoye las estrategias y políticas trazadas por la dirección del país y el MINSAP; de manera que se logre la incorporación progresiva y sistemática de las Tecnologías de la Información y las Comunicaciones (TIC) en función de la adquisición y gestión del conocimiento y los servicios de salud. Con esto se propone aportar rapidez, fiabilidad y un mayor grado de acceso a la información pertinente y necesaria para las modernas técnicas de dirección y que esta información además de que llegue con la puntualidad necesaria a los niveles requeridos, sirva de referencia a otras áreas con similares problemas para la solución de sus necesidades.

Este proyecto de informatización se ha estructurado sobre una arquitectura Basada en Componentes (CBA) y Orientado a Servicios (SOA), estos componentes no son más que los registros de información básicos que se gestionan en este nivel de atención, siendo uno de ellos el Registro de Enfermedades de Declaración Obligatoria (REDO).

Existe la necesidad imperiosa de optimizar el procesamiento de la información, permitir la introducción o recepción de datos remotamente y que los datos estén organizados de manera que puedan ser recuperados o modificados de forma rápida y sin tener que acceder a otros componentes de información del sistema.

Problema a Resolver

¿Cómo gestionar de manera automatizada la información relacionada con las Enfermedades de Declaración Obligatoria en el Sistema Nacional de Salud?

Objetivo General

Diseñar e implementar un sistema informático para la gestión de la información estadística de las Enfermedades de Declaración Obligatoria en el Sistema Nacional de Salud.

Idea a Defender

Al realizar el diseño e implementación de la aplicación, se garantizará una mejor gestión de la información relacionada con las Enfermedades de Declaración Obligatoria en el Sistema Nacional de Salud, permitiendo un control epidemiológico más eficiente.

Objeto de estudio

Proceso de gestión de la información en el Sistema Nacional de Salud cubano.

Campo de acción

Proceso de gestión de la información estadística relacionada con las Enfermedades de Declaración Obligatoria en el Sistema Nacional de Salud.

Tareas de la investigación

1. Estudiar el estado del arte del tema tratado en la investigación.
2. Realizar un estudio de las tendencias y tecnologías actuales para realizar el proceso de implementación de dicha solución automatizada como por ejemplo PHP, XML, XSLT, XML Web Services, MySQL, XHTML, JavaScript.
3. Asimilar de la Plataforma de Servicio (PlaSer), librería de clases, de la empresa SOFTEL, mediante la cual se desarrollan los procesos de implementación.
4. Modelar el flujo de diseño e implementación utilizando la metodología Proceso Unificado de RationalDiseñar e Implementar las funcionalidades del sistema en la capa de negocio entregadas por el analista y conectarlas con la capa de presentación.
5. Diseñar una Base de Datos capaz de organizar y almacenar de manera eficiente la información que se manipula sobre los pacientes con EDO, que soporte además integridad referencial y transacciones.
6. Aplicar los estándares existentes para el diseño, codificación y tratamiento de errores o excepciones en las capas de datos, negocio y presentación de la aplicación a implementar.
7. Implementar los servicios Web del Registro de Enfermedades de Declaración Obligatoria.
8. Realizar la Integración con los componentes del Sistema de Información para la Salud (SISalud).

Estructura del Documento

El documento se estructura en tres capítulos, los cuales, incluyen la fundamentación teórica, así como la documentación del diseño e implementación de la solución informática que se propone.

Capítulo 1. Fundamentación Teórica: Se enmarca al lector en los conceptos básicos del Sistema Nacional de Salud y del Sistema de Información para la Salud (SISalud) para una mayor comprensión del tema tratado. Se analizan sistemas automatizados existentes en Cuba y en el mundo y se realiza un estudio de la situación actual del tema a nivel de país y mundial. Por último se presenta un estudio de las tecnologías, lenguajes y metodologías utilizadas para el diseño e implementación de aplicaciones Web.

Capítulo 2. Diseño del Sistema: Se realiza el flujo de trabajo del diseño partiendo de la base del análisis propuesto por el analista. Para ello se comienza por explicar los patrones de diseño utilizados y luego se modelan todos los diagramas propios de este flujo de trabajo mediante estereotipos Web. Posteriormente se describen las clases del diseño y se muestra el modelo de datos de la aplicación. Por último se describen las clases y atributos de las tablas de la base de datos.

Capítulo 3. Implementación: Se comienza este capítulo explicando la integración con los componentes de SISalud de los cuales se consumen servicios en las distintas funcionalidades del sistema. Se definen los componentes y se realiza el diagrama de despliegue como parte del modelo de implementación. Luego se pasa a explicar los métodos y agentes, cuya comprensión puede resultar engorrosa y por último se describen los estándares de diseño, codificación y tratamiento de errores.

Capítulo 1. Fundamentación Teórica

Introducción

En este capítulo se trata de dar al lector los elementos necesarios para una mejor comprensión del tema. Se describe la estructura del Sistema Nacional de Salud cubano y del Sistema de Información para la Salud (SISalud). Se describen brevemente algunos sistemas informáticos para la vigilancia de enfermedades a nivel mundial y en el país que pueden servir como referencia a la hora de elaborar la solución propuesta.
Se trata también la situación del control epidemiológico mediante un estudio de la situación actual global y nacional del tema.
Las técnicas, lenguajes y metodologías utilizadas actualmente para la realización de aplicaciones Web se describen y se analizan en la parte final de este capítulo.

1.1 Sistema Nacional de Salud (SNS)

El Sistema Nacional de Salud se puede clasificar mediante la distribución administrativa en:

- **Nivel Nacional**: Es el Ministerio de Salud Pública como órgano rector de la salud cubana en sí. Incluye en él todas sus funciones inherentes (metodológicas, normativas, de coordinación y control).
- **Nivel Provincial**: En cada provincia del país existe una dirección de salud (Dirección Provincial de Salud), la cual se subordina a la Asamblea Provincial del Poder Popular ya sea en lo relativo a administración, como al financiamiento.
- **Nivel Municipal**: Al igual que en la provincia, en los municipios existe una dirección de salud que se subordina a la instancia municipal del Poder Popular que le corresponde. En los municipios también los llamados Consejos Populares.

Y según los niveles de atención se puede dividir en:

- **Atención Primaria de Salud (APS):** Este nivel, cuya esencia es la participación activa de la comunidad en la solución de sus problemas de salud, es el nivel más

básico de atención al paciente, el cual no sólo se restringe al consultorio médico o al policlínico, sino que puede darse en cualquier institución con carácter médico. Su función está dada por brindar servicios médicos de carácter ambulatorio y de consultoría generalmente, como pueden ser emitir un diagnóstico o un tratamiento con poca complejidad. Si se sospecha una enfermedad grave en una persona se remite al nivel de atención superior que corresponda.

Este nivel comprende tanto a personas sanas como enfermos dentro de su jurisdicción. En cuanto a los enfermos se les da seguimiento a sus padecimientos, ya sean crónicos o no, y se les proporciona bienestar de vida a pacientes con patologías incurables.

- **Atención Secundaria de Salud:** A este tipo de atención se llega generalmente mediante remisión por parte del médico en la atención primaria. Ya aquí se ofrecen servicios técnicos y terapéuticos de gran complejidad frente a carencias de salud de moderadas a graves. Estos servicios de salud se obtienen de manera ambulatoria en policlínicos u hospitales o mediante la hospitalización en sí.

- **Atención Terciaria de Salud:** Este nivel de atención se caracteriza por la especialización en el servicio y sólo se brinda en centros específicos con funciones bien enfocadas como centros o institutos de investigación. Ejemplo de esta atención también pueden ser Servicios de Neurocirugía, Nefrología, Cirugía Cardiovascular o Trasplante Renal.

1.1.1 Informatización del SNS

A partir del año 2003 el Ministerio de Salud Pública de la República de Cuba (MINSAP) define como una de sus prioridades la informatización, como recurso para añadir un valor agregado al potente sistema de salud cubano, con nivel comparado con los del primer mundo. Es por este motivo que se decide desarrollar bajo una arquitectura basada en componentes y orientada a servicios un sistema que permitiera integrar la información de diferentes áreas de una manera sencilla pero eficiente. [1]

Es aquí cuando comienza el desarrollo de aplicaciones basadas en “software libre” y orientadas a este fin.

1.1.2 Registro Informatizado de Salud (RIS)

El RIS surge como respuesta a la necesidad planteada en el epígrafe anterior. Se decide su creación para integrar información de las diferentes áreas de salud, las cuales se tenían como partes aisladas y que no garantizaban la disponibilidad de información única, confiable y en tiempo real para la toma de decisiones.

El RIS es en sí la solución informática integral para la Salud Pública, acorde con los objetivos de la informatización de la sociedad cubana. Constituido por un conjunto de aplicaciones independientes (módulos del sistema) que se interconectan según las necesidades del flujo de información. Es además la herramienta que permite a los usuarios autorizados combinar la información de los diferentes módulos que lo componen, para obtener una información integral en tiempo real para la toma de decisiones en los diferentes niveles de dirección, la docencia, investigación y la gestión en salud.[2]

1.1.3 El Sistema de Información para la Salud (SISalud)

Posteriormente se vio la necesidad de crear otros sistemas de información con nivel paralelo al RIS pero que manejaran otras informaciones como el Sistema Informatizado de Atención Primaria (SIAP) y el Sistema de Gestión Hospitalaria (SIGH). El primero contiene los módulos propios de la APS, como el Registro de Población, mientras que el segundo contempla los de la Atención Secundaria de Salud como el Registro de Autopsias.

El SISalud surge de la integración de estos componentes, los cuales se relacionan entre sí mediante el intercambio de información necesaria para realizar sus funciones en los diferentes niveles.

1.2 Estado del arte a nivel internacional, nacional y de la Universidad

A continuación se exponen observaciones relacionadas con el estado actual del tema de esta investigación. Algunas características de esta situación actual, o sea las deficiencias o amenazas de epidemia, son las razones por las cuales se elabora este proyecto informático y otras características evidencian las ventajas que se deben potenciar en este sector de la salud.

El control de las Enfermedad de Declaración Obligatoria o estudio epidemiológico está dado por el análisis de la diseminación y causas de las situaciones o sucesos relacionados con la salud en una población específica, y la aplicación de ese estudio al control de los problemas de salud.

Los brotes y epidemias de enfermedades transmisibles, nuevas o conocidas, por su forma súbita de ocurrencia, su expansión en cortos períodos de tiempo, por sus enormes repercusiones sociales y políticas, demandan respuesta inmediata de las autoridades sanitarias encargadas de su vigilancia.

Debido al rápido incremento de los viajeros, así como la velocidad y la frecuencia con que las personas se mueven, un evento epidémico local puede transformarse rápidamente en una amenaza global. Por lo cual y apoyados en la velocidad con que se mueve la información a nivel mundial actualmente, un brote de una enfermedad que ocurra en algún lugar distante del planeta puede ser percibido como una amenaza.

Actualmente existe cierta ineficacia a nivel mundial en cuanto a evitar que las enfermedades se diseminen entre países, sin embargo, Cuba presenta una infraestructura fuerte y preparada para detectar e intervenir oportunamente en situaciones de emergencia causadas por epidemias de forma efectiva, eficiente y sostenida. Aunque, en algunas ocasiones, la percepción de los riesgos en salud, muchas veces no coinciden con los datos epidemiológicos recibidos mediante los mecanismos pasivos de notificación. A través de esta situación, dada en el ámbito internacional, y de la cual la tierra cubana no ha estado exenta, sólo se confirma la necesidad de fortalecer el sistema de vigilancia epidemiológica.

Haciendo un poco de historia, en 1958 existían apenas 400 unidades médicas con un total de 32 500 camas para la atención médica. Esto, por supuesto, imposibilitaba el acceso a los servicios básicos y por tanto impedía enfrentar los males que afectaban al país. Además de que no existían en ese entonces institutos de investigaciones médicas.

El control epidemiológico tenía un comportamiento a nivel municipal, mediante una jefatura local de sanidad, existiendo un cuerpo de inspectores cuyas acciones eran muy limitadas y con un nivel de corrupción bastante elevado. En aquel entonces, estas jefaturas se encargaban de recoger la poca información de algunas de las enfermedades transmisibles y la participación en campañas de vacunación frente a las situaciones epidémicas que con frecuencia se presentaban. El control de las enfermedades y de la higiene- epidemiología era bastante pobre y muchas veces estaba acompañado de un amplio registro, característico de esa época en las estadísticas de morbilidad y en menor grado en las de mortalidad.

Hoy en día, la epidemiología se enfrenta a nuevos desafíos y oportunidades. La aplicación de la informática a grandes archivos de datos ha ampliado la función y la capacidad de los estudios epidemiológicos.

Cuba, ha experimentado una total transformación en cuanto a la situación de salud de la población. Teniendo en consideración la sensible disminución de la incidencia de las enfermedades infecciosas, unido a la relativamente alta vigencia de enfermedades no transmisibles los cuales han generado cambios considerables percibiéndose estos en el decrecimiento de los patrones de mortalidad y morbilidad. La evolución en la infraestructura de servicios de salud, con su organización y con poder resolutivo, contando con sobrados recursos humanos y científicamente calificados, así como el equilibrio entre el uso de tecnologías de punta junto a la ética, el humanismo y la solidaridad, han sido elementos básicos de un desarrollo abarcador y tecnológico en el sector de la salud.

En la Universidad de las Ciencias Informáticas (UCI), un grupo de estudiantes en conjunto con especialistas de la empresa Softel, del Ministerio de la Informática y las Comunicaciones (MIC), y expertos funcionales del MINSAP, son los encargados de poner a disposición de la comunidad aplicaciones desarrolladas por estos para beneficio de la atención médica que sea

pertinente. Con esto se cumple con el deber de mejorar mediante herramientas informáticas la infraestructura de la salud cubana. Realizando este proyecto de tesis se dará solución a una gama de problemas que afectan particularmente a la gestión concerniente a la información de las enfermedades que deben declararse de forma obligatoria en las áreas de salud de cada persona, teniendo esto repercusión a nivel nacional.

1.3 Sistemas Informáticos para la Vigilancia de Enfermedades

Estos sistemas automatizados surgen desde finales del siglo pasado para la detección e intervención urgente en situaciones de emergencia, para desarrollar las estrategias de investigación y establecer las medidas de control necesarias, proporcionando a las autoridades sanitarias información oportuna y retroalimentación periódica.

La característica más importante de estos sistemas lo constituye la rapidez en la detección. Muchos de ellos funcionan en tiempo real, o sea, al mismo tiempo que se recibe la información se van disparando los mecanismos para la alerta inmediata.

Sin embargo, la necesidad de elaborar una solución propia se deriva de que los sistemas no responden a las necesidades del proyecto necesidades o están fuera de alcance. En el caso de los sistemas de otros países son en su mayoría software propietario y bastantes caros, además de que por supuesto no se identifican con la estructura de salud tan particular que posee Cuba. Muchas de estas aplicaciones, tanto cubanas como extranjeras, se enfocan sólo en el control de una enfermedad, y lo que se busca es la vigilancia del total de elementos de la lista de Enfermedades de Declaración Obligatoria.

Por último, pero no menos importante, las aplicaciones que hoy en día existen en el país no se integran satisfactoriamente con el Sistema de Información para la Salud.

1.3.1 Antecedentes en el mundo

Leaders (Lightweight Epidemiology Advanced Detection & Emergency Response System): Uno de los primeros sistemas automatizados de alerta, fue desarrollado por la Agencia de Proyectos de Investigación para la Defensa de los Estados Unidos. Utiliza información de los registros

hospitalarios de varios hospitales de emergencia seleccionados y rastrea el reporte de síntomas inusuales. [3]

RODS (Real Time Outbreak and Disease Surveillance): La primera versión de este sistema fue creado por investigadores norteamericanos de la Universidad de Pittsburg. Este sistema recolecta información de las principales causas de admisión en los servicios de urgencia de hospitales seleccionados y en tiempo real clasifica estas causas en categorías de síndromes basado en la Novena Revisión de la CIE y alerta cuando existen anomalías que pueden ser brotes epidémicos. [4]

ALERT (Advanced Logic for Event Detection in Real Time): Su funcionamiento se basa en la información recogida de variables "señales": Órdenes de cultivo de líquido cerebro espinal, hemocultivos, coprocultivos y radiografías de tórax positivas a neumonía. [5]

GPHIN (Global Public Health Intelligence Network) Red de Información Mundial en Salud Pública: Este sistema fue creado en Canadá y es utilizado por la OMS desde 1997. Es un sistema de alerta temprana, seguro, basado en Internet. Busca reportes preliminares significativos de problemas de salud, en tiempo real. Este sistema único y multilingüe busca y provee información importante no verificada acerca de brotes de enfermedades y otros eventos en salud pública, monitoreando los medios de comunicación a escala mundial en seis idiomas. [6]

NDRM (Nacional Retail Data Monitor): Es un sistema automatizado que desde diciembre del año 2002 en Estados Unidos, analiza las ventas diarias de medicamentos sin prescripción en tiendas pertenecientes a varias cadenas de establecimientos minoristas de casi todo el país. [7]

Sentiweb: Sistema Centinela de Vigilancia que funciona desde 1984 en Francia, con una muestra de médicos generales (1%) del país conectados mediante una computadora personal y un MODEM, los que transmiten la incidencia de varias enfermedades transmisibles seleccionadas a un servidor. [8]

FormSUS (Notificação de Surtos e Emergências em Saúde Pública) Notificación de brotes y emergencias en Salud Pública: Sistema online de uso público para la creación de formularios para diversos tipos de información en el sector salud, con la finalidad de agilizar, estructurar y mejorar la calidad del proceso de recolección y diseminación de datos en Internet. [9]

DengueNet*:* Sistema automatizado en tiempo real basado en Internet creado por la OMS para la vigilancia global del dengue y el dengue hemorrágico. Las principales características de este sistema de vigilancia son: servicio protegido por contraseña que permite la introducción de datos a distancia, inclusión de subdivisiones por estado o provincia de los países, para los que se introducen o se calculan indicadores, dispositivo dinámico de consulta y preguntas, con el análisis y la presentación de los datos en gráficos, tablas, textos y utilización de herramientas SIG para elaborar mapas. [10]

Sistema de Información en el Sistema Nacional de Vigilancia de Salud: Surge en Argentina con el propósito de automatizar totalmente el ingreso de datos, el análisis epidemiológico básico (tablas y gráficos) de comportamiento de los daños bajo vigilancia y monitorea el comportamiento de los atributos (aceptabilidad, oportunidad, cobertura) del sistema. [11]

1.3.2 Antecedentes en Cuba

SAVT (El Sistema Automatizado de la Vigilancia de la Tuberculosis): Este sistema fue uno de los primeros confeccionados en el país por investigadores del Instituto de Medicina Tropical "Pedro Kourí" en 1994. [12]

VIGILA: Software para la vigilancia epidemiológica de enfermedades transmisibles que permite registro y mantenimiento de datos, creado en 1998 en el Instituto de Medicina Tropical "Pedro Kourí". [13]

WINEDOS: Sistema automatizado para el análisis de las bases de datos de las enfermedades de declaración obligatoria desarrollado en la UATS de la provincia Las Tunas, en el año 2000. Tiene entre sus funciones captar datos preliminares, importar datos que se oficialicen en los departamentos de estadísticas, captar poblaciones para el cálculo de las tasas, generar canales por diferentes tipos de métodos de cálculos, tablas o reportes de salida con tasas,

proporciones y el diagnóstico de una epidemia por municipios y provincia, con frecuencia semanal. [14]

Vigired: Este sistema automatizado creado en el municipio de San Cristóbal, provincia de Pinar del Río en el 2005, garantiza la inmediatez de la información al actualizar simultáneamente la base de datos, cada vez que se introduce nueva información, está diseñado en formato Web y puede ser solicitado y consultado desde cualquier lugar y momento. [15]

SIVD: (Sistema Integrado de Vigilancia de Dengue) Es un sistema que se utiliza desde el año 2005 en los municipio Cotorro y en Centro Habana. Permite agilizar en gran medida el procesamiento de la información obtenida mediante la recogida de información clínico-epidemiológica de los pacientes que eran diagnosticados como sospechosos de dengue, además de información ambiental recogida por parte de los compañeros de la Campaña de Lucha Antivectorial y de datos entomológicos provenientes del laboratorio a tal efecto. [16]

1.4 Estado del arte de técnicas, tecnologías, metodologías y software usados en la actualidad

A continuación, se hace un análisis de cómo se encuentran las tecnologías y técnicas que pueden ser adecuadas para llevar a cabo el sistema que se pretende desarrollar. De igual forma se fundamenta la propuesta final.

1.4.1 Internet: TCP/IP, WWW

Es una realidad cuanto Internet ha evolucionado las comunicaciones modernas, nada importante está fuera de la red hoy en día. Basados en la infraestructura global, numerosos servicios se han ido desarrollando, creciendo así las exigencias para las aplicaciones, en cuanto a facilidad y transparencia para los usuarios, así como la necesidad de ser eficientes en todo sentido.

La posibilidad de conectar prácticamente cualquier dispositivo a la red, es un hecho. Actualmente casi todo incluye una conexión de red: Automóviles, Cámaras domésticas, Consolas de Juego, Hornos de Microondas, entre otros. Esto significa que la red no es solamente de las personas, sino también de "las cosas".

Luego de una primera generación en las tecnologías de comunicaciones aplicadas a la transmisión de datos caracterizada por sistemas centralizados, la segunda generación nos trajo sistemas distribuidos, la imposición del cable UTP, la fibra óptica y la modificación de los canales físicos en las urbes, así como el surgimiento del cable submarino. A mediados de los 90s una capacidad de 45 Mbps era considerada como bastante alta, actualmente se instalan sistemas de hasta 40 Gbps.

En cuanto a Internet existen muchas tecnologías emergentes como son el Modo de transferencia asincrónico (ATM, siglas en inglés), el cual permite operar a velocidades de 40 Gbps, hace posible además, la transmisión de cualquier tipo de servicio y presenta QoS (Calidad de Servicios) para cada tipo de servicio; VoIP, o sea, voz sobre IP e IPv6. Con estas tecnologías se ofrecen al usuario QoS configurable, ancho de banda a demanda así como bajos costos de instalación y operación.

Los dispositivos móviles (celulares, PDAs, entre otros) han demostrado su funcionalidad y flexibilidad en un nuevo modelo de comunicación en la productividad o en la vida personal. Por este motivo se ha producido un crecimiento vertiginoso, de usuarios, computadoras, dispositivos móviles, aparatos domésticos, y aplicaciones y servicios de todo tipo, en apenas una década de la etapa comercial de Internet. El incremento constante del número de host y del requerimiento de ancho de banda, la necesidad de contar con un nivel mayor de seguridad y un espacio de direcciones suficiente ha motivado un cambio en el protocolo TCP/IP evidenciado en el desarrollo y la aceptación de protocolo IPv6, como el protocolo para la nueva era de Internet.

IPv6 brinda direcciones de 128 bits, así como encabezados flexibles, por lo tanto un número casi infinito de direcciones IP. El mismo se aplica sobre Internet2, el cual es un proyecto que nació entre las universidades de EEUU y tiende a extenderse a todo el mundo. Su estructura básica es similar a Internet, pero con un ancho de banda muy superior, está constituido por “gigapops” (Gigabits Capacity Point Of Presence) que unen diferentes regiones en forma directa evitando los grandes nudos de concentración de la red. Utiliza exclusivamente IPv6, que

mejora la confiabilidad y la seguridad de la red. Los enlaces son ATM que garantizan un acceso mínimo de 34 Mbps.

Estos años definirán en buen grado, el proceso de estandarización de IPv6. Algunos factores que deben resultar catalizadores del proceso, son los avances obtenidos en su adopción y desarrollo en Asia y Europa, la salida este año, del nuevo sistema operativo IPv6 Ready, de Microsoft (Windows Vista) y la celebración de los Juegos Olímpicos en Beijing, China (bautizados como Olimpiada Digital, basada puramente en IPv6), y la exigencia de que todas las agencias federales norteamericanas estén corriendo el protocolo IPv6 en ese año.

Cuba planea la incorporación de estas nuevas tendencias, en particular de IPv6 dentro de un rango de tres años. Con vistas a este desarrollo en mayo del año pasado el Grupo IPv6 Cuba participó en la X Reunión de LACNIC, en la cual se expuso el trabajo realizado y se intercambiaron fuerzas para realizar tareas IPv6 en la región.

Como se puede observar, existe toda una nueva generación de servicios avanzados en los que la transferencia simultánea de estímulos múltiples, permitirá desarrollar aplicaciones “virtuales” todavía no imaginadas en todos los campos (educación, medicina, negocios, etc.). Con estos cambios se podrán alcanzar una gran variedad de novedosos servicios como Teleinmersión, Laboratorios Virtuales o Bibliotecas Digitales.

No obstante de las ventajas antes expuestas por estas nuevas tendencias, el proyecto que aquí se presenta se desarrolla basado en IPv4 debido a que el framework designado por la empresa Softel, PlaSer, trabaja hoy en día sobre IPv4.

1.4.2 Arquitectura en Capas

Actualmente, el diseño de aplicaciones modernas involucra casi siempre una arquitectura basada en capas. Existen tres propuestas para la arquitectura en capas, donde las capas reciben a veces el nombre de niveles (dos capas, tres capas y cuatro capas). [17]

Por supuesto, los elementos en las diferentes capas tienen la necesidad de comunicarse, lo que varía son los permisos de comunicación entre estos elementos.

Existen tres posibilidades:

- Arquitectura top-down de capas: Los elementos pueden enviar solicitudes a las capas inferiores. Esto genera una cascada de solicitudes de las capas superiores hacia las capas inferiores. Una arquitectura top-down es *no estricta* si los elementos de una capa superior pueden enviar peticiones directamente a cualquier elemento de cualquiera de las capas inferiores.
- Arquitectura bottom-up de capas: Los elementos de capas inferiores alertan a las capas superiores que ha ocurrido un evento de interacción, ejemplo de esta capa son los drivers de dispositivos. Esta arquitectura puede, igualmente, ser *no estricta* en el caso de que los eventos se notifiquen a cualquiera de las capas superiores.
- Arquitectura bidireccional de capas: Es común su implementación mediante dos pilas de capas que se comunican entre sí, mayormente usada en los protocolos de redes de computadoras. [18]

El módulo REDO se desarrolló siguiendo una arquitectura de tres capas y top-down en su forma de comunicación donde las capas definidas de superior a inferior son:

- Capa de Presentación: Contiene todos lo relacionado con las interfaces de interacción con los usuarios. En general incluye el manejo y aspecto de las ventanas, así como sus funcionalidades.
- Capa de Negocio (o Acceso a Datos): Contiene los elementos o métodos que automatizan los procesos del negocio que se ejecutan desde el cliente.
- Capa de Datos (o Repositorio): Contiene los datos persistentes de la aplicación agrupados en tablas de la base de datos.

1.4.3 Arquitectura Basada en Componentes

Esta arquitectura habla de cómo construir aplicaciones complejas a través de ensamblar módulos, o componentes, diseñados previamente con el fin de ser reusados. Se hace énfasis en esta arquitectura en la estandarización de las interfaces de comunicación de los componentes. Estos módulos se diseñan desde un comienzo para integrarse en una gran variedad de configuraciones. [19]

Una de las ventajas de los componentes es la reusabilidad, aunque depende en buen grado de la correcta identificación de los componentes, así como del modo en que se organicen estos. El Desarrollo Basado en Componentes (DBC) brinda el soporte para la integración de las partes en sistemas mayores, por lo cual se requiere de mucho cuidado a la hora del modelado de la arquitectura con fines de asegurar la compatibilidad entre los componentes que interactúan. [20]

Este enfoque se diferencia de otros en la que separa la especificación de componentes de la implementación y divide la especificación en interfaces. Existen dos categorías para los componentes de acuerdo con el grado de abstracción: componentes conceptuales, del análisis y el diseño, y componentes de implementación, de despliegue.

Desde un punto de vista de aplicación el módulo REDO es un componente del SISalud que interactúa con varios otros módulos de este sistema general.

1.4.4 Arquitectura Orientada a Servicios

No existe una definición de amplia aceptación de este tipo de arquitectura de software que no sea su comprensión literal como una arquitectura soportada sobre una orientación a servicios como su principio de diseño fundamental. La interacción entre estos servicios se produce utilizando lenguajes de tipo descriptivo, y la misma es independiente de cualquier otra interacción. [21]

La Arquitectura Orientada a Servicios (SOA) es comúnmente implementada mediante Web-Services basados en Simple Object Access Protocol (SOAP) el cual es un protocolo que dice como dos objetos pueden comunicarse a través de XML. [22]

Para la implementación de los Servicios Web también se utilizan los WSDL (Web Services Description Language), los cuales describen la interfaz pública de los servicios y el formato que han de tener los mensajes que van a interactuar con los servicios disponibles. [23]

Tanto SOAP como WSDL en su versión 1.0 se encuentran bajo el auspicio de World Wide Web Consortium, o sea W3C, el cual es un consorcio internacional que produce estándares para la red mundial. [24]

1.4.5 Metodologías de desarrollo

En un proyecto de desarrollo de software la metodología define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. Una metodología es un proceso.
No existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable.

UML

UML, cuyas siglas en inglés significan Unified Modeling Language, es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Se utiliza para modelar sistemas orientados a objetos e incluye un conjunto de diagramas y notaciones estándar y su significado. UML cuenta con varios tipos de diagramas para definir sistemas de software y hardware, detallar artefactos en los sistemas y documentar la programación. Puede ser utilizado en variadas metodologías de software aunque se utilizará específicamente en el Proceso Unificado de Desarrollo (RUP) en el actual proyecto.

RUP

El Rational Unified Process (RUP) es un proceso de ingeniería de software que mejora la productividad del equipo de trabajo y entrega las mejores prácticas del software a todos los miembros del mismo. Proporciona una guía específica en áreas tales como la de Modelado de Negocios, Arquitecturas Web, Pruebas y Calidad.

Basado en las mejores prácticas adoptadas en miles de proyectos por todo el mundo, utiliza como lenguaje de modelación el anteriormente expuesto UML. Contiene patrones que permiten a los gestores de proyectos añadir o quitar rápidamente piezas que resuelvan problemas comunes y enfocarlas en el proyecto específico. Permite manejar contenidos en diferentes dominios como el modelado de bases de datos o control de requisitos.

Los casos de uso, o sea, lo que los usuarios futuros necesitan o desean, guían el proceso de desarrollo. Los modelos que se obtienen como resultado de los diferentes flujos de trabajo representan la realización de un Caso de Uso (CU).

La arquitectura muestra la visión común del sistema, describe los elementos del modelo que son más importantes para su construcción. El RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura.

RUP propone que cada fase del proyecto se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo aunque se enfoca en unos más que en otros.

1.4.6 Lenguajes de Programación Web

Los lenguajes de programación Web se clasifican en dos partes fundamentales, los lenguajes del lado del Servidor y los lenguajes del lado del Cliente.

Entre los lenguajes del lado del servidor se pueden presentar algunos significativos como PHP, ASP.NET, PERL, Java, JSP, los módulos CGIs e ISAPIs entre otros. Estos se caracterizan por desarrollar la lógica de negocio dentro del Servidor, además de ser los encargados del acceso a Bases de Datos, tratamiento de la Información, etc. Del lado del cliente se encuentran principalmente el JavaScript y el Visual Basic Script, que son los encargados de aportar dinamismo a la aplicación en los navegadores.

Esta distinción en los lenguajes ha sido necesaria debido a que la Web funciona en modo “Desconectado”, o sea, un usuario a través de un navegador hace una petición de una página Web a un Servidor Web (Request); el Servidor recepciona la petición, la procesa y le envía la Respuesta al Cliente (Response), este realiza la recepción y se desconecta.

PHP

Es el acrónimo recursivo de Hypertext Preprocessor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Es también un lenguaje interpretado y embebido en el HTML.

PHP es un lenguaje de programación muy potente que, junto con HTML, permite crear sitios Web dinámicos. PHP se instala en el servidor y funciona con versiones de Apache, Microsoft IIS, Netscape Enterprise Server y otros. [25]

La forma de usar PHP es insertando código PHP dentro del código HTML de un sitio Web. Cuando un cliente (cualquier persona en la Web) visita la página Web que contiene éste código, el servidor lo ejecuta y el cliente sólo recibe el resultado. Su ejecución, es por tanto en el servidor, a diferencia de otros lenguajes de programación que se ejecutan en el navegador. PHP permite la conexión a numerosas bases de datos, incluyendo MySQL, Oracle, ODBC, etc. y puede ser ejecutado en la mayoría de los sistemas operativos (Windows, Mac OS, Linux, Unix. [26]

Resumiendo, el PHP corre en 7 plataformas, funciona en 11 tipos de servidores, ofrece soporte sobre unas 20 Bases de Datos y contiene unas 40 extensiones estables sin contar las que se están experimentando, además de que: [27]

- Es software libre y abierto, lo que implica menos costes y servidores más baratos que otras alternativas.
- Es muy rápido. Su integración con la base de datos MySQL y el servidor Apache, le permite constituirse como una de las alternativas más atractivas del mercado.
- Su sintaxis está inspirada en C, ligeramente modificada para adaptarlo al entorno en el que trabaja, de modo que si se está familiarizado con esta sintaxis, resultará muy fácil aprender PHP.
- Su librería estándar es realmente amplia, lo que permite reducir los llamados "costes ocultos", uno de los principales defectos de ASP.
- PHP tiene una de las comunidades más grandes en Internet, por lo que no es complicado encontrar ayuda, documentación, artículos, noticias, y más recursos.
- Posee una potente variedad de extensiones para el acceso a la mayoría de los sistemas de gestión de bases de datos, por lo que una migración a otro sistema de gestión es mucho menos costosa que en otras plataformas.

ASP.NET

Es un conjunto de tecnologías de desarrollo de aplicaciones Web comercializado por Microsoft. Es usado por programadores para construir sitios Web domésticos, aplicaciones Web y servicios XML. Forma parte de la plataforma .NET de Microsoft y es la tecnología sucesora de la tecnología Active Server Pages (ASP). [28]

Esta tecnología genera páginas dinámicas en el servidor y las envía al cliente (navegador Web) que las ha solicitado, ejecutando previamente el código que contienen (código Visual Basic, C#, etc.) y convirtiendo el resultado a código HTML, que es el único que puede interpretar adecuadamente el cliente. La clave de este proceso es que, a diferencia de la navegación clásica en que el servidor espera (*escucha*) peticiones del cliente y, cuando las recibe, envía automáticamente la respuesta (generalmente un documento HTML que es mostrado en el navegador Web), aquí el servidor ejecuta algún tipo de procesamiento a raíz de la petición del cliente y elabora dinámicamente la respuesta que devuelve.
En esquema sería:

- El cliente (explorador Web) se conecta al servidor Web a través de su URL.
- El cliente envía una petición al servidor (solicita una página Web). El cliente puede ejecutar código (*en el lado del cliente*), con secuencias de comandos que se utilizan para generar efectos dinámicos en el cliente (DHTML). Este código no se procesa en el servidor.
- El servidor busca la página solicitada y ejecuta el código contenido en ella. Este código del lado del servidor se escribe en secuencias de comandos (para tareas simples), pero también en lenguajes compilados (Visual Basic, C#, etc., para tareas complejas como son cálculos matemáticos, recuperación de datos, gestión de sitios, comercio electrónico, etc.).
- El servidor traduce el resultado de la ejecución del código a HTML y lo envía al cliente.
- El cliente muestra al usuario el documento recibido. Si el usuario consulta el código fuente del documento que visita, no verá el código original del archivo que reside en el servidor, sino solamente código HTML en que la tecnología ASP .NET ha convertido el resultado del procesamiento. [29]

PERL

Lenguaje de programación de scripts multiplataforma desarrollado por Larry Wall en 1987. Toma características del lenguaje C, del shell, de AWK, Lisp, etc. Se utiliza para manipular textos y ciertos procesos, y especialmente para la creación de CGI. Perl es un lenguaje imperativo, con variables, expresiones, asignaciones, bloques de código delimitados por llaves, estructuras de control y subrutinas. [30]

El diseño de Perl puede ser entendido como una respuesta a tres amplias tendencias de la industria informática: rebaja de los costes en el hardware, aumento de los costes laborales y las mejoras en la tecnología de compiladores. Anteriormente, muchos lenguajes de ordenador como el Fortran y C, fueron diseñados para hacer un uso eficiente de un hardware caro. En contraste, Perl es diseñado para hacer un uso eficiente de los costosos programadores de ordenador.

Es un lenguaje libre de uso, eso quiere decir que es gratuito. Antes estaba muy asociado a la plataforma Unix, pero en la actualidad está disponible en otros sistemas operativos como Windows.

JAVA

Lenguaje de programación orientado a objeto parecido al C++. Usado en WWW para la telecarga y telejecución de programas en el ordenador cliente. Desarrollado por Sun Microsystems. [31]

Pero lo que hace de Java un concepto diferente es que, en un segundo nivel, es también un entorno para la ejecución de programas, englobado en la llamada máquina virtual de Java. Este entorno es un software que permite que las aplicaciones escritas en Java se ejecuten en cualquier ordenador, independientemente del sistema operativo y de la configuración de hardware utilizados. [32]

Este enfoque consiste en introducir como elementos de las páginas Web los llamados applets, o "programitas" escritos en Java. El navegador carga los applets junto con el resto de la página, y una vez cargados se ejecutan en el ordenador del usuario, visualizándose el resultado en la pantalla del navegador.

La filosofía en la programación de los applets debe ser la misma que la que presidía la programación de los chips de los electrodomésticos: los programas deben ser pequeños (para que se carguen en poco tiempo) y rápidos (para que se visualicen en la página Web sin demoras), a la par que robustos y fiables (son inadmisibles errores o fallos del applet que pudieran modificar los ficheros del usuario o bloquear su sistema operativo); y sobre todo: han de funcionar en cualquier tipo de plataforma, con cualquier sistema operativo bajo el que funcione el navegador. El lenguaje y la plataforma Java permiten todo ello. [33]

Entre 1996 y 1997 mejora drásticamente Java como lenguaje de programación, al incluir facilidades de impresión, el estándar JDBC de acceso a bases de datos, así como el estándar JavaBeans ("granos de café"): éste potencia el carácter de lenguaje orientado a objetos de Java, al tiempo que proporciona un entorno visual de programación que facilita y simplifica el desarrollo del software escrito en Java. De este modo, Java se convierte en un poderoso entorno de programación y de ejecución de programas para producir no sólo applets aislados para páginas Web, sino también verdaderas aplicaciones de interés comercial, comparables a las programadas hasta ahora en otros lenguajes, pero con las ventajas mencionadas de independencia de plataforma (transportabilidad), robustez y fiabilidad. [34]

XSLT

XSL Transformaciones. Hojas de estilo que transforman documentos empleando reglas de plantillas. Es un estándar de la W3C que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML. La unión de XML y XSLT permite separar el contenido y su presentación.

Fundamentación del Lenguaje a utilizar

Hasta el momento se han analizado las características fundamentales de los lenguajes de programación candidatos para la implementación de la propuesta de este trabajo, para fundamentar la elección se hará una comparación teniendo en cuenta algunas características que influyen directamente en el ambiente de trabajo donde se va a desarrollar la propuesta. En cuanto a:

- **Características multiplataformas:** Menos el ASP, que es solamente soportado por la plataforma Windows, los demás lenguajes están soportados en múltiples plataformas.
- **Velocidad de ejecución:** La velocidad es mayor en PHP, seguidos por PERL y JSP.
- **Disponibilidad de recursos:** Actualmente los más utilizados en Internet son PHP y JSP, siendo más utilizado en la publicación de artículos y códigos de ejemplos. PHP tiene una de las comunidades más grandes en Internet, al igual que la de Java.
- **Familiaridad con el lenguaje:** En la universidad los lenguajes más utilizados por los programadores son el ASP y el PHP.

De acuerdo a estas comparaciones, PHP resulta mucho más favorecido, por tanto se piensa que es el adecuado para implementar la propuesta de sistema de este trabajo, además y principalmente por ser el lenguaje solicitado por el Cliente. También se utiliza el lenguaje XSLT para transformar los documentos XML en páginas HTML y JavaScript para aportar dinamismo a las páginas HTML.

1.4.7 Librería utilizada

PlaSer

Acrónimo de Plataforma de Servicio, constituye una plataforma sobre la que se pueden desplegar aplicaciones XML Web Services. Este sistema está concebido completamente sobre Arquitectura Basada en Componentes y Orientada a Servicios, usando el paradigma de XML Web Services, específicamente SOAP. En su concepción se han utilizado estándares actuales

y normas abiertas. Todo el código ha sido programado en PHP y pertenece a la empresa Softel del Ministerio de la Informática y las Comunicaciones (MIC).

Desde el punto de vista estructural permite trabajar con cualquier base de datos que cumpla con el estándar SQL-92; pero desde el punto de vista de implementación sólo trabaja con las bases de datos soportadas por el componente DBX, ya que encapsula a dicho componente y lo utiliza para el acceso a bases de datos.

PlaSer, en su configuración ideal, se distribuye en tres servidores, los cuales se encuentran conectados en cascada y sólo el primero está conectado a Internet con una IP real. Este primero contiene la capa de presentación (layout), el segundo contiene a ProxPla y los demás componentes, incluido el SAAA (Single Authorization Authentication and Account) componente de seguridad, que autentifica y autoriza, además tiene registrados los nombres de los métodos. El tercer servidor contiene las bases de datos desarrolladas en MySQL. Para aumentar la seguridad se pudiera colocar a ProxPla sólo en un cuarto servidor separado de lo demás componentes.

También PlaSer permite ser desplegado totalmente en un sólo servidor, aunque esto no es recomendable por motivos de seguridad.
PlaSer en su estructura encapsula varios componentes:

- XSL: Extensión para transformaciones de ficheros XML.
- Pear SOAP: Biblioteca para la comunicación SOAP.
- DBX: Extensión para la abstracción del acceso a datos.

El Sistema de Seguridad de **PlaSer** comprende 3 capas o “rejillas”.

1. Primera Rejilla: Niveles:

Existen 4 niveles:

- Nivel 1: Nacional
- Nivel 2: Provincial
- Nivel 3: Municipal
- Nivel 4: Unidades de Salud

2. Segunda Rejilla: Tipos de Usuarios

Existen 3 tipos de usuarios:

- Administrador: Manipula los usuarios
- Editor: Modifica información en el Sistema.
- Visualizador: Sólo puede visualizar información.

3. Tercera Rejilla: Especifica a que módulo se pertenece.

1.4.8 Sistemas Gestores de Bases de Datos

SQL

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos. Es un lenguaje declarativo de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o más programas que utilicen un lenguaje de bajo nivel orientado a registro. Es un lenguaje de cuarta generación (4GL).

El SQL proporciona una rica funcionalidad más allá de la simple consulta (o recuperación) de datos. Asume el papel de lenguaje de definición de datos (**LDD**), lenguaje de definición de vistas (**LDV**) y lenguaje de manipulación de datos (**LMD**). Además permite la concesión y denegación de permisos, la implementación de restricciones de integridad y controles de transacción, y la alteración de esquemas. Las primeras versiones del SQL incluían funciones propias de lenguaje de definición de almacenamiento (**LDA**), pero fueron suprimidas en los estándares más recientes con el fin de mantener el lenguaje sólo a nivel conceptual y externo.
El SQL permite fundamentalmente dos modos de uso:

- Un uso **interactivo**, destinado principalmente a los usuarios finales avanzados u ocasionales, en el que las diversas sentencias SQL se escriben y ejecutan en línea de comandos, o un entorno semejante.

- Un uso **integrado**, destinado al uso por parte de los programadores dentro de programas escritos en cualquier lenguaje de programación anfitrión. En este caso el SQL asume el papel de sub-lenguaje de datos.

En el caso de hacer un uso embebido del lenguaje se pueden utilizar dos técnicas alternativas de programación. En una de ellas, en la que el lenguaje se denomina **SQL estático**, las sentencias utilizadas no cambian durante la ejecución del programa. En la otra, donde el lenguaje recibe el nombre de **SQL dinámico**, se produce una modificación total o parcial de las sentencias en el transcurso de la ejecución del programa. La utilización de SQL dinámico permite mayor flexibilidad y mayor complejidad en las sentencias, pero como contra punto se obtiene una eficiencia menor y el uso de técnicas de programación más complejas en el manejo de memoria y variables.

MySQL

MySQL Database Server es la base de datos de código fuente abierto más usada del mundo. Su ingeniosa arquitectura lo hace extremadamente rápido y fácil de personalizar. La extensiva reutilización del código dentro del software y una aproximación minimalística para producir características funcionalmente ricas, ha dado lugar a un sistema de administración de la base de datos incomparable en velocidad, compactación, estabilidad y facilidad de despliegue. La exclusiva separación del core server del manejador de tablas, permite funcionar a MySQL bajo control estricto de transacciones o con acceso a disco no transaccional ultrarrápido.

Características:

Interioridades y portabilidad:

- Escrito en C y en C++.
- Probado con un amplio rango de compiladores diferentes.
- Funciona en diferentes plataformas.
- Usa GNU Automake, Autoconf, y Libtool para portabilidad.
- APIs disponibles para C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, y Tcl.
- Uso completo de multi-threaded mediante threads del kernel. Pueden usarse fácilmente múltiple CPUs si están disponibles.
- Proporciona sistemas de almacenamientos transaccionales y no transaccionales.

- Usa tablas en disco B-tree (MyISAM) muy rápidas con compresión de índice.
- Relativamente sencillo de añadir otro sistema de almacenamiento. Esto es útil si desea añadir una interfaz SQL para una base de datos propia.
- Un sistema de reserva de memoria muy rápido basado en threads.
- Joins muy rápidos usando un multi-join de un paso optimizado.
- Las funciones SQL están implementadas usando una librería altamente optimizada y deben ser tan rápidas como sea posible. Normalmente no hay reserva de memoria tras toda la inicialización para consultas.
- El servidor está disponible como un programa separado para usar en un entorno de red cliente/servidor. También está disponible como biblioteca y puede ser incrustado (linkado) en aplicaciones autónomas. Dichas aplicaciones pueden usarse por sí mismas o en entornos donde no hay red disponible.

Seguridad:

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

Escalabilidad y límites:

- Soporte a grandes bases de datos. Se usa MySQL Server con bases de datos que contienen 50 millones de registros. También se tienen usuarios que usan MySQL Server con 60.000 tablas y acerca de 5.000.000 de registros.
- Se permiten hasta 64 índices por tabla (32 antes de MySQL 4.1.2). Cada índice puede consistir desde 1 hasta 16 columnas o partes de columnas. El máximo ancho de límite son 1000 bytes (500 antes de MySQL 4.1.2).Un índice puede usar prefijos de una columna para los tipos de columna CHAR, VARCHAR, BLOB, o TEXT.

Conectividad:

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

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

Localización:

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

Clientes y herramientas:

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

Fundamentación del Sistema Gestor de Bases de Datos a utilizar

Hasta el momento se han planteado algunas de las características de los dos sistemas gestores de base de datos (SGBD) más usados en el mundo y en la universidad, a continuación se fundamenta el la selección del gestor a utilizar:

En concreto, usar MySQL tiene ventajas adicionales:

- Escalabilidad: Es posible manipular bases de datos enormes, del orden de seis mil tablas y alrededor de cincuenta millones de registros, y hasta 32 índices por tabla.
- MySQL está escrito en C y C++ y probado con multitud de compiladores y dispone de APIs para muchas plataformas diferentes.
- Conectividad: Es decir, permite conexiones entre diferentes máquinas con distintos sistemas operativos. Es corriente que servidores Linux o Unix, usando MySQL, sirvan datos para ordenadores con Windows, Linux, Solaris, etc. Para ello se usa TCP/IP, tuberías, o sockets Unix.
- Es multi-hilo, con lo que puede beneficiarse de sistemas multiprocesador.
- Permite manejar multitud de tipos para columnas.
- Permite manejar registros de longitud fija o variable.
- FREE: MySQL es libre, o sea no hay que pagar por su utilización.

De acuerdo a estas ventajas, MySQL resulta ser el propuesto para implementar el sistema además de ser el gestor solicitado por el Cliente.

Conclusiones

En este capítulo se ha realizado la Fundamentación Teórica del trabajo, quedando descritos todos los factores alrededor de la gestión de la información de las EDO. Además se aborda la situación y las tendencias de la producción de aplicaciones Web y las herramientas y lenguajes seleccionados para la elaboración de la aplicación que son PHP como lenguaje de programación Web, MySQL Server como gestor de base de datos y PLASER para la seguridad y la arquitectura de 3 capas.

Capítulo 2. Diseño del sistema

Introducción

A lo largo del presente capítulo, se plantea la solución propuesta por el equipo, al problema planteado y se argumentan los resultados obtenidos, principalmente durante el diseño. Este flujo es muy importante para crear una arquitectura sólida, que representa un plano para la etapa de implementación.

Para esto se hace un análisis de los patrones de diseño utilizados y el por qué de su uso. Se definen los elementos de diseño y se exponen algunos diagramas, los más representativos, de las clases del diseño. Además, se hace una descripción de las clases y atributos del diseño.

Para el modelado del sistema se utiliza el Lenguaje Unificado de Modelado (UML), el cual permite la representación de las clases y sus relaciones en los diferentes tipos de diagramas.

2.1 Modelo de diseño

El diseño se basa en aplicar ciertas técnicas y principios con el propósito de definir el sistema, tan detalladamente como sea posible, para así facilitar su interpretación y realización física.

El modelo de diseño está muy cercano al de implementación, lo que es natural para guardar y mantener el modelo de diseño a través del ciclo de vida completo del software. Entre los propósitos de este modelo se encuentran:

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.

- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.
- Capturar las interfaces entre los subsistemas del software, lo cual es muy útil cuando se utiliza interfaces como elementos de sincronización entre diferentes equipos de desarrollo. [35]

2.1.1 Fundamentación del uso de Patrones

Los patrones de diseño son búsquedas de soluciones a problemas comunes en el desarrollo de software. Un patrón es una solución efectiva de un problema, que lo ha resuelto en otras ocasiones y que puede ser reusada en otras circunstancias.

2.1.1.1 GRASP

Es el acrónimo para “General Responsibility Assignment Software Patterns”. Aunque más que patrones propiamente dichos, se les considera prácticas recomendables a la hora del diseño del software. Como bien indica su nombre orientan su función a la asignación de responsabilidades en los objetos.

Se utilizan los patrones de Alta Cohesión y Bajo Acoplamiento, los cuales están bien ligados entre sí, en una relación de proporcionalidad invertida en cuanto al resultado esperado. O sea, que si se enfoca la atención solo en uno, el otro queda en muy malas condiciones.

Esto está dado por el hecho de que si se aumenta mucho la cohesión, aumenta también el acoplamiento entre las clases y si por el contrario se reduce demasiado el acoplamiento, también se ve disminuida la cohesión.

Cohesión

Frente a un objeto que tiene bien delimitadas sus responsabilidades, uno se encuentra ante una clase que tiene alta cohesión. En una programación bien diseñada un objeto o una clase tiene una responsabilidad que cumplir dentro de la aplicación.

Si por ejemplo se crea una clase Profesor, el cual tiene la responsabilidad de dar clases. Si además se le asignan las tareas de limpiar la escuela y entregar el correo se dice que la clase tiene baja cohesión pues realiza demasiadas tareas ajenas a su función principal.

La alta cohesión se expresa en términos de cuán bien enfocadas y relacionadas se encuentren las tareas o métodos que realizan una acción de una clase. Desde el momento en que se asigna a un objeto tareas ajenas a su naturaleza se complica el sistema de objetos y las oportunidades de mantenerlo en el futuro.

Se tiene entonces como una buena práctica que no se deben sobrecargar las responsabilidades de un objeto.

Este factor viene dado en la aplicación por el hecho de que cada módulo del sistema tiene bien delimitadas sus responsabilidades, enmarcadas en los dominios de su negocio. Y a su vez, cada método o agente no tiene funcionalidades más allá de las de su razón de ser.

Acoplamiento

Este término se refiere a las relaciones entre objetos, dentro de un sistema. Cuando un objeto se relaciona con otros en el sistema y estos últimos cambian, necesariamente el primero tiende a verse afectado. De aquí se deriva la necesidad de que un objeto tenga el mínimo de relaciones posibles con otros, para así, poder realizar cambios en partes del programa sin tener que hacer luego grandes modificaciones en la aplicación.

Si un objeto tiene que conocer demasiados detalles internos de otro para su funcionamiento se rompe el encapsulamiento del segundo. Un sistema bien diseñado es por supuesto un sistema con bajo acoplamiento.

Esta técnica trae grandes beneficios a los programadores del módulo desde el punto de vista de que es más fácil reusar un objeto con bajo acoplamiento.

Los métodos no tienen relaciones innecesarias con otros, siempre que puedan resolver ellos el problema y sin disminuir la cohesión, se encargan del asunto.

2.1.1.2 Patrones Estructurales

La función de estos patrones es mayormente la de modelar las relaciones entre los objetos para crear estructuras con cierto grado de complejidad. Aunque existen también dentro de este grupo otros que se encargan de las relaciones de herencia y composición entre las clases.

Fachada

El objetivo de este patrón es simplificar la interface entre sistemas o componentes de software ocultando tras una clase fachada todo un sistema de interacción complejo. Se trata de ocultar todo lo posible la complejidad del sistema, conjunto de componentes que lo forman, para así ofrecer el mínimo posible de puntos de entrada al sistema oculto por fachada.

Además de lo anterior, el patrón ofrece la ventaja de aislar los posibles cambios que tengan lugar en algunas de las partes de los sistemas que interactúan. Si por ejemplo se decidiese migrar la capa de datos de la aplicación en un futuro hacia php5, la capa de presentación no tiene que afectarse.

Proxy

Al igual que Fachada este patrón se encuentra entre los patrones estructurales. Este patrón obliga a que las llamadas a un objeto ocurran indirectamente a través de un objeto proxy, el cual, actúa sustituyendo al original y luego delegando las llamadas a los métodos de las clases respectivas.

Este patrón se usa frecuentemente incluso en otros patrones. El mismo resuelve el problema de cuando se tienen objetos que consumen mucha memoria o recursos de la máquina, ya que se instancian estos objetos sólo en el momento que se solicita por el cliente.
Los objetos del cliente llaman a los métodos del objeto proxy, el objeto proxy recibe llamadas a métodos que pertenecen a otros objetos, con lo cual invoca los métodos del objeto específico que brinda ese servicio.

La razón fundamental para usar un proxy es la transparencia en la administración de los servicios de otros objetos. Con este patrón se fuerza a que todos los accesos a servicios provistos por objetos pasen a través de un objeto proxy sin que los objetos clientes se den

cuenta de que en realidad no están invocando a los métodos directamente en el objeto respectivo.

Los métodos en la aplicación no se invocan directamente sino a través de ProxPla que es el "Proxy de PlaSer". Este componente es el encargado de administrar los servicios y las acciones a las que tiene derecho el cliente.

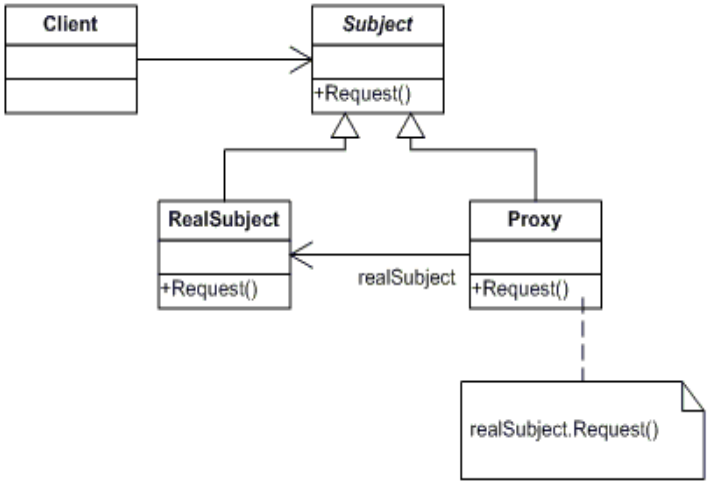


Figura 1. Diagrama UML del patrón Proxy

2.1.2 Definición de elementos de diseño

2.1.2.1 Subsistemas de diseño

La organización de los artefactos del modelo de diseño en piezas que se puedan manejar más fácilmente es posible mediante el uso de los subsistemas. Cada subsistema puede estar contenido por clases del diseño, realización de casos de usos, interfaces e incluso otros subsistemas.

En la Figura 2 se muestra como están distribuidos estos subsistemas en la aplicación.

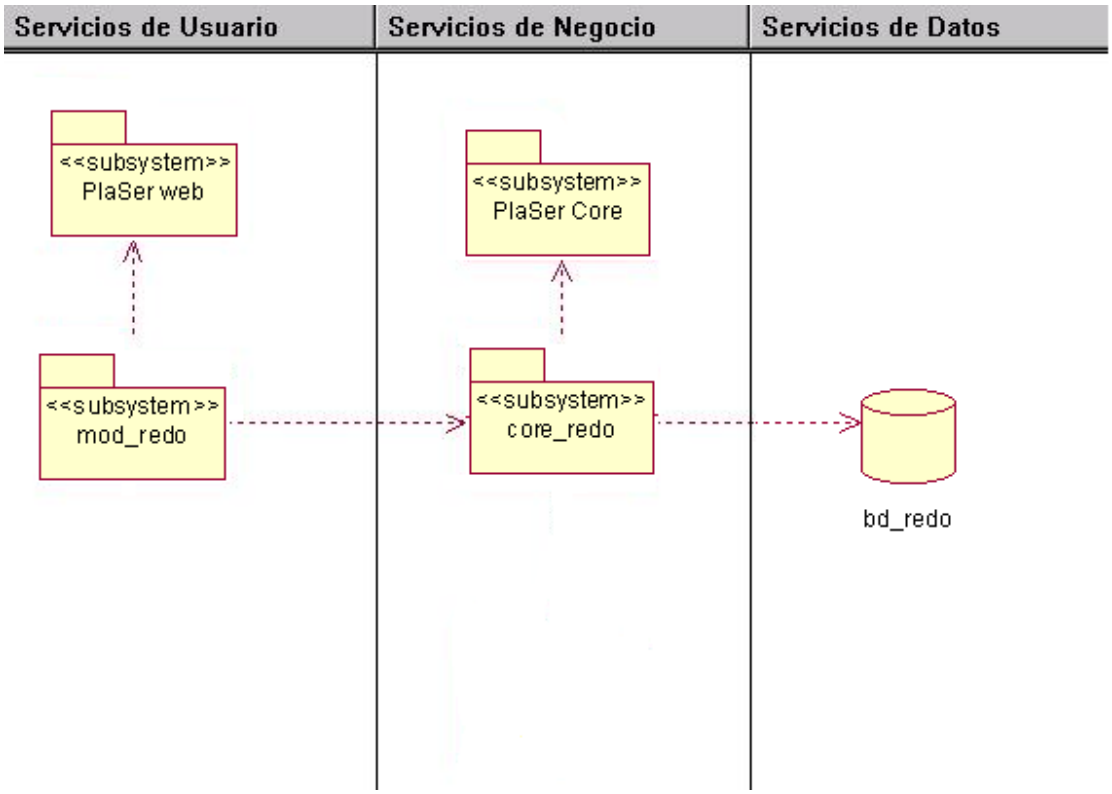


Figura 2. Diagrama de subsistemas del diseño

- El primer subsistema, mod_redo corresponde a la capa superior de la arquitectura con el nombre Capa de Presentación, por lo cual almacena todos componentes y clases que intervienen en las funcionalidades que se brindan mediante las interfaces de usuario.
- Complementando mod_redo, el subsistema PlaSer Web contiene las clases de PlaSer de la capa superior y que intervienen en el desarrollo de esta capa. Un ejemplo de estas clases es Fachada.
- El tercer subsistema, core_redo contiene los objetos correspondientes a la Capa de Negocio (o “core”) de la aplicación.
- PlaSer Core es el equivalente de PlaSer Web pero en la capa de negocio conteniendo clases de acceso a datos como dbz_class.
- El último de los subsistemas, bd_redo, aglomera las clases que representan los datos físicos de la aplicación en cuestión.

2.1.2.2 Modelado mediante Estereotipos Web

A finales de los 90, cuando el desarrollo de aplicaciones Web se hizo más importante, se comienza a hacer uso de las facilidades de extensión brindadas por el UML para basado en este lenguaje modelar aplicaciones Web. [36]

Se considera necesario efectuar una acotación. En sus inicios lenguajes como ASP y PHP, los cuales resultaban frecuentemente empleados para la construcción de aplicaciones Web, no brindaban soporte para la orientación a objetos (OO). UML hace su aparición para el modelado de aplicaciones OO, por lo que existía una brecha relacionada con el lenguaje para modelar los flujos de trabajo y el lenguaje de programación Web. [37]

Esta brecha comienza a disminuirse teniendo en cuenta que un fichero script interpretable con extensión php por ejemplo, contiene código que se ejecuta en el contexto del servidor WWW (código php) y código que se ejecuta en el contexto de la máquina cliente (JavaScript, Visual Basic Script). Por lo cual se establece que como parte de su extensión, cada fichero script interpretable podría modelarse a partir de varias clases de UML, una clase representaría el código puramente servidor, una clase representaría el código puramente cliente, y una clase representaría los formularios presentes en el código cliente. [38]

Así su extensión presenta como elementos más significativos a 3 clases de UML estereotipadas con los siguientes estereotipos “Server Page”, “Client Page” y “Form” empleados para el código servidor, código cliente y formularios respectivamente. [39]

Las relaciones posibles a establecerse entre los tres elementos claves son:

Hasta Desde	**Client Page**	**Form**	**Server Page**
Client Page	<<Link>> , <<redirect>>	Contiene	<<Link>> , <<redirect>>
Form	Agregado por.	---	<<Submit>>
Server Page	<<Build>>, <<redirect>>		<<Redirect>>

Tabla 1. Relaciones entre las clases principales que conforman la extensión de UML para Web.

Una página Cliente podría contener varios formularios e incluso estos pueden enviar sus datos a distintas páginas servidoras encargadas de procesarlos.

Una página Servidora puede para completar su funcionamiento incluir código existente en otra página servidora, a partir de la sentencia “include(../../FachadaEDO.php)” en este caso, podría representarse dicha relación en el diagrama de clases.

Sin embargo, como la herramienta seleccionada para el modelado fue el Rational Rose 2003 de IBM, y la tecnología empleada para la implementación fue libre, al llegar al punto del diseño y el paso a la implementación de la aplicación Web, dicha herramienta no es suficiente, ya que no tiene soporte para generar código en PHP, ni bases de datos en MySql. Por tanto, se culmina el proceso con un documento que contiene los modelos de diseño e implementación sin la posibilidad de generar el código fuente.

2.1.3 Diagramas de Clases del Diseño

En los diagramas de clases se pueden observar claramente las relaciones que unen todas las clases que intervienen en un momento u otro en el funcionamiento del sistema, así como las operaciones y atributos de las mismas.

A continuación se presentan los diagramas de clases del diseño, para el modelado de los cuales se usaron estereotipos Web y son una representación de la mayoría de los tipos de funcionalidades que presenta la aplicación (listar, agregar, modificar, eliminar y generar reportes).

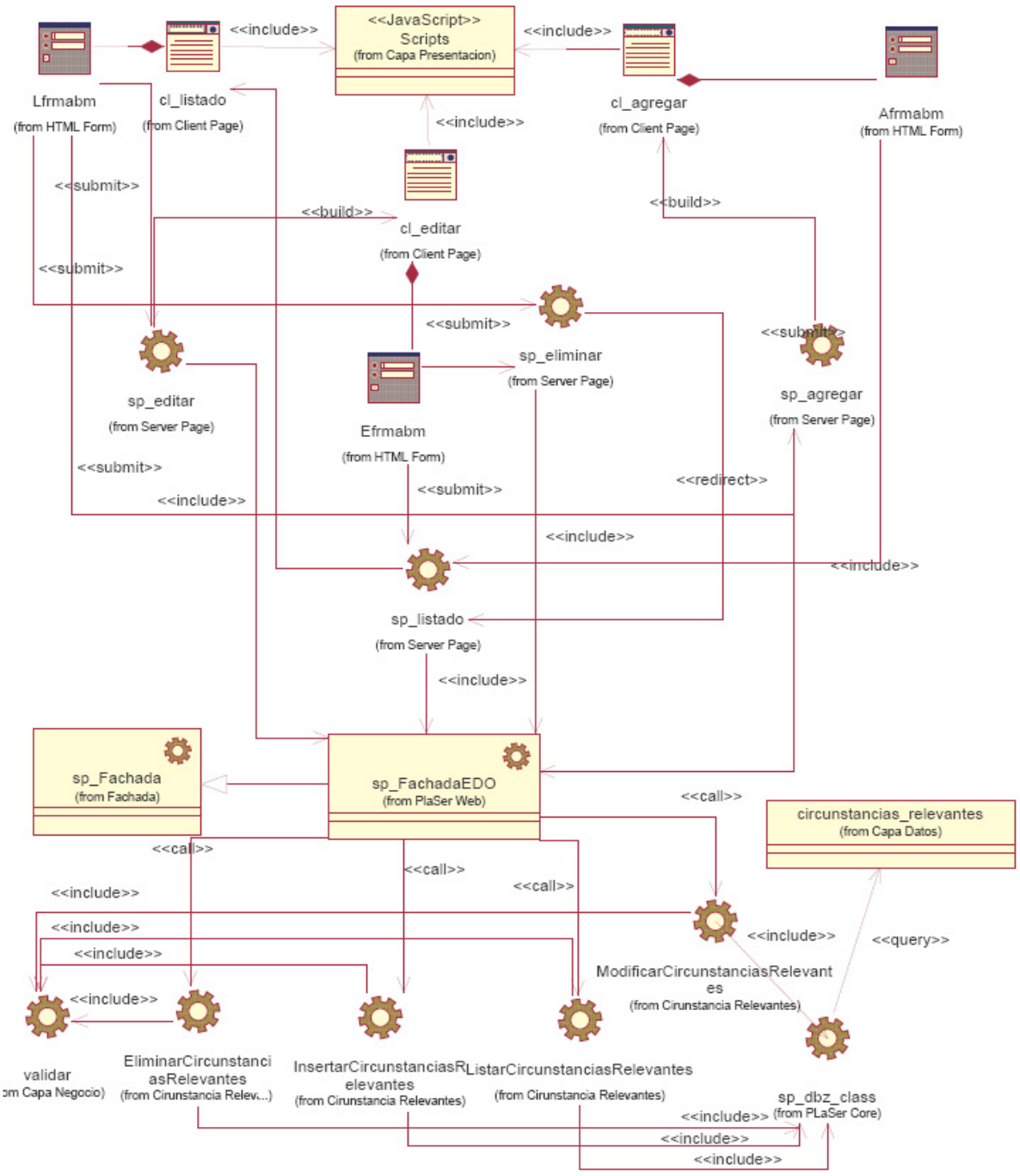


Figura 3. Diagrama de Clases del Caso de Uso "Configurar Circunstancias Relevantes"

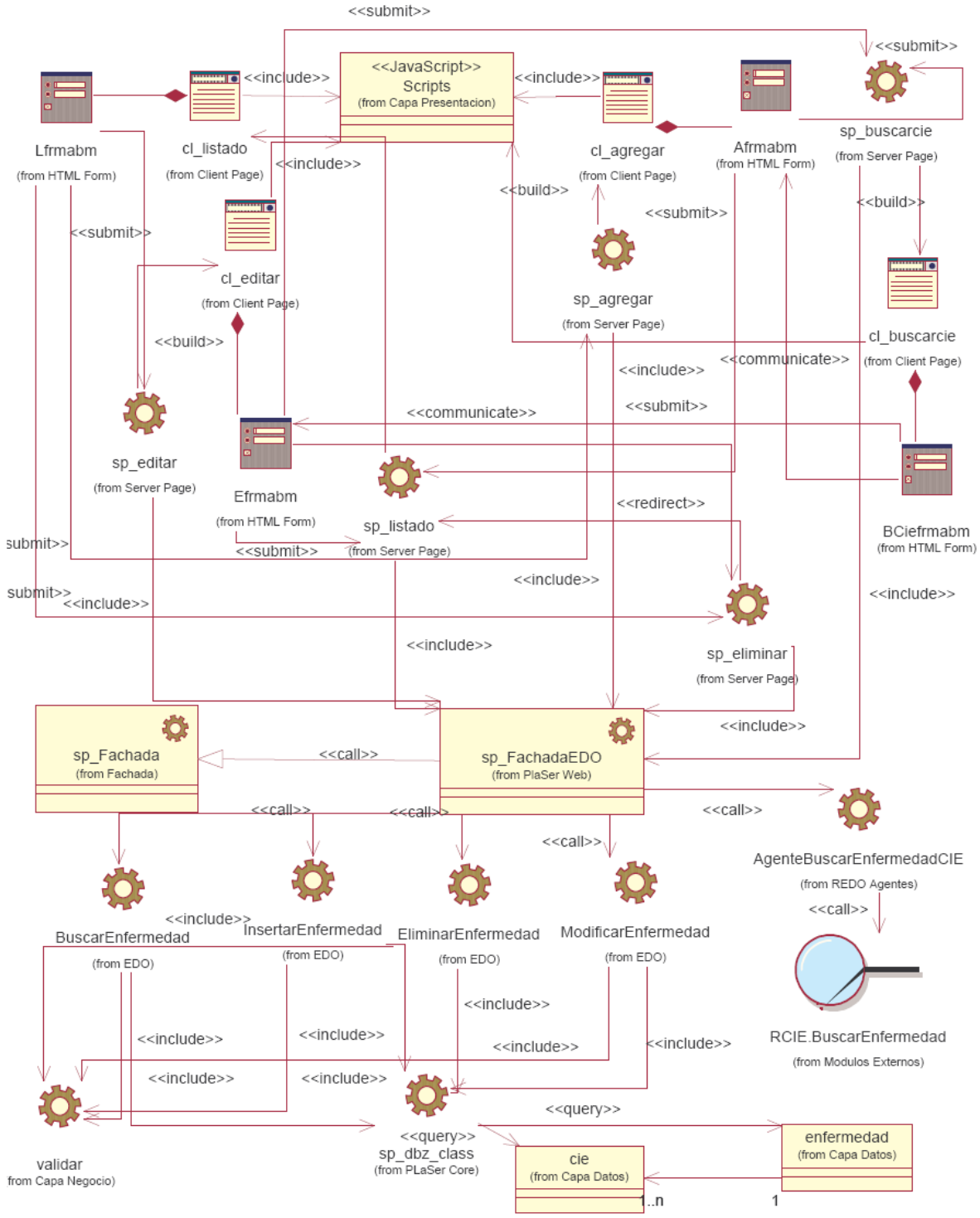


Figura 4. Diagrama de Clases del Caso de Uso "Gestionar Enfermedades"

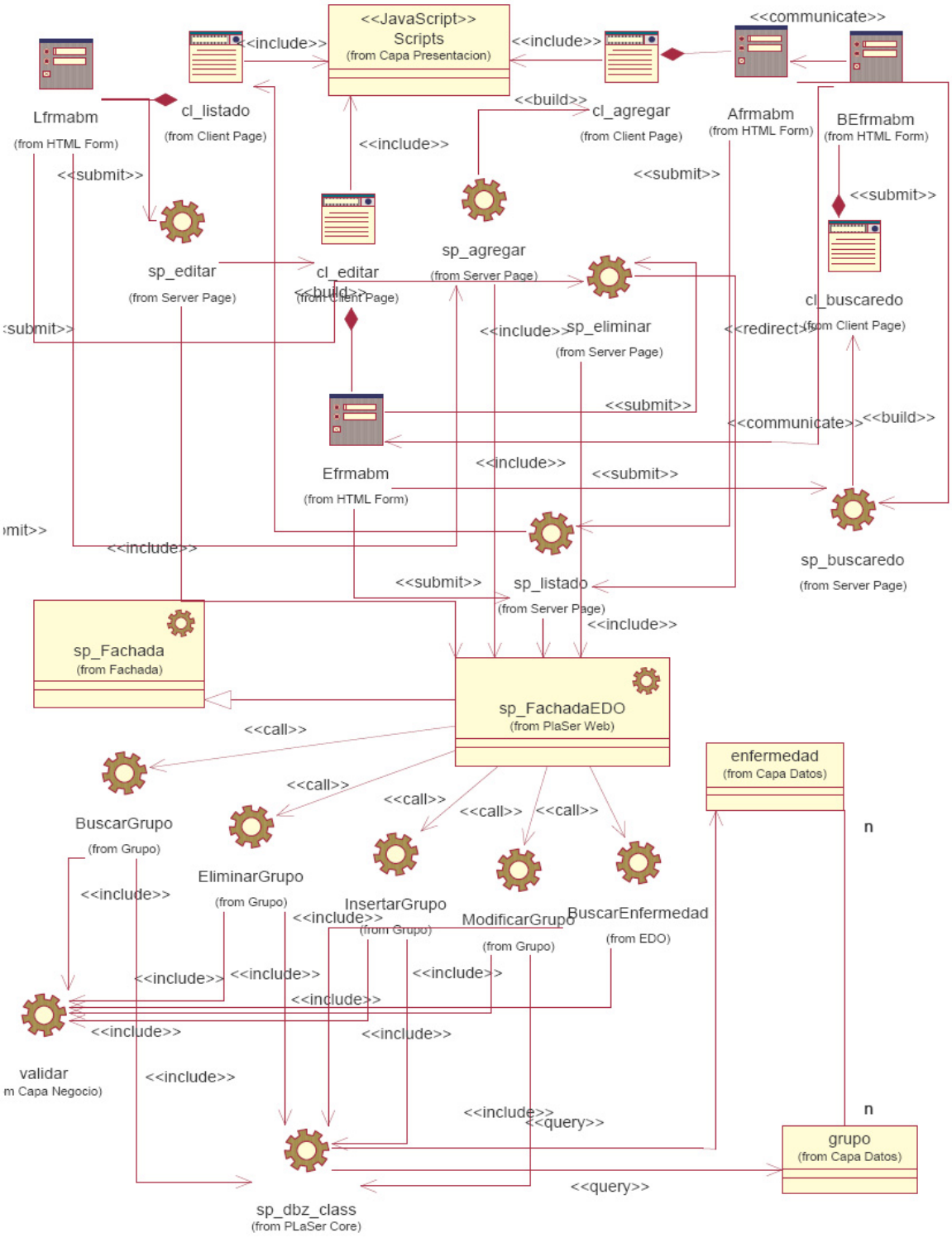


Figura 5. Diagrama de Clases del Caso de Uso "Gestionar Grupos"

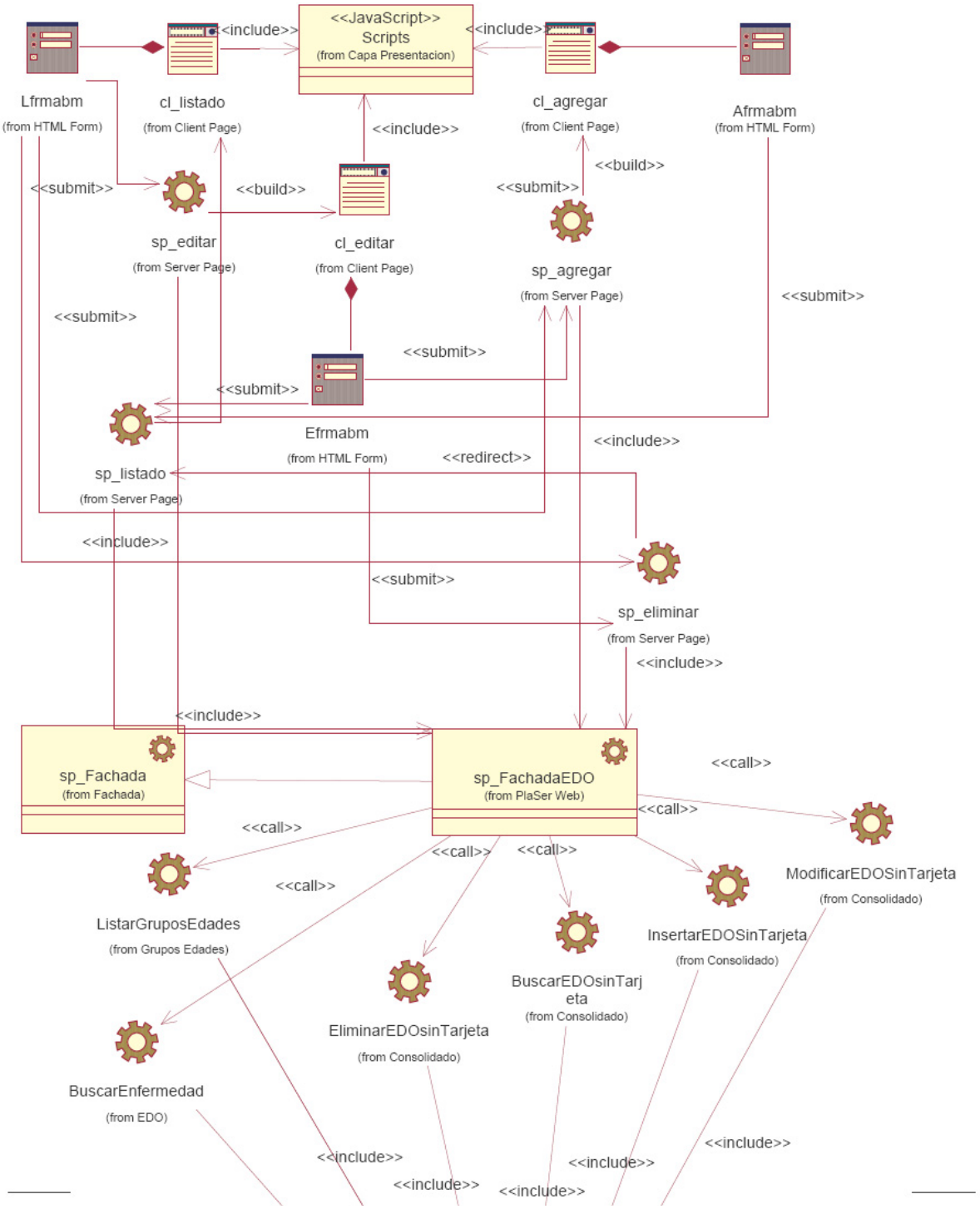


Figura 6.1. Diagrama de Clases del Caso de Uso "Gestionar Consolidado" (Parte 1)

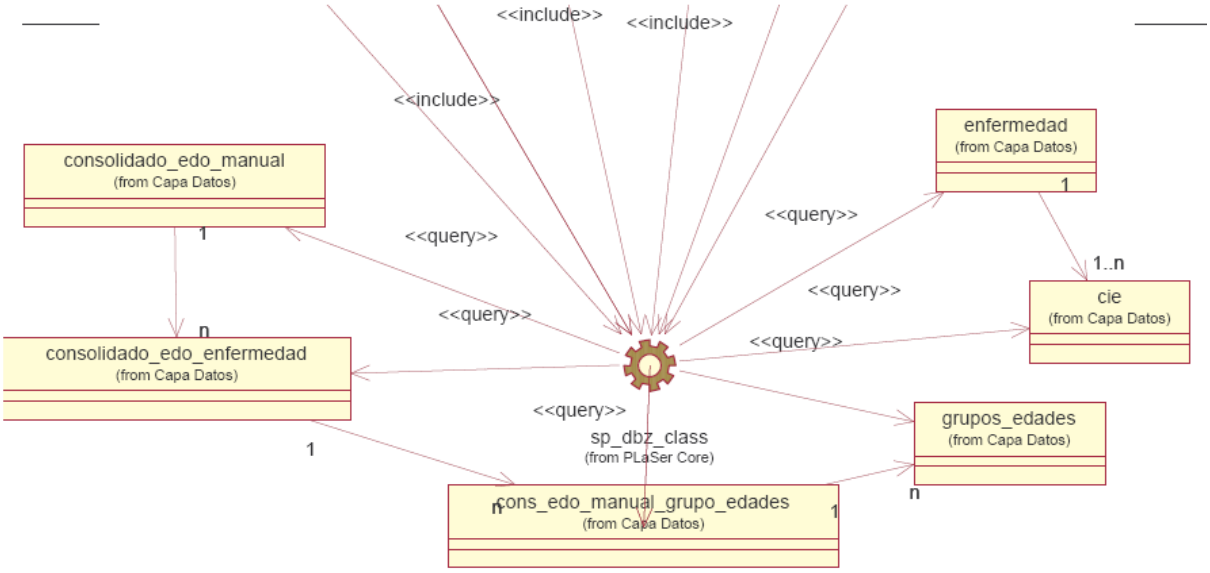


Figura 6.2. Diagrama de Clases del Caso de Uso "Gestionar Consolidado" (Parte 2)

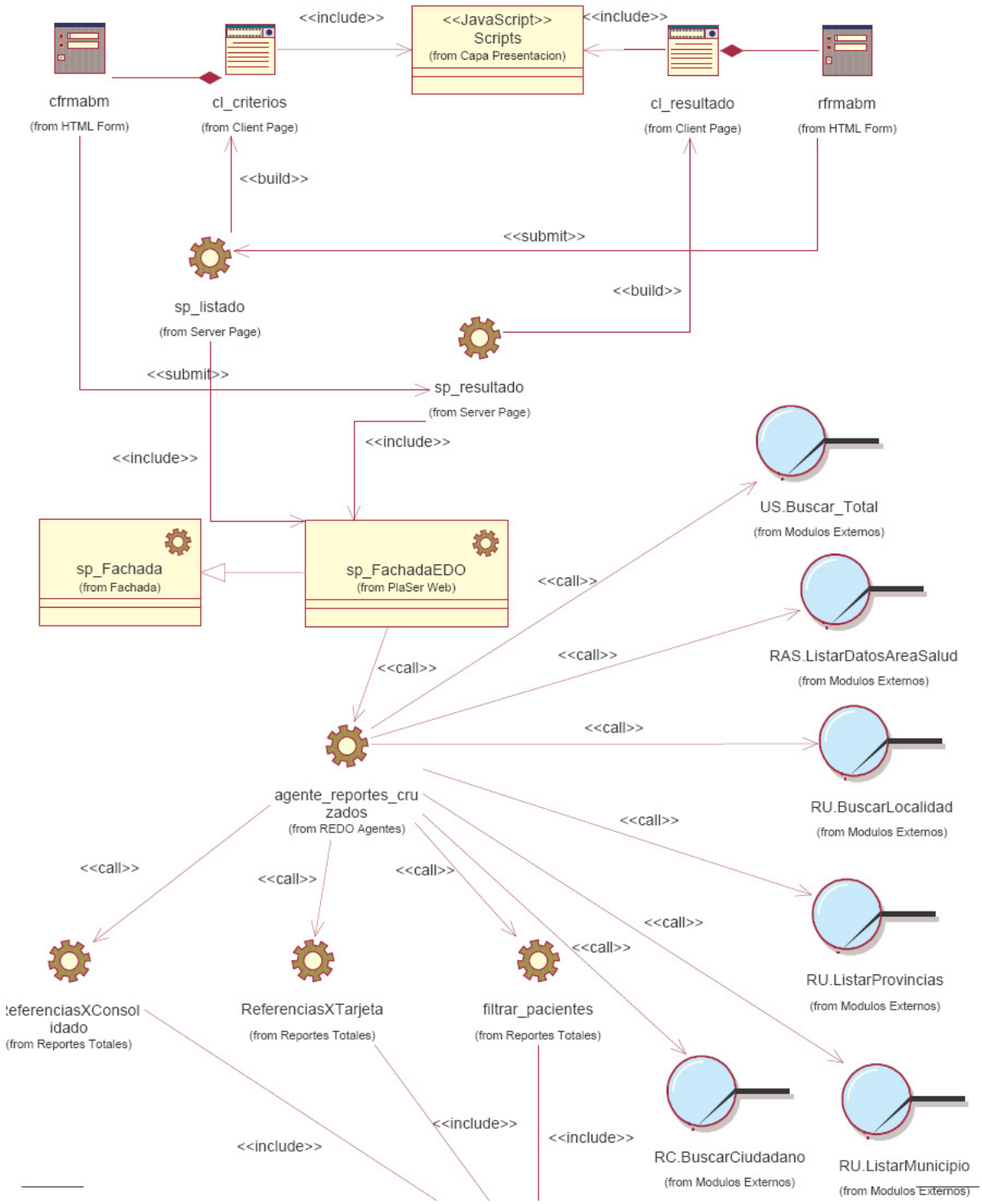


Figura 7.1. Diagrama de Clases del Caso de Uso "Reportes Totales" (Parte 1)

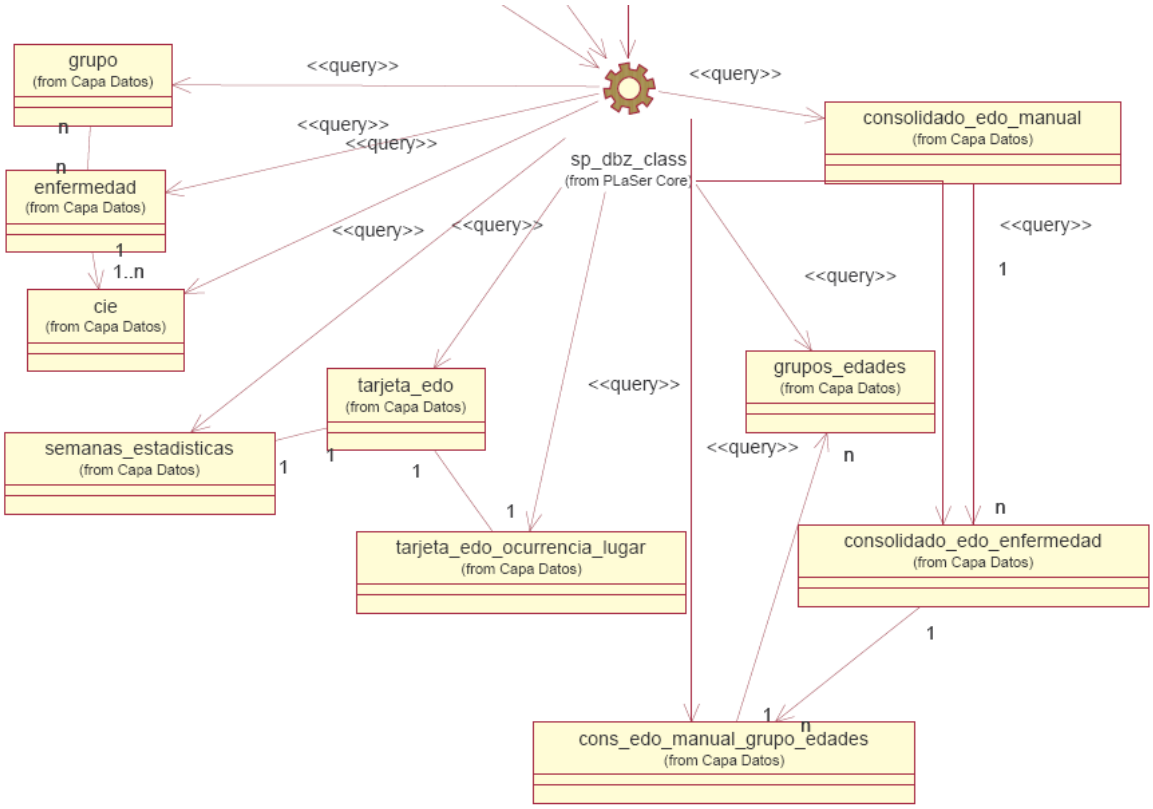


Figura 7.2. Diagrama de Clases del Caso de Uso "Reportes Totales" (Parte 2)

En este último diagrama de clases del caso de uso Reportes Totales, están incluidos todos los posibles resultados a modo de escenarios, o sea se tuvieron en cuenta todos los reportes relacionados con el anuario que ofrece la aplicación.

2.1.4 Descripción de las clases y atributos

Páginas Clientes (*Client Page*)

Nombre: cl_listado
Tipo de Clase: Client Page
Descripción General
Es una página Web que se ejecuta del lado del cliente que permite visualizar los datos generales de los elementos listados en su totalidad o sólo los que generan una búsqueda. Los mismos se presentan al usuario organizados por páginas, en su mayoría de 10 elementos. A partir de esta página, y en dependencia de los derechos con los que cuenta, el usuario puede acceder a las páginas clientes de

cl_agregar o cl_editar, incluso puede eliminar información desde esta página si tiene permisos de editor.
Casos de Uso que involucran a la clase:
✓ Gestionar Enfermedades. ✓ Gestionar Grupos. ✓ Configurar Circunstancias. ✓ Gestionar Consolidado.

Tabla 1. Descripción de la clase del diseño *cl_listado.*

Nombre: cl_agregar
Tipo de Clase: Client Page
Descripción General
Esta es otra página cliente al igual que listado, pero su funcionalidad es la de introducir o seleccionar los datos de un nuevo elemento a crear en una tabla. Claramente esta clase sólo está accesible con los permisos adecuados de usuario. A partir de esta página el sistema envía al usuario a la página cl_listado para que corrobore su acción.
Casos de Uso que involucran a la clase:
✓ Gestionar Enfermedades. ✓ Gestionar Grupos. ✓ Configurar Circunstancias. ✓ Gestionar Consolidado.

Tabla 2. Descripción de la clase del diseño *cl_agregar.*

Nombre: cl_editar
Tipo de Clase: Client Page
Descripción General
A esta página cliente se accede a través de cl_listado y luego de mostrar la información asociada al elemento seleccionado permite modificar los datos de éste. La página cl_editar muestra datos con un nivel de especificidad mayor que el de cl_listado. El usuario con permisos de visualizador tendrá acceso a esta página, pero se verá imposibilitado de modificar la información del elemento que aparezca en pantalla.
Casos de Uso que involucran a la clase:
✓ Gestionar Enfermedades. ✓ Gestionar Grupos. ✓ Configurar Circunstancias.

✓ Gestionar Consolidado.

Tabla 3. Descripción de la clase del diseño *cl_editar.*

Nombre: cl_criterios
Tipo de Clase: Client Page
Descripción General
Esta página que se ejecuta del lado del cliente es la encargada de recepcionar los datos por los cuales el usuario desea que se le genere el reporte de manera dinámica.
Casos de Uso que involucran a la clase:
• Reportes Totales

Tabla 4. Descripción de la clase del diseño *cl_criterios.*

Nombre: cl_resultado
Tipo de Clase: Client Page
Descripción General
Esta página recibe los datos del reporte generado dinámicamente mediante los datos seleccionados en cl_criterios y los organiza en una tabla de resultados. Esta tabla varía su estructura en dependencia de la selección hecha anteriormente. Desde esta página se puede regresar a la página anterior en caso de que se desee cambiar los parámetros por lo que se genera el reporte.
Casos de Uso que involucran a la clase:
• Reportes Totales

Tabla 5. Descripción de la clase del diseño *cl_resultado.*

Páginas Servidoras (Server Page) de la Capa de Presentación

Nombre: sp_listado
Tipo de Clase: Server Page
Descripción General
Esta clase se ejecuta del lado servidor y se encarga de construir la página XHTML cl_listado tras aplicar un documento XSLT llamado listado.xsl, al documento XML que contiene los datos. En caso de que el resultado a generar sea una búsqueda, primero se reciben y se procesan los datos para luego formar el XML y el proceso de transformación.
Casos de Uso que involucran a la clase:
✓ Gestionar Enfermedades.
✓ Gestionar Grupos.
✓ Configurar Circunstancias.

✓ Gestionar Consolidado.

Tabla 6. Descripción de la clase del diseño *sp_listado.*

Nombre: sp_agregar
Tipo de Clase: Server Page
Descripción General
Otra página ejecutada desde el servidor es la clase sp_agregar. Construye la página cliente cl_agregar mediante un XSLT llamado agregar.xsl. Esta página genera los datos necesarios en caso de que haya información que se manipule mediante selección y no por introducción de los datos.
Casos de Uso que involucran a la clase:
✓ Gestionar Enfermedades. ✓ Gestionar Grupos. ✓ Configurar Circunstancias. ✓ Gestionar Consolidado.

Tabla 7. Descripción de la clase del diseño *sp_agregar.*

Nombre: sp_editar
Tipo de Clase: Server Page
Descripción General
Esta clase se ejecuta del lado servidor y su acción fundamental es la de construir la página cl_editar. Para esta acción utiliza un XSLT llamado editar.xsl para transformar el XML que contiene los datos del elemento a modificar en una página XHTML.
Casos de Uso que involucran a la clase:
✓ Gestionar Enfermedades. ✓ Gestionar Grupos. ✓ Configurar Circunstancias. ✓ Gestionar Consolidado.

Tabla 8. Descripción de la clase del diseño *sp_editar.*

Nombre: sp_eliminar
Tipo de Clase: Server Page
Descripción General

Se ejecuta del lado del servidor y se encarga de enviar a la capa inferior los datos del elemento que se debe eliminar. Esta página no construye ningún cliente, sino que luego de ejecutar la acción, refresca la pantalla desde la cual se eliminó, puede ser desde cl_agregar o cl_listado. Se comunica con los métodos de eliminación mediante la clase sp_FachadaEDO.
Casos de Uso que involucran a la clase:
✓ Gestionar Enfermedades. ✓ Gestionar Grupos. ✓ Gestionar Consolidado.

Tabla 9. Descripción de la clase del diseño *sp_eliminar.*

Nombre: sp_criterios
Tipo de Clase: Server Page
Descripción General
Se ejecuta del lado del servidor y se encarga de construir la página cl_criterios mediante un fichero XSLT con nombre criterios.xsl. Llama a los métodos del negocio y agentes necesario para conformar el arreglo de datos a mostrar como elementos de selección obtenidos en la página cliente.
Casos de Uso que involucran a la clase:
• Reportes Totales.

Tabla 10. Descripción de la clase del diseño *sp_criterios*

Nombre: sp_resultado
Tipo de Clase: Server Page
Descripción General
Desde el lado del servidor, esta página a través de sp_FachadaEDO, llama al agente necesario para conformar el XML resultado de los parámetros introducidos en el formulario cl_criterios. A partir de esta respuesta construye una página XHTML cl_resultado utilizando el XSLT resultado.xsl y el XML que creó anteriormente.
Casos de Uso que involucran a la clase:
• Reportes Totales.

Tabla 11. Descripción de la clase del diseño *sp_resultado.*

Otras Páginas de uso general

Nombre: Scripts
Tipo de Clase: Fichero

Descripción General
Esta clase es un fichero que contiene código JavaScript interpretado por cualquier navegador y que corre del lado del cliente. Proporciona funciones para la validación de datos de entrada y para la dinámica en las páginas web clientes en que se muestran los datos. Otras de estas funciones nos permiten interactuar con páginas servidoras.

Tabla 12. Descripción de la clase del diseño *Scripts.*

Nombre: sp_Fachada
Tipo de Clase: Server Page
Descripción General
Se encuentra dentro de PlaSer y es la típica fachada que simplifica la comunicación de sistemas complicados y brinda los puntos de acceso entre estos. Se utiliza para separar la Capa de Presentación de la capa de acceso a datos.

Tabla 13. Descripción de la clase del diseño *sp_Fachada.*

Nombre: sp_FachadaEDO
Tipo de Clase: Server Page
Descripción General
Es la representante de sp_Fachada dentro del módulo y hereda de ésta. Cada uno de los módulos contiene la suya. Esta clase es en sí la que modela la comunicación con la Capa de Negocio.

Tabla 14. Descripción de la clase del diseño *sp_FachadaEDO.*

Nombre: sp_dbz_class
Tipo de Clase: Server Page
Descripción General
Se encuentra en la capa de negocio de todos los módulos y es utilizada por todos los métodos de esta capa que interactúan con la de datos. Crea la conexión a la base de datos MySQL mediante el módulo dbx de PlaSer. Su funcionamiento se basa en crear un objeto de conexión que tiene las funcionalidades necesarias para el trabajo con los datos, como puede ser filtrar resultados, insertar, actualizar o eliminar datos.

Tabla 15. Descripción de la clase del diseño *sp_dbz_class.*

Páginas Servidoras (Server Page) de la Capa de Negocio

Nombre: ListarCircunstanciasRelevantes

Tipo de Clase: Server Page
Descripción General
La clase ListarCircunstanciasRelevantes es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Su actividad es devolverle al usuario un listado de las circunstancias relevantes según los parámetros solicitados.

Tabla 16. Descripción de la clase del diseño *ListarCircunstanciasRelevantes.*

Nombre: InsertarCircunstanciasRelevantes
Tipo de Clase: Server Page
Descripción General
La clase InsertarCircunstanciasRelevantes es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Su actividad es insertar en la base de datos los parámetros que recibe de la Capa de Presentación.

Tabla 17. Descripción de la clase del diseño *InsertarCircunstanciasReleventes.*

Nombre: ModificarCircunstanciasRelevantes
Tipo de Clase: Server Page
Descripción General
La clase ModificarCircunstanciasRelevantes es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Su actividad es modificar en la base de datos los parámetros que recibe de la Capa de Presentación.

Tabla 18. Descripción de la clase del diseño *ModificarCircunstanciasReleventes.*

Nombre: EliminarCircunstanciasRelevantes
Tipo de Clase: Server Page
Descripción General
La clase EliminarCircunstanciasRelevantes es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Su actividad es eliminar la circunstancia relevante cuyo identificador se recibe de Capa de Presentación.

Tabla 19. Descripción de la clase del diseño *EliminarCircunstanciasReleventes.*

Nombre: BuscarEnfermedad
Tipo de Clase: Server Page
Descripción General

La clase BuscarEnfermedad es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Tiene la responsabilidad de traer de la base de datos las enfermedades EDO que se soliciten de la Capa de Presentación.

Tabla 20. Descripción de la clase del diseño *BuscarEnfermedad.*

Nombre: InsertarEnfermedad
Tipo de Clase: Server Page
Descripción General
La clase InsertarEnfermedad es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Tiene la responsabilidad de insertar en la base de datos las enfermedades EDO que se envíen de la Capa de Presentación.

Tabla 21. Descripción de la clase del diseño *InsertarEnfermedad.*

Nombre: ModificarEnfermedad
Tipo de Clase: Server Page
Descripción General
La clase ModificarEnfermedad es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Tiene la responsabilidad de cambiar en la base de datos los elementos de las enfermedades EDO que se envíen de la Capa de Presentación.

Tabla 22. Descripción de la clase del diseño *ModificarEnfermedad.*

Nombre: EliminarEnfermedad
Tipo de Clase: Server Page
Descripción General
La clase EliminarEnfermedad es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Tiene la responsabilidad de eliminar en la base de datos los elementos de las enfermedades EDO cuyos identificadores se envíen de la Capa de Presentación.

Tabla 23. Descripción de la clase del diseño *EliminarEnfermedad.*

Nombre: BuscarGrupo
Tipo de Clase: Server Page
Descripción General
La clase BuscarGrupo es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Su función es listar todos los grupos que se soliciten por el usuario mediante la Capa de Presentación.

Tabla 24. Descripción de la clase del diseño *BuscarGrupo.*

Nombre: InsertarGrupo
Tipo de Clase: Server Page
Descripción General
La clase InsertarGrupo es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Su función es insertar todos los grupos que se soliciten por el usuario mediante la Capa de Presentación.

Tabla 25. Descripción de la clase del diseño *InsertarGrupo.*

Nombre: ModificarGrupo
Tipo de Clase: Server Page
Descripción General
La clase ModificarGrupo es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Su función es modificar los datos de los grupos que se soliciten por el usuario mediante la Capa de Presentación.

Tabla 26. Descripción de la clase del diseño *ModificarGrupo.*

Nombre: EliminarGrupo
Tipo de Clase: Server Page
Descripción General
La clase EliminarGrupo es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Su función es eliminar los grupos cuyos identificadores se envíen por el usuario de la Capa de Presentación.

Tabla 27. Descripción de la clase del diseño *EliminarGrupo.*

Nombre: ListarGruposEdades
Tipo de Clase: Server Page
Descripción General
La clase ListarGruposEdades es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Se encarga de listar todos los grupos de edades que aparecen en la base de datos.

Tabla 28. Descripción de la clase del diseño *ListarGruposEdades.*

Nombre: BuscarEDOsinTarjeta
Tipo de Clase: Server Page
Descripción General

La clase BuscarEDOsinTarjeta es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Se encarga de traer de la base de datos todos los consolidados cuyos datos coinciden con los introducidos por el usuario en la Capa de Presentación.

Tabla 29. Descripción de la clase del diseño *BuscarEDOsinTarjeta.*

Nombre: InsertarEDOsinTarjeta
Tipo de Clase: Server Page
Descripción General
La clase InsertarEDOsinTarjeta es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Se encarga de introducir en la base de datos todos los consolidados cuyos datos se pasan de la Capa de Presentación.

Tabla 30. Descripción de la clase del diseño *InsertarEDOsinTarjeta.*

Nombre: ModificarEDOsinTarjeta
Tipo de Clase: Server Page
Descripción General
La clase ModificarEDOsinTarjeta es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Se encarga de cambiar los elementos del consolidado que se envían desde la Capa de Presentación.

Tabla 31. Descripción de la clase del diseño *ModificarEDOsinTarjeta.*

Nombre: EliminarEDOsinTarjeta
Tipo de Clase: Server Page
Descripción General
La clase EliminarEDOsinTarjeta es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Se encarga de eliminar los consolidados que el usuario decida y de los cuales pasa a través de presentación sus identificadores.

Tabla 32. Descripción de la clase del diseño *EliminarEDOsinTarjeta.*

Nombre: ReferenciasXTarjeta
Tipo de Clase: Server Page
Descripción General
La clase ReferenciasXTarjeta es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Se configurar el reporte de las tarjetas de acuerdo a los parámetros introducidos por el usuario en la

Capa de Presentación y devolver el SOAP de resultado.

Tabla 33. Descripción de la clase del diseño *ReferenciasXTarjeta.*

Nombre: ReferenciasXConsolidado
Tipo de Clase: Server Page
Descripción General
La clase ReferenciasXConsolidado es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Se configurar el reporte de los consolidados de acuerdo a los parámetros introducidos por el usuario en la Capa de Presentación y devolver el SOAP de resultado.

Tabla 34. Descripción de la clase del diseño *ReferenciasXConsolidado.*

Nombre: filtrar_pacientes
Tipo de Clase: Server Page
Descripción General
La clase filtrar_pacientes es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Como bien dice su nombre se utiliza para devolver los identificadores de los pacientes que se encuentran en la base de datos para así reducir el dominio de la búsqueda sólo a los que tengan EDO específicas o que pertenezcan a lugares determinados.

Tabla 35. Descripción de la clase del diseño *filtrar_pacientes.*

Nombre: AgenteBuscarEnfermedadCIE
Tipo de Clase: Server Page
Descripción General
La clase AgenteBuscarEnfermedadCIE es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Se encarga de consumir el servicio de un sistema externo, en este caso RCIE, para traer las enfermedades del Clasificador Internacional de Enfermedades que se asociarán a las EDO.

Tabla 36. Descripción de la clase del diseño *AgenteBuscarEnfermedadCIE.*

Nombre: agente_reportes_cruzados
Tipo de Clase: Server Page
Descripción General
La clase agente_reportescruzados es una clase que se ejecuta del lado del servidor en la Capa de Negocio. Este agente se utiliza en la generación de los reportes, a la hora de llevar los datos a las existencias en la base de datos de EDO.

Tabla 37. Descripción de la clase del diseño *agente_reportes_cruzados.*

Clases Entidades

Nombre: enfermedad
Tipo de Clase: Entidad
Descripción General
Almacena las enfermedades EDO definidas por el SNS como de seguimiento.

Tabla 38. Descripción de la clase del diseño *enfermedad.*

Nombre: grupo
Tipo de Clase: Entidad
Descripción General
Almacena los grupos de EDO.

Tabla 39. Descripción de la clase del diseño *grupo.*

Nombre: cie
Tipo de Clase: Entidad
Descripción General
Almacena los datos de las enfermedades CIE que componen las EDO.

Tabla 40. Descripción de la clase del diseño *cie.*

Nombre: circunstancias_relevantes
Tipo de Clase: Entidad
Descripción General
Almacena las circunstancias relevantes a tener en cuenta durante la creación de una tarjeta.

Tabla 41. Descripción de la clase del diseño *circunstancias_relevantes.*

Nombre: grupos_edades
Tipo de Clase: Entidad
Descripción General
Almacena los grupos de edades que definen los usuarios.

Tabla 42. Descripción de la clase del diseño *grupos_edades.*

Nombre: consolidado_edo_manual
Tipo de Clase: Entidad
Descripción General

Almacena los datos de los casos de enfermedades EDO sin tarjeta.

Tabla 43. Descripción de la clase del diseño *consolidado_edo_manual.*

Nombre: consolidado_edo_enfermedad
Tipo de Clase: Entidad
Descripción General
Almacena las enfermedades de un consolidado.

Tabla 44. Descripción de la clase del diseño *consolidado_edo_enfermedad.*

Nombre: cons_edo_manual_grupo_edades
Tipo de Clase: Entidad
Descripción General
Almacena los grupos de edades a los que se le diagnosticó una enfermedad en un consolidado.

Tabla 45. Descripción de la clase del diseño *cons_edo_manual_grupo_edades.*

Conclusiones

En este capítulo se planteó el modelado del flujo de trabajo de diseño del sistema como continuación del análisis, basado en la propuesta del sistema hecha por el analista. Se modeló utilizando estereotipos Web, lo cual ha dado una mejor perspectiva y entendimiento de los diagramas antes presentados, para los pasos siguientes correspondientes a la implementación. Se presentaron diagramas de clases del diseño y la descripción de las clases utilizadas en los mismos, correspondientes a los casos de uso que se explican en el documento.

Capítulo 3. Implementación

Introducción

En este capitulo se le da continuación al flujo de trabajo de diseño. Se plantean las estrategias de integración con otros módulos del SISalud. También se presentan los diagramas de despliegue y componentes, los cuales permiten tener una mejor comprensión del sistema implementado. De la misma manera se describen algunos de los algoritmos más complejos que intervienen en la funcionalidad del módulo Registro de EDO, en especial agentes y métodos que se utilizarán para generar los reportes estadísticos.

3.1 Fundamentación de integración con otros sistemas

El Registro de Enfermedades de Declaración Obligatoria tiene relación con otros módulos dentro del Sistema de Información para la Salud (SISalud). Una de las estrategias de esta aplicación es la integración y relación con otros módulos y aplicaciones existentes para evitar la duplicación de la información mediante la reutilización de los datos ya existentes en otros módulos y que conciernen más específicamente a su negocio. Esta estrategia se ejemplifica en la obtención y emisión de información necesitada por estos módulos. A continuación se relacionan algunas de las aplicaciones de las cuales el sistema consume servicios.

SAAA (Single Authorization, Authentication and Account).
Este módulo es el encargado, como bien refiere su nombre, de controlar la autenticación de los usuarios mediante un usuario único y su password en el sistema, dando un error de acceso a las personas que intenten entrar al sistema sin estar aún registrados. En el momento en que un usuario accede a la aplicación se genera un certificado digital conformado por un identificador único o “token” de 32 caracteres, un identificador del usuario, el nivel de acceso (Nacional, Provincial, Municipal o Unidad de Salud), el identificador de nivel de acceso, así como un listado de los módulos a los cuales el usuario tiene acceso y que tipo de acceso es el que

posee en ellos, los tipos de acceso para el Registro de EDO pueden ser Editor, Visualizador, Médico o Técnico en Estadísticas.

En el módulo, se verifican los permisos a los que un usuario tiene derechos mediante el certificado digital que se creó a la hora de su entrada. Así también se controla si la persona es editor del sistema y tiene la posibilidad de configurar los datos del sistema, siempre teniendo en cuenta el nivel de acceso de la misma. Por otra parte, si el usuario tiene derechos de visualizador sólo podrá comportarse de manera pasiva frente a la información a la que tenga acceso dentro de Registro de EDO. Los otros tipos de usuarios, o sea, Médico y Técnico en Estadística, tendrán privilegios similares a los de editor, pero restringidos, en caso del primero sólo para las tarjetas, y para tarjetas y consolidados en el caso del último.

El sistema SAAA, controla todas las acciones que realizan los usuarios y en el momento exacto de tiempo en que estas fueron efectuadas o si se le denegó el acceso. La utilización de este módulo de seguridad permite una programación transparente en cuanto a términos de seguridad, ya que todo este negocio recae sobre el SAAA.

RUS (Registro de Unidades de Salud).

Este registro controla la información de todas las unidades de salud del país, así como toda su información relacionada. El Registro de EDO utiliza muy a menudo los servicios de búsqueda que brinda este módulo para localizar las unidades donde se genera la captación de las enfermedades.

Las búsquedas de las unidades de salud se pueden efectuar usando los servicios Web que ofrece el RUS. En el Registro de EDO estos servicios se consumen teniendo en cuenta el nivel en que se haya autenticado el usuario que se encuentra trabajando, sólo se localizan los datos del nivel correspondiente al usuario.

RPS (Registro de Personal de la Salud).

Este codificador maneja los datos del personal médico y no médico del país que se encuentre trabajando para el SNS. A la hora de captación de una EDO, es necesario tener en cuenta qué personal de la salud fue el que dio lugar a esta notificación. El RPS brinda servicios de búsqueda de la información que se procesa en él, y son utilizados por el Registro de EDO para tomar la información de quién generó la tarjeta o el consolidado.

RCIE (Registro de la Clasificación Internacional de Enfermedades y Problemas Relacionados con la Salud).
El negocio de este codificador trata todas las enfermedades que aparecen en el libro del CIE-10, almacenando los datos de estas y agrupándolas según criterios específicos. De este registro, se reciben las enfermedades CIE que serán relacionadas con una EDO específica para facilitar la creación de reportes estadísticos a organizaciones internacionales, ya que se estandariza la información a la nomenclatura internacional.

RPOB (Registro de Población).
Este registro gestiona las Historias de Salud Familiar de cada población del área de salud. Los datos del paciente al cual se le detecta una EDO, por ejemplo la historia clínica del paciente, se localizan mediante servicios de búsqueda que brinda este registro.

RC (Registro del Ciudadano).
Este registro es una simulación de lo que en un futuro será el codificador con la información personal de todo residente en la Isla. Este módulo se utiliza a la hora de generar reportes por edades o sexo de los pacientes a los que se les haya diagnosticado una EDO.

RAS (Registro de Áreas de Salud).
El RAS es el encargado de gestionar la información de la distribución de las áreas de salud del país. A la hora de captar una tarjeta o un consolidado es necesario tomar los datos del Equipo Básico de Salud (EBS), e incluso del Grupo Básico de Trabajo (GBT), del cual forma parte el personal que hace la notificación. De ahí la necesidad de utilizar los servicios de búsqueda de EBS y GBT que brinda este módulo.

3.2 Modelo de implementación

Mediante este modelo se implementa el sistema en términos de componentes, los cuales pueden ser ficheros de código fuente, scripts, ejecutables y similares. Este flujo de trabajo parte del diseño y describe como los elementos del modelo del diseño se implementan en términos de componentes.

Este modelo está fuertemente determinado por el lenguaje de programación elegido y está conformado por los diagramas de despliegue e implementación, los cuales indican cómo se construye y se organiza la aplicación, así como la dependencia que se establece entre los nodos físicos en que estará distribuida la aplicación.

3.2.1 Diagramas de Componentes

Es un diagrama que muestra un conjunto de elementos del modelo, tales como componentes, subsistemas de implementación y sus relaciones.

Estos diagramas se utilizan para modelar la vista estática de un sistema. Muestra la organización y las dependencias lógicas entre un conjunto de componentes software, sean éstos componentes de código fuente, librerías, binarios o ejecutables. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema.

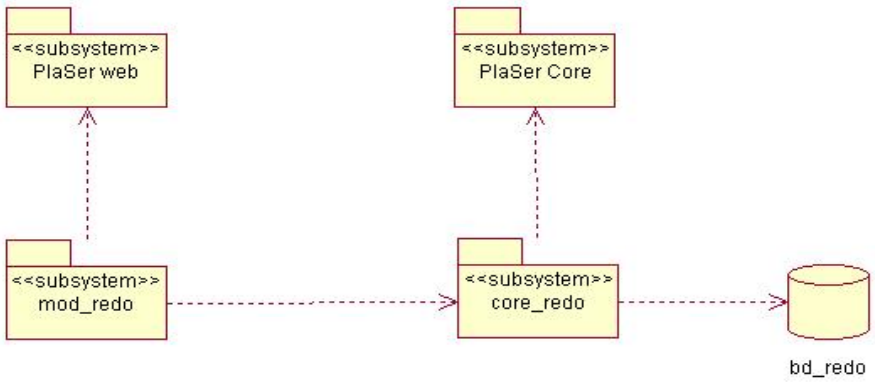


Figura 8. Diagrama General de Componentes de los Subsistemas de REDO

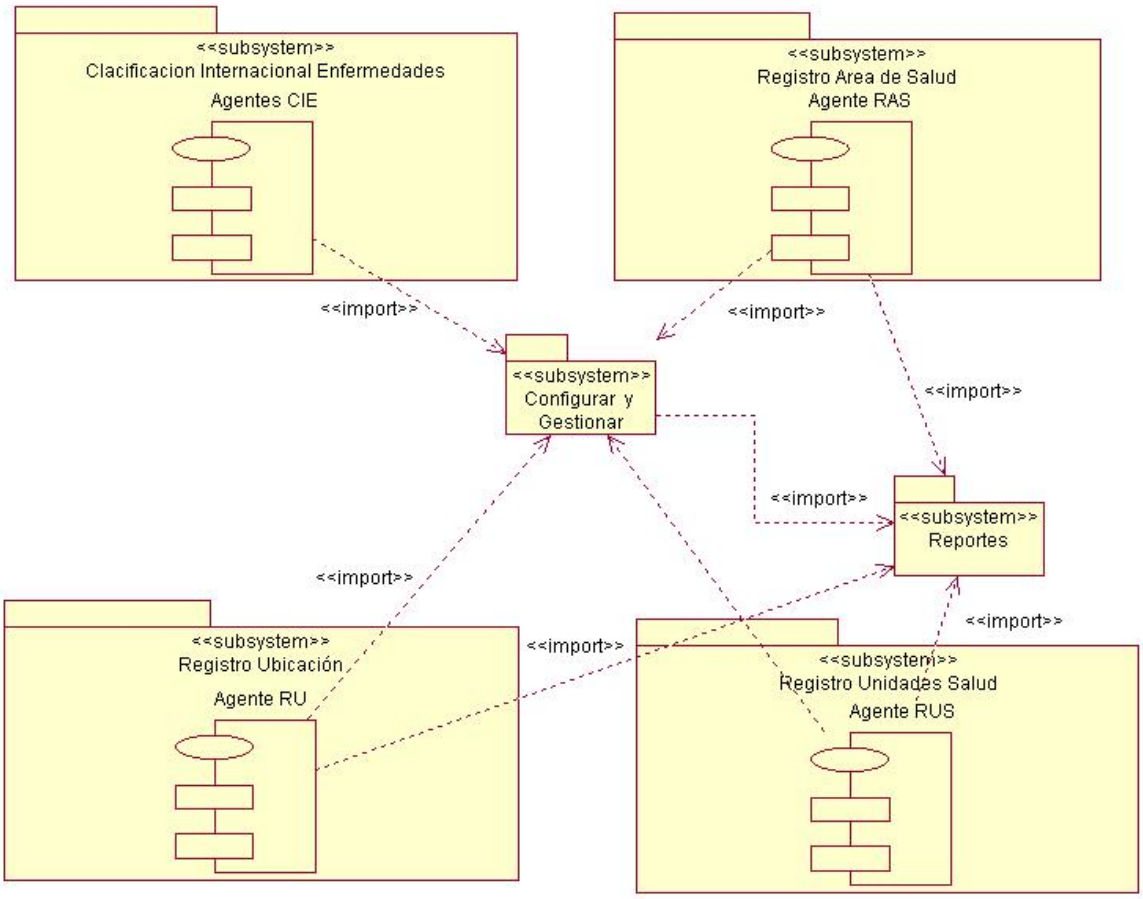


Figura 9. Diagrama de Componentes del Subsistema mod_redo

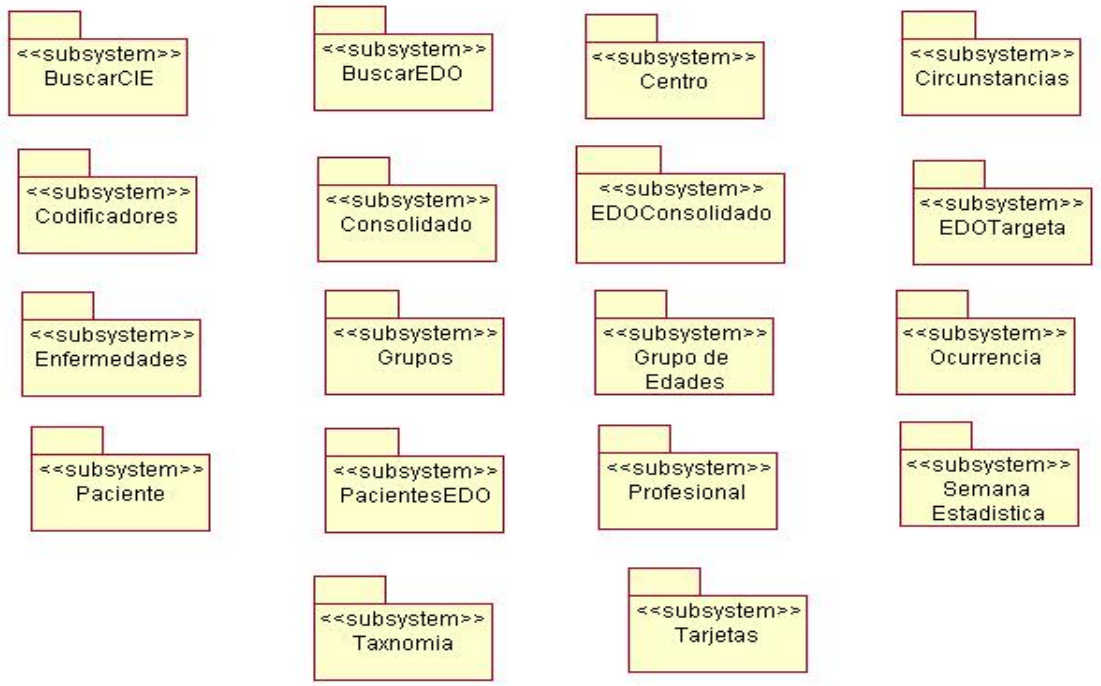


Figura 10. Diagrama de Componentes del Subsistema Gestionar y Configurar

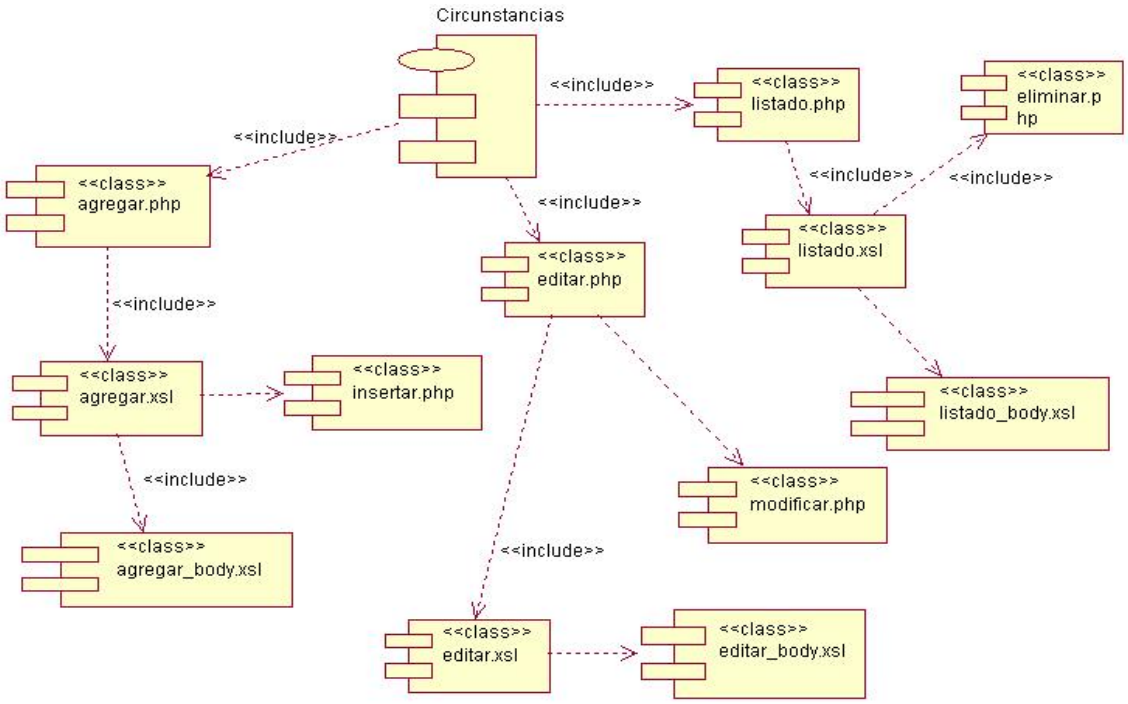


Figura 11. Diagrama de Componentes del Subsistema Configurar Circunstancias

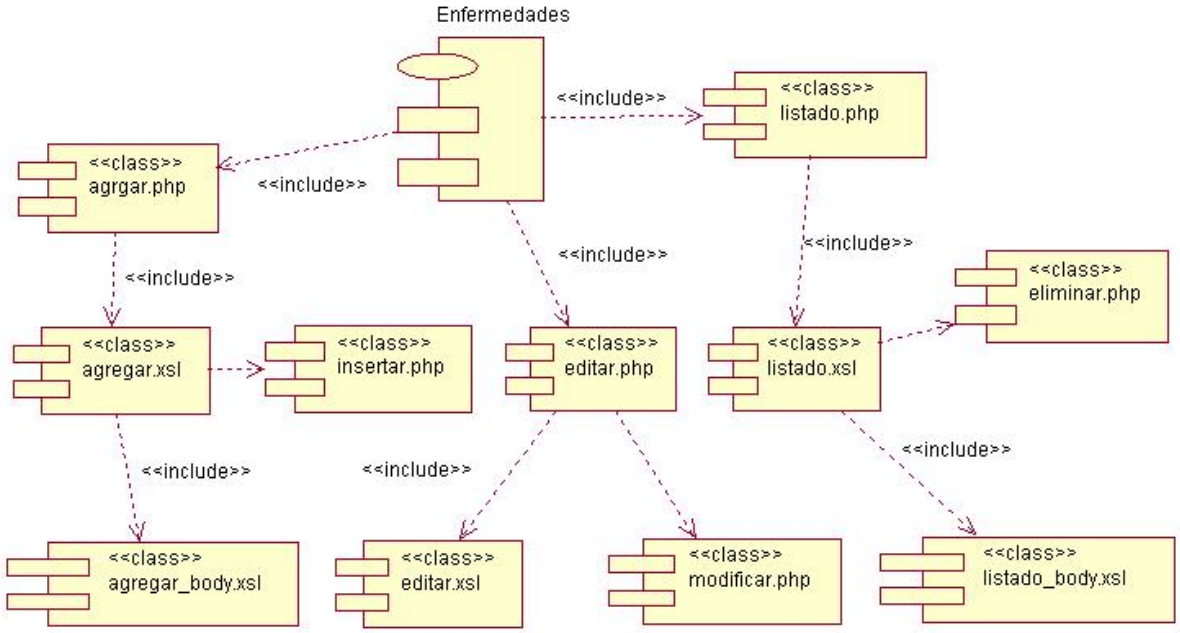


Figura 12. Diagrama de Componentes del Subsistema Gestionar Enfermedades

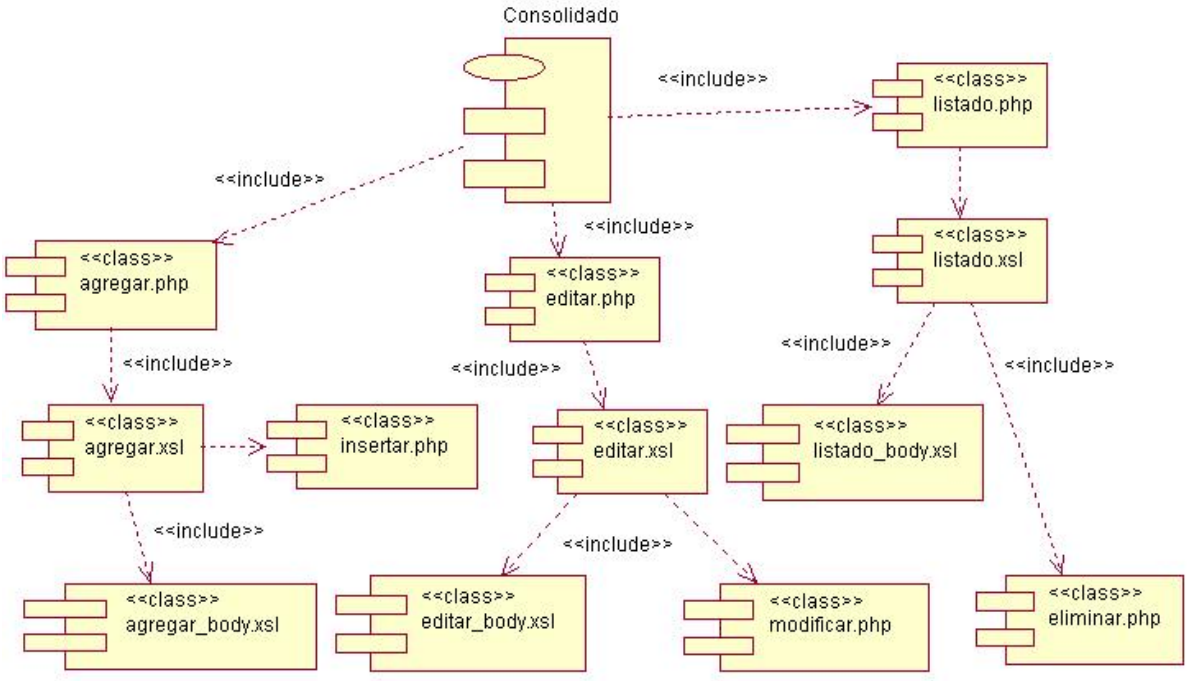


Figura 13. Diagrama de Componentes de Subsistema Gestionar Consolidado

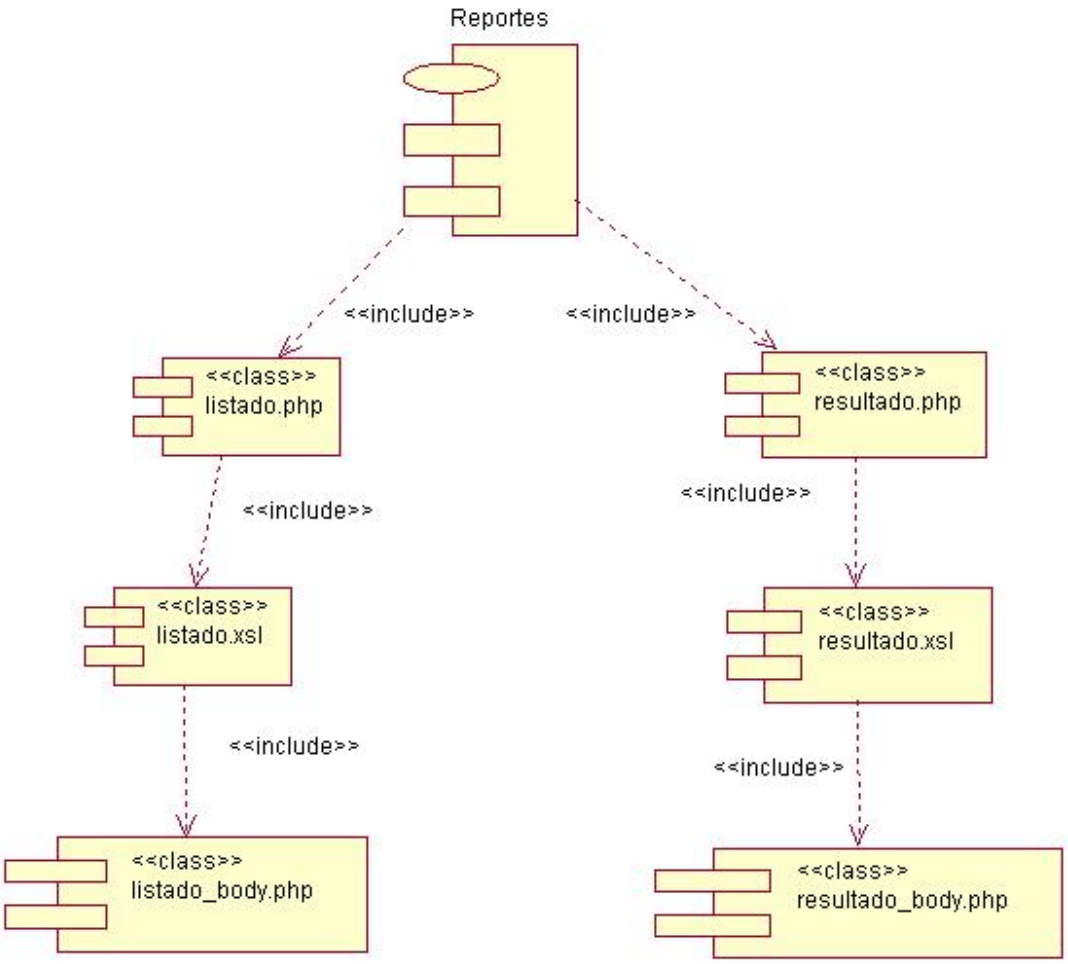


Figura 14. Diagrama de Componentes de Subsistema Reportes Totales

3.2.2 Diagrama de Despliegue

Estos diagramas describen cómo se relacionan los procesadores, dispositivos y componentes de hardware durante la ejecución del programa. Los nodos en el diagrama representan recursos computacionales que existen en tiempo de ejecución como pueden ser computadoras, procesadores, impresoras.

El modelo de despliegue se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño.

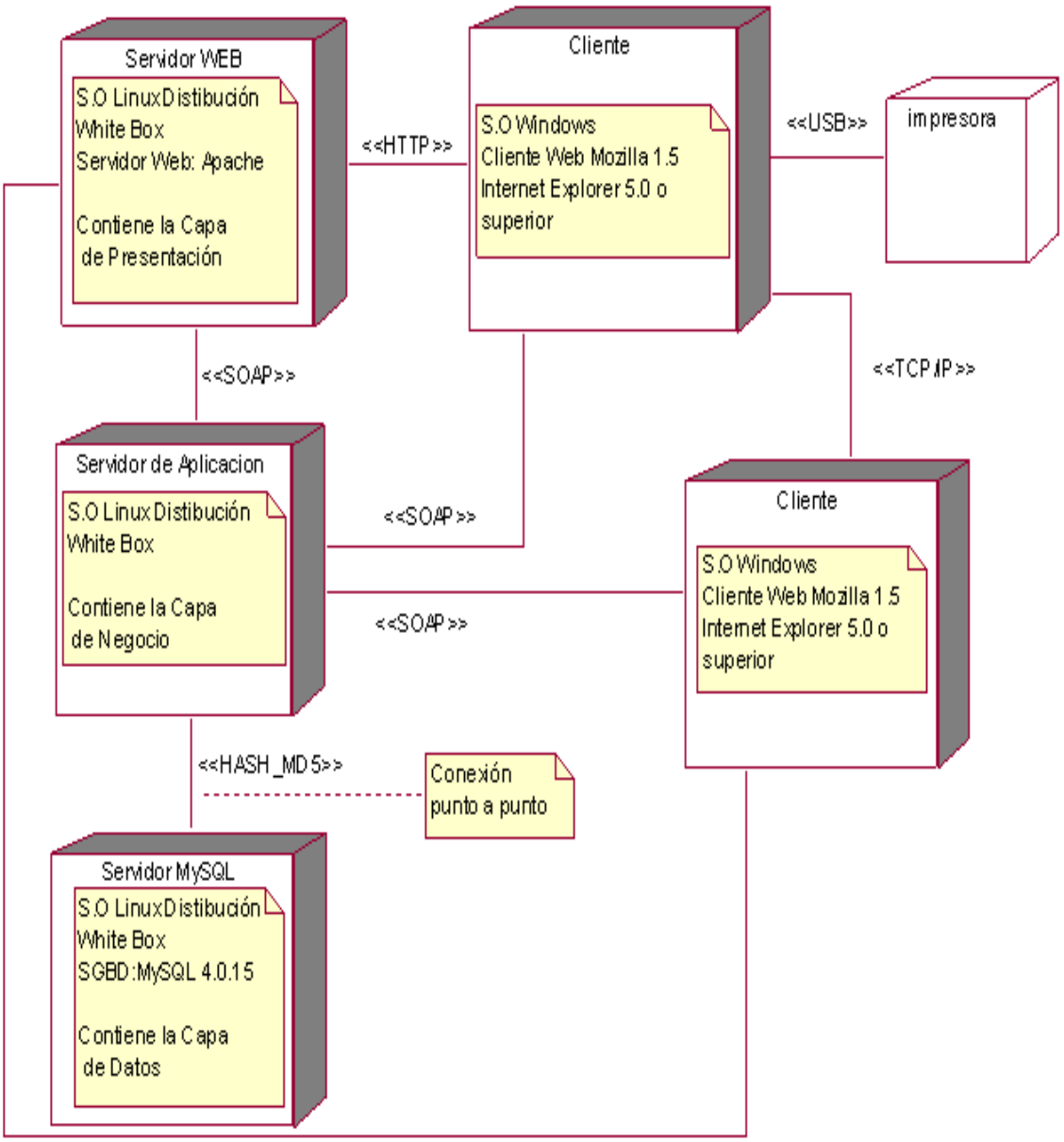


Figura 15. Diagrama de Despliegue de la aplicación

3.3 Descripción de métodos y agentes más complejos

Métodos del Negocio REDO

GenerarSemanaEstadistica

Este método genera las 52 semanas estadísticas para un año determinado. Recibe como parámetro el año del cual se desea crear las semanas estadísticas. Si no se encuentra en la base de datos ninguna semana de ese año se procede a crearlas. Para ello se crean dos arreglos, uno con las fechas de inicio y otro con las fechas de fin de las semanas respectivas.

Se toma como primera fecha de inicio el 1 de enero del año en cuestión y se procede a buscar que día es el que termina esa semana para introducirlo en el arreglo de fechas finales como el primer elemento. Esto se hace así porque la primera y última semana del año pueden tener de 4 a 11 días. A partir de aquí se comienza a sumar siete al número del día y a generar fechas de inicio y fin hasta la semana 51.

Como último paso de esta parte del proceso se toma como fecha de inicio de la semana 52 el día después de donde se detuvo el bucle anterior y como fecha de fin el 31 de diciembre de ese año.

Teniendo los dos arreglos de fechas, se procede a introducirlos en la base de datos mediante un ciclo, en el que se pasan las fechas de inicio y fin y el número de la semana en el año. Este número viene dado por la posición en el arreglo de la fecha (los índices de los arreglos comienzan por 1 hasta 52).

filtrar_pacientes

Este método devuelve el arreglo de id_paciente en dependencia de los parámetros por los cuales se quiera hacer el filtrado. Estos parámetros pueden ser: grupos de enfermedades, enfermedades EDO y/o enfermedades CIE en cuanto a enfermedades, y en cuanto al filtrado por lugar puede ser por unidades de salud, EBS o localidades.

El método construye la consulta en dependencia de los valores de filtrado anteriormente expuestos y devuelve el arreglo de los pacientes que se corresponden con los parámetros de entrada.

ReferenciasXTarjeta y ReferenciasXConsolidado

Ambos métodos son utilizados en los reportes cruzados a la hora de hacer las consultas en las bases de datos, aunque el primero las ejecuta en las tarjetas y el segundo en los consolidados.

Como parámetros de entrada se les pasan tres arreglos: las filas, las columnas y las variantes. Las primeras representan el valor por el cual se genera la tabla del reporte en las filas. El segundo qué columnas tiene la tabla de salida y por las que se efectúa el filtrado. Las últimas significan acotaciones adicionales que se le pueden hacer a las consultas, como una tercera dimensión, que no se representa en la tabla, pero sí están implícitas en el valor de resultado.

Los valores en cada uno de estos arreglos pueden ser:
En el caso de las filas.

- Enfermedad: en las cuales se pueden incluir grupos, enfermedades EDO y enfermedades CIE y sus posibles combinaciones.
- Lugar: comenzando por provincias hasta el nivel más bajo EBS.

En el caso de las columnas.

- Período: se puede expresar en semestres, meses y semanas estadísticas.
- Edad: se le pasan rangos de edades, o edades específicas. Este caso no se tiene en cuenta en el consolidado.
- Lugar: tiene las mismas posibilidades de parámetros que el lugar en las filas, pero de estar un arreglo, no está presente en los otros, ya que por ejemplo, no tiene sentido un reporte de municipios por municipios.

Las variantes incluyen todos los anteriores aunque, si están seleccionados como variantes no lo están en las filas y columnas y viceversa. Además se agregan los siguientes:

- Sexo: es el único que tiene representación en la salida como elemento de la tabla pero sólo cuando se elige la opción de desglosado, no así cuando se selecciona masculino o femenino específicamente.

- Tipo de enfermedad: aquí es donde se hace la distinción de cuál de los métodos es el que toma partido en el resultado, se puede elegir entre tarjeta o consolidado, o ninguno, en cuyo caso se muestra el resultado por ambos métodos.
- Tipo de diagnóstico: se deriva de Tipo de enfermedad en el caso de que se elija tarjeta, y puede seleccionarse entre presuntivo o confirmado.
- Tipo de notificación: se da en las mismas condiciones que el caso anterior y las posibilidades son ocurrencia o residencia.

Luego de que se tienen los parámetros se pasa a confeccionar dos arreglos por las filas. El primero contiene los valores de filtrado del tipo que se paso en las filas y el segundo las descripciones de éstos.

En las columnas la operación es similar, pero se va configurando el XML de las columnas con los valores de las descripciones.

Por último se tienen las variantes, las cuales se van agregando a una cadena que posteriormente se le agrega al “where” de la consulta a la vez que se va configurando una parte del SOAP de salida que almacena qué variantes se seleccionaron.

Con todas las condiciones creadas se pasa entonces a crear un arreglo con un número de elementos igual a la cantidad de valores en las filas x cantidad de valores en las columnas, el mismo contiene todas las posibilidades de consultas que se generan en las combinaciones de filas con columnas. A estos elementos del arreglo se les va agregando la cadena de filtrado que se construyó con las variantes.

El último paso es llevar a cabo las consultas e ir creando el SOAP de salida con los valores que devuelven éstas.

Agentes de REDO

Es necesario explicar que un agente es un método en el core o Capa de Negocio que se encarga de comunicarse con otros módulos y consumir los servicios necesarios para el funcionamiento del módulo al que pertenece. La razón de ser de los agentes es la de no

romper con el encapsulamiento de los módulos, ya que los métodos del negocio no se comunican directamente con el resto de los módulos.

Casi todos los agentes del Registro de EDO tienen funcionamiento similar, se pasan los parámetros necesarios al método de otro módulo que se invoca y se trae la respuesta de éste para el consumo dentro del negocio del Registro de EDO.

agente_reportes_cruzados

Este agente se utiliza en la generación de los reportes para llevar los parámetros seleccionados en la Capa de Presentación a los datos que existen en la base de datos de EDO, hacer el llamado a los métodos de REDO que construyen el reporte y luego conformar la salida.

Este proceso es necesario ya que los métodos **ReferenciasXTarjeta** y **ReferenciasXConsolidado** que se explicaron con antelación, sólo se comunican con la base de datos por cuestiones de acoplamiento y en esta no existen muchos de los parámetros que se introducen como criterios para generar los reportes, por ejemplo idprovincia e idmunicipio. El primer paso es transformar los datos, se verifica si las filas, columnas o variantes contienen datos de lugar, si estos son de tipo provincia o municipio se llevan a unidades de salud que es lo que contienen la tabla tb_tarjeta_edo y tb_consolidado_edo_manual.

Para esto se hace uso del método Buscar_Total.php del Registro de Unidades de Salud (RUS). El otro caso posible es que se introduzcan GBT como datos de lugar para lo cual se hace un llamado al método ListaDatosAreaSalud.php del Registro de Áreas de Salud (RAS). En el caso de que se seleccione ocurrencia como tipo de notificación, los datos que se introducen por las provincias o los municipios se deben relacionar con las localidades de la tabla de tb_tarjeta_edo_ocurrencia_lugar. Las cuales se obtienen mediante el método BuscarLocalidad.php del Registro de Ubicación (RU).

También se modifican las edades, en caso de que se introduzcan por las columnas o las variantes. El proceso sólo se necesita para el caso de las tarjetas pues en esta tabla sólo se tiene el identificador del paciente al que se le detectó la enfermedad y no la edad de éste, la cual es necesario obtenerla mediante cálculos con el número del carné de identidad

proveniente del método BuscarCiudadano.php del Registro del Ciudadano (RC). En el caso del consolidado sólo se puede buscar por rangos de edades, los cuales si están definidos en las tablas del módulo, por lo cual no se necesita de ninguna transformación.

El caso del sexo es el mismo que el de la edad, y se utiliza el mismo método del RC sólo que no se hacen cálculos, ya que este método devuelve el sexo de las personas encontradas.

Luego de las transformaciones, se hace el llamado al **ReferenciasXTarjeta** y/o **ReferenciasXConsolidado** y se guardan los resultados en una variable. A partir de aquí sólo se tienen identificadores en el caso de las provincias, los municipios y las unidades de salud, por lo cual hay que buscar las descripciones respectivas y cambiarlas por los id en el SOAP que anteriormente se guardó.

Para el caso de las provincias se utiliza el método ListarProvincias.php del Registro de Ubicación (RU) y el ListarMunicipio.php del mismo módulo para los municipios. Por último, en el caso de las unidades de salud se utiliza nuevamente el Buscar_Total.php, pero ahora, para obtener el nombre de la unidad.

En el caso de que se invoque a la vez a los dos métodos de reportes antes referidos, porque no se especifique cuál de los dos se quiere, se unen los SOAP de salida y se muestra al fin el resultado.

3.4 Estándares de diseño, codificación y tratamiento de errores

3.4.1 Estándares de Codificación

Es muy común actualmente encontrar estándares de codificación para gran parte de los lenguajes de programación. Estos permiten un mayor entendimiento entre programadores, facilitando la reusabilidad y el mantenimiento de sistemas. El estilo de codificación definido fue el estándar para aplicaciones PHP (PHP CODIG Standard).

Las etiquetas de apertura y cierre serán de la forma “<?php” y “?>” respectivamente. Para el manejo de los valores que envíe el usuario se hará uso de los arreglos predefinidos: $_GET, $_POST, $_FILES, evitando el uso de $_REQUEST.

Los nombres de variables se escribirán con letras minúsculas en español y separando las palabras por “_”. Se tratará de usar nombres sugerentes en cuanto al contenido de la variable.

Todos los campos identificadores van a comenzar con el identificador (id) seguido del nombre del campo. **Ejemplo: id_tarjeta**.

Los arreglos empezarán con el identificador array y las palabras no se separarán con el carácter “_”. **Ejemplo: arrayidenfermedad**.

Las estructuras se identificarán poniendo al final del nombre struct. **Ejemplo: paginadostruct**. En el caso de las clases se pondrá delante la letra C. Ejemplo: **CFachadaEDO** y en el de los métodos no se usarán abreviaturas y las palabras continuas deben comenzar con mayúsculas. **Ejemplo: BuscarEnfermedad**.

Para comentar el código se utilizará, en el caso de una línea, al final de la misma el carácter “//“ y seguido el comentario y en el caso de un bloque se utilizarán los caracteres “/*” y “*/”.
Ejemplos:

```
//llevar los datos de entradas a la existencia en mis tablas
/*el filtrar lo que hace es limitar los id ciudadano a los que existen
en las tablas de edo en
dependencia de los parametros de las filas.*/
```

Las llaves se usarán poniendo la llave inicial en una línea para ella sola, y en su respectiva columna la llave final también en una línea.

El idioma de las clases auxiliares como sesión y error, será el inglés para garantizar la homogeneidad con las programadas en este ámbito en el mundo, en el caso de los Servicios

Web y la interface de administración se usará el español para esclarecer los objetivos de cada método o script a utilizar.

Para lograr que las comparaciones sean seguras, se colocarán siempre los valores constantes a la izquierda de la comparación "if (6 == $variable)", con esto se garantizará la generación de un error cuando por error escriba '=' y no '=='. Se utilizará el operador “?” para sentencias cortas, preferiblemente que ocupen una sola línea. La sentencia switch siempre tendrá la opción default y se evitará el uso de continue y break, ya que podrían perder la vista lógica del código fuente.

El almacenamiento de la información será en scripts SQL para construir la base de datos e interactuar con ella desde las aplicaciones.

Las palabras correspondientes a las sentencias SQL y sus parámetros deben ir en mayúsculas **Ejemplo: SELECT * FROM tb_tarjeta_edo**

Los nombres de las tablas deben ir en minúsculas y cada palabra separada por línea abajo "_". **Ejemplo: id_nombre_tabla**

En el caso de los XSL será con el mismo nombre que el fichero de la capa de presentación. Los controles seguirán el siguiente tratamiento:

Control	Prefijo	Ejemplo
Botón	Btn	btnAceptar
Etiqueta	Lbl	lblNombre
Lista/Menú	Mn	mnPrincipal
Campo de Texto	Txt	txtFecha
Botón de Opción	Opt	optSexo
Casilla de Verificación	Chx	chxBorrar
Grid o rejilla	Grid	grUsuario

Las páginas HTML se harán sin incluir código y todas las funciones JavaScript que se usarán se escribirán dentro de ficheros “*.js*”.

Para la capa de datos hay que nombrar la base de datos poniendo el identificador del proyecto "APS" seguido del carácter "_" y el nombre del módulo. **Ejemplo: bd_edo**. Las tablas se identificarán con el acrónimo tb_Nombre. **Ejemplo: tb_consolidado_edo_manual**. Los campos de la base de datos se nombrarán igual que las variables.

3.4.2 Tratamiento de Excepciones

Se utilizará JavaScript para depurar los errores. Por medio de este lenguaje serán informados la mayoría de los errores de la página, como apoyo a las validaciones de entrada de datos, garantizando que los datos introducidos por los usuarios sean válidos, o les sea posible corregirlos en caso contrario.

Otros errores en la capa de negocio serán tratados devolviendo un Fault cuyos elementos FaultCode, FaultString, FaultAutor se describen a continuación:

FaultCode:
Código de texto utilizado para indicar la clase de error, será codificado de la siguiente manera. Código del proyecto-código del módulo (:) número del método (.) número del error. **Ejemplo: APS-REDO: 1.5** que indica error 5 en el método 1 del módulo Registro de EDO perteneciente al Proyecto APS.

FaulString:
Una explicación del error asequible al humano (explicativo). Debe tenerse en cuenta que este texto puede ser mostrado al operador final del sistema. **Ejemplo: Formato de entrada no válido para la fecha de cierre estadístico**.

FaultActor:
Un texto que indica quién provocó el error, siempre será el nombre del método que eleva la excepción. **Ejemplo: REDO_BuscarEnfermedad.**

Detail:

Este elemento se usa para llevar mensajes de errores específicos de las aplicaciones, se empleará únicamente en errores cuya resolución depende del Centro de Control, en cualquier otro caso este elemento debe estar vacío.

También se utilizaron codificadores para evitar posibles errores por parte del usuario al registrar información de poca variabilidad, se puede citar el caso de los grupos de edades.

Ejemplo:

3.4.3 Tratamiento de excepciones en la Capa de Presentación

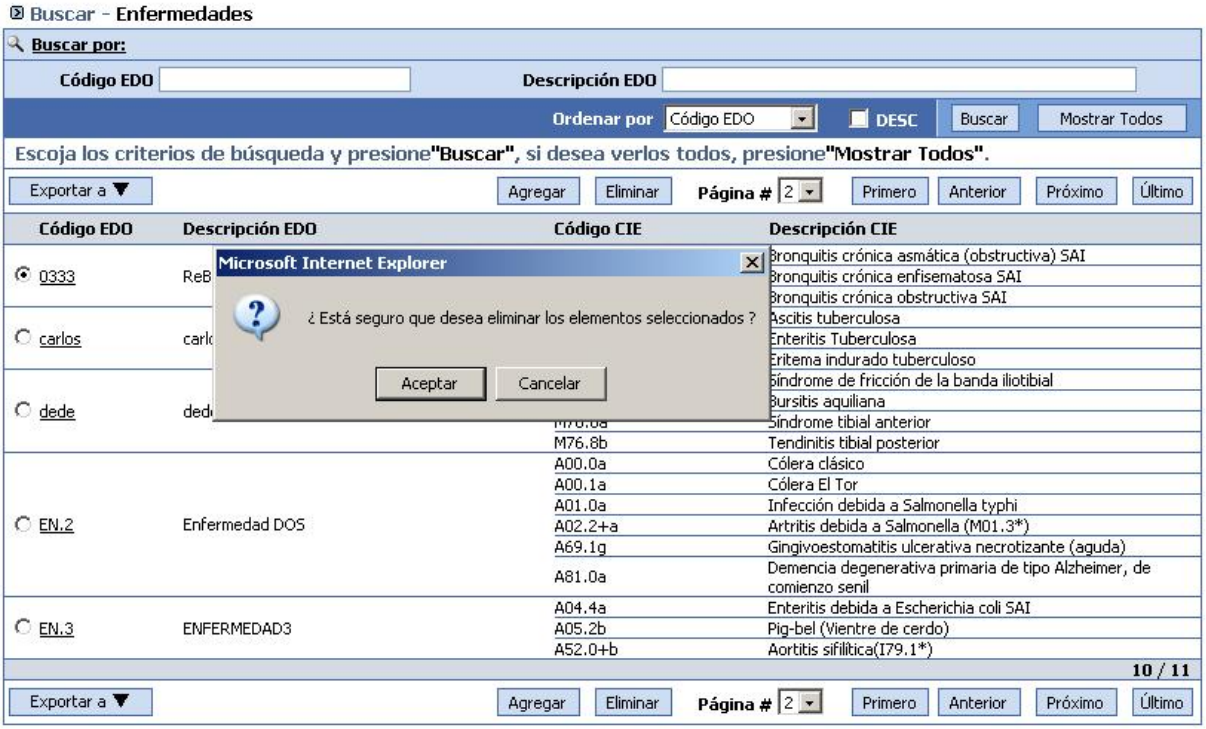
Antes de eliminar o modificar cualquier información se deben hacer las siguientes preguntas:

¿Está seguro que desea Actualizar el elemento seleccionado?

Ejemplo:

¿Está seguro que desea Eliminar los elementos seleccionados?

Ejemplo:

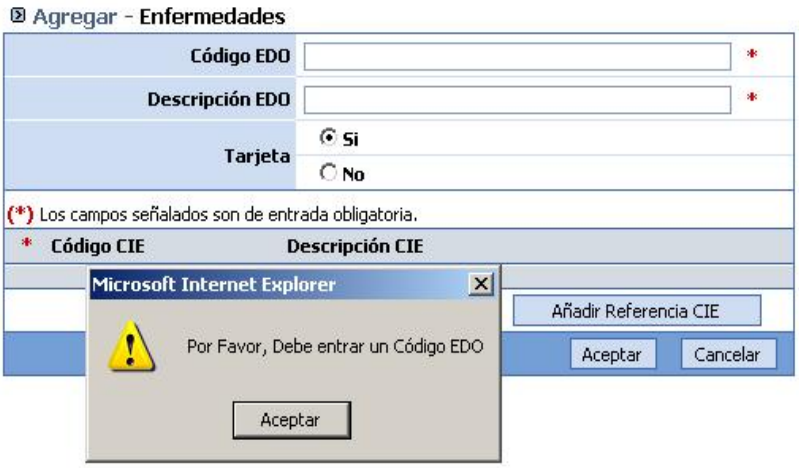


En el caso de agregar algún elemento en las tablas y falten campos obligatorios por llenar, se hace de forma particular para cada caso, pero siempre siguiendo un mismo estándar:

Por favor, se requiere “nombre del campo”.

Por favor, especifique “nombre del campo”.

Ejemplo:



Al no existir resultados al realizar una Búsqueda dichos mensajes no se emiten en nuevas ventanas, sino en el mismo lugar donde se mostrarían los resultados en caso de existir:
No existen “Elementos” registrados hasta el momento para estos criterios de búsqueda.
Ejemplo:

Los mensajes de errores que llegan a la capa de presentación desde la capa de negocio se muestran en la página de error del módulo. Estos mensajes quedan bien definidos para el programador, pero al usuario no le aporta nada porque este no entiende de nombres de métodos, ni de componentes y menos del número del error. Es por eso que se hace necesario llevar a cabo un proceso de actualización con respecto a cómo mostrar los mensajes de error en la capa de presentación, de forma tal que el error se muestre en la página donde esté trabajando el usuario y no en otra página que obliga al usuario a presionar un botón más.

3.4.4 Estándares en la Interfaz de la aplicación

Los módulos que conforman el Proyecto APS mantienen una profunda interrelación, desde el punto de vista que se acoplan para solucionar el extenso negocio de la Atención Primaria de Salud. Por este motivo es necesario que tengan rasgos característicos para así facilitar su identificación y manejo.

El todo general lo conforma el **Sistema de Información para la Salud (SISalud)**, el cual se compone por el Registro Informatizado de Salud (RIS), el Sistema Informatizado de Atención Primaria (SIAP), el Sistema Informatizado de Gestión Hospitalaria (SIGH) y el Sistema Informatizado de Atención Especializada (SIAE). A continuación se describen las pautas a seguir en a la hora de hacer el diseño.

3.4.5 Diseño de Interfaz Gráfica del Proyecto APS

Se definen un grupo de pautas en el diseño grafico de los módulos con el propósito de estandarizar estos y que sean identificados como partes de un todo. Estas pautas permitirán también una mayor facilidad en el manejo de los sistemas implementados.

Se diseñará una Pantalla Inicial global del **Sistema de Información para la Salud,** desde la cual se accederá a los diferentes módulos del RIS, del SIAP, del SIGH y del SIAE. Esta pantalla contará con accesos a los diferentes módulos, informaciones generales, guías de ayuda, sistema de avisos que genera cada registro y enlaces definidos. Así mismo será diseñada una Pantalla Inicial para cada una de las aplicaciones, que contará con accesos a todas las utilidades, avisos, ayuda y un enlace para regresar a la Pantalla Inicial del **SISalud**.

La estructura base de las aplicaciones es la misma para todos los módulos: las pantallas más usadas, los modelos establecidos, las rutas de navegación, las utilidades básicas, la organización de los elementos en pantalla y el diseño de identificadores serán comunes para todos.

Para particularizar el diseño de cada módulo se ha definido entonces una pauta de dos colores básicos en este caso, azul y azul oscuro, con sus degradaciones hacia blanco y negro, así como la diferenciación por logotipo e imagen principal del cabezal, que identificará a cada módulo.

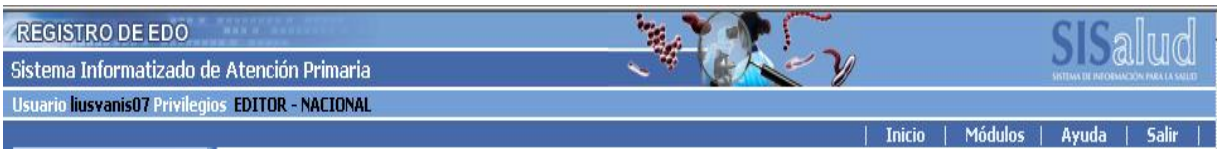
Su diseño está determinado fundamentalmente por el principio de la usabilidad, teniendo en cuenta que no se trata de un sitio Web, sino de una aplicación de trabajo donde el diseño tiene como principal propósito facilitar su uso, comprensión y navegación, por encima de ornamentos inútiles, aunque manteniendo pautas estéticas, orgánicas y agradables.

La resolución óptima para la cual está diseñada la aplicación es de 800 x 600 px. El fondo siempre será blanco y los elementos de pantalla de los colores definidos para el módulo. Se ha definido un cabezal pequeño de 65 px de altura, más pequeño que el utilizado en las páginas Web, que recomiendan cabezales de hasta 80 px de altura.

El menú principal siempre estará situado en una barra superior horizontal de sólo 15 px de altura. Existirá una barra vertical de menú situada la izquierda de la página (como usualmente se hace) pero para ampliar el espacio, esta podrá replegarse. Gracias a esto, el espacio de trabajo estará reservado lo más amplio posible para la inserción de grandes tablas y formularios que constituyen la base fundamental de estas aplicaciones.

El logo siempre estará ubicado en el extremo superior izquierdo de la página, es una imagen que cuenta con un ancho de 270 px y se corresponde con el nombre de cada módulo. Estará constituido por un juego tipográfico en *Frankling Gothic Medium*, y en el caso de las aplicaciones propias del Proyecto APS, estando especificado dentro del logo como una especie de genérico.

Ejemplo: Menú del Registro de EDO.



Bajo el logo existirá una barra de ubicación dentro del sitio, funcionando como hipervínculo, que servirá como referencia para saber dónde se encuentra el usuario o para acceder rápidamente a cualquiera de los niveles superiores de navegación dentro de los que se encuentra. Además se encontrará destacado dentro del menú principal (con un destaque en el color secundario) en cuál de los elementos del menú se encuentra el usuario en ese momento.

La tipografía será siempre Tahoma, por su amplia legibilidad y por las facilidades conocidas que brinda para la lectura digital.

El menú principal será a 7 ptos y los submenús a 6 ptos. Los demás puntajes se definirían en dependencia de las necesidades puntuales de cada pantalla. El espacio de trabajo comienza 33 px por debajo del menú. El espacio intermedio que queda es también con fondo blanco y está reservado para el texto de ubicación dentro del sitio (justificado a la izquierda) y para ubicar los botones propios de la pantalla (justificados a la derecha). Estos se organizarán en

una o dos filas, de hasta cuatro botones (13 x 72 px) cada una. Los botones se corresponden también con los colores pautados.

Entre los elementos comunes del menú principal se encuentran *Inicio* para regresar a la página inicial del módulo, *Salir* para desconectarse del sistema, y *Otros Módulos* para facilitar los enlaces a otros módulos necesarios. Son también comunes a casi todos los botones del menú principal *Configurar* para la configuración de codificadores, *Cierre* para la realización de cierre estadístico y *Reportes* para generar reportes de actividades u operaciones.

Es común para todos los módulos el diseño de una serie de ventanas, en las que sólo cambiarían los colores, en dependencia de cada uno. Son estas las ventanas de precaución, error, validación de datos, etc.

En cuanto a los elementos de diseño del interior de las pantallas, es decir, de las tablas, formularios, etc., se definen los edit que se utilicen con una altura de 16 px y la separación entre estos y entre ellos y los bordes de tablas será de 8 px. Será de 8 px la separación entre el texto y el edit. Los textos de estos campos serán justificados siempre a la derecha, es decir, justificados a 8 ptos de cada edit.

En el caso de tablas generadas por búsquedas, que ordenan una serie de elementos, y necesiten selección, se harán a través de checkboxs justificados a la izquierda de la tabla. Siempre habrá un checkbox en la fila de título, también a la izquierda, que facilite *seleccionar todos.* Es necesario destacar que estas tablas pueden tener una cantidad grande de líneas generadas por la búsqueda, por lo que debe quedar pautado que hasta 25 resultados la tabla funcione con scroll, pero más de esta cantidad será entonces por paginado, al estilo de *Google*, con 25 resultados por página.

Existen detalles que serán definidos particularmente en cada uno de los módulos, ya que satisfacen a necesidades específicas de los mismos. El interés general es mantener el diseño y la estructura del sitio lo más simple posible, la simplicidad es entendimiento del contenido, de la estructura, es facilidad para encontrar lo que se busca, es también velocidad de descarga.

Conclusiones

En este capítulo se aborda la integración del módulo con varios componentes y se modelan los diagramas de componentes más representativos, así como el diagrama de despliegue. Del mismo modo se han descrito algunos métodos, debido a su complejidad e importancia. Por último se exponen las pautas a seguir en la confección del módulo para lograr una estandarización entre los sistemas del Sistema de Información para la Salud y poder abordar la etapa de mantenimiento con mayor facilidad.

Conclusiones

1. Se realizó un estudio del estado del arte del tema de las Enfermedades de Declaración Obligatoria, de las tendencias y tecnologías actuales, así como de la arquitectura definida por el MINSAP para realizar el proceso de implementación del Registro de EDO.
2. Se modelaron los flujos de trabajo de diseño e implementación utilizando la metodología RUP y el lenguaje UML para el modelado.
3. Se aplicaron, para la confección del módulo, los estándares de diseño, codificación y tratamiento de errores definidos en el Proyecto APS.
4. Se implementaron los servicios Web del Registro de Enfermedades de Declaración Obligatoria (REDO).
5. Se Integró el módulo en el conjunto de aplicaciones del Sistema de Información para la Salud (SISalud), definiendo los servicios que se necesitan de otros componentes, así como los que se brindan.
6. Con el registro de REDO el Sistema Nacional de Salud contará con una herramienta para agilizar el control y vigilancia epidemiológica.

Recomendaciones

Considerando la experiencia alcanzada durante la confección de este trabajo se derivan las siguientes recomendaciones:

1. Concluir el desarrollo de los reportes dinámicos que permita contar con información oportuna para la toma de decisiones.
2. Realizar la implementación de la opción Cierre Estadístico en la próxima iteración de la aplicación propuesta, para una mejor gestión de la información de las EDO en los departamentos de estadística a todos los niveles de dirección del MINSAP.
3. Implementar en el Componente de Avisos del Sistema de Información para la Salud los mensajes que se derivan de la información que se gestiona en el Registro de EDO, para activar el sistema de vigilancia de forma oportuna en todos los niveles de atención médica a la población.
4. Concluir el despliegue en Infomed del Registro de EDO para comenzar a utilizarlo en los cuatro policlínicos del piloto del municipio Cerro.

Referencias Bibliográficas

[1]. **Delgado Ramos, Dr. Ariel, Cabrera Hernández, Ing. Mirna, Juncal, Dra. Virginia** (2005). Registro Informatizado de Salud (RIS). http://72.14.205.104/search?q=cache:eLdOEBnyCnIJ:www.sld.cu/galerias/pdf/sitios/dne/ris.pdf+Registro+Informatizado+de+Servicios+Medicos&hl=es&ct=clnk&cd=1&gl=cu&client=firefox-a [En Línea] [Citado el: 22 Mayo del 2007)

[2]. Ídem a la Referencia 1.

[3]. **Coutin Marie, Dra. Gisele**. Utilidad de la Informática para la Vigilancia de enfermedades en el tiempo. [En Línea] [Citado el: 22 de Mayo del 2007).

[4]. Ídem a la Referencia 3.

[5]. Ídem a la Referencia 3.

[6]. Ídem a la Referencia 3.

[7]. Ídem a la Referencia 3.

[8]. Ídem a la Referencia 3.

[9]. Ídem a la Referencia 3.

[10]. Ídem a la Referencia 3.

[11]. **Rico Cordeiro, Dr. Osvaldo**. TECNOLOGIA Y GESTION DE LA INFORMACION BASES PARA EL DESARROLLO DE LA VIGILANCIA DE LA SALUD EN ARGENTINA. [En Línea] [Citado el: 21 de Mayo del 2007].

[12]. Ídem a la Referencia 3.

[13]. Ídem a la Referencia 3.

[14]. Ídem a la Referencia 3.

[15]. Ídem a la Referencia 3.

[16]. **Suárez Medina, Dr. Ramón**. SOFTWARE PARA EL SISTEMA INTEGRADO DE VIGILANCIA DE DENGUE. [En Línea] [Citado el: 21 de Mayo del 2007).

[17]. **Teruel, Prof. Alejandro**. http://www.ldc.usb.ve/~teruel/ci3715/clases/arqCapas.html. [En línea] [Citado el: 05 de Mayo del 2007.].

[18]. Ídem a la Referencia 17.

[19]. **Salazar Alea, MsC. Caridad, Cortina Rodríguez, MsC. Antonio**. Diseñando Aplicaciones Distribuidas. http://www.monografias.com/trabajos14/aplicacion-distrib/aplicacion-distrib.shtml. [En línea] [Citado el: 05 de Mayo del 2007.].

[20]. Ídem a la Referencia 19.

[21]. **Barco, Antonio**. Arquitectura Orientada a Servicios (SOA). http://arquitecturaorientadaaservicios.blogspot.com. [En línea] [Citado el: 05 de Mayo del 2007.].

[22]. Ídem a la Referencia 21.

[23]. Ídem a la Referencia 21.

[24]. Ídem a la Referencia 21.

[25]. S/N. ¿Qué significa php? - Definición de php. http://www.masadelante.com/faq-php.htm. [En línea]. [Citado el: 8 de febrero del 2007].

[26]. Ídem a la Referencia 25.

[27]. S/N. Lenguajes de Programación (PHP). http://es.wikipedia.org/wiki/PHP. [En línea] [Citado el: 8 de febrero del 2007].

[28]. **Gassmann, Alejandro**. Artículos de Programación Web | .NET. http://www.gamarod.com.ar/articulos/introduccion_a_aspnet.asp. [En línea] [Citado el: 08 de Febrero del 2007].

[29]. Ídem a la Referencia 28.

[30]. S/N. Lenguajes de Programación (PERL). http://es.wikipedia.org/wiki/Perl. [En línea]. [Citado el: 8 de Febrero del 2007].

[31]. S/N Definición y Significado de JAVA. http://www.mastermagazine.info/termino/5470.php [En línea]. [Citado el: 8 de Febrero del 2007].

[32]. S/N Beneficios del uso de JAVA en las aplicaciones modernas de Bibliotecas. http://fesabid98.florida-uni.es/Comunicaciones/m_enjolras.htm [En línea]. [Citado el: 8 de Febrero del 2007]

[33]. Ídem a la Referencia 31.

[34]. Ídem a la Referencia 31.

[35]. S/N Resumen PN. UCI. [En Línea][Citado el: 01 de Junio del 2007.].

[36]. **Navarro, Franco, Ing José Ángel**. UML en Acción. Modelando Aplicaciones Web. [Citado el: 25 de Junio del 2007.].

[37]. Ídem a la Referencia 36.

[38]. Ídem a la Referencia 36.

[39]. Ídem a la Referencia 36.

Bibliografía

alegsa.com.ar [Online] // alegsa.com.ar. - 8 de Febrero de 2007. - http://www.alegsa.com.ar/Dic/perl.php.

alegsa.com.ar [Online] // alegsa.com.ar. - 8 de Febrero de 2007. - http://www.alegsa.com.ar/Dic/xslt.php

es.wikipedia.org [Online] // es.wikipedia.org. -8 de Febrero de 2007 http://es.wikipedia.org/wiki/PHP

masadelante.com [Online] // masadelante.com -8 de Febrero de 2007 http://www.masadelante.com/faq-php.htm

gamarod.com.ar [Online] / aut: Ing. Alejandro Gassmann // gamarod.com.ar -8 de Febrero de 2007 http://www.gamarod.com.ar/articulos/introduccion_a_aspnet.asp

es.wikipedia.org [Online] // es.wikipedia.org -8 de Febrero de 2007 http://es.wikipedia.org/wiki/Perl

mastermagazine.info [Online] // mastermagazine.info -8 de Febrero de 2007 http://www.mastermagazine.info/termino/5470.php

fesabid98.florida-uni.es [Online] // fesabid98.florida-uni.es -8 de Febrero de 2007 http://fesabid98.florida-uni.es/Comunicaciones/m_enjolras.htm

dev.mysql.com [Online] // dev.mysql.com. -12 de Febrero de 2007. - http://dev.mysql.com/doc/refman/5.0/es/features.html

ciao.es [Online] // ciao.es. -12 de Febrero de 2007. - http://www.ciao.es/Opiniones/MySQL__153895

wikipedia.org [Online] // es.wikipedia.org. -14 de Mayo de 2007. - http://es.wikipedia.org/wiki/Patrón_de_diseño

programacion.com [Online] // programacion.com. -14 de Mayo de 2007. - http://www.programacion.com/java/articulo/joa_patrones4

jorgesaavedra.wordpress.com [Online] /aut. Jorge Saavedra // jorgesaavedra.wordpress.com. -14 de Mayo de 2007. - http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman

blogs.3devnet.com [Online] / aut. Sergio Tarrillo // blogs.3devnet.com . -14 de Mayo de 2007. - http://blogs.3devnet.com/blogs/starrillo/archive/2006/03/28/70.aspx

72.14.205.104 [Online] // 72.14.205.104. -21 de Mayo de 2007. - http://72.14.205.104/search?q=cache:6lCu-gnKF30J:eisc.univalle.edu.co/materias/Material_Desarrollo_Software/exposiciones2005B/polimorfismoFabricacionPura_G01/presentacion.ppt+patr%C3%B3n%2Balta+cohesion&hl=es&ct=clnk&cd=1&gl=cu&client=firefox-a

es.wikipedia.org [Online] // es.wikipedia.org/wiki/Grasp. -21 de Mayo de 2007. - http://es.wikipedia.org/wiki/Grasp

www.mcc.unam.mx [Online] // www.mcc.unam.mx**.** -21 de Mayo de 2007. - http://www.mcc.unam.mx/~cursos/Objetos/Cap8/cap8.html

es.tldp.org [Online] // es.tldp.org**.** -21 de Mayo de 2007. -http://es.tldp.org/Tutoriales/NOTAS-CURSO-BBDD/notas-curso-BD/node18.html

72.14.205.104 [Online] // 72.14.205.104**.** -21 de Mayo de 2007. - http://72.14.205.104/search?q=cache:PtUGWIT2tOUJ:www-gris.det.uvigo.es/~avilas/ISII/L507.pdf+diagrama+de+despliegue%2Bdefinicion&hl=es&ct=clnk&cd=3&gl=cu&client=firefox-a

scielo.sld.cu [Online] / aut. Denis Berdasquera Corcho1 // Rev Cubana Med Gen Integr v.18 n.1 Ciudad de La Habana ene.-feb. 2002 // art. La vigilancia en salud. Elementos básicos que debe conocer el Médico de Familia -21 de Mayo de 2007. - http://scielo.sld.cu/scielo.php?pid=S0864-21252002000100011&script=sci_arttext&tlng=es

www.ldc.usb.ve [Online] / aut. Prof. Alejandro Teruel // www.ldc.usb.ve -21 de Mayo de 2007. - http://www.ldc.usb.ve/~teruel/ci3715/clases/arqCapas.html

arquitecturaorientadaaservicios [Online] / aut. Antonio Barco // arquitecturaorientadaaservicios.blogspot.com -11 de Junio de 2007. - http://arquitecturaorientadaaservicios.blogspot.com/

Glosario de Términos

Área de Salud: Área geográfica a la que presta sus servicios una unidad de salud que puede ser policlínico, policlínico con camas u hospital rural. Cada área de salud contempla el programa de trabajo del medico y la enfermera de la familia. En ellas se atiende la mayoría de los problemas de salud de la población.

Consultorio Médico de la Familia: Establecimiento donde se realizan las actividades asistenciales del Equipo Básico de Salud (EBS) de la Atención Primaria de Salud. Puede estar ubicado en la comunidad, centros educacionales o de trabajo y puede ser de vivienda o no. En un consultorio puede existir más de un Equipo Básico de Salud.

Equipo Básico de Salud: Binomio conformado por el médico y enfermera de la familia, que atiende una población geográficamente determinada, que pueden estar ubicados en la comunidad, en centros laborales o educacionales.

Grupo Básico de Trabajo: Es el equipo multidisciplinario integrado por diferentes Equipos Básicos de Salud, por un especialista en Medicina Interna, Gineco-Obstetricia, Pediatría, Medicina General Integral (Jefe del grupo) y Psicología, por una enfermera supervisora, un técnico estadístico y un técnico de higiene y epidemiología, además de un trabajador social, que cumple funciones asistenciales y docentes dirigidas a incrementar la calidad de la atención de la población.

Grupos Edades: Es una agrupación que se realiza por edades, estas pueden ser de 5 años generalmente, aunque esta agrupación depende del investigador o de los indicadores que se buscan.

Incidencia: La incidencia de una enfermedad se define como el número de casos nuevos de esa enfermedad que afecta a una población durante un periodo determinado.

Nuevo caso: Casos registrados en la semana en curos de nuevos brotes o epidemias, de enfermedades nuevas o conocidas. Registrados a través del Sistema Nacional de Vigilancia Epidemiológica.

Paciente: Clasificación que se le da a la persona que utiliza los servicios del Sistema Nacional de Salud. Es todo individuo que busque asistencia médica.

Semanas estadísticas: Es la división que se realiza a todas las semanas del año con fines estadísticos. La semana se divide en 52 semanas estadísticas, donde todas, excepto la primera y la ultima comienzan el domingo y terminan el sábado. En el caso de la primera, comienza siempre por el primero de enero y la última siempre termina el 31 de diciembre, por lo cual estas semanas pueden tener desde 4 hasta 11 días.

Vigilancia: Se trata del conocimiento del comportamiento de las enfermedades en cualquier país o región. Conocer el riesgo de las distintas poblaciones frente a enfermedades bajo vigilancia. La vigilancia de salud se encarga de intervenir ante problemas de salud haciendo énfasis en los problemas que requieran de continua atención. Crea y toma las medidas de promoción, prevención y curación en los territorios que controla.