UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 6

Título: “Propuesta de Arquitectura Orientada a Servicios para alasMEDIGEN.Sistema Informático de Genética Médica

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Marisel Santana Rodríguez
Renier Batista Nagarián

Tutores: Ing. Elvismary Molina De Armas
Ing. Yusdenis Sánchez Parodin
Ing. Félix Mario Marrero Hernández

Ciudad de la Habana, Cuba

Junio 2010

DECLARACIÓN DE AUTORÍA

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

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

Marisel Santana Rodríguez

_____

Firma del Autor

Renier Batista Negarían

_____

Firma del Autor

Elvismary Molina de Armas

_____

Firma del Tutor

Yusdenys Sánchez Parodin

_____

Firma del Tutor

Félix M. Marrero Hernández

_____

Firma del Tutor

DATOS DE CONTACTO

TUTORES:

Tutor: Ing. Elvismary Molina de Armas

Especialidad de graduación: Ingeniería en Ciencias Informáticas

Categoría docente: Instructor en Adiestramiento

Categoría Científica: Ingeniero

Años de experiencia en el tema: 2

Años de graduado: 2

Correo Electrónico**:** emolina@uci.cu

Tutor: Ing. Yusdenys Sánchez Parodin

Especialidad de graduación: Ingeniería en Ciencias Informáticas

Categoría docente: Instructor

Categoría Científica: Ingeniero

Años de experiencia en el tema: 4

Años de graduado: 3

Correo Electrónico: ysanchezp@uci.cu

Tutor: Ing. Félix Mario Marrero Hernández

Especialidad de graduación: Ingeniería en Ciencias Informáticas

Categoría docente: Instructor en Adiestramiento

Categoría Científica: Ingeniero

Años de experiencia en el tema: 1

Años de graduado: 1

Correo Electrónico: fmmarrero@uci.cu

RESUMEN

La investigación surge de la necesidad por parte de alasMEDIGEN.Sistema Informático de Genética Médica, de cumplir las normas establecidas por el Marco Regulatorio del Grupo de Integración de Soluciones (GIS), encargado de supervisar la integración de las aplicaciones desarrolladas para el Sistema Nacional de Salud.

En el presente trabajo de diploma se expone una propuesta de Arquitectura Orientada a Servicios para alasMEDIGEN.Sistema Informático de Genética Médica, de modo que le posibilite al mismo ser seguro, portable e integrable con otros sistemas (Registro Informatizado de la Salud, entre otros).

Para satisfacer el objetivo de esta investigación se realizó una investigación de los principales componentes de la Arquitectura Orientada a Servicios, sus características y las tendencias actuales según los autores más prestigiosos del tema, así como la experiencia del desarrollo de alasMEDIGEN.Sistema Informático de Genética Médica, desarrollado en la Facultad 6 de la Universidad de las Ciencias Informáticas. La descripción y construcción de la solución propuesta se presenta a través de las vistas arquitectónicas definidas por Philipp Kruchten e incorporadas a la metodología de desarrollo RUP.

Para validar la propuesta realizada se aplicó el método de evaluación Revisión Activa para Diseños Intermedios (en inglés: Active Reviews for Intermediate Designs, ARID), obteniéndose resultados satisfactorios para los atributos de calidad definidos por el Modelo ISO 9126-1:2001.

Este documento puede ser fuente de consulta para el desarrollo de sistemas con características similares en diversos entornos.

PALABRAS CLAVES

Arquitectura Orientada a Servicios, arquitectura, evaluación, servicios, Sistema Informático de Genética Médica.

TABLA DE CONTENIDO

INTRODUCCIÓN

La Universidad de la Ciencias Informáticas como centro productor de software ha desarrollado varias aplicaciones para el Ministerio Nacional de Salud Pública (MINSAP), entre las que se encuentra alasMEDIGEN.Sistema Informático de Genética Médica. En el año 2008 se crea dicho sistema, el cual se encarga de recoger los datos de los diferentes estudios genéticos realizados en el país por el Centro Nacional de Genética Médica (CNGM), entre los cuales se encuentra el Registro Cubano de Gemelos y Discapacidad Física e Intelectual.

El sistema alasMEDIGEN está compuesto por siete módulos que gestionan de forma independiente su negocio, pero comparten la misma base de datos. Los módulos del sistema son:

- ✓ Módulo Registro Cubano de Enfermedades Genéticas (RECUEGEN).
- ✓ Módulo Registro Cubano de Malformaciones Congénitas (RECUMAC).
- ✓ Módulo Registro Cubano de Discapacitados (RECUDIS).
- ✓ Módulo Registro Cubano de Retraso Mental (RECURM).
- ✓ Módulo Registro Cubano de Gemelos (RECUGEM).
- ✓ Módulo Registro Cubano de Historias Clínicas (RECUHCL).
- ✓ Módulo Teleconsulta.

En la actualidad existen nuevos sistemas que se están desarrollando con el CNGM que tienen la posibilidad de integrarse a alasMEDIGEN, entre los cuales se encuentran:

- ✓ Registro de Anomalías Cromosómicas.
- ✓ Registro de Enfermedades Comunes en la Familia Cubana.
- ✓ alasARBOGEN 2.0.

alasMEDIGEN fue creado cuando se comenzaba el desarrollo de aplicaciones para la salud de forma centralizada y no existían pautas definidas ni evaluadas. En el 2009 tras una reestructuración organizacional de las entidades que rigen el proceso de desarrollo de software para la salud cubana, el MINSAP y el Centro de Desarrollo Informático para la Salud Pública (CEDISAP), institución encargada de coordinar el Programa de la Informatización de la Salud, junto al Grupo de Integración de Soluciones Informáticas (GIS) de la empresa Softel, crean el Marco Regulatorio del GIS. En el mismo se recogen todas las políticas y estrategias de desarrollo, especificaciones a cumplir, arquitectura, estándares, requerimientos y regulaciones de los sistemas informáticos desarrollados para la informatización del Sistema Nacional de Salud.

alasMEDIGEN se debe desplegar en Infomed, integrándose específicamente al Registro Informatizado de Salud (RIS), guiado por el Marco Regulatorio del GIS. En la actualidad este despliegue se va a realizar aun cuando el sistema no cumple con el Marco Regulatorio del GIS debido a que cuando surgió, alasMEDIGEN se encontraba en un 90% de su desarrollo. Después de conversaciones con las autoridades del MINSAP se decidió aceptar el despliegue del sistema en estas circunstancias.

A pesar de ello, se considera que la integración de nuevos sistemas con alasMEDIGEN resultará muy engorrosa debido a la arquitectura Modelo Vista Controlador (dada por el framework Symfony) que actualmente presenta la aplicación, ya que para realizar cualquier cambio o integración de los nuevos desarrollos sería necesaria una reestructuración de dicha arquitectura, rediseñar las bases de datos y las vistas, perdiéndose en este caso interoperabilidad y reutilización.

Por lo anteriormente expuesto se plantea como **problema científico:** ¿cómo garantizar la integración de los nuevos módulos y los ya existentes de alasMEDIGEN.Sistema Informático de Genética Médica al Registro Informatizado de Salud cumpliendo con el Marco Regulatorio del GIS?

Se reconoce como **objeto de estudio:** la Arquitectura de Software; dentro del mismo se selecciona como **campo de acción:** la Arquitectura Orientada a Servicios (SOA), planteándose como **objetivo general:** proponer una Arquitectura Orientada a Servicios para alasMEDIGEN.Sistema Informático de Genética Médica, del cual se derivan los siguientes **objetivos específicos:**

- ✓ Caracterizar la arquitectura del Registro Informatizado de Salud y de alasMEDIGEN.Sistema Informático de Genética Médica.
- ✓ Definir una propuesta de Arquitectura Orientada a Servicios para alasMEDIGEN.Sistema Informático de Genética Médica.
- ✓ Evaluar la arquitectura propuesta.

Con el propósito de dar cumplimiento a estos objetivos específicos se definen las siguientes tareas de la investigación:

1. Análisis de las tendencias actuales de la arquitectura de software.
2. Análisis del Marco Regulatorio del GIS para el desarrollo de software de la salud.
3. Revisión de las técnicas y métodos de evaluación de la arquitectura de software.
4. Estudio de la arquitectura de alasMEDIGEN.Sistema Informático de Genética Médica.
5. Diseño de una propuesta arquitectónica para alasMEDIGEN.Sistema Informático de Genética Médica basada en una Arquitectura Orientada a Servicios.
6. Evaluación de la arquitectura definida aplicando los métodos de evaluación.

El presente trabajo de diploma se estructura en 3 capítulos:

- ✓ **Capítulo 1:** En este capítulo se plantean los principales conceptos relacionados con el objeto de estudio, los que resultan vitales para entender la solución propuesta. Además se realiza un análisis de los elementos fundamentales de la arquitectura de software entre los que se encuentra definiciones, patrones y métodos. Finalmente se efectúa un estudio de la arquitectura del Sistema Nacional de Salud, así como del Marco Regulatorio del GIS y de la arquitectura de alasMEDIGEN.Sistema Informático de Genética Médica.

- ✓ **Capítulo 2:** En este capítulo se presenta el diseño de la propuesta arquitectónica orientada a servicios de alasMEDIGEN.Sistema Informático de Genética Médica, tributando al cumplimiento de las pautas arquitectónicas que propone el Marco Regulatorio del GIS. Para ello se especifica el ciclo de vida de la arquitectura a diseñar y se definen las vista de casos de uso, vista lógica y de despliegue del sistema.

- ✓ **Capítulo 3:** En este capítulo se presentan aspectos a tener en cuenta para la evaluación de la arquitectura de software. Se realiza la evaluación de la propuesta arquitectónica a través del método ARID basado en los atributos de calidad definidos por el Modelo ISO 9126-1:2001 y se usa como técnicas de evaluación las cualitativas.

CAPÍTULO 1
FUNDAMENTACIÓN TEÓRICA

En este capítulo se plantean los principales conceptos relacionados con el objeto de estudio, los que resultan vitales para entender la solución propuesta. Además se realiza un análisis de los elementos fundamentales de la arquitectura de software entre los que se encuentra definiciones, patrones y métodos. Finalmente se efectúa un estudio de la arquitectura del Sistema Nacional de Salud, así como del Marco Regulatorio del GIS y de la arquitectura de alasMEDIGEN.Sistema Informático de Genética Médica.

1.1 ARQUITECTURA DE SOFTWARE

La Arquitectura de Software se asemeja a los planos de un edificio o construcción, porque ella indica la estructura, funcionamiento e interacción entre las partes del software. Tiene sus orígenes en la década del 60, pero no fue hasta los años 90 que Dewayne Perry y Alexander Wolf, la exponen en el sentido que hoy se conoce.

1.1.1 DEFINICIÓN DE ARQUITECTURA DE SOFTWARE

La arquitectura de software permite representar la estructura de un sistema a un nivel mayor que el dado por la programación e incluso por el diseño.

Para que la arquitectura se convierta en una herramienta útil dentro del desarrollo y mantenimiento de los sistemas de software es necesario que se cuente con una manera precisa de representarla.

La arquitectura, conformada por diferentes visiones del sistema, constituye un modelo de cómo está estructurado dicho sistema, sirviendo de comunicación entre las personas involucradas en el desarrollo y ayudando a realizar diversos análisis que orienten el proceso de toma de decisiones.

A continuación se presentan los principales conceptos de Arquitectura de Software:

Según Len Bass, Paul Clement, Rich Kazman:

“La arquitectura de software consiste en la estructura o sistema de estructuras, que comprenden los elementos de software, las propiedades externas visibles de esos elementos y la relación entre ellos”. (1)

Paul Clement, Rich Kazman y Mark Klein plantean:

“La arquitectura de software constituye una vista del sistema que incluye los componentes principales según se le percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión del detalle inherente a la mayor parte de las abstracciones”. (2)

El Ingeniero en Ciencias de la Computación Ivar Hjalmar Jacobson plantea que “la arquitectura abarca decisiones importantes sobre:

- ✓ La organización de un sistema de software.
- ✓ Los elementos estructurales que compondrían el sistema y de sus interfaces, junto con sus comportamientos.
- ✓ La composición de los elementos estructurales y del comportamiento en subsistemas progresivamente más grandes.
- ✓ El estilo de la arquitectura que guía esta organización: los elementos y sus interfaces, sus colaboraciones y su composición”. (3)

Según Pressman “…es una descripción de los subsistemas y los componentes de un sistema informático y las relaciones entre ellos (...)”. (4)

La definición oficial de arquitectura del software es la IEEE Std 1471-2000 que reza así: “La arquitectura del software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución”. (5)

1.1.2 PATRONES ARQUITECTÓNICOS

En un principio, se puede pensar en un patrón como una manera especialmente inteligente e intuitiva de resolver una clase de problema en particular. El patrón es una descripción del problema y la esencia de su solución, de forma que la solución pueda reutilizarse en diferentes situaciones, es una solución adecuada a un problema común.

Los patrones se definen por un nombre, un problema, una solución y las consecuencias de su aplicación. En dependencia del problema que solucionan, estos pueden ser clasificados en: patrones de arquitectura, patrones de análisis, patrones de diseño, entre otros.

Los patrones arquitectónicos “expresan el esquema de organización estructural fundamental para sistemas de software, proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y pautas para la organización de las relaciones entre ellos”. (6)

A continuación se listan los patrones arquitectónicos más conocidos y utilizados:

Layers (Capas): Descompone una aplicación en un conjunto de capas independientes y ordenadas jerárquicamente según el nivel de abstracción que tenga cada una de ellas. Cada nivel o capa usa los servicios del nivel inmediatamente inferior y ofrece servicios a la capa inmediatamente superior. La comunicación sólo se establece entre componentes de la misma capa, o entre componentes de capas

contiguas. Este patrón simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle de las superiores. (7)

Pipes and Filtres (Tuberías y Filtros): Esta arquitectura está constituida por "una tubería (pipeline) que conecta componentes computacionales (filtros) a través de conectores (pipes), de modo que las operaciones se ejecutan a la manera de un flujo". (8) Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas.

Es habitualmente utilizado en compiladores, sistemas de UNIX y tratamiento de documentos en XML.

Blackboard (Pizarra): La arquitectura en pizarra consta de "una estructura de datos que representa el estado actual (pizarra) y una colección de componentes independientes que operan sobre él, denominados agentes." (9) Los agentes son programas especializados en una tarea concreta. Todos ellos cooperan para alcanzar una meta común, sin embargo, sus objetivos individuales no están aparentemente coordinados. La pizarra tiene un doble papel. Por una parte, coordina a los distintos agentes, y por otra, facilita su intercomunicación. El estado inicial de la pizarra es una descripción del problema a resolver. El estado final será la solución del problema. Este patrón es habitualmente utilizado en sistemas expertos de Inteligencia Artificial.

Service Oriented Architecture (Arquitectura Orientada a Servicios, SOA): SOA es una arquitectura que permite desglosar las aplicaciones en términos de servicios, resultando ser más flexibles y por tanto son más fáciles de modificar ante los cambios del negocio.

Según W3C: "Conjunto de componentes que pueden ser invocados, cuyas descripciones de interfaces se pueden publicar y descubrir". (10)

La Arquitectura Orientada a Servicios fomenta la creación de aplicaciones que se adaptan fácilmente a los inevitables y frecuentes cambios del negocio. Propone desglosar las funcionalidades principales de las aplicaciones en servicios autónomos y luego orquestarlos para formar nuevas aplicaciones de negocio. Estos servicios se encuentran disponibles en la red y presentan interfaces estándares bien definidos que permiten el flujo de mensajes entre proveedores y consumidores.

Model View Controller (Modelo Vista Controlador, MVC): El patrón MVC modela el sistema en tres componentes fundamentales: Modelo, Vista y Controlador. Se utiliza frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la Lógica de negocio.

Descripción del patrón

- ✓ **Modelo:** Ésta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.
- ✓ **Vista:** Éste presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- ✓ **Controlador:** Éste responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. (11)

1.2 ARQUITECTURA DEL SISTEMA INFORMÁTICO DE GENÉTICA MÉDICA

Los especialistas del CNGM solicitaron un sistema que integrara todos los estudios genéticos realizados en el país, siendo el mismo alasMEDIGEN.Sistema Informático de Genética Médica. Este sistema está constituido por un conjunto de módulos que gestionan de forma independiente su negocio, pero comparten la misma base de datos. El objetivo es tener concentrada toda la información obtenida en los estudios realizados por el CNGM, de forma que se pueda gestionar y hacer reportes sobre la información. Cada módulo gestiona los datos correspondientes a su estudio, lo cual incluye insertar, modificar y realizar determinados reportes sobre los datos. Los módulos del sistema son: RECUEGEN, RECUMAC, RECUDIS, RECURM, RECUGEM, RECUHCL y Teleconsulta.

El sistema desarrollado constituye una aplicación informática para la salud en Cuba, y por tanto debe adoptar las normas establecidas por el Grupo de Arquitectura MINSAP-MIC para este tipo de aplicaciones, cumpliendo los siguientes requerimientos:

- ✓ Desarrollado en PHP5.
- ✓ Funcionar sobre servidor Linux, distribución Debian (Sarge).
- ✓ Funcionar sobre Servidor Web Apache 2.0.
- ✓ Utilizar como sistema gestor de base datos MySQL5.0.
- ✓ Consumir o brindar servicios web para interactuar con otros sistemas utilizando el componente SAAA.
- ✓ Ser capaz de desplegarse sobre la estructura de servidores presente en Infomed.

Dentro de los requerimientos arquitectónicos anteriormente mencionados, también se insertan un grupo de requisitos no funcionales que el sistema debe cumplir:

Apariencia o interfaz externa: Se deben utilizar imágenes y colores identificados con el negocio del sistema. La interfaz externa debe estar diseñada para verse en cualquier resolución igual o superior a 1024x768.

Usabilidad: La aplicación informática debe garantizar un acceso fácil y rápido, contando con un menú que satisfaga las necesidades de los usuarios. Éste podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de una computadora y del ambiente web.

Rendimiento: Los tiempos de respuestas deben ser generalmente rápidos al igual que la velocidad de procesamiento de la información.

Soporte: Se debe asegurar el soporte para los usuarios de manera que se puedan satisfacer sus necesidades a partir de mejoras una vez puesta en marcha la aplicación. Para ello se crearán una serie de manuales de usuarios y videos tutoriales, y se mantendrá la asistencia a los usuarios.

Seguridad: El sistema necesita información que contienen algunos de los registros del SISalud, específicamente el Registro de Ciudadanos y el Registro de Unidades de Salud. Para consumir los servicios web que brindan estos componentes, es necesario utilizar el componente de seguridad SAAA.

La información genética, por su naturaleza, corresponde al tipo de información más sensible, dado que no sólo desvela información del individuo al que se realiza el análisis (información que es permanente y que no se modifica con el tiempo), sino que también proporciona información de los familiares o su descendencia. Debido a la gran preocupación relacionada con posibles problemas de discriminación y estigmatización debidos al mal uso de la información genética, es necesaria la implantación de un mecanismo de seguridad estricto que regule el acceso a la misma. Este debe ser administrado de forma centralizada e independiente de cualquier sistema, y sus implicados deben estar bien identificados y comprometidos con el resguardo de la información. Por lo anteriormente expuesto, el sistema debe tener un mecanismo propio para gestionar la seguridad a través de niveles de acceso a la información. Los permisos al ejecutar cualquier acción deben estar de acuerdo con el nivel jerárquico de acceso que presente el usuario en cada módulo, el cual es definido por los administradores del sistema.

Software: Se requiere para el funcionamiento del sistema disponer de un servidor que cuente con Sistema Operativo Linux, Apache 2.0 y MySQL 5.0 o versiones superiores. Los usuarios del sistema deberán contar con un navegador Internet Explorer 5.5 o Mozilla Firefox 2.0 o superior, para poder acceder a las opciones que brinda el sistema.

Hardware:

Para el desarrollo y ejecución de la aplicación se necesitará:

Para el servidor de aplicación:

- ✓ Microprocesador Pentium IV a 3.0 GHz o superior.
- ✓ 1GB de RAM o superior.

Para el Cliente:

- ✓ Microprocesador Pentium a 233 MHz (Recomendado: Pentium a 500MHz o superior).
- ✓ 64 MB de RAM (Recomendado*:* 128 MB RAM o superior).
- ✓ 52 MB de espacio de disco duro. (Recomendado: 120MB para la instalación completa del Internet Explorer 5.5).
- ✓ Conexión al servidor a través de MODEM o tarjeta de red.

Además es necesario contar con una impresora para poder imprimir los diferentes tipos de reportes.

Disponibilidad: El sistema debe ser capaz de funcionar por sí solo en caso de que los servicios de los diferentes componentes que utiliza SISalud no estén disponibles. Se debe garantizar el funcionamiento de la aplicación durante las 24 horas del día y los siete días de la semana, con el menor tiempo posible de recuperación ante fallos. Se deben crear copias de respaldo periódicas que puedan restaurar el sistema en caso de fallo crítico o pérdida total de la información.

Requisitos Legales: Las tecnologías y herramientas en que estará basada la aplicación informática deberán cumplir con las licencias de software libre.

Persistencia: La información debe almacenarse en bases de datos con carácter permanente con el objetivo de poder realizar análisis de la misma con el transcurso de los años.

Para el desarrollo de alasMEDIGEN se seleccionó un grupo de herramientas de soporte, dicha selección se basa principalmente en que deben ser herramientas libres. En la siguiente tabla se exponen las herramientas seleccionadas:

Lenguaje de Programación	PHP 5
Gestor de Bases de Datos	MySQL 5
Control de Versiones	Subversion 1.4
Herramienta Case	Visual Paradigm for UML Community Edition 3.1
Framework de Desarrollo	Symfony 1.0
Entorno de Desarrollo	Eclipse 3.3.1

Para la Arquitectura de alasMEDIGEN se seleccionó como patrón arquitectónico el Modelo Vista Controlador (MVC). En la siguiente imagen se muestra la organización estructural alasMEDIGEN teniendo en cuenta el patrón MVC y las características fundamentales del framework de desarrollo:

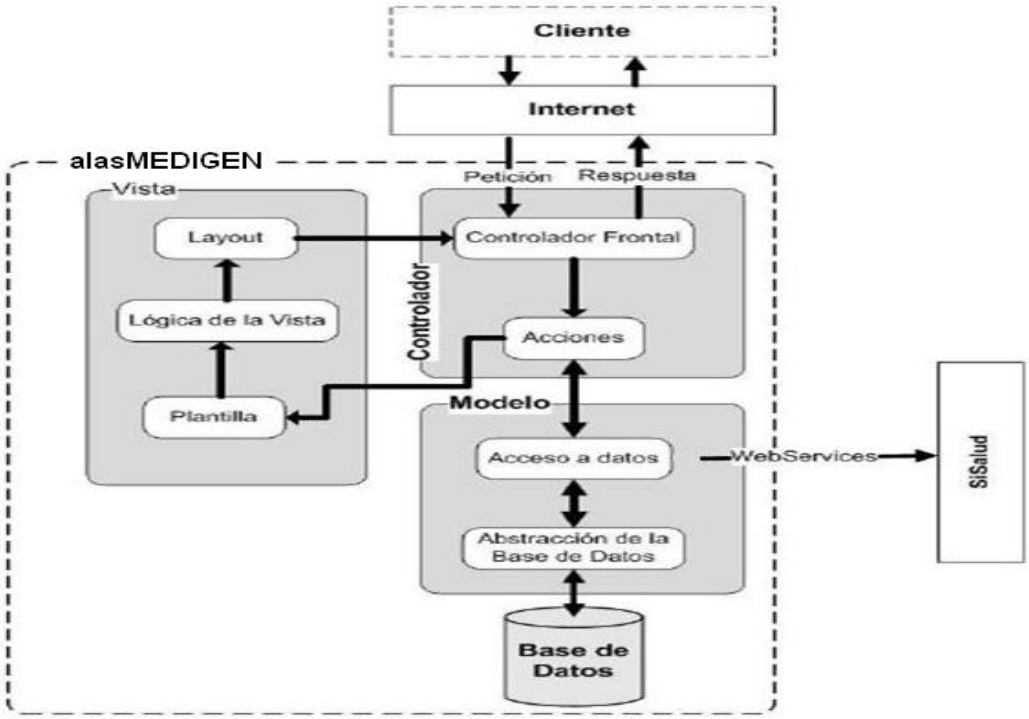

Figura 1. La organización estructural de alasMEDIGEN teniendo en cuenta el patrón MVC y las características fundamentales de Symfony.

Este patrón separa el modelado del dominio, la presentación y las acciones realizadas sobre los datos en tres elementos diferentes. El modelo administra el comportamiento y los datos del dominio de la aplicación, y responde a instrucciones de cambiar su estado desde el controlador. La vista tramita la visualización de la información, y el controlador interpreta las acciones de los dispositivos de entrada de datos, realiza los procesos propios de la aplicación e informa al modelo y/o a la vista para que cambien su estado según resulte apropiado.

Este patrón arquitectónico brinda soporte para múltiples vistas, pues la vista es independiente del modelo, posibilitando por ejemplo que múltiples páginas de una aplicación web puedan utilizar el mismo modelo de objetos, mostrado de maneras diferentes. Además permite una adaptación total al cambio. Debido a que los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios,

y los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs, este patrón permite agregar nuevas opciones de presentación, que generalmente no afectan al modelo, ya que este último no depende de las vistas.
alasMEDIGEN presenta una estructura flexible que haciendo uso del patrón modelo vista controlador, permite que los elementos que lo componen puedan ser modificados y se pueda reutilizar el código implementado.

1.3 INFORMATIZACION DEL SISTEMA NACIONAL DE SALUD

Con la misión principal de trazar la política nacional de introducción de la informática al Sistema de Salud Cubana, se crea en 1988 el Centro de Desarrollo Informático para la Salud Pública (CEDISAP). El rápido desarrollo de la tecnología y la inevitable descentralización del procesamiento de la información, determinó, que a fines de 1988, la dirección del CEDISAP iniciara una reorientación de las funciones y direcciones estratégicas del trabajo de la institución para los años posteriores.
CEDISAP se convierte entonces en una institución especializada en servicios profesionales de informática para el sector salud, que por orientación de la Dirección del MINSAP, abarca además Servicios de Consultoría.
El proceso de informatización de la Salud Pública se lleva a cabo, como parte de la informatización de la sociedad, aplicando las Tecnologías de la Informática y las Comunicaciones (TIC), con lo cual las instituciones de salud del país deben lograr un incremento de la calidad, efectividad y eficiencia de los servicios que prestan a la población.
La informatización del Sistema Nacional de Salud está dada por el conjunto de métodos, técnicas, procederes y actividades gerenciales dirigidas al manejo de la información en salud. En este proceso se utilizan las TIC para el desarrollo de las aplicaciones que se integrarán al Sistema Nacional de Salud, las mismas deben estar sobre una arquitectura basada en componentes y orientada a servicios.
Partiendo de lo anterior se obtendrán componentes con un alto nivel de integración e interoperabilidad que permitirán garantizar la información oportuna, consistente y confiable para la toma de decisiones y el mejoramiento continuo de la calidad de la atención médica cubana.
Hasta el momento se han desarrollado 18 aplicaciones informáticas para la Salud, en coordinación con diferentes entidades como el MINSAP y el Centro de Informática Médica (CESIM), entre éstas se encuentran: Sistema Automatizado Cubano para el Control de Equipos Médicos, Sistema de Colaboración

Médica, Red Nacional de Nefrología, Control Sanitario Internacional, Sistema Integral para la Atención Primaria de Salud, entre otras.

En el desarrollo de las aplicaciones para la salud, después de adquirir cierto grado de experiencia, el Grupo de Integración de Soluciones Informáticas (GIS), creó el Marco Regulatorio pautando las normas arquitectónicas que de una forma u otra han regido el desarrollo de software de salud, pero no habían sido estandarizadas, ni supervisadas. A partir del año 2009 todos los sistemas que se integren al Registro Informatizado de Salud (RIS) deben cumplir con las mismas, por las cuales el GIS velará que se cumplan estrictamente.

1.3.1 REGISTRO INFORMATIZADO DE SALUD

Dando cumplimiento a la informatización de Salud en el 2003 se crea el Registro Informatizado de Salud (RIS), "El RIS es 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". (12)

El RIS es una aplicación orientada a servicios bajo una arquitectura moderna y versátil, es un producto portable que no depende del motor de base de datos escogido para su desarrollo y funciona tanto con el Sistema Operativo "Windows" como en "Linux". En la actualidad el Registro Informatizado de Salud (RIS) cuenta con 10 módulos. Estos son:

Componente de Seguridad (SAAA): Este componente está basado en el modelo de Autenticación, Autorización y Auditoría (AAA). La autenticación debe ser la primera acción del usuario en el sistema y consiste en suministrar un nombre de usuario único y una contraseña que debe ser de conocimiento exclusivo de la persona que se autentica. Si el usuario autenticado no se encuentra registrado se reporta un error de acceso. En caso contrario, se autoriza su acceso y se crea un certificado digital y se retornan todos los datos y permisos del usuario. Cada petición del usuario, autorizada o no, es registrada, así como el día, mes, año, hora, minuto y segundo en que se efectuó la acción, y si fue o no autorizada.

Módulo Registro Unidades de Salud (RUS): En este registro se tiene toda la información detallada de las Unidades de Salud. A través de este sistema se canaliza y gestiona todo el flujo de la información: ya sea de creación, modificación o eliminación de las unidades de salud.

Módulo Registro de Ciudadanos (RC): Contiene los datos personales de cualquier ciudadano que trabaje en la salud y/o reciba sus servicios.

Registro de Personal de Salud (RPS): Mantiene el registro actualizado con la información detallada del Personal de la Salud con que cuenta nuestro país. Orientado a registrar los profesionales médicos y no médicos.

Módulo Registro de Ubicación (RU): Posee toda la información detallada de la división política-administrativa del país, las provincias con sus municipios, localidades, calles y manzanas.

Módulo Registro de Localidades (RL): Permite mantener una información detallada de los Consejos Populares, Circunscripciones, Zonas y CDR con que cuenta nuestro país, con vistas a poder configurar las Áreas de Salud.

Módulo Registro de Equipos Médicos (REM): Contiene el registro de los equipos médicos que se encuentran en las unidades de salud y en las otras dependencias del MINSAP.

Módulo Registro de Equipos no Médicos (RENM): Mantiene el registro de los equipos no médicos que se encuentran en las unidades de salud y en las otras dependencias del MINSAP.

Módulo Registro de Facultades (RF): Permite registrar los egresados de las diferentes Facultades, así como asignarle el número de registro a cada profesional.

Módulo Registro del Clasificador Internacional de Enfermedades y Problemas de Salud (RCIEPS): Gestiona la estructura de la Clasificación Internacional de Enfermedades y Problemas Relacionados con la Salud y permite realizar búsquedas dinámicas de acuerdo a criterios seleccionados por el usuario.

Módulo Registro de Problemas de Salud de la Atención Primaria (RPSAP): Gestiona una clasificación especial para los problemas de salud que se presentan en el nivel de Atención Primaria de Salud. Permite realizar búsquedas dinámicas de acuerdo a criterios seleccionados por el usuario.

Módulo Registro de Servicios Médicos: Mantiene actualizado el registro de Servicios Médicos para configurar los mismos de manera uniforme en las unidades de salud.

Módulo Registro de Áreas de Salud: Gestiona la información de las Áreas de Salud a nivel nacional, permitiendo un control de las mismas, su composición según la estructura organizativa propuesta por la Atención Primaria de Salud, así como sus integrantes según la plantilla del Grupo Básico de Trabajo y Equipo Básico de Salud.

Módulo Registro de Enfermedades de Declaración Obligatoria: Gestiona en tiempo real y con alcance nacional la información sobre la incidencia de las Enfermedades de Declaración Obligatoria, facilitando así la vigilancia epidemiológica de dichas enfermedades.

1.3.2 MARCO REGULATORIO

El Marco Regulatorio del Grupo de Integración de Soluciones Informáticas (GIS) surge a raíz de la reestructuración organizacional de las entidades que rigen el proceso de desarrollo de software para la salud cubana MINSAP y CEDISAP, en el mismo se recogen todas las políticas y estrategias de desarrollo, especificaciones a cumplir, arquitectura, estándares, requerimientos y regulaciones de los sistemas informáticos desarrollados para la informatización del Sistema Nacional de Salud.

A partir del momento en que fue creado el Marco Regulatorio, es inviolable su aplicación para nuevos desarrollos, y todo sistema que quiera integrarse al RIS y al Sistema Nacional de Salud tendrá que cumplir con todas las pautas que el mismo plantea. Entre ellas están:

Las políticas y estrategias de desarrollo definidas para la informatización del Sistema Nacional de Salud por la Dirección de Informática del MINSAP(DI MINSAP), las cuales establecen que todos los proyectos que se coordinen con otros organismos y entidades dentro o fuera del país están en la obligación de reconocer y cumplir las políticas e intereses del MINSAP y admitir la evaluación, control y certificación de las soluciones informáticas para el sector de la salud pública; así como las políticas y estrategias de desarrollo definidas por el Ministerio de la Informática y las Comunicaciones (MIC) en cuanto a Seguridad, Soberanía Tecnológica y Software Libre, siendo este el rector y regulador de las infocomunicaciones cubanas.

Dentro de sus especificaciones, el Marco Regulatorio establece que todo sistema debe contar con una arquitectura sólida y robusta, integrando diferentes aspectos arquitectónicos, los cuales han sido evaluados en la siguiente tabla con respecto a su estado actual en alasMEDIGEN:

Requerimientos arquitectónicos	Marco Regulatorio	Arquitectura alasMEDIGEN	Cumple
Seguridad	El sistema deberá contar con un mecanismo de seguridad basado en el modelo de Autenticación, Autorización y Auditoría (Authentication, Authorization and Accounting, AAA) con Autenticación de firma única (Single Sign On).	Consume o brinda servicios web para interactuar con otros sistemas utilizando el componente SAAA.	Si

Confiabilidad	El sistema deberá ser capaz de prevenir o recuperarse ante posibles fallos.	El sistema cuenta con un registro de eventos que le permite recuperarse ante posibles fallos.	Si
Interfaz interna	Las operaciones que puedan ser reutilizables entre varios sistemas deberán ser desarrolladas como XML Web Services.	El sistema no cuenta con servicios web que le permitan reutilizar sus funcionalidades.	No
Software	-El software base debe responder a las políticas de Software Libre y de Fuente Abierta. -Sistema Operativo: Linux Debian Etch 4.0 rev 5. -Servidor Web: Apache 2.2.3 -Lenguaje programación: PHP 5.2.0-8 -Servidor de Bases de Datos: Mysql 5.0.32 -El acceso a los datos debe prever el balanceo de carga para una configuración de replicación Maestro – Esclavo entre servidores mysql. -Las consultas de inserción, actualización y eliminación (INSERT, UPDATE, DELETE) deben hacerse en el servidor maestro. -Las consultas de selección (SELECT) deben hacerse en el servidor esclavo. -Los clientes tendrán acceso al Sistema a través de cualquier navegador Web y debe ser operable tanto en Mozilla como Internet Explorer.	-Las tecnologías y herramientas en que está basada la aplicación informática deberán cumplir con las licencias de software libre. -Sistema Operativo: Linux Debian (Sarge). -Servidor Web: Apache 2.0. -Lenguaje de programación: PHP5. -Servidor de Bases de Datos MySQL5.0.	Si
Hardware	-La infraestructura de hardware deberá	Se especifican los	No

	permitir aumentar la cantidad de servidores o adicionar componentes de hardware en función de disminuir la carga, sin que sea necesario realizar modificaciones al software. -Las Capas de Presentación, Negocio y Datos estarán desplegadas en tres servidores dedicados. -El servidor que contiene la Capa de Presentación deberá poseer conectividad a la Red Privada Virtual de INFOMED. -El servidor que contiene la Lógica de Negocio poseerá conectividad punto a punto únicamente con el servidor que hospeda la capa de presentación y el servidor de datos. -El servidor que contiene la Capa de Datos poseerá únicamente conectividad punto a punto con el servidor que hospeda la Capa de Negocio.	requisitos se hardware siguientes Servidor de aplicación: -Microprocesador Pentium IV a 3.0 GHz o superior. -1GB de RAM o superior. Máquina cliente: -Microprocesador Pentium a 233 MHz (Recomendado: Pentium a 500MHz o superior) -64 MB de RAM (Recomendado*:* 128 MB RAM o superior) -52 MB de espacio de disco duro. (Recomendado: 120MB para la instalación completa del Internet Explorer 5.5) -La conexión al servidor debe ser a través de MODEM o tarjeta de red. -Es necesario contar con una impresora para poder imprimir los diferentes tipos de reportes.	
Diseño e implementación	-El Sistema debe ser diseñado con SOA y debe desplegarse en tres capas bien definidas. -La Capa de Presentación contendrá la interfaz de usuario y negocio de validación	-El sistema fue desarrollado con el *"framework"* Symfony, está diseñado con el patrón arquitectónico MVC y desplegado en 2	No

	de formularios de entrada de datos o funcionalidades que garanticen la usabilidad del sistema. -La Capa de Negocio contendrá los servicios que dan cumplimiento a los Requisitos Funcionales del Sistema. -La Capa de Datos contiene la Base de Datos relacional del Sistema y negocio únicamente que garantice la integridad referencial de los datos. -Todos los componentes del sistema deben desarrollarse siguiendo el principio de máxima cohesión y mínimo acoplamiento.	servidores.	
Pautas de Diseño	La resolución utilizada para el diseño de la aplicación debe ser de 800 x 600 pixel.	-La interfaz externa debe estar diseñada para verse en cualquier resolución igual o superior a 1024 x 768 pixel.	No
Ayuda y Manual de Usuario	En la Ayuda se plasmará una explicación sencilla del funcionamiento de las opciones de la aplicación y en el Manual de Usuario se dará una explicación más detallada. Debe contener por ejemplo una explicación general de la aplicación, un Glosario de términos, los Usuarios y accesos por niveles y una explicación de cada opción de la aplicación.	-Están creados una serie de manuales de usuarios y videos tutoriales, y se mantiene la asistencia a los usuarios.	Si

1.3.3 ARQUITECTURA ORIENTADA A SERVICIOS

El principal aspecto que plantea el marco regulatorio es que los sistemas para poder integrarse al RIS tienen que tener una Arquitectura Orientada a Servicios (SOA).

La Arquitectura Orientada a Servicio, es un concepto que define la utilización de servicios para todos los requisitos del negocio, es una estrategia tecnológica por la cual las aplicaciones hacen uso de los servicios disponibles en una red. SOA consiste en un método de diseño de software donde las aplicaciones de negocio se descomponen en "servicios" individuales que pueden ser utilizados independientemente de las aplicaciones de las que forman parte y de las plataformas informáticas sobre las que se ejecutan.

Las definiciones más importantes de la Arquitectura Orientada a Servicios, se exponen en los siguientes párrafos:

Según Billy Reinoso: "SOA es una arquitectura de aplicación en la cual todas las funciones se definen como servicios independientes con interfaces invocables bien definidas, que pueden ser llamadas en secuencias definidas para formar procesos de negocio". (10)

IBM plantea que: "…SOA es un modelo de componentes que interrelaciona las diferentes unidades funcionales de las aplicaciones, denominadas servicios, a través de interfaces y contratos bien definidos entre esos servicios. La interfaz se define de forma neutral, y debería ser independiente de la plataforma hardware, del sistema operativo y del lenguaje de programación utilizado. Esto permite a los servicios, construidos sobre sistemas heterogéneos, interactuar entre ellos de una manera uniforme y universal." (11)

CARACTERÍSTICAS DE LA ARQUITECTURA ORIENTADA A SERVICIOS

Una Arquitectura Orientada a Servicios define las siguientes capas de software:

Aplicativa básica: Sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad.

Exposición de funcionalidades: Donde las funcionalidades de la capa aplicativas son expuestas en forma de servicios (servicios web).

Integración de servicios: Facilitan el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración.

Composición de procesos: Define el proceso que varía en función del negocio de entrega. Además al contrario de las arquitecturas orientada a objetos, las arquitecturas SOA están formadas por servicios de aplicación débilmente acoplados y altamente interoperables.

Beneficios de una arquitectura SOA:

- ✓ Permite justificar más claramente las inversiones en tecnologías de información, ya que éstas están más alineadas con el negocio.

- ✓ Permite la creación y cambio de servicios de forma incremental, evitando proyectos de larga duración y alto costo.
- ✓ Las tecnologías de información de las empresas están compuestas por un gran número de sistemas interdependientes, heterogéneos y muchas veces redundantes.
- ✓ La rapidez con que las tecnologías de información pueden adaptarse a los cambios en las necesidades del negocio no siempre es suficiente (falta de agilidad y flexibilidad).
- ✓ La estructura actual de las tecnologías de información puede hacer que los cambios introducidos cuesten más que los beneficios que aportan (baja eficiencia de costos). (11)

FUNCIONAMIENTO DE LA ARQUITECTURA SOA

Las colaboraciones en SOA siguen el paradigma *publicar*, *descubrir* e *invocar*, donde un consumidor de servicios realiza la localización dinámica de un servicio consultando el registro de servicios para hallar uno que cumpla con un determinado criterio. Si el servicio existe, el registro proporciona al consumidor la interfaz de contrato y la dirección del servicio proveedor. (13)

En la figura se muestran las entidades (roles, operaciones y artefactos) de una Arquitectura Orientada a Servicios:

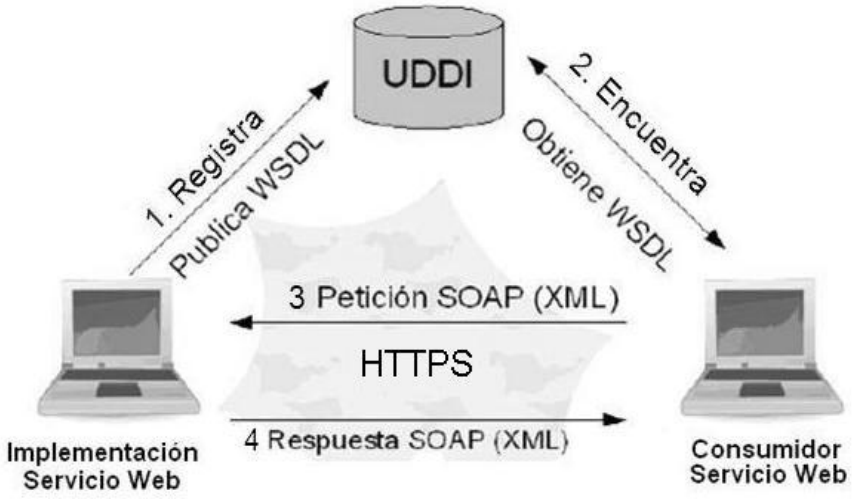

Figura 2. Colaboraciones y roles de una arquitectura SOA.

Los roles son:

Cliente del servicio: Es el que solicita la ejecución del servicio web, y por lo tanto el que lo consume.

Proveedor del servicio: Es el encargado de implementar el servicio web y ofrecerlo a los clientes.

Registro del servicio: Es un repositorio donde se almacenan las descripciones de los servicios, para que así los clientes puedan buscar el servicio web que mejor se adapte a sus necesidades.

La secuencia de ejecución más detallada es la siguiente:

1. El proveedor del servicio da de alta el servicio web en el registro. Para realizar esto, el proveedor almacena en el registro el documento de descripción de éste.
2. El solicitante del servicio busca en el registro un servicio web que pueda adaptarse a sus necesidades.
3. Una vez seleccionado el servicio, el solicitante lo invoca mediante el envío de un mensaje SOAP, en el cual se indica la acción a realizar y los datos de entrada.
4. El servicio web recibe la petición y ejecuta la funcionalidad. Para finalizar envía un mensaje SOAP al solicitante con los resultados obtenidos.

ELEMENTOS DE UNA ARQUITECTURA SOA

Existen diferentes elementos SOA agrupados en dos bloques, uno que abarca los aspectos funcionales de la arquitectura y otro que abarca aspectos de calidad de servicio.

Figura 3. Componentes de una arquitectura SOA.

Aspectos funcionales de la arquitectura

PROTOCOLO DE TRANSPORTE: Es el mecanismo utilizado para llevar las demandas de servicio de un consumidor a un proveedor de servicio y devolver las respuestas. En una arquitectura SOA existen diferentes protocolos de transporte, ellos son:

- ✓ **Protocolo Seguro de Transferencia de Hipertexto (HTTPS: Hypertext Transfer Protocol Secure):** "Es un protocolo de red basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, utiliza un cifrado basado en las Secure Socket Layers (SSL) para crear un canal cifrado (cuyo nivel de cifrado depende del servidor remoto y del navegador utilizado por el cliente) más apropiado para el tráfico de información sensible". (14) La idea principal de HTTPS es la de crear un canal seguro sobre una red insegura. Es utilizado principalmente por entidades bancarias, tiendas en línea, y cualquier tipo de servicio que requiera el envío de datos personales o contraseñas.
- ✓ **Servicio de Mensajes Java (JMS: Java Message Service):** Este es un estándar de mensajería que permite a los componentes de aplicaciones basados en la plataforma Java2 crear, enviar, recibir y leer mensajes. También hace posible la comunicación confiable de manera sincrónica y asincrónica.
- ✓ **Protocolo Simple de Transferencia de Correo (SMTP: Simple Mail Transfer Protocol):** Protocolo de red basado en texto utilizado para el intercambio de mensajes de correo electrónico entre computadoras u otros dispositivos. SMTP se basa en el modelo cliente-servidor, donde un cliente envía un mensaje a uno o varios receptores. La comunicación entre el cliente y el servidor consiste enteramente en líneas de texto compuestas por caracteres ASCII.
- ✓ **Protocolo de Transferencia de Hipertexto (HTTP: Hypertext Transfer Protocol):** Es un protocolo orientado a transacciones y sigue el esquema petición-respuesta entre un cliente y un servidor. HTTP es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores. El desarrollo de aplicaciones web necesita frecuentemente mantener estado. Para esto se usan las *"cookies"*, que es la información que un servidor puede almacenar en el sistema cliente. Esto le permite a las aplicaciones web instituir la noción de "sesión", y también permite rastrear usuarios ya que las cookies pueden guardarse en el cliente por tiempo indeterminado.

El protocolo de transporte que se va a utilizar es HTTPS por la sensibilidad de la información que se maneja.

PROTOCOLO DE COMUNICACIÓN DE SERVICIOS: Es el mecanismo por el que proveedor y consumidor se comunican cuando se solicita o responde, respectivamente. El protocolo de comunicación de la arquitectura SOA es SOAP:

- ✓ **Protocolo de Acceso a Servicios (SOAP: Services Object Access Protocol):** Es un protocolo para el intercambio de mensajes basados en XML (normalmente bajo HTTP), permite la transmisión de información. El elemento básico en esta transmisión es el mensaje SOAP (Anexo 1). Un mensaje está compuesto por tres partes fundamentales y una cuarta que se omite en la mayoría de los mensajes:
- ✓ **El sobre o envoltura:** Encapsula el contenido del mensaje, identifica si es un mensaje SOAP o cualquier otro tipo de mensaje.
- ✓ **El encabezado:** Contiene las extensiones definidas para usar en SOAP. Puede invocar actividades adicionales como autenticación, también contiene información de seguridad como los *"tokens"* y puede que algunos mensajes no porten ningún encabezado.
- ✓ **Cuerpo:** Contiene el mensaje, ya sea la petición de servicio o la respuesta.
- ✓ **Fallos:** Es el responsable de los mensajes de error que genere el mensaje. En la mayoría de los mensajes no aparece.

Por ser basado en XML es independiente de sistemas operativos o plataformas y puede viajar a través de cualquier protocolo de comunicación (ejemplo: HTTP o TCP/IP). Aunque en SOA lo más común es el mensaje SOAP viaje utilizando HTTP como protocolo de comunicación porque trabaja bien y es rápido para documentos estáticos.

EL mensaje SOAP interviene en el descubrimiento y la publicación de los servicio en la UDDI y en el intercambio de peticiones/respuestas entre el consumidor y el proveedor de servicios. Los proveedores de servicio pueden enviar un mensaje SOAP a la UDDI para publicar sus servicios. Mediante mensajes SOAP los clientes pueden comunicarse con la UDDI en busca de determinados servicios y luego de encontrar el servicio, el cliente se comunica con su proveedor a través de mensajes SOAP. Como se puede apreciar es un estándar muy usado en un entorno SOA.

Algunas de las Ventajas de SOAP son:

- ✓ **No está asociado con ningún lenguaje:** Los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el último y mejor lenguaje de programación que exista pero los desarrolladores responsables de mantener antiguas aflicciones heredadas podrían no poder hacer esta elección sobre el lenguaje de programación que utilizan. SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación.
- ✓ **No se encuentra fuertemente asociado a ningún protocolo de transporte:** La especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de

SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.

- ✓ **No está atado a ninguna infraestructura de objeto distribuido:** La mayoría de los sistemas de objetos distribuidos se pueden extender, y ya están algunos de ellos para que admitan SOAP.
- ✓ **Aprovecha los estándares existentes en la industria:** Los principales contribuyentes a la especificación SOAP evitaron, intencionadamente, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML. Y como ya se ha mencionado SOAP no define un medio de transporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.
- ✓ **Permite la interoperabilidad entre múltiples entornos:** SOAP se desarrolló sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dichos estándares pueden comunicarse mediante mensajes SOAP con aplicaciones que se ejecuten en otras plataformas.

DESCRIPCIÓN DE SERVICIO: Es un esquema acordado para describir el servicio, cómo debe invocarse, y qué datos requiere para invocarse correctamente. En la arquitectura SOA la descripción de servicios viene dada por dos protocolos:

- ✓ **Lenguage de Marcado Extensible (XML: Extensible Markup Language):** Es un conjunto de reglas para definir etiquetas semánticas que organizan un documento en diferentes partes. XML es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados. Integra los datos de las fuentes más dispares y permite realizar el intercambio de documentos entre las aplicaciones tanto en la propia PC como en una red local o extensa. La codificación del contenido web en XML permite que la estructura de la información resulte más accesible.
- ✓ **Lenguaje de descripción de Servicios Web (WSDL: Web Services Description Language):** Un Lenguaje de Descripción de Servicios Web no es más que un lenguaje basado en XML para describir el comportamiento de un servicio web. Contiene las operaciones y los tipos de datos necesarios para definir dichas operaciones. Todo servicio web debe proveer su WSDL para que otros puedan invocarlo.

Es muy importante entender WSDL porque es la parte fundamental para desarrollar servicios. No es necesario saber construir un documento WSDL (ya que lo construyen automáticamente las herramientas de desarrollo), pero sí entenderlo.

WSDL es un lenguaje basado en XML creado para definir la interfaz de los servicios. Un documento WSDL está divido en dos partes claramente diferenciadas:

Parte concreta: Es la parte que define el "cómo" y "dónde".

Parte abstracta: Es la parte que define qué hace el servicio a través de los mensajes que envía y recibe.

A continuación se detallarán brevemente cada una de estas partes que componen un documento WSDL:

En la parte abstracta se tiene:

portType: Define el conjunto de operaciones que soporta el Servicio Web. Una operación no es más que un grupo de mensajes que serán intercambiados. Cada operación puede enviar o recibir al menos un mensaje cada vez.

types: Esta etiqueta define las estructuras de datos que se utilizarán para construir los mensajes tanto de petición como de respuesta. Estas estructuras de datos pueden construirse con cualquier lenguaje, pero lo más normal es hacerlo con XML Schema. Este apartado, es el más complicado sobre todo cuando se tenga que construir estructuras de datos muy complejas.

message: Describe los mensajes que se van a intercambiar entre el cliente y el Servicio Web. Un mensaje puede estar dividido en varias partes, por ejemplo, si en un mensaje se quieren enviar datos y una imagen al mismo tiempo.

En la parte concreta se tiene:

binding: Describe como formatear los mensajes para interactuar con un Servicio determinado. WSDL no define un estándar para formatear mensajes. Para ello utiliza la extensibilidad permitiendo definir como intercambiar los mensajes usando SOAP, HTTP, MIME, etc.

services: Este elemento indica dónde se encuentra el Servicio usando la etiqueta . Cada etiqueta define el formato de los mensajes, y la dirección donde se encuentra el servicio que acepta mensajes en ese formato.

En la presente investigación se va a utilizar WSDL como descripción de servicios para la propuesta arquitectónica.

SERVICIOS (ESB): Una Empresa de Servicios o Enterprise Service Bus es una plataforma de integración de sistemas centrada en estándares abiertos que implementa interfaces para proveer comunicación, conectividad, transformación, portabilidad y seguridad. Se ha convertido en una de las partes más

importantes de una arquitectura SOA ya que se presenta como el "mediador" multiprotocolo y multipropósito para dicha arquitectura.

Cuando se habla de ESB se refiere a la comunicación, con esto se tiene necesidad de más cosas como enrutamiento y trasformación de mensajes, mediación de protocolos y manejo de eventos, componentes esenciales de un ESB. A continuación se muestra qué son cada uno de estos términos.

ESB tiene cuatro funciones fundamentales:

Enrutamiento de Mensajes: El ruteo es una función clave para la virtualización de servicios. Estableciendo un nivel intermedio de abstracción entre el invocador y el servicio a invocar por medio de otro encargado de la localización de este que debe satisfacer el requerimiento.

Transformación de mensajes: Un mensaje entrante es enviado ocupando un protocolo diferente al original. Ejemplo un mensaje en formato largo-posición puede ser transformado en un mensaje en formato XML para la invocación de un Web Service.

Mediación de protocolos: Una aplicación cliente envía un requerimiento usando un protocolo diferente al de invocación del servicio requerido. Ejemplo, el mensaje de la aplicación cliente podría utilizar HTTP, mientras que el mensaje saliente requerido para invocar el servicio podría usar WebSphere MQ.

Manejo de eventos: Un mensaje entrante para un evento es enviado a una serie de puntos finales ("*endpoints*"), normalmente empleando métodos de publicación y suscripción.

No existe un estándar para implementar y configurar un ESB, este se crea de acuerdo a la conectividad que exista para integrar los sistemas de una empresa. En un sistema como alasMEDIGEN en el que se tienen gran cantidad de servicios expuestos y numerosos consumidores de estos, no es aconsejable dejar que estas dependencias se realicen punto a punto aun garantizando la estabilidad de las comunicaciones, por lo que se recomienda crear una capa lógica (ESB) para mediar entre estos servicios.

PROCESOS DE NEGOCIO (BPM): Gestión de Procesos de Negocio (BPM: Business Process Management): Es el conjunto de servicios y herramientas que facilitan la administración de procesos de negocio. BPM también es vista como una disciplina de administración, que requiere que las organizaciones se cambien a un pensamiento centrado en los procesos y que reduzcan su dependencia de estructuras tradicionales de territorio y funcionalidad. Es un enfoque estructurado que emplea métodos, políticas, métricas, prácticas de administración, y herramientas de software para mejorar la agilidad y el desempeño operacional.

En la presente investigación no se hará uso de un BMP para modelar los procesos del negocio debido a que los mismos ya se encuentran descritos utilizando la metodología RUP.

REGISTRO DE SERVICIOS: Es un repositorio de descripciones de servicios y datos que pueden utilizar proveedores de servicios para publicar sus servicios, así como consumidores de servicios para descubrir o hallar servicios disponibles. En SOA este registro es UDDI.

- ✓ **Descripción, Descubrimiento e Integración Universal (UDDI: Universal Description, Discovery and Integration):** UDDI es un registro donde se publica y descubre información referente a los servicios, información técnica de las características de los servicios y sus interfaces de comunicación y las reglas de la empresa para el uso de los mismos.

Es un registro público para que las empresas puedan acceder a él en busca de los servicios que necesitan o publicar los suyos. Brinda dos interfaces, una para el proveedor de servicios y otra para el consumidor de dichos servicios.

UDDI está estrechamente relacionada con el estándar de descripción WSDL. Ella es quien posibilita que las empresas coloquen sus archivos WSDL en la red, así los usuarios pueden conocer cuáles son los servicios que brinda cada empresa y además encontrar en la descripción WSDL la forma de invocar esos servicios.

Actualmente el Ministerio de Salud Pública no tiene una UDDI implementada para dar soporte a sus servicios web, debido a esto no se registrarán los propuestos en la presente investigación.

1.4 MÉTODOS BASADOS EN LA ARQUITECTURA DE SOFTWARE

Existen diversos métodos para evaluar la arquitectura de software, para la presente investigación se analizaron los que se muestran a continuación:

Análisis de Compensación de Arquitectura (Architecture Tradeoff Analysis Method, ATAM)

Este método está inspirado en tres áreas distintas: los estilos arquitectónicos, el análisis de atributos de calidad y el método de evaluación **Método de Análisis de Arquitectura de Software (Software Architecture Analysis Method, SAAM)**. El método surge del hecho de que revela la forma en que una arquitectura específica satisface ciertos atributos de calidad, y provee una visión de cómo estos atributos interactúan con otros; o sea, los tipos de *acuerdos ("trade-off")* que se establecen entre ellos.

ATAM se concentra en la identificación de los estilos arquitectónicos ya que estos elementos representan los medios empleados por la arquitectura para alcanzar los atributos de calidad, así como también permiten describir la forma en la que el sistema puede crecer, responder a cambios, e integrarse con otros sistemas.

El ATAM confía fuertemente en la identificación de los puntos de sensibilidad, los riesgos y fortalezas de la arquitectura. Un aspecto del método es que no solo encuentra riesgos, sino que también localiza no riesgos, ya que ambos están relacionados con las respuestas arquitectónicas provenientes de las decisiones arquitectónicas.

Talleres de Atributos de Calidad (Quality Attribute Workshop, QAW)

Los Talleres de Atributo de Calidad fueron creados para complementar ATAM. Este método se basa en la identificación de los atributos de calidad que dirigen el proceso de diseño de la arquitectura, por lo que su aplicación es previa al diseño arquitectónico. Este método pretende definir el significado de los atributos de calidad en el contexto del sistema en desarrollo, la manera de descubrir, caracterizar y priorizar estos atributos, y la forma de utilizar esta información. El resultado de este proceso es una lista de factores que van a dirigir el diseño de la arquitectura y una lista de escenarios que servirán para evaluar los atributos de calidad.

Revisiones Activas para Diseños Intermedios (Active Reviews for Intermediate Designs, ARID)

ARID es el resultado de combinar ATAM y **Revisiones de Diseño Activas (Active Design Reviews, ADR)**.

Por su parte, ADR es utilizado para la evaluación de diseños detallados de unidades del software como los componentes o módulos. Este método se basa en que los participantes realicen tareas que son cuidadosamente estructuradas para realizarlas en entrevistas y que las respuestas obliguen a utilizar el diseño mostrando el estado actual de este.

ARID toma lo mejor de ATAM y ADR, constituyendo un método conveniente para realizar la evaluación de diseños parciales en las etapas tempranas del desarrollo, aunque se puede aplicar sobre el sistema completo. Su objetivo no es la sustitución de ATAM, sino que más bien prepara el camino en sistemas en los que por su complejidad, puedan ser necesarias revisiones de la arquitectura en etapas intermedias del diseño, tomando de ATAM la idea de la generación de escenarios por parte de los interesados, donde los usuarios podrán decirle a los diseñadores que es lo que necesitan. ARID, de forma similar toma lo mejor de las ADR, como la participación activa de los entrevistados lo cual asegura alta fidelidad en las respuestas.

La evaluación con ARID se basa en detallar el funcionamiento del sistema en una serie de escenarios predefinidos, a través de lo cual se pueden identificar fortalezas y debilidades en la arquitectura.

Un escenario consta de tres partes: el estímulo, el ambiente y la respuesta. Estos describen una utilización anticipada o deseada del sistema, representando una abstracción de los requerimientos más importantes que el sistema debe cumplir.

1.5 CONCLUSIONES

Previendo la posibilidad de incluir nuevos sistemas integrados y siguiendo las políticas del Marco Regulatorio del GIS surge la necesidad de redefinir la arquitectura de alasMEDIGEN.Sistema Informático de Genética Médica. Para ello se asume la arquitectura SOA, la cual constituye un modelo escasamente acoplado de gran flexibilidad para la definición, implementación, sustitución, evolución de los servicios y funcionalidades del sistema. Dando cumplimiento a lo mismo se define como protocolo de transporte HTTPS, de comunicación SOAP y para la descripción de servicios WSDL, y se realiza un análisis de otros elementos que dan soporte a los mismos como UDDI y ESB.

Después de realizar un estudio de los métodos basados en la evaluación de la arquitectura de software, se decide utilizar ARID, con el cual se realiza la evaluación temprana de la arquitectura sobre diseños intermedios de la misma, permitiendo modificarla en caso de encontrar riesgos que puedan traer un posterior fracaso del proyecto.

CAPÍTULO 2

DISEÑO DE LA PROPUESTA ARQUITECTÓNICA

En este capítulo se presenta el diseño de la propuesta arquitectónica orientada a servicios de alasMEDIGEN.Sistema Informático de Genética Médica, tributando al cumplimiento de las pautas arquitectónicas que propone el Marco Regulatorio del GIS. Para ello se especifica el ciclo de vida de la arquitectura a diseñar y se definen las vista de casos de uso, vista lógica y de despliegue del sistema.

2.1 CICLO DE VIDA DE UNA ARQUITECTURA ORIENTADA A SERVICIOS

Según Daniels: “El ciclo de vida define las fases y las tareas esenciales para el desarrollo de sistemas, sin importar el tipo o la envergadura del sistema que se intenta construir”. (15)

Cuando se habla de una arquitectura SOA, refiriéndose a servicios y asociado entre ellos, se tiene que hablar de ciclo de vida. El ciclo de vida de SOA se adecua a las necesidades del proyecto y generalmente está compuesto de seis fases: Análisis, Diseño, Desarrollo, Prueba, Implantación y Administración, las cuales son respaldadas por un grupo de procesos que permiten asegurar que haya concordancia y que las políticas operacionales no sean rebasadas, que los cambios ocurran en un ambiente controlado y con la autoridad apropiada como fue establecido en el diseño del negocio.

Las fases son las siguientes:

Análisis: En esta primera fase se determina el alcance del proyecto SOA. Antes de modelar el esquema de servicios se comienza por analizar en detalle el flujo. Luego surgen los principales servicios candidatos, y se definen las capas a utilizar.

Diseño: Una vez definido el análisis, se puede comenzar a diseñar de qué forma implementarlo. Esta fase es por lo general dirigida en base a estándares e incorpora principios y convenciones establecidas para sistemas orientados a servicios.

Desarrollo: Una vez determinadas las tecnologías sobre las cuales se construirán los componentes de la arquitectura orientada a servicios, sólo basta construirlos. Existe una gran diversidad de tecnologías, herramientas, *“frameworks”* y plataformas para simplificar el desarrollo del proyecto.

Pruebas: Debido a que los servicios serán potencialmente módulos reutilizables en una gran variedad de escenarios, su calidad debe ser rigurosamente controlada.

Implantación: En base a las tecnologías específicas seleccionadas para el desarrollo se podrán definir las actividades que formen parte de la fase de implantación.

Administración: La naturaleza de los factores a administrar para este tipo de sistemas va a ser muy similar a la utilizada para sistemas distribuidos basados en componentes. La administración incluye el monitoreo de servicios, control de versiones, seguimiento de mensajes, detección de cuellos de botella.

Si bien las 6 fases son de importancia debido a que proporcionan una guía para el desarrollo e implantación de una arquitectura SOA en un proyecto, hay que partir diciendo que alasMEDIGEN ya pasó el proceso de análisis.

El presente trabajo de diploma se centrará en dos etapas del ciclo de vida, el análisis que se realizará sobre la base de los Requisitos Funcionales (RF) capturados y el análisis realizado al sistema alasMEDIGEN y en el diseño que se desarrollará en este capítulo.

Después de la evaluación de la presente investigación por el GIS se pasará a la implementación. El sistema de gobierno se establece a través de la SAAA, y la estructura jerárquica de salud, donde el profesional puede tener 3 niveles, entre estos se encuentran: Genetista Nacional, Provincial y Municipal y 2 tipo de acceso, los cuales son: Editor y Visualizador.

2.2 PRINCIPIOS DEL DISEÑO PARA LA ORIENTACIÓN A SERVICIOS

El principal componente de una Arquitectura Orientada a Servicios lo constituyen los servicios. A raíz de esto se plantearán: Cuáles son los principios de la orientación a servicios, qué tipos de servicios existen, qué características debe cumplir un servicio.

No existe una definición estándar de cuáles son los principios de una arquitectura SOA, por lo tanto, se proporcionan un conjunto de principios que guían la etapa de diseño. Estos principios son:

Los servicios deben ser reusables: Todo servicio debe ser diseñado y construido pensando en su reutilización dentro de la misma aplicación, dentro del dominio de aplicaciones de la empresa o incluso dentro del dominio público para su uso masivo.

Los servicios deben proporcionar un contrato formal: Todo servicio desarrollado, debe proporcionar un contrato en el cual figuren: el nombre del servicio, su forma de acceso, las funcionalidades que ofrece, los datos de entrada de cada una de las funcionalidades y los datos de salida. De esta manera, todo consumidor del servicio, accederá a éste mediante el contrato, logrando así la independencia entre el consumidor y la implementación del propio servicio. En el caso de los servicios web, esto se logrará mediante la definición de interfaces con WSDL.

Los servicios deben tener bajo acoplamiento: Los servicios tienen que ser independientes los unos de los otros. Para lograr ese bajo acoplamiento, lo que se realizará es que cada vez que se vaya a ejecutar

un servicio, se accederá a él a través del contrato, logrando así la independencia entre el servicio que se va a ejecutar y el que lo llama. Si se consigue este bajo acoplamiento, entonces los servicios podrán ser totalmente reutilizables.

Los servicios deben permitir la composición: Todo servicio debe ser construido de tal manera que pueda ser utilizado para construir servicios genéricos de más alto nivel, el cual estará compuesto de servicios de más bajo nivel. En el caso de los servicios web, esto se logrará mediante el uso de los protocolos para orquestación y coreografía.

Los servicios deben de ser autónomos: Todo servicio debe tener su propio entorno de ejecución. De esta manera el servicio es totalmente independiente y se podrá asegurar que sea reutilizable desde el punto de vista de la plataforma de ejecución.

Los servicios no deben tener estado: Un servicio no debe guardar ningún tipo de información. Esto es así porque una aplicación está formada por un conjunto de servicios, lo que implica que si un servicio almacena algún tipo de información, se pueden producir problemas de inconsistencia de datos. La solución: que un servicio contenga la lógica y toda la información se almacene en algún sistema de información.

Los servicios deben poder ser descubiertos: Todo servicio debe poder ser descubierto de alguna forma para que pueda ser utilizado, consiguiendo así evitar la creación accidental de servicios que proporcionen las mismas funcionalidades. En el caso de los servicios web, el descubrimiento se logrará publicando las interfaces de los servicios en registros UDDI.

Cuando se desarrollan aplicaciones SOA es muy útil y necesario tener en cuenta siempre estos principios, ya que van a dar las pautas necesarias para tomar ciertas decisiones de diseño.

2.3 REPRESENTACIÓN ARQUITECTÓNICA

RUP propone 4+1 vistas arquitectónicas para la descripción del sistema, en la presente investigación solamente se utilizan las vistas correspondientes a las fases de Análisis y Diseño:

- ✓ **Vista de casos de uso:** “Muestra los casos de uso arquitectónicamente significativos del modelo de caso de uso del sistema”. (16) RUP cataloga esta vista como obligatoria en el proceso de desarrollo.
- ✓ **Vista lógica:** “Proporciona la base para comprender la estructura y la organización del diseño del sistema”. (16) Describe las clases más importantes que formarán parte del ciclo de

desarrollo atendiendo a los niveles de abstracción. RUP define a esta vista como obligatoria en el proceso de desarrollo.

- ✓ **Vista de despliegue:** "Muestra la distribución física de procesos del sistema" (16). Da una medida de la tecnología necesaria para el correcto funcionamiento del sistema a desarrollar. RUP define esta vista como opcional.

Las vistas anteriormente explicadas responden a las necesidades de las distintas partes interesadas: clientes, programadores, administradores, etc. Para cada uno de estos, se presenta una visión resumida del sistema con la información que requieren para satisfacer sus necesidades.

2.4 VISTA DE CASOS DE USO

Partiendo de la vista de CU de la arquitectura de alasMEDIGEN se reestructuran las dependencias entre las funcionalidades ya definidas, las que se mantendrán disponibles para cada registro del RIS, ahora de forma independiente.

El caso de uso más importante es el Gestionar Datos Primarios el mismo es implementado por el módulo RECUHCL y reutilizado por los módulos RECUDIS, RECUGEM, RECUMAC, RECURM y Teleconsulta.

La vista de casos de uso del sistema está definida como se muestra en la siguiente figura:

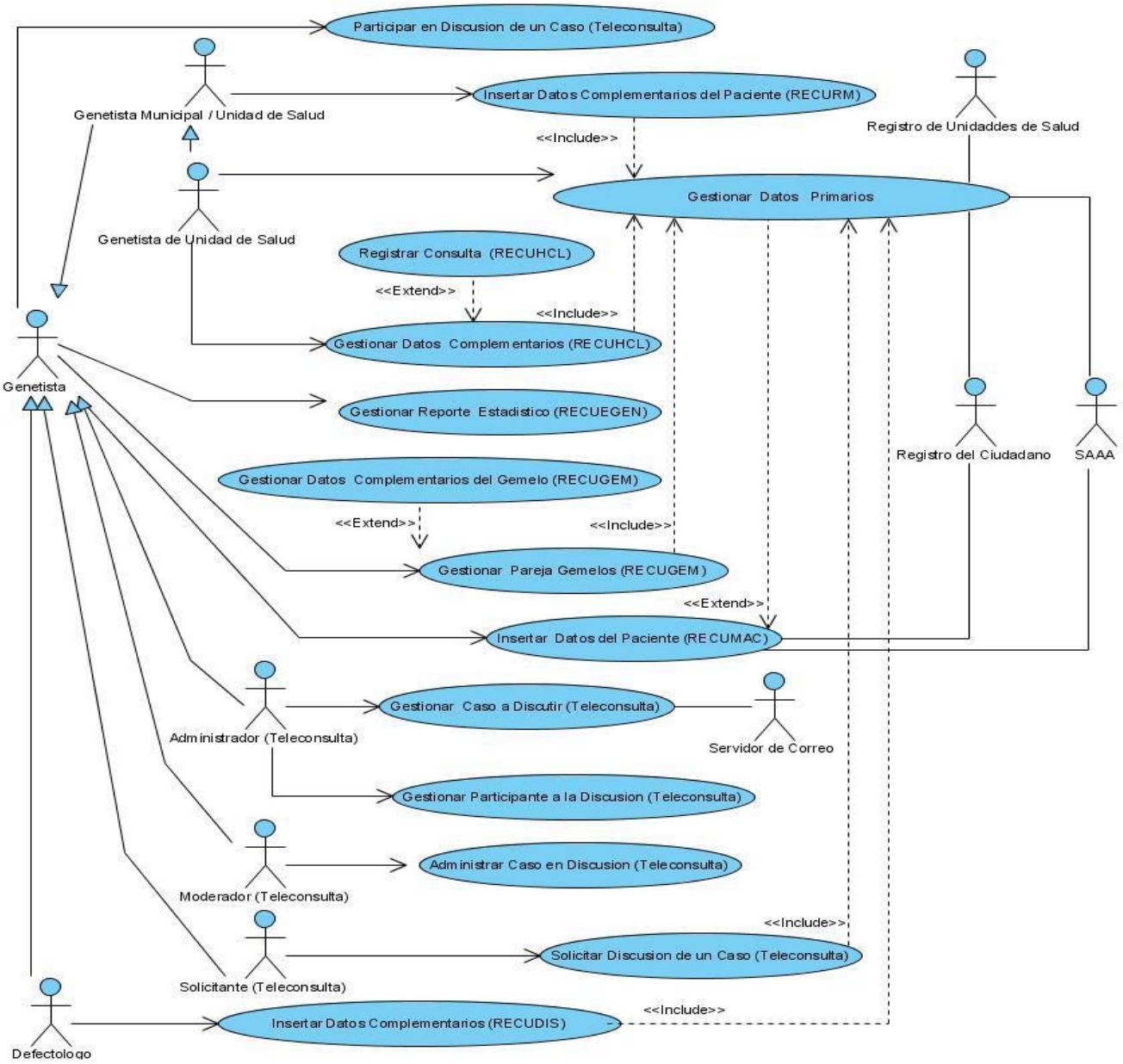

Figura 4. Vista de casos de uso del alasMEDIGEN.

2.5 REDEFINICIÓN DE LA VISTA DE CASOS DE USOS

En la presente investigación se redefine la vista de casos de uso por la integración que se va a realizar entre los diferentes sistemas a través de servicios web, los mismos se van a modelar como actores.

Para mayor detalle se realiza la explicación del CU RECUHCL.

El actor Genetista de Unidad de Salud inicia el caso de uso Gestionar Datos Primarios el cual es el CU más importante debido a que el mismo gestiona toda la información de los pacientes, este se conecta mediante el protocolo soap con los sistemas del RIS: SAAA componente de Seguridad encargado de permitir el acceso a los demás registros del RIS, la autenticación debe ser la primera acción del usuario en el sistema, la misma consiste en suministrar un nombre de usuario único y una contraseña que debe ser de conocimiento exclusivo de la persona que se autentica. Si el usuario autenticado no se encuentra registrado se reporta un error de acceso. En caso contrario, se autoriza su acceso y se crea un certificado digital y se retornan todos los datos y permisos del usuario.

Registro de Ciudadanos contiene toda la información y los datos personales de cualquier ciudadano que trabaje en la salud y/o reciba sus servicios.

Registro Unidades de Salud se tiene toda la información detallada de las Unidades de Salud.

Registro Localidades almacena información detallada de los Consejos Populares, Circunscripciones, Zonas y CDR del país.

Registro Ubicación posee toda la información detallada de la división política-administrativa del país, las provincias con sus municipios, localidades, calles y manzanas.

Registro del Clasificador Internacional de Enfermedades y Problemas de Salud (RCIEPS) este registro gestiona la estructura de la Clasificación Internacional de Enfermedades y Problemas Relacionados con la Salud.

El CU Gestionar Datos Primarios incluye el CU Gestionar Datos Primarios y extiende el CU Registrar Consulta.

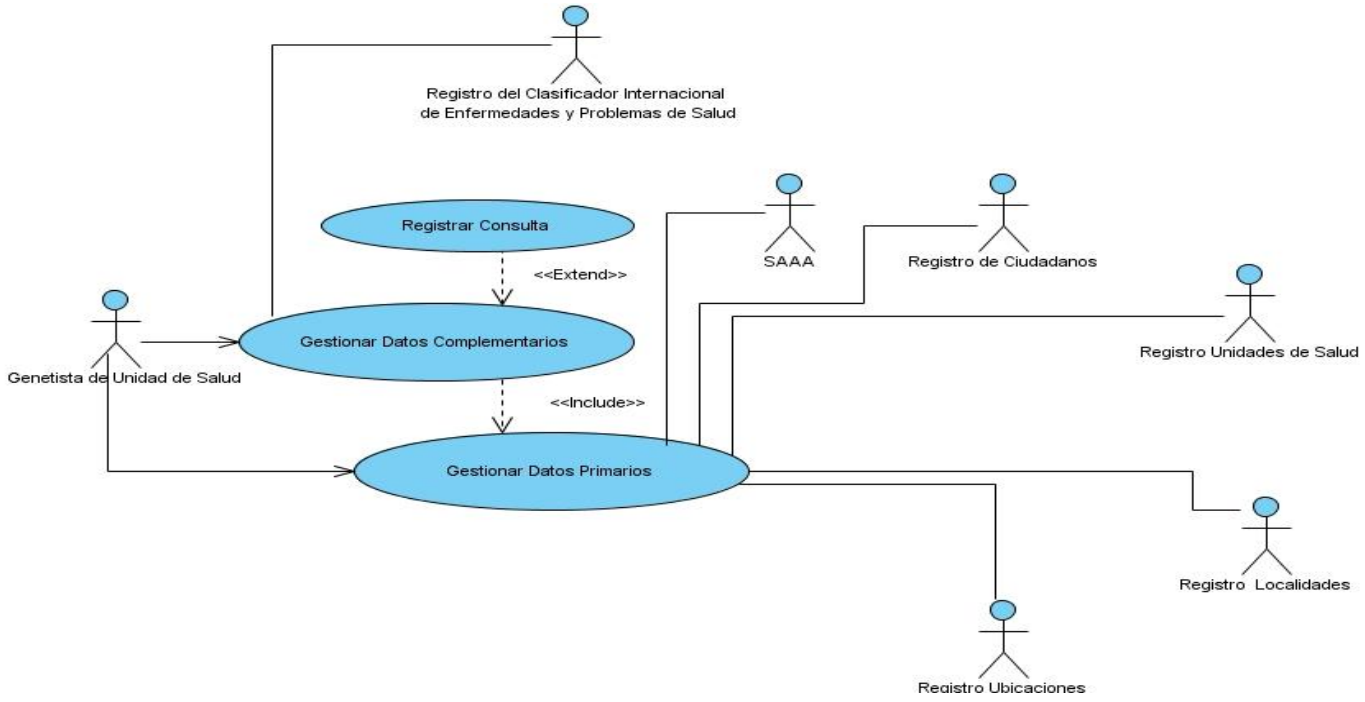

Figura 5. Diagrama de CU. RECUHCL.

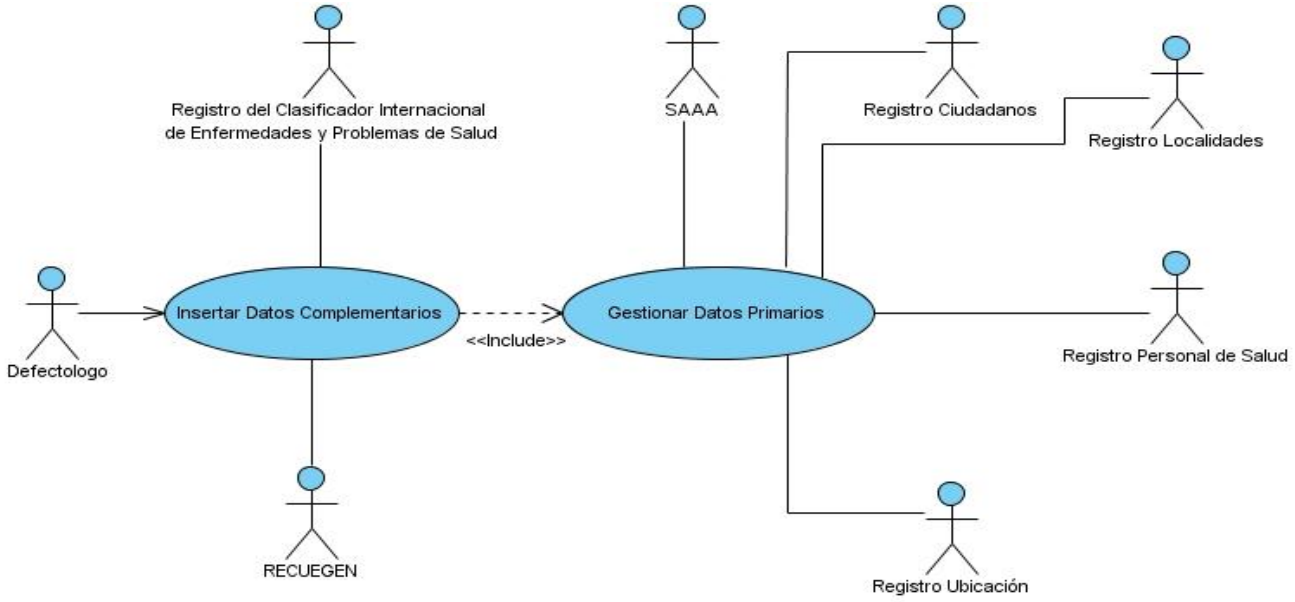

Figura 6. Diagrama de CU Críticos. RECUDIS.

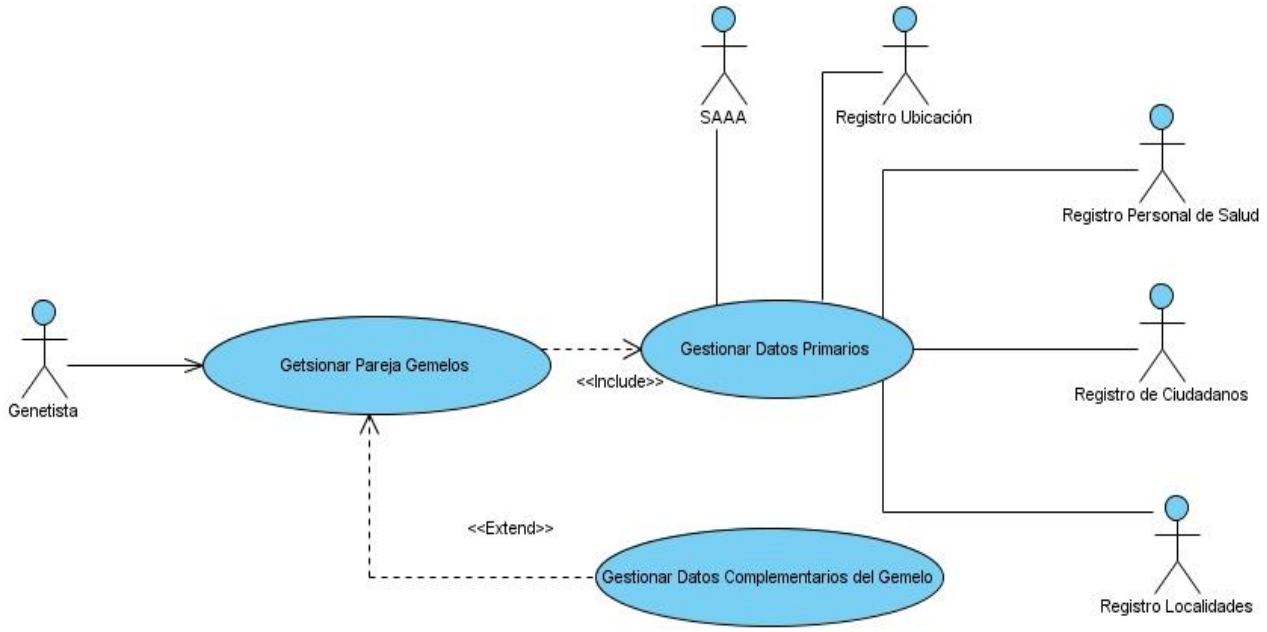

Figura 7. Diagrama de CU Críticos. RECUGEM.

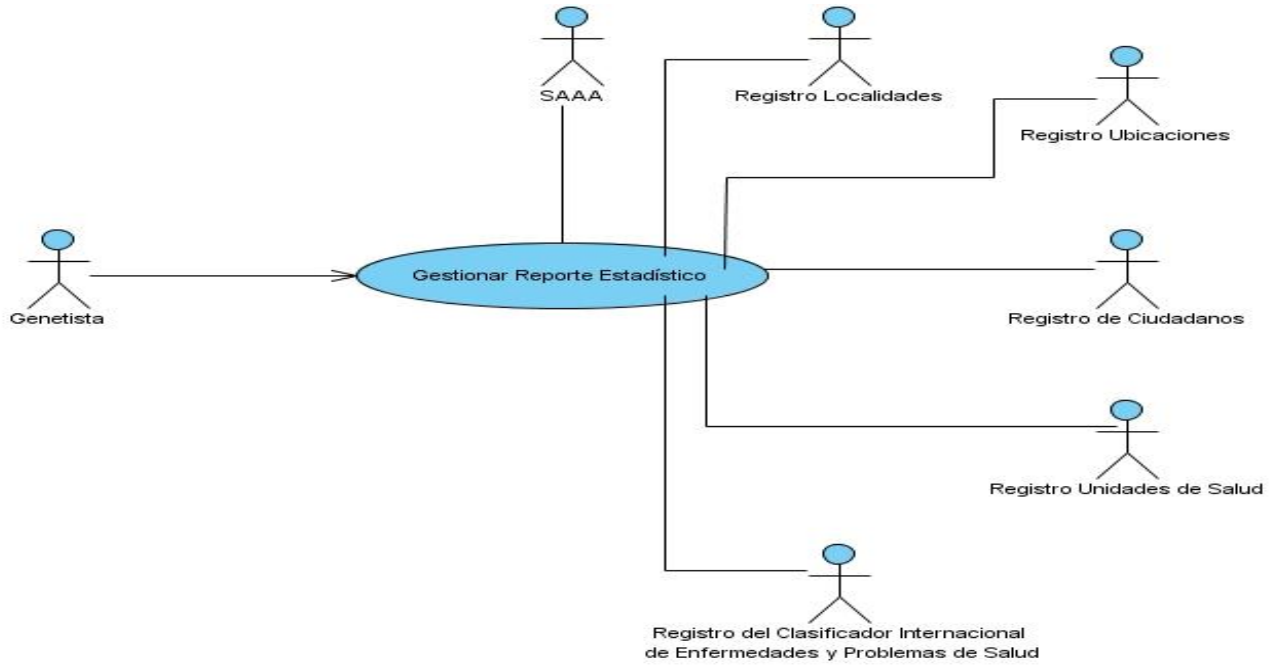

Figura 8. Diagrama de CU Críticos. RECUEGEN.

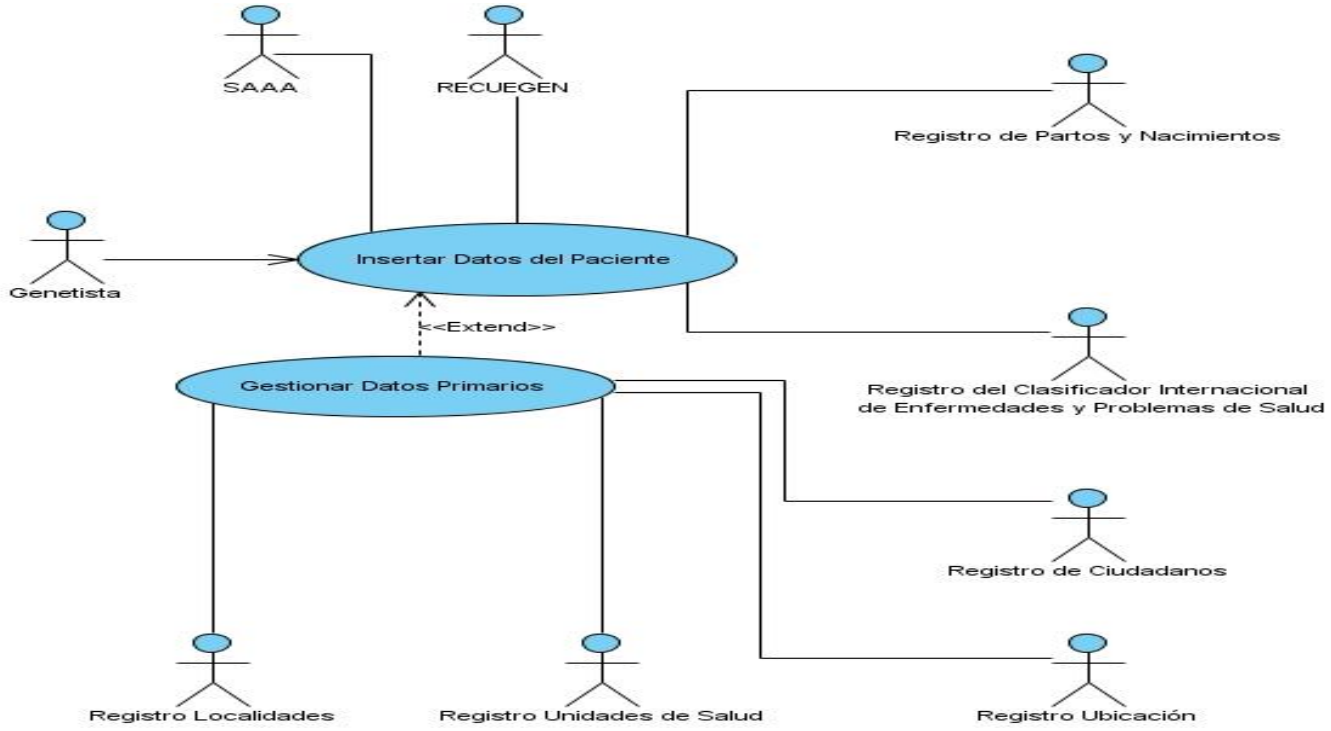

Figura 9. Diagrama de CU Críticos. RECUMAC.

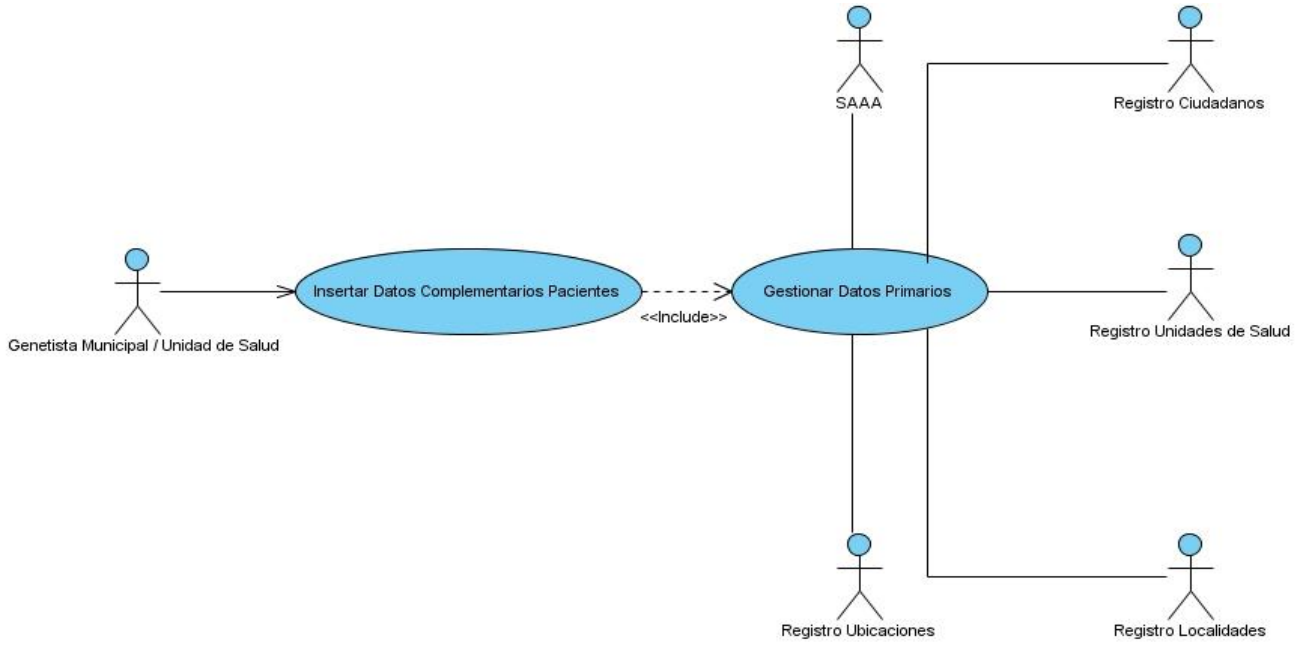

Figura 10. Diagrama de CU Críticos. RECURM.

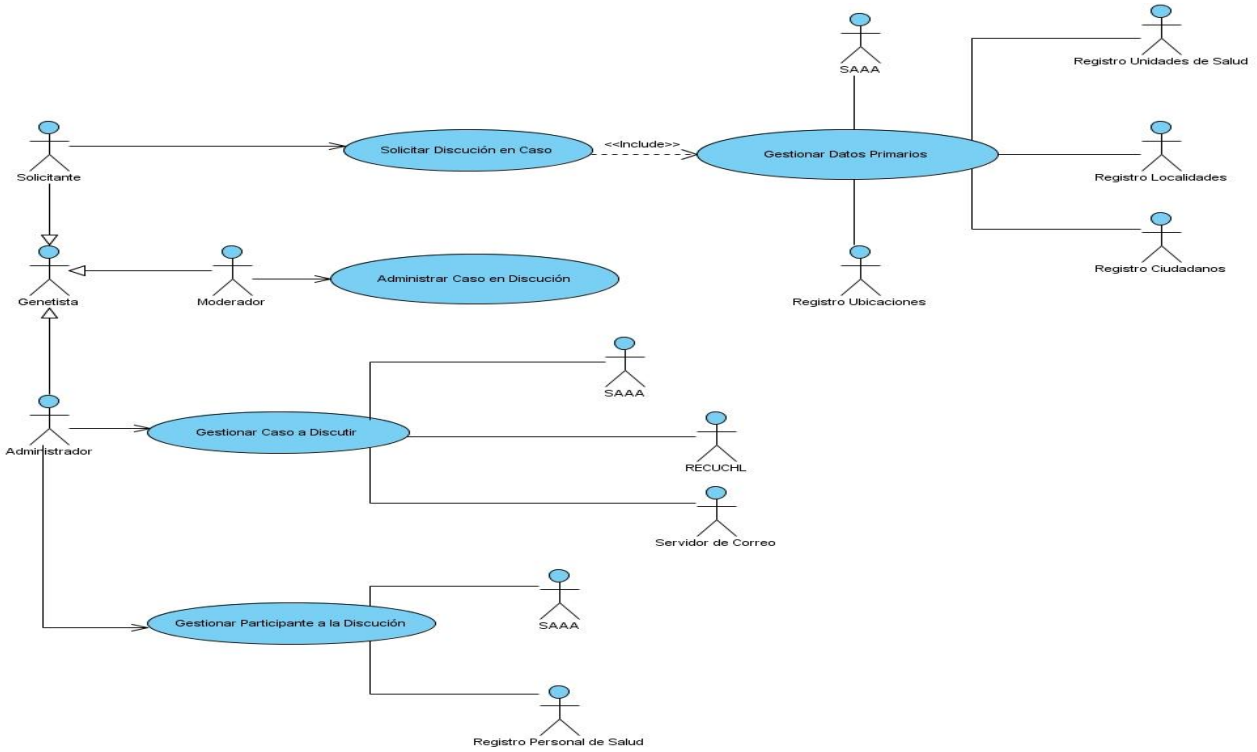

Figura 11. Diagrama de CU Críticos. TELECONSULTA

2.5 VISTA LÓGICA

En la descripción de la arquitectura, la vista lógica describe el sistema mostrando las clases más importantes y su organización en paquetes y subsistemas. En esta vista se describen los paquetes utilizados, y las relaciones que entre ellos existen ya sea de dependencia o de uso; en la misma se refleja el patrón de arquitectura definido SOA y para cada sistema independiente el patrón de diseño es MVC.

En la figura 12 se ilustra la división en módulos de alasMEDIGEN de forma independiente y los servicios que estos brindan y consumen, con el objetivo de hacer el sistema más reusable y facilitar su mantenimiento, logrando que cualquier cambio en los procesos del negocio se realice solo en el módulo al que pertenece la funcionalidad que debe ser cambiada. Esto facilita el trabajo del equipo de desarrollo ya que permite que se puedan ir desarrollando módulos paralelamente y determinar la dependencia entre los mismos.

Además se propone la utilización de un ESB (figura 13) que media entre los servicios web los cuales son expuestos y consumidos entre los sistemas de alasMEDIGEN y el RIS, para así evitar la comunicación punto a punto.

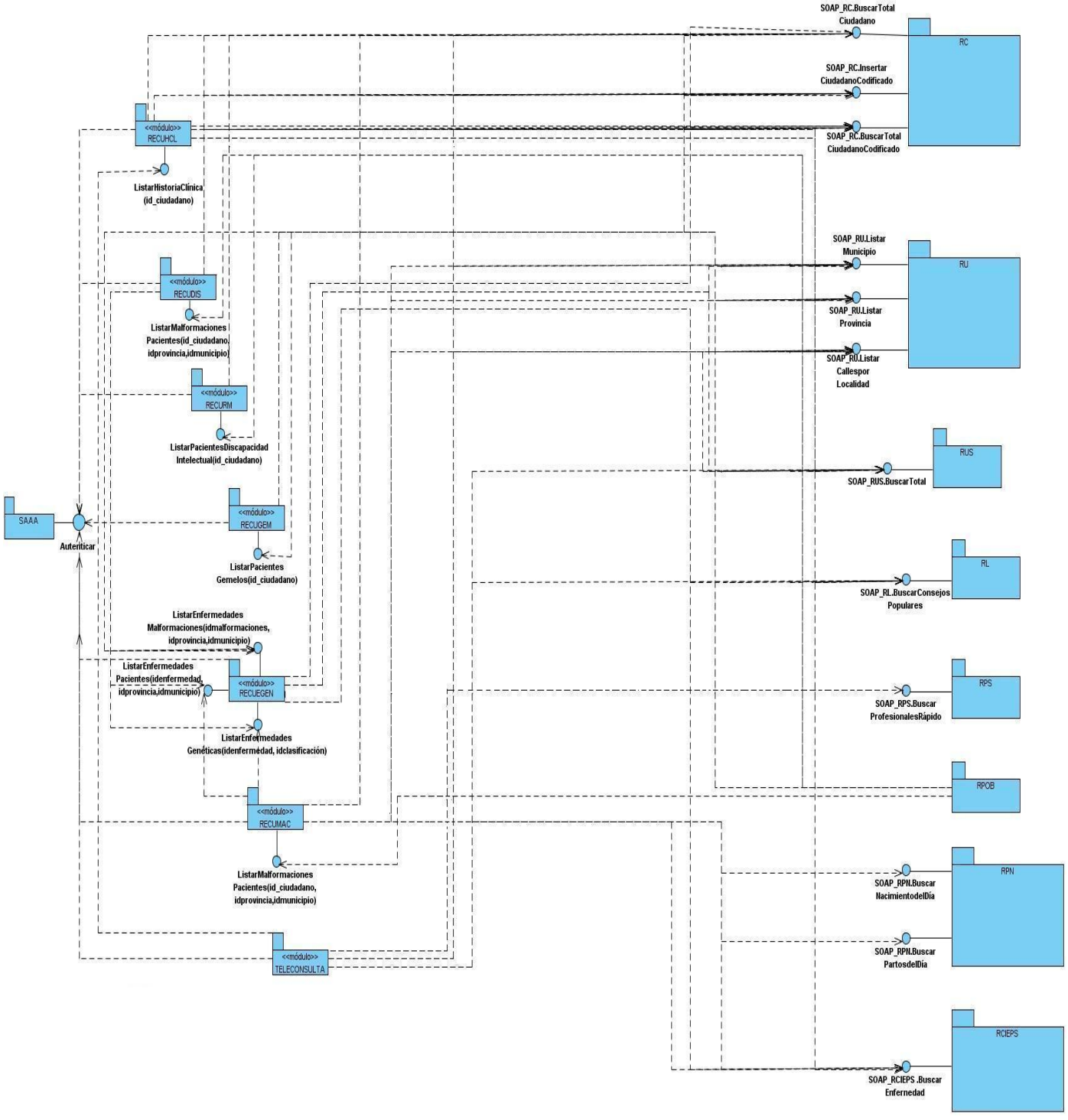

Figura 12. Vista Lógica de los Módulos de alasMEDIGEN.Sistema Informático de Genética Médica.

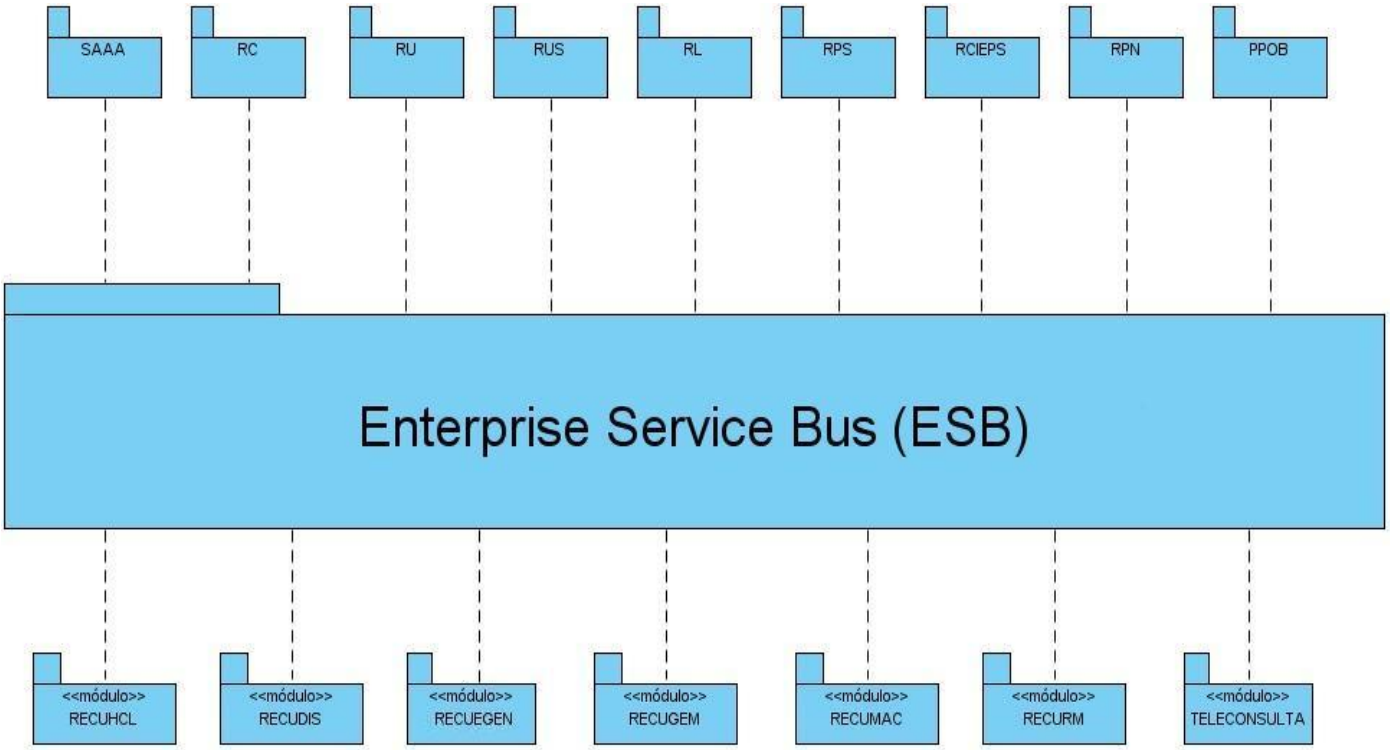

Figura 13. Vista Lógica de los módulos de alasMEDIGEN con un ESB.

Los módulos identificados son:

RECUHCL: Este módulo solicita servicios del componente de Seguridad (SAAA), del Registro de Ciudadanos (RC), Registro Unidades de Salud (RUS), Registro de Ubicación (RU), Registro Localidades (RL), Registro Internacional de Clasificación de Enfermedades y Problemas de Salud (RCIEPS) y le ofrece servicios al módulo Teleconsulta.

RECUDIS: Este módulo solicita servicios del componente de Seguridad (SAAA), del Registro de Ciudadanos (RC), Registro Unidades de Salud (RUS), Registro de Ubicación (RU), Registro Localidades (RL), del módulo RECUEGEN, además le ofrece servicios al Registro de Población (RPOB).

RECURM: Este módulo solicita servicios del componente de Seguridad (SAAA), del Registro de Ciudadanos (RC), Registro Unidades de Salud (RUS), Registro de Ubicación (RU), Registro Localidades (RL), además le ofrece servicios al Registro de Población (RPOB).

RECUGEM: Este módulo solicita servicios del componente de Seguridad (SAAA), del Registro de Ciudadanos (RC), Registro Unidades de Salud (RUS), registro de Ubicación (RU), Registro Localidades (RL), además le ofrece servicios al Registro de Población (RPOB).

RECUEGEN: Este módulo solicita servicios del componente de Seguridad (SAAA), del Registro de Ciudadanos (RC), Registro Unidades de Salud (RUS), Registro de Ubicación (RU), Registro Localidades (RL), Registro Internacional de Clasificación de Enfermedades y Problemas de Salud (RCIEPS), además le ofrece servicios al Registro de Población (RPOB), a los módulos RECUMAC y RECUDIS.

RECUMAC: Este módulo solicita servicios del componente de Seguridad (SAAA), del Registro de Ciudadanos (RC), Registro Unidades de Salud (RUS), Registro de Ubicación (RU), Registro Localidades (RL), del módulo RECUEGEN, Registro nacional de Partos y Nacimientos (RPN), además le ofrece servicios al Registro de Población (RPOB).

Teleconsulta: Este módulo solicita servicios del componente de Seguridad (SAAA), del Registro de Ciudadanos (RC), Registro Unidades de Salud (RUS), Registro de Ubicación (RU), Registro Localidades (RL), Registro Personal de Salud (RPS) y del módulo RECUHCL.

Para mayor detalle, se presentan a continuación un grupo de vistas específicas que reflejan algunas relaciones que no se evidencian en la vista lógica general, para esto se expone la descripción de los paquetes y subsistemas del diseño que conforman la vista lógica del sistema RECUHCL.

Paquete Controlador: Agrupa las clases que implementan la lógica de la aplicación.

Paquete Acciones: Encierra la clase Actions de cada módulo. Estas clases definen las acciones que incluyen el código específico del controlador de cada página.

Paquete Vista: Incluye las clases necesarias para presentar al usuario la lógica de la aplicación.

Paquete del Modelo: Contiene las clases que encapsulan la lógica del dominio, y se encargan del acceso a los datos almacenados en el gestor de base de datos. Además se muestran las clases ***proxyRegCiudadanos*** que permite recuperar los datos brindados por los servicios web ***SOAP_RC.BuscarTotalCiudadanosCodificados***, ***SOAP_RC.InsertarCiudadanoCodificado, SOAP_RC.BuscarTotalCiudadano*** expuestos por el Registro de Ciudadanos; la clase ***proxyRegUbicación*** que permite recuperar los datos brindados por los servicios web ***SOAP_RU.ListarMunicipio***, ***SOAP_RU.ListaProvincia, SOAP_RU.ListarCallesporLocalidad*** expuesto por el Registro Ubicaciones. De igual forma se presentan las clases ***proxyRegUnidadesSalud, proxyRegLocalidades y proxyRegInternacEnfermedades*** que permiten obtener la información de los servicios web brindados por las operaciones ***SOAP_RUS.BuscarTotal, SOAP_RL.BuscarConsejosPopulares y SOAP_RCIEPS.BuscarEnfermedad*** expuestos por los Registro de Unidades de Salud, Registro Localidades y Registro del Clasificado Internacional de

Enfermedades y Problemas de Salud respectivamente. Para la utilización de estos servicios cada una de las clases utiliza los servicios del componente de seguridad SAAA. Consumiendo el servicio Autenticar, el sistema se autentifica con la SAAA suministrando un nombre de usuario único y una contraseña.
De forma análoga a RECUHCL se presenta el diseño de los demás sistemas de alasMEDIGEN.

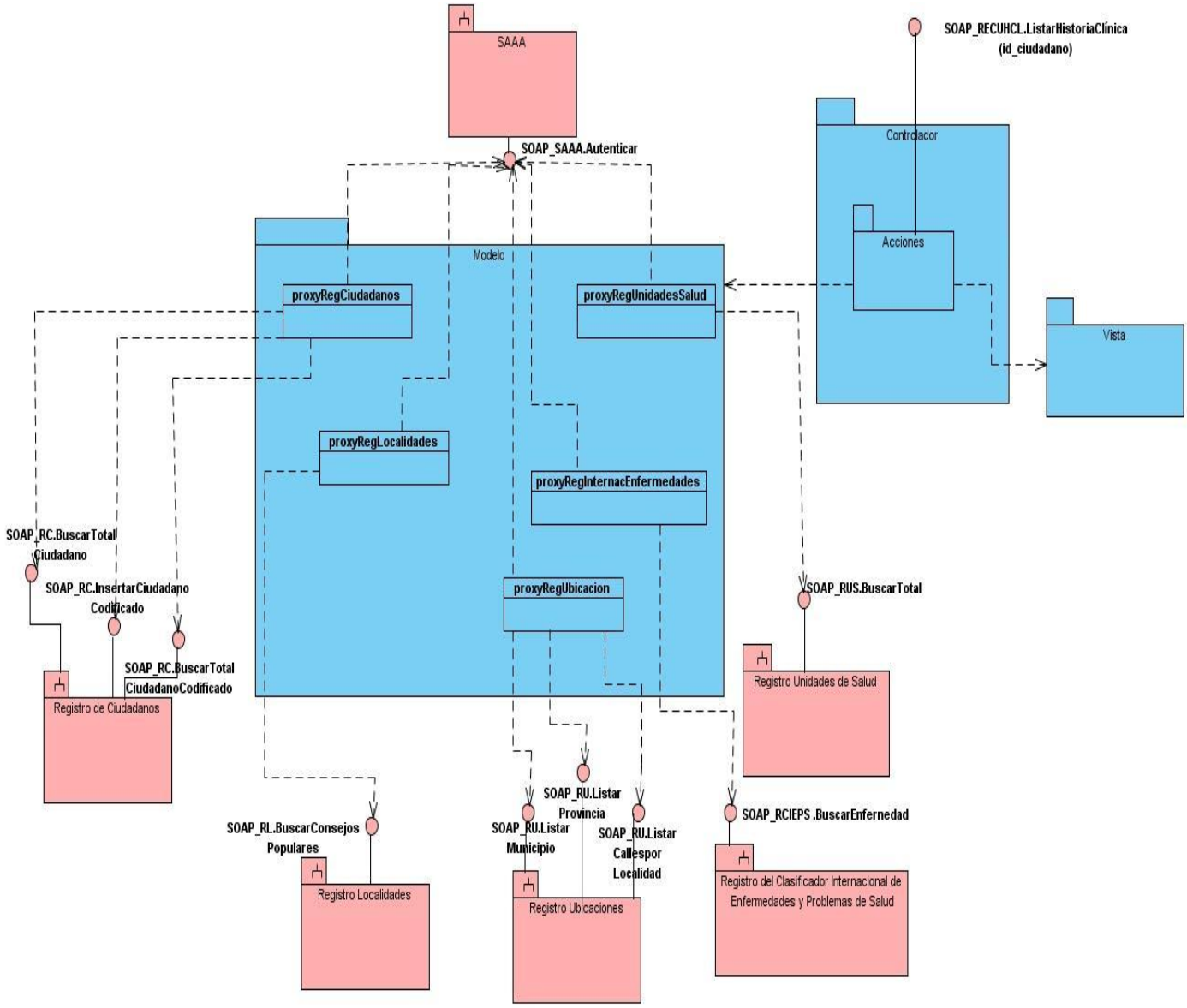

Figura 14. Vista Lógica del sistema RECUHCL.

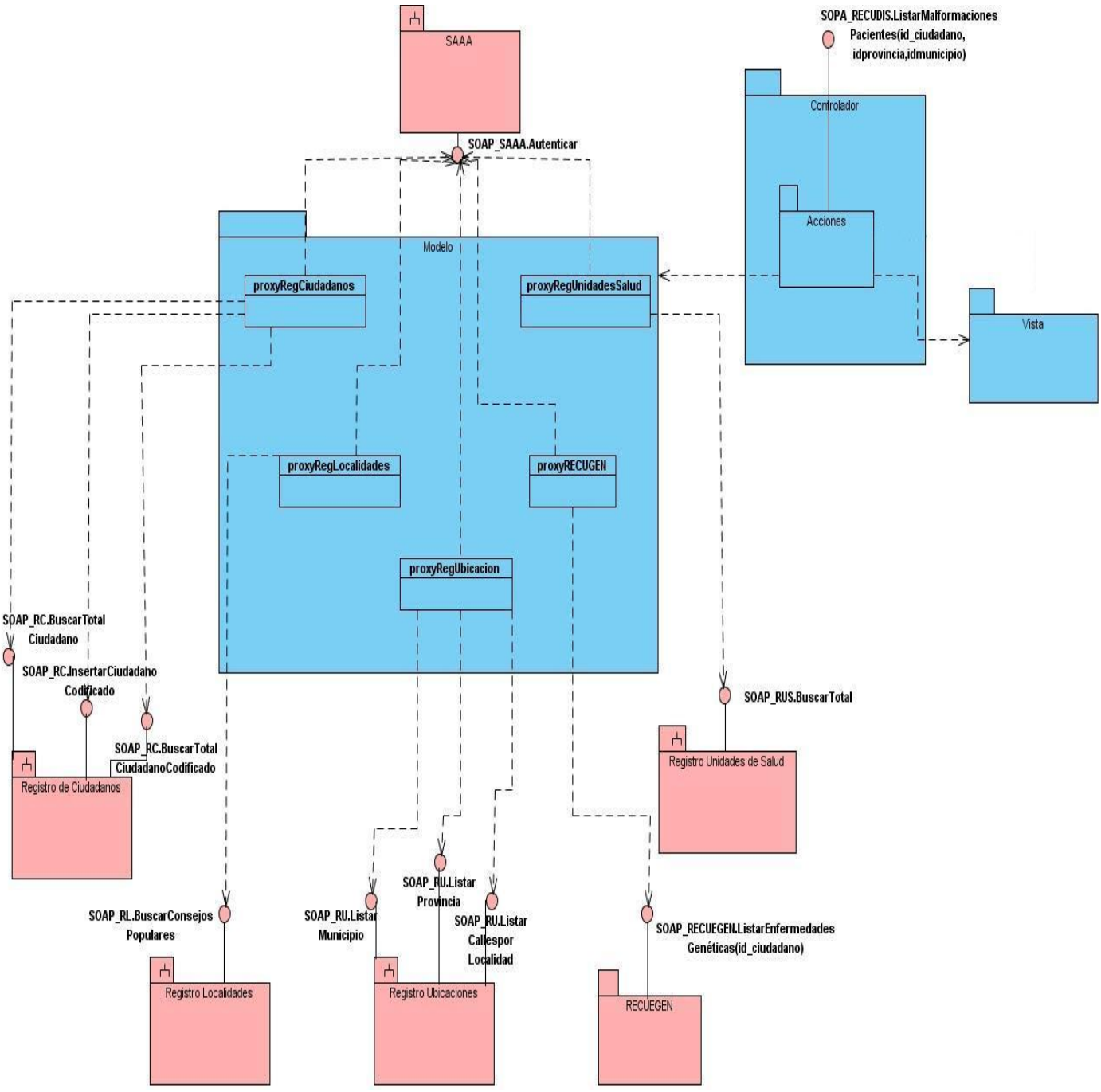

Figura 15. Vista Lógica del sistema RECUDIS.

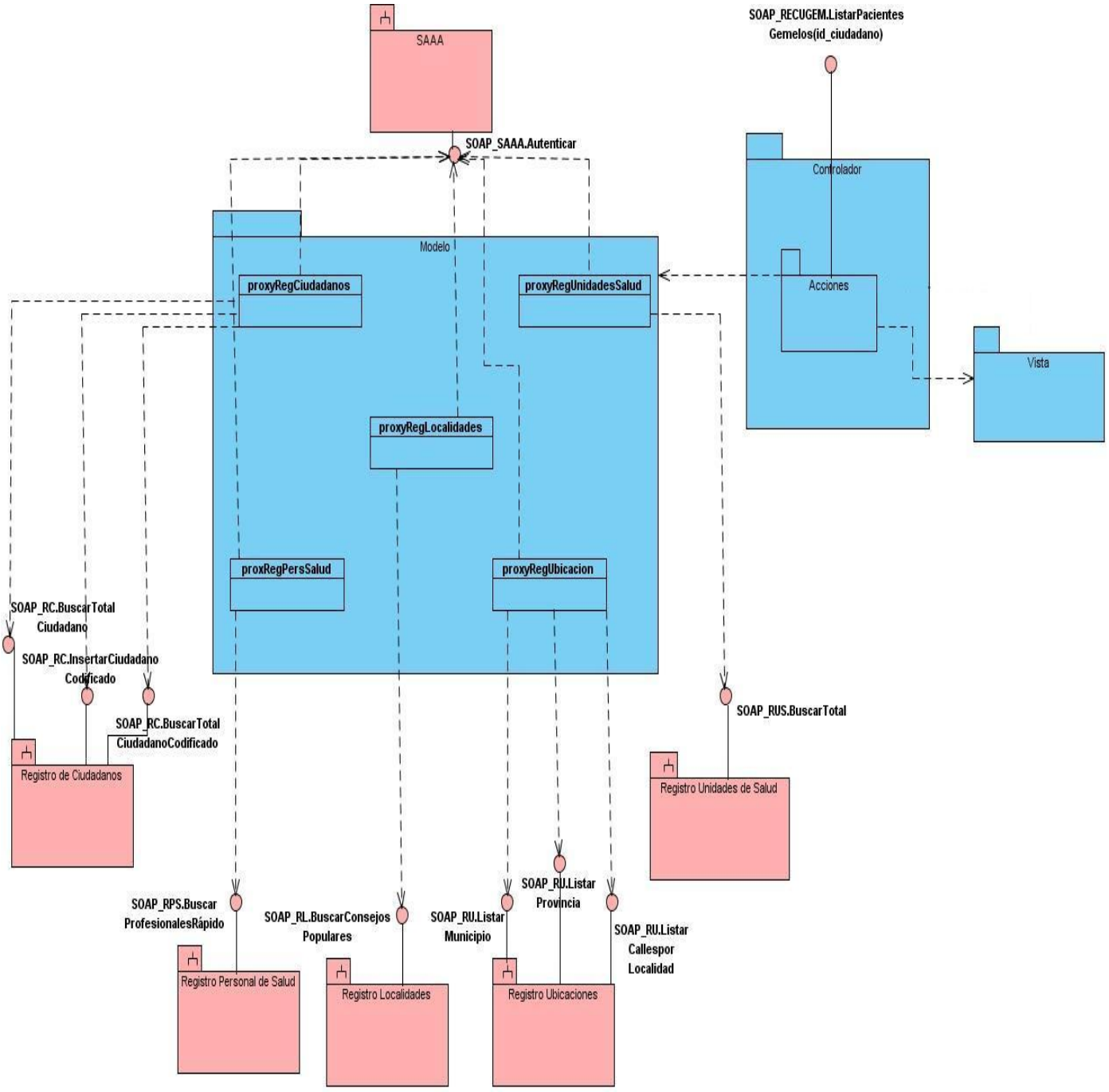

Figura 16. Vista Lógica del sistema RECUGEM.

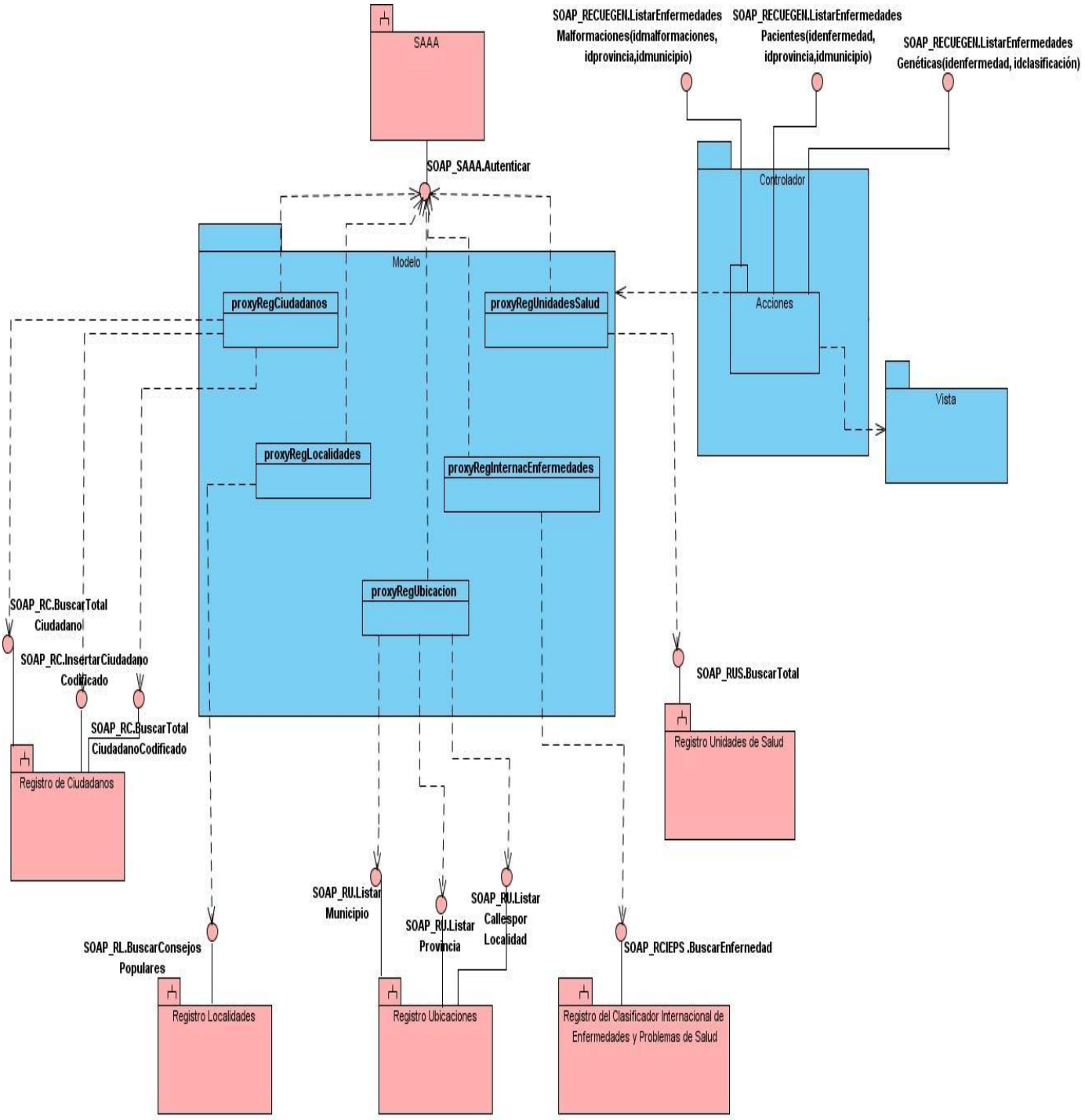

Figura 17. Vista Lógica del sistema RECUEGEN.

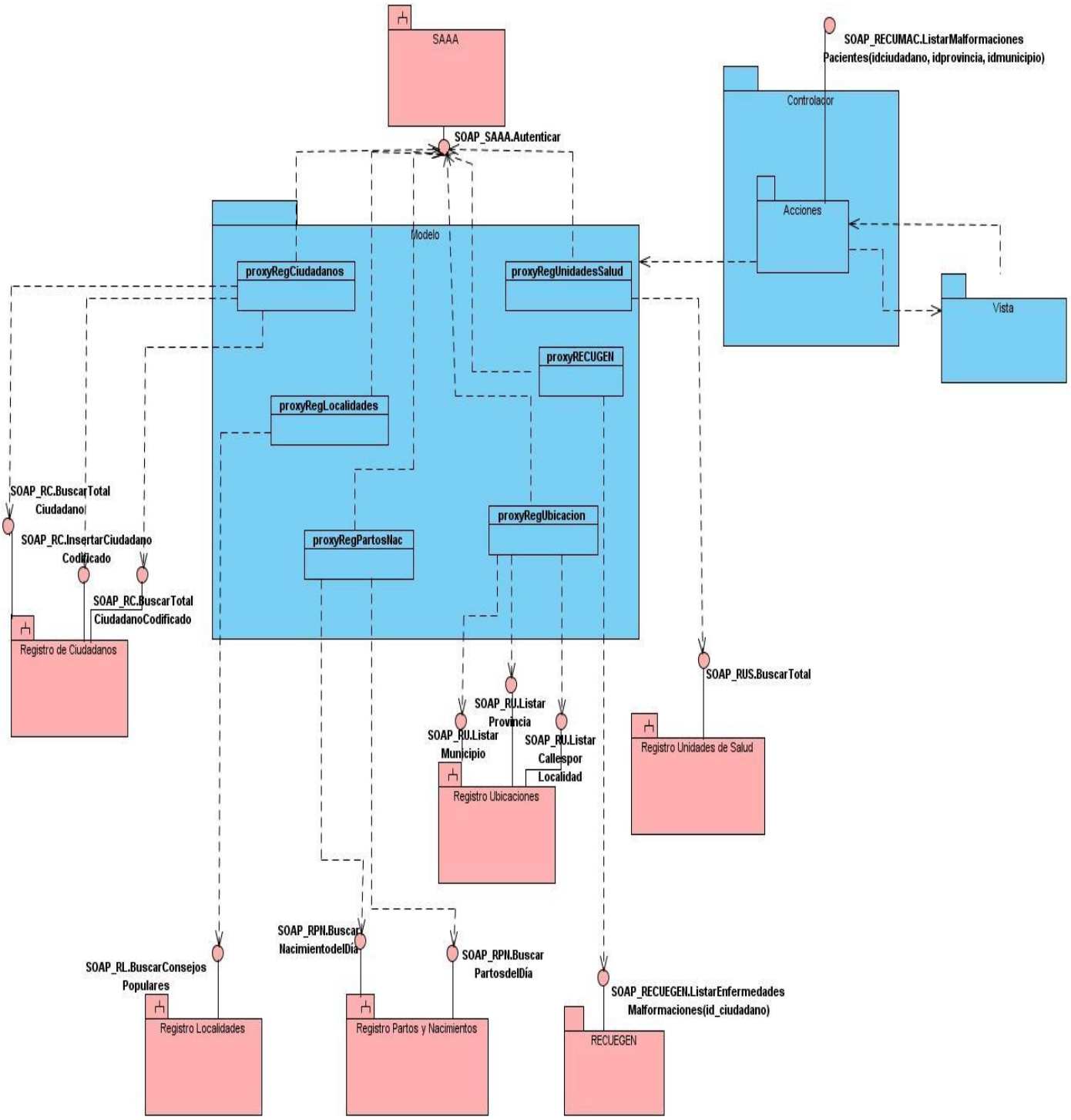

Figura 18. Vista Lógica del sistema RECUMAC

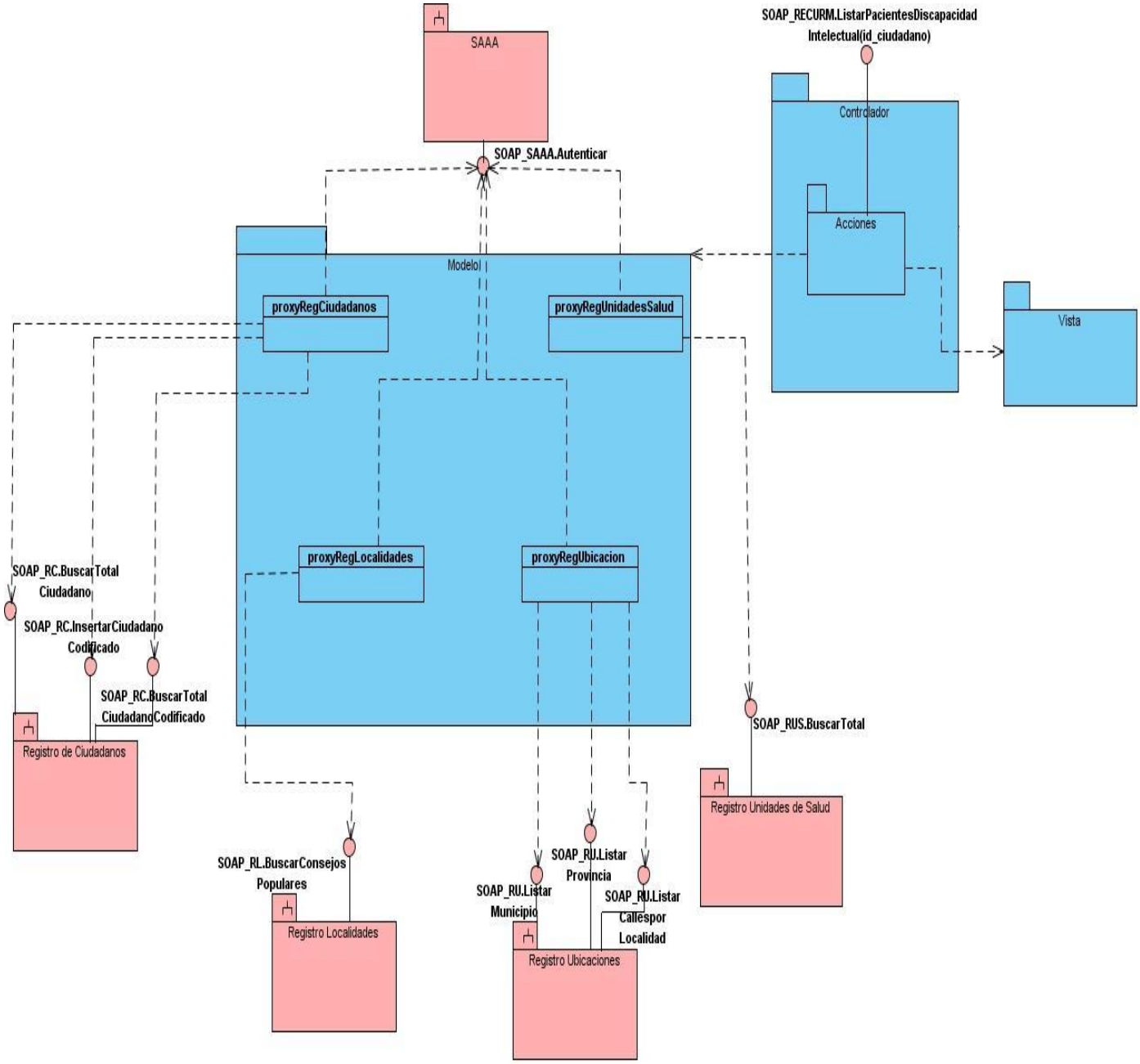

Figura 19. Vista Lógica del sistema RECURM.

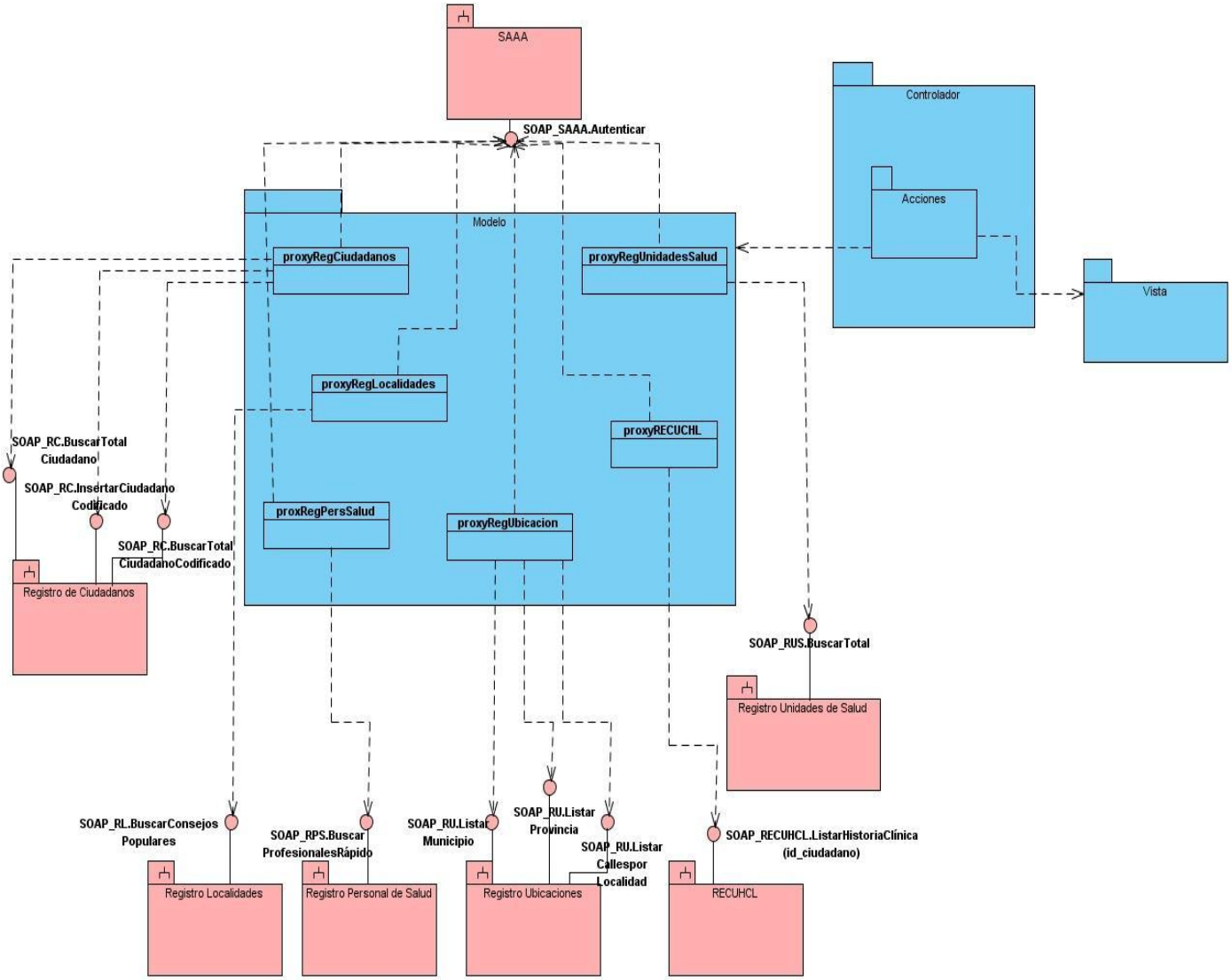

Figura 20. Vista Lógica del sistema Teleconsulta.

2.6 DEFINICIÓN DE LOS SERVICIOS UTILIZANDO WSDL

En el capítulo anterior se realizó un análisis de las principales características de los WSDL. En la siguiente figura se expone el WSDL del módulo RECUHCL el cual brinda el servicio ListarHistoriaClínica que dado el identificador de un ciudadano (id_ciudadano) retorna una lista con datos fundamentales de una historia clínica, los cuales son: identificador de una persona (id_persona), nombre y apellidos (nombre, apellidos), dirección donde reside actualmente (dirección), edad (edad), sexo (sexo) y enfermedad que padece (enfermedad_padece).

```
<? xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2010 (http://www.altova.com) by MESMERiZE (MiZE) -->
<!-- RECUHCL (Registro Cubano de Historia Clínica) -->
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:tns="urn:RECUHCL" targetNamespace="urn:RECUHCL">
	<wsdl:types>
		<xsi:schema xmlns:xsi="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:RECUHCL"
elementFormDefault="qualified">
			<xsi:complexType name="Token">
				<xsi:attribute name="actor" use="required"/>
			</xsi:complexType>
			<xsi:complexType name="ListarHistoriaClinica">
				<xsi:sequence>
					<xsi:element name="id_persona" type="xsi:int"/>
					<xsi:element name="nombre" type="xsi:string"/>
		<xsi:element name="apellidos" type="xsi:string"/>
		 <xsi:element name="direccion" type="xsi:string"/>
					<xsi:element name="edad" type="xsi:int"/>
					<xsi:element name="sexo" type="xsi: string "/>
					<xsi:element name="enfermedad_padece" type="xsi:string"/>
				</xsi:sequence>
			</xsi:complexType>
		</xsi:schema>
	</wsdl:types>
	<wsdl:message name="ListarHistoriaClinicaRequest">
		<wsdl:part name="id_ciudadano" type="xs:int"/>
	</wsdl:message>
	<wsdl:message name="ListarHistoriaClinicaResponse">
		<wsdl:part name="listadoRetorno" type="tns:ListarHistoriaClinica"/>
	</wsdl:message>
	<wsdl:message name="Header">
		<wsdl:part name="Token" type="xs:string"/>
	</wsdl:message>
	<wsdl:portType name="RECUHCLServicePort">
		<wsdl:operation name="ListarHistoriaClinica">
			<wsdl:input message="tns:ListarHistoriaClinicaRequest"/>
			<wsdl:output message="tns:ListarHistoriaClinicaResponse"/>
		</wsdl:operation>
	</wsdl:portType>
	<wsdl:binding name="RECUHCLServiceBinding" type="tns:RECUHCLServicePort">
		<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
		<wsdl:operation name="ListarHistoriaClinica">
			<soap:operation soapAction="RECUHCL.ListarHistoriaClinica"/>
			<wsdl:input>
				<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
				<soap:header message="tns:Header" part="Token" use="literal"/>
			</wsdl:input>
			<wsdl:output>
				<soap:body use="literal"/>
			</wsdl:output>
		</wsdl:operation>
	</wsdl:binding>
	<wsdl:service name="SOAP_RECUHCL">
		<wsdl:port name="RECUHCLServicePort" binding="tns:RECUHCLServiceBinding">
			<wsdl:documentation>servicioRECUHCL</wsdl:documentation>
			<soap:address location="http://201.220.222.147/core/WSDL/RECUHCL.wsdl"/>
		</wsdl:port>
	</wsdl:service>
</wsdl:definitions>
```

2.7 VISTA DE DESPLIEGUE

La vista de despliegue describe la configuración física sobre la que será desplegado el software. Ésta presenta los nodos computacionales que intervienen en el funcionamiento del sistema, las conexiones entre estos y los protocolos de comunicación, estableciendo posibles configuraciones que se ilustran mediante los diagramas de despliegue.

Para el funcionamiento el sistema, es necesario que se despliegue en tres servidores existentes en Infomed, corriendo sobre el sistema operativo Linux, distribución Debian (Sarge). Estos servidores presentan la siguiente estructura:

- ✓ Un servidor denominado Presentación que contiene la lógica de la presentación y se conecta a la red virtual privada de Infomed. Constituye el punto de entrada del usuario al sistema, pues es el nodo con que el cliente se conecta directamente.
- ✓ Un servidor llamado Aplicación que contiene la lógica de la aplicación, y posee conectividad punto a punto únicamente con el servidor que hospeda la capa de presentación y el servidor de datos.
- ✓ Un servidor llamado Datos que contiene los datos, y posee únicamente conectividad punto a punto con el servidor de aplicación.

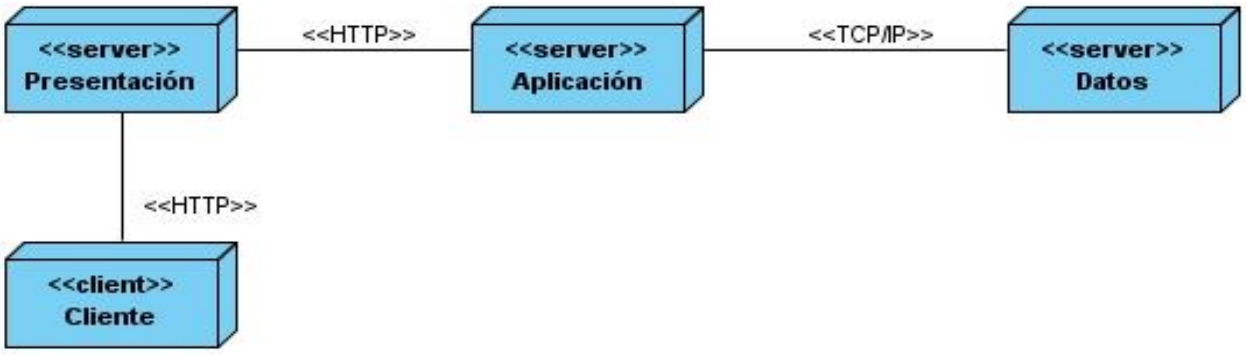

Figura 21. Estructura básica de los servidores de Infomed propuesta por el Marco Regulatorio del GIS.

Esta estructura básica es justificada por Infomed, y aceptada por la propuesta arquitectónica planteada en la presente investigación, debido a las ventajas de seguridad y rendimiento que posibilita utilizar un despliegue basado en tres servidores.

Para dar cumplimiento a los requisitos del cliente se representa una impresora como dispositivo externo.

El único punto de acceso del cliente a la aplicación es a través del nodo de Presentación, el servidor de Aplicación se encuentra en un segundo nivel y el de Datos en un tercer nivel de acceso, implementando

una barrera física de conexión difícil de violar por usuarios maliciosos. Además todos los procesos que lleva a cabo el sistema se dividen en tres servidores, aumentando el rendimiento. Asumiendo la estructura anteriormente planteada, el diagrama de despliegue del sistema quedaría como se muestra en la siguiente figura:

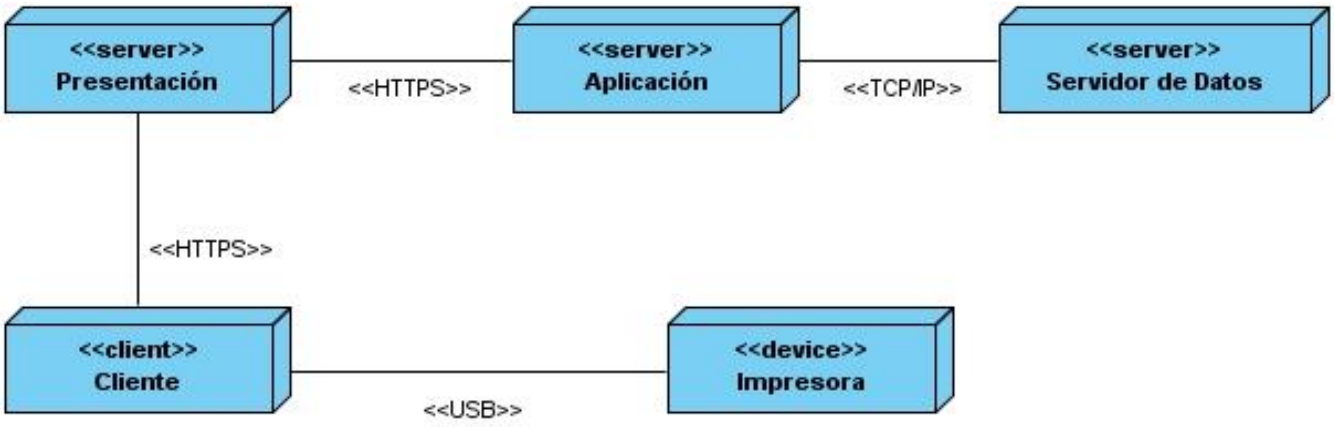

Figura 22. Vista de despliegue del sistema.

2.8 CONCLUSIONES

En el capítulo se realizó un análisis acerca del ciclo de vida de una arquitectura SOA y los principios del diseño de la orientación a servicios.

Se evidenció el uso del patrón arquitectónico SOA, analizado y propuesto en el capítulo anterior. Además se realizó la descripción de la arquitectura del sistema a través de la vista de de casos de uso, la lógica y la de despliegue. En la vista de caso de uso del sistema se agruparon las principales funcionalidades que debe cumplir el mismo, evidenciándose la integración de los diferentes módulos que actualmente componen a alasMEDIGEN, como sistemas independientes relacionados a través de servicios web. Para la realización de la vista lógica se tuvieron en cuenta las clases, paquetes y subsistemas de diseño más significativos, permitiendo observar la forma en que están estructuradas las funcionalidades en el interior del mismo. Finalmente en la vista de despliegue se representaron los nodos y conexiones que intervienen en el funcionamiento del sistema, mostrando la configuración física sobre la que será desplegado el software.

CAPÍTULO 3

EVALUACIÓN DE LA ARQUITECTURA PROPUESTA

En este capítulo se presentan aspectos a tener en cuenta para la evaluación de la arquitectura de software. Se realiza la evaluación de la propuesta arquitectónica a través del método ARID basado en los atributos de calidad definidos por el Modelo ISO 9126-1:2001 y se usa como técnicas de evaluación las cualitativas.

3.1 ASPECTOS PARA EVALUAR UNA ARQUITECTURA DE SOFTWARE

A la hora de evaluar una arquitectura se pueden plantear las siguientes interrogantes: ¿por qué evaluar una arquitectura?, ¿quiénes intervienen en la evaluación? y ¿cuándo una arquitectura puede ser evaluada?

"El propósito de realizar evaluaciones a la arquitectura, es para analizar e identificar riesgos potenciales en su estructura y sus propiedades, que puedan afectar al sistema de software resultante, verificar que los requerimientos no funcionales estén presentes en la arquitectura, así como determinar en qué grado se satisfacen los atributos de calidad. Cabe señalar que los requerimientos no funcionales también son llamados atributos de calidad". (18)

Generalmente las evaluaciones a la arquitectura se hacen por miembros del equipo de desarrollo, arquitecto, diseñador, entre otros. Sin embargo puede haber también situaciones en las que intervengan personas especialistas en el tema. El cliente es uno de los más interesados en los resultados de una evaluación, ya que en dependencia de los mismos puede tomar decisiones sobre continuar o no con el proyecto.

Es posible realizar la evaluación en cualquier momento, pero se proponen dos variantes según las etapas de desarrollo: temprana y tarde. (19)

- ✓ **Temprana:** No es necesario que la arquitectura esté completamente especificada para efectuar la evaluación, y esto abarca desde las fases tempranas de diseño y a lo largo del desarrollo.
- ✓ **Tarde:** Cuando la arquitectura se encuentra establecida y la implementación se ha completado. Este es el caso general que se presenta al momento de la adquisición de un sistema ya desarrollado.

3.2 EVALUACIÓN DE LA ARQUITECTURA.

En el presente epígrafe se utiliza el método ARID seleccionado en el capítulo uno para evaluar la actual propuesta de arquitectura de alasMEDIGEN.

“Un atributo de calidad es una característica de calidad que afecta a un elemento. Donde el término “característica” se refiere a aspectos no funcionales y el término “elemento” a componente”. (18)

Los requerimientos no funcionales son conocidos también como atributos de calidad, pues son aspectos que afectan en gran medida la calidad de los componentes que integran un sistema. El trabajo con los atributos de calidad es un tanto engorroso, pues los cambios realizados en la arquitectura con el objetivo de mejorar un atributo específico puede afectar de forma negativa a otro, siendo necesario establecer un balance entre los mismos.

Debido a que las decisiones en la arquitectura determinan los atributos de calidad del sistema, será posible saber que la arquitectura seleccionada es correcta evaluando el resultado que han producido dichas decisiones sobre los atributos de calidad.

Existen diferentes atributos de calidad que se pueden tomar en cuenta a la hora de evaluar una arquitectura. En la presente investigación se decidió utilizar los atributos definidos por el Modelo ISO 9126-1:2001 (20) y como técnica de evaluación, las cualitativas basadas en escenarios. Este identifica seis atributos básicos de calidad que pueden estar presentes en cualquier producto de software, los cuales se relacionan a continuación:

1. Funcionalidad
2. Confiabilidad
3. Eficiencia
4. Usabilidad
5. Mantenibilidad
6. Portabilidad

Además ARID define tres grupos de roles que se relacionan a continuación:

- ✓ El equipo de verificación, el cual está formado por tres roles: líder de la evaluación que se encarga de preparar conjuntamente con el arquitecto la evaluación, un secretario que se encarga de recolectar los resultados y realizar las anotaciones, y un conjunto de interrogadores que ayudan a obtener los escenarios durante las entrevistas. Opcionalmente se podrá tener un observador para mejorar el proceso de evaluación.

- ✓ El arquitecto, el cual es responsable de presentar el diseño y a quien se la dará los resultados de la evaluación.
- ✓ Revisores, son los interesados en la viabilidad y adecuación del diseño que se presentan. Son las personas que van a utilizar el diseño (ingenieros de software) y son las personas más adecuadas para juzgar su calidad.

A continuación se realiza la evaluación aplicando el método seleccionado.

FASE 1: PRE-REUNIÓN

Paso #1: Identificar los encargados de la revisión: Definir quiénes llevarán a cabo la evaluación de la arquitectura.

En la presente investigación los encargados de realizar la evaluación de la arquitectura propuesta son los arquitectos que la definieron, pues tienen pleno dominio del diseño arquitectónico y los conocimientos del método a aplicar para la evaluación.

Paso #2: Preparar el informe del diseño: El diseñador es el encargado de elaborar un informe donde se explique detalladamente el diseño, para capacitar a los encargados de la revisión sobre las cuestiones fundamentales del sistema.

En este paso no se va a preparar un informe puesto que el diseño de la arquitectura del sistema ya fue elaborado, explicado y detallado en el capítulo anterior.

Paso #3: Preparar los escenarios base.

A continuación se presentan los escenarios por atributo de calidad propuestos para realizar la evaluación:

La **Funcionalidad** es el conjunto de atributos que se refieren a la existencia de un conjunto de funciones y sus propiedades específicas. Las funciones cumplen unos requerimientos o satisfacen unas necesidades implícitas. Las características específicas de la Funcionalidad son: Aptitud, Precisión, Interoperabilidad, Conformidad, Seguridad y Trazabilidad.

Escenario #1: El sistema cumple con los requisitos funcionales solicitados por el cliente y solo con estos.

Escenario #2: Se debe garantizar el acceso al sistema de forma segura en todo momento.

Escenario #3: Se emplea un mecanismo de cierre de sesión de usuario en caso de pasado determinado tiempo de inactividad.

Escenario #4: Los usuarios sólo pueden acceder a los datos del sistema que están autorizados.

Escenario #5: El sistema no es vulnerable a ataques de tipo XSS (cross site scripting) y CSRF (cross-site request forgery).

Escenario #6: Existe un mecanismo de registro de eventos.

La **Confiabilidad** es el conjunto de atributos que se refieren a la capacidad del software de mantener su nivel de rendimiento bajo unas condiciones especificadas durante un período definido. Las características específicas de la Confiabilidad son: Madurez, Tolerancia a fallas, Facilidad de Recuperación, Disponibilidad y Degradabilidad.

Escenario #1: Los datos introducidos por el usuario deben ser los más correctos posible.

Escenario #2: El sistema estará disponible las 24 horas del día.

La **Eficiencia** es el conjunto de atributos que se refieren a las relaciones entre el nivel de rendimiento del software y la cantidad de recursos utilizados bajo unas condiciones predefinidas. Las características específicas de la Eficiencia son: Respecto al tiempo y Respecto a los recursos.

Escenario #1: Los tiempos de respuesta a las peticiones de los usuarios deben ser mínimos.

Escenario #2: Búsqueda de información de los usuarios de forma rápida.

La **Usabilidad** es el conjunto de atributos que se refieren al esfuerzo necesario para usarlo, y sobre la valoración individual de tal uso, por un conjunto de usuarios definidos e implícitos. Las características específicas de la Usabilidad son: Comprensibilidad, Facilidad de aprendizaje, Operatividad, Explicitud, Adaptabilidad al usuario, Atractivo, Claridad, Facilidad de ayudas y Amistoso al usuario.

Escenario #1: El sistema debe ser entendible y fácil de usar.

Escenario #2: El sistema debe tener una interfaz amigable.

Escenario #3: Se cuenta con manuales de usuarios y videos tutoriales.

La **Mantenibilidad** es el conjunto de atributos que se refieren al esfuerzo necesario para hacer modificaciones especificadas. Las características específicas de la Mantenibilidad son: Facilidad de análisis, Facilidad de cambio, Estabilidad y Facilidad de prueba.

Escenario #1: Agregar nuevos módulos al sistema sin dificultad.

Escenario #2: Agregar funcionalidades a los módulos existentes sin dificultad.

Escenario #3: El sistema podrá cambiar de gestor de base de datos.

La **Portabilidad** es el conjunto de atributos que se refieren a la habilidad del software para ser transferido desde un entorno a otro. Las características específicas de la Portabilidad son: Adaptabilidad, Facilidad de instalación, Conformidad y Facilidad de Reemplazo.

Escenario #1: Es fácil instalar las actualizaciones del sistema.

Escenario #2: El sistema debe ser flexible ante cualquier cambio de entorno.

Escenario #3: Se puede acceder al sistema desde cualquier lugar del país.

Escenario #4: El sistema puede ser instalado en varios sistemas operativos.

Paso #4: Preparar los materiales, copias de la presentaciones y escenarios: se realiza la agenda invitando a interesados externos y asegurando la presencia de los revisores.

Todos los materiales están asegurados para llevar a cabo la realización del método. Se han hecho copias de las presentaciones para la exposición del método y de los escenarios propuestos en el paso tres.

FASE 2: EVALUACIÓN

Paso #5: Presentación del método ARID: a continuación se explican los pasos de la evaluación con el método a los participantes.

Una evaluación ARID progresa a través de dos fases que abarcan nueve pasos.

Fase1: PRE-reunión

Primero una reunión entre el arquitecto y el líder de la evaluación se lleva a cabo para preparar el ejercicio. Esta reunión dura un día aproximadamente y en la misma se llevan a cabo los primeros 4 pasos.

1. **Identificar los revisores**, son los ingenieros de software que van a usar el diseño.
2. **Preparar la presentación del diseño**, el arquitecto prepara un informe que explica el diseño, el mismo deberá ser lo suficientemente detallado como para que una capacitada audiencia pueda usar el diseño.

   El arquitecto presenta el informe al líder de la evaluación, esta presentación es provechosa por varios motivos:

   - ✓ El diseño es mostrado al líder de la evaluación y éste realizará una serie de preguntas que los revisores van a realizarle, preparando al arquitecto.
   - ✓ Ayuda a identificar áreas donde la presentación puede ser mejorada.
   - ✓ Ayuda a tomar el tiempo de la presentación.
   - ✓ Ayuda al arquitecto a practicar la presentación que luego dará a una audiencia más crítica.
3. **Preparar los escenarios**, el arquitecto y el líder preparan los escenarios que sirven para ilustrar los conceptos a los revisadores.
4. **Preparar los materiales,** copias de la presentaciones, escenarios, se realiza la agenda invitando a interesados externos y asegurando la presencia de los revisores.

Fase 2: Evaluación

Durante la fase dos, los revisores son reunidos y la evaluación comienza. Esta durará un día y medio y las 5 actividades restantes son completadas.

5. **Presentación del método ARID**, el líder utiliza 30 minutos para explicar los pasos de la evaluación a los participantes.

6. **Presentación del diseño**, el arquitecto realiza una presentación de dos horas del diseño mostrando los ejemplos. Durante la presentación no se podrán hacer preguntas concernientes a la implementación ni tampoco se proponen diseños alternativos. El objetivo es ver si el diseño es adecuado, no saber por qué el diseño se hizo de esa manera u obtener información para la futura implementación. Preguntas para clarificar el diseño son permitidas. El secretario es el encargado de recolectar las preguntas y registrar las veces que el arquitecto evadió una respuesta afirmando que estaba pensado pero no disponible.
7. **Tormenta de ideas y priorización de escenarios**, entre todos los presentes se proponen escenarios relevantes para solucionar problemas que esperan hacer frente. Luego los revisores pueden sugerir que dos escenarios son versiones del mismo o un escenario es parte de otro y deben ser unidos. A continuación cada revisor puede votar, siendo la cantidad de votos un 30% de la totalidad de los escenarios, los revisores pueden utilizar todo sus votos en un escenario o repartirlos entre varios. Los escenarios con más votos son usados para probar la adecuación del diseño.
8. **Se realiza la revisión**, comenzando con el escenario que tuvo la mayor cantidad de votos, el líder de la evaluación le pide a los revisores que trabajando como grupo usen el diseño para resolver el problema presentado en el escenario. Durante este trabajo el arquitecto no podrá ayudar a los revisores, sin embargo si el líder ve que los revisores van por un mal camino, puede pedirle al arquitecto que los guíe o les brinde cierta información para no perder el tiempo (todo esto debe ser registrado por el secretario). Todas las discrepancias también son registradas.

   La etapa continúa hasta que alguno de los siguientes eventos ocurre:

   - ✓ El tiempo reservado para la revisión se acaba.
   - ✓ Los escenarios con más votos han sido analizados.
   - ✓ El grupo se siente satisfecho tanto porque vio que el diseño era correcto o porque no lo aprobó.
9. **Conclusiones**, finalmente, el líder recolecta los ejercicios y pregunta a los revisores una opinión sobre la eficacia de la evaluación y se agradece su participación.

Paso #6: Presentación del diseño: el arquitecto realiza una presentación de dos horas del diseño mostrando los ejemplos.

En este paso del método se realizó la presentación del diseño propuesto en la investigación y todas las partes implicadas estuvieron de acuerdo con el mismo.

Paso #7: Tormenta de ideas y priorización de escenarios: entre todos los presentes se proponen escenarios relevantes para solucionar problemas que esperan hacer frente.

Una vez que se tienen los escenarios por cada uno de los requisitos se procede a priorizarlos mediante la técnica de los 100 puntos. (21) A cada implicado presente en el Taller se le da la posibilidad de distribuir 100 puntos entre todos los requisitos que fueron detectados dándole mayor puntuación a los más importantes teniendo en cuenta su criterio personal.

A continuación se muestran aquellos escenarios relevantes a los que se les determinarán sus prioridades:

No	Escenario
R1	Se debe garantizar el acceso al sistema de forma segura en todo momento.
R2	Los usuarios solo pueden acceder a los datos del sistema que están autorizados.
R3	El sistema no es vulnerable a ataques de tipo XSS y CSRF.
R4	Existe un mecanismo de registro de eventos.
R5	Los tiempos de respuesta a las peticiones de los usuarios deben ser mínimos.
R6	El sistema debe ser entendible y fácil de usar.
R7	Agregar nuevos módulos al sistema sin dificultad.
R8	El sistema podrá ser instalado en varios sistemas operativos.
R9	Es fácil instalar las actualizaciones del sistema.

Una vez realizada la votación se procede a determinar el nivel de prioridad de cada requisito a través de la matriz de cálculo de prioridad mostrada en la siguiente tabla:

	R1	R2	R3	R4	R5	R6	R7	R8	R9
Arquitecto1	21	11	10	8	8	7	16	13	6
Arquitecto2	23	9	10	9	7	7	15	12	8
Prioridad final	22	10	10	8.5	7.5	7	15.5	12.5	7

Teniendo en cuenta el siguiente resultado se ordenó de manera decreciente los requisitos, de forma tal que la prioridad será dada de acuerdo al orden en que fueron ubicados. La siguiente tabla representa lo explicado anteriormente:

Requisito	Valor Prioridad	Índice Prioridad
R1	22.0	5.0
R7	15.5	4.4
R8	12.5	3.9

R2	10.0	3.3
R3	10.0	2.7
R4	8.5	2.2
R5	7.5	1.6
R6	7.0	1.1
R9	7.0	0.5

Paso #8: Se realiza la revisión: se comienza con el escenario que tuvo la mayor cantidad de votos.

Escenario #1	Se debe garantizar el acceso al sistema de forma segura en todo momento.
Estímulo	Persona o usuario no autenticado intenta acceder a las funcionalidades del sistema.
Ambiente	Operación normal.
Respuesta	Se muestra la página de autenticación para que el usuario introduzca sus credenciales.
Explicación	La autenticación será la primera acción del usuario en el sistema y consistirá en suministrar un nombre de usuario único y una contraseña que debe ser de conocimiento exclusivo de la persona que se autentica. Cada acción antes de ejecutarse es pasada por un filtro especial que verifica si el usuario actual está autenticado y tiene privilegios de acceder a la acción requerida. Dichos privilegios se gestionan a través de credenciales que se definen bajo un nombre, y permiten organizar la seguridad en grupos. Las mismas se definen y gestionan por el componente SAAA (registro que garantiza que cada usuario acceda solamente a los datos del sistema que está autorizado). Para cada petición, el sistema se encarga de buscar la acción solicitada por el usuario y verifica que se satisfagan los permisos necesarios para acceder a ella. El comportamiento del sistema ante el intento de acceso de un usuario a una acción depende de las credenciales de este.

Escenario #7	Agregar nuevos módulos al sistema sin dificultad.
Estímulo	Ampliar el sistema.
Ambiente	Operación normal.
Respuesta	Desplegar una nueva versión del proyecto.
Explicación	El sistema cuenta con una arquitectura SOA, un modelo que expone bajo acoplamiento entre sus partes lo cual le permite que estas sean reutilizables. Además, presenta un alto grado de modularidad posibilitando añadir nuevos módulos y funcionalidades al sistema sin un alto costo en implementación o afectación total del mismo, ya que cualquier cambio en los procesos de negocio se realiza justo en el módulo y funcionalidad especifica que debe ser cambiada, abstrayéndose de las dependencias y demostrando alta interoperabilidad.

Escenario #8	El sistema podrá ser instalado en varios sistemas operativos.
Estímulo	El administrador cambia el sistema operativo donde radica el servidor Web.

Ambiente	Operación normal.
Respuesta	No hay cambios en la funcionalidad de la aplicación.
Explicación	Para implementar el sistema se definió como lenguaje de programación PHP5 que es multiplataforma y garantiza la portabilidad de la aplicación, posibilitando que el producto se pueda instalar tanto en servidores Linux como Windows.

Escenario #2	Los usuarios solo pueden acceder a los datos del sistema que están autorizados.
Estímulo	El usuario se encuentra autenticado en el sistema.
Ambiente	Operación normal.
Respuesta	Solo se muestran las funcionalidades y los datos a los que el usuario puede acceder.
Explicación	Los datos del sistema deben mostrarse de acuerdo al usuario que este activo, garantizando de este modo que la aplicación es utilizada en correspondencia con los derechos permitidos a cada tipo de usuario. Esta accesibilidad es garantizada por el registro de seguridad SAAA que es el encargado de gestionar las credenciales de cada usuario, permitiéndole el acceso al sistema de acuerdo a su nivel: Genetista Nacional, Provincial o Municipal.

Escenario #3	El sistema no es vulnerable a ataques de tipo XSS y CSRF.
Estímulo	Agente externo intenta violar seguridad del sistema.
Ambiente	Operación normal.
Respuesta	No tiene efecto el intento de ataque.
Explicación	El sistema presenta un mecanismo de escape ante el riesgo de que los contenidos introducidos puedan incluir scripts y otros elementos maliciosos que se encargan de realizar ataques de tipo XSS y CSRF.

Escenario #4	Existe un mecanismo de registro de eventos.
Estímulo	Registrar todas las acciones realizadas por cada usuario.
Ambiente	Operación normal.
Respuesta	El sistema se recupera ante cualquier falla en la ejecución de una petición.
Explicación	En caso de fallas en la ejecución de una petición, el sistema puede comprobar mediante los registros de *"logs"*, la traza generada por el proceso que intentó ejecutarse. Cada evento es introducido en el archivo de log de la aplicación como una línea de código, que incluye la fecha y hora en que ha sido creada, el tipo de evento, el objeto que se ha procesado y otros detalles relevantes que dependen del tipo de evento registrado. Los archivos de log contienen información como las consultas que se han enviado a la base de datos, las plantillas que han sido procesadas, las llamadas que se han realizado entre objetos, entre otros. El componente encargado de realizar esta operación es la SAAA a quien se le informa de los métodos de los demás módulos que se desean registrar.

Escenario #5	Los tiempos de respuesta a las peticiones de los usuarios deben ser mínimos.
Estímulo	El usuario está realizando operaciones en el sistema
Ambiente	Múltiples usuarios conectados concurrentemente a la aplicación.
Respuesta	El sistema responde forma rápida.
Explicación	El sistema presenta un tiempo de respuesta a las peticiones relativamente rápido, gracias al mecanismo de cache de configuración que le facilita Symfony. Además garantiza que algunas páginas web respondan más rápido, debido a que no se hace necesario recargarla completamente cada vez que el usuario realiza cambios con el uso de Ajax, incrementando la interactividad y la rapidez.

Escenario #6	El sistema debe ser entendible y fácil de usar.
Estímulo	Facilidad de uso por parte de los usuarios del sistema.
Ambiente	Operación normal
Respuesta	El sistema cuenta con manuales de usuarios y videos tutoriales.
Explicación	El sistema presenta al usuario una interfaz atractiva e interactiva, con un menú general que lo guía en la búsqueda de la información que necesita. Además implementa patrones de diseño gráfico que garantizan una estructura afín a todas las páginas de los módulos, estableciendo iguales íconos para acciones similares y regularidades en el orden de aparición de algunos campos de formularios. Otra característica que contribuye a la facilidad de uso del sistema es la forma en que trata el enrutamiento. Se trata de un mecanismo encargado de reescribir las URL para hacer más entendible su aspecto. La URL que se le muestra al usuario se relaciona con el recurso solicitado, y no está obligada a guardar relación con la instrucción del servidor necesaria para completar su petición. Configurar la estructura interna de la aplicación no afecta el aspecto hacia el exterior que presentan las URL.

Escenario #9	Es fácil instalar las actualizaciones del sistema.
Estímulo	Actualizar el sistema.
Ambiente	Operación normal.
Respuesta	Mejora el rendimiento del sistema.
Explicación	El sistema, una vez desplegado y en ejecución, permite realizar las transferencias de archivos de actualización de forma incremental mediante el uso de la herramienta sincronización *"rsync"* que brinda Symfony mediante SSH. *"Rsync"* es una utilidad de línea de comandos que permite realizar una transferencia incremental de archivos de manera rápida y segura, donde solamente se transfieren los datos que han sido modificados, y en caso de que un archivo solo haya sufrido cambios parciales, solo se envían los cambios efectuados. Gracias a este mecanismo las sincronizaciones requieren del envío de muy pocos datos, sin el consiguiente gasto de tiempo y ancho de banda que provocaría hacerlo si se

	utilizara FTP, que además es muy propenso a cometer errores y puede ocasionar que el sitio web no esté disponible durante el traspaso de la información.

Paso #9: Conclusiones

El método se aplicó de forma correcta cumpliendo las metas trazadas en su comienzo, como son: la evaluación satisfactoria del diseño de la arquitectura propuesta y el trabajo en equipo teniendo en cuenta que era la primera vez que los revisores llevaban a cabo esta tarea. Se priorizaron y describieron nueve escenarios que habilitaron los atributos de calidad definidos en el Modelo ISO 9126-1:2001, concluyendo que el diseño propuesto cumple con los requisitos establecidos. De esta forma el equipo de revisión quedó satisfecho con el trabajo realizado, a pesar de no contar con una amplia experiencia por parte de los participantes que realizaron la evaluación.

3.3 CONCLUSIONES

Después de la selección de un método de evaluación de arquitectura y su especificación en el presente capitulo, se realizó la evaluación de la arquitectura con resultados cualitativos satisfactorios, aplicando una serie de escenarios guiados por el grupo de atributos de calidad definidos en el Modelo ISO 9126-1:2001. Dentro de los escenarios aplicados prevalecieron aquellos que exponen situaciones relacionadas con la seguridad del sistema dada la sensibilidad de la información que manipula el mismo, así como la facilidad de integración que debe prevalecer para la interoperabilidad y reutilización de funcionalidades.

CONCLUSIONES

Dado el estudio del Marco Regulatorio del GIS se evidencia que el sistema alasMEDIGEN.Sistema Informático de Genética Médica no cumple con el diseño e implementación que establece el mismo. Para dar solución se analizó la arquitectura de alasMEDIGEN y el Registro Informatizado de Salud permitiendo conocer las funcionalidades descritas y los componentes a reutilizar para desarrollar un diseño arquitectónico más flexible e interoperable.

Como resultado de la investigación se definió y describió la propuesta de Arquitectura Orientada a Servicios para el sistema alasMEDIGEN donde se exponen las vistas arquitectónicas de la metodología RUP con el propósito de lograr un sistema con un alto grado de modularidad e interoperabilidad. La vista de casos de uso permitió evidenciar la reestructuración de las dependencias entre las funcionalidades ya definidas para los sistemas. A su vez, la vista lógica permite mostrar la integración entre los sistemas de alasMEDIGEN con los registros del RIS. Finalmente la vista de despliegue propone la realización del mismo en tres servidores, brindando mayor seguridad y rendimiento al sistema.

A través de la evaluación de la arquitectura mediante el método ARID y la selección de los atributos de calidad definidos en el MODELO ISO 9126-1:2001. Se obtuvieron resultados positivos a través de la priorización de escenarios que alcanzaron su mayor peso en relación a la seguridad del sistema debido a la necesidad de restringir su acceso por el alto nivel de sensibilidad que presenta la información de alasMEDIGEN, y en la facilidad de cambio e integración que debe prevalecer para la interoperabilidad y reutilización de las funcionalidades.

RECOMENDACIONES

- ✓ Se recomienda completar la definición de los servicios propuestos a través de su descripción mediante WSDL.
- ✓ Se recomienda implementar y desplegar en el ambiente del RIS los servicios web definidos en la investigación.
- ✓ Se recomienda realizar la implementación y configuración de un ESB y una UDDI en el ambiente del RIS, para facilitar la integración y descubrimiento de los servicios web.

REFERENCIAS BIBLIOGRÁFICAS

(1). **BASS, Len; CLEMENT, Paul Y KAZMAN, Rich**. Software Architecture in Practice. s.l.: Addison-Wesley, 2003.

(2). **CLEMENTS, Paul; KAZMAN, Rich Y KLEIN, Mark**. Evaluating Software Architectures: Methods and Case Studies. s.l.: Addison Wesley, 2000.

(3). **JACOBSON, Ivar, BOOCH, Grady Y RUMBAUGH, James**. El Proceso Unificado de desarrollo de software. s.l.: Addison Wesley, 1999.

(4). **UCI**. Conferencia de Arquitectura de Software. Introducción a la Arquitectura de Software. Ciudad de la Habana: UCI, 2009.

(5). **1471-2000, IEEE Std**. Recommended Practice for Architectural Description of Software-Intensive Systems. 2000.

(6). **BUSCHMANN, F; MEUNIER, R; ROHNERT, H; SOMMERLAD, P; STAL, M.** Pattern – Oriented Software Architecture. A System of Patterns. Inglaterra: John Wiley & Sons, 1996.

(7). **RIVERA, Ignacio; ROBACIO, Luis**. Patrones Arquitectónicos, Layers. 2006.

(8). **DOBERKAT, Ernst-Erich.** Pipes and filters: Modelling a Software Architecture through relations. 2002, Junio.

(9). **STIGER, P. R Y GAMBLE, R. F**. Blackboard systems formalized within a software architectural style.

.

(10). **REINOSO, Carlos Billy.** Arquitectura Orientada a Servicios (SOA). [En línea]. [Consultado el: 23 de Febrero de 2010]. Disponible en: http://www.microsoft.com/spanish/arquitectura/roadmap_arq/Arquitectura-de-Software-Arquitecturas-Orientada-a-Servicios-SOA

(11). **UCI**. Conferencia Ingeniería de Software II. Arquitectura y Patrones de diseño. Ciudad de la Habana: UCI, 2007-2008**.**

(12). **DELGADO RAMOS, Ariel; CABRERA HERNÁNDEZ, Mirna; JUNCALS, Virginia**. Registro Informatizado de Salud (RIS), Ciudad de la Habana, 2006. [En línea]. [Consultado el: 22 de Febrero de 2010]. Disponible en: http://www.informatica2007.sld.cu/Members/arieldr/el-registro-informatizado-de-salud-ris-solucion-informatica-integral-para-el-sistema-nacional-de-salud/

(13). **CHAVIANO GÓMEZ, Enrique; CARRASCOSO PUEBLA, Yoan Arlet**. Propuesta de arquitectura orientada a servicios para el módulo de inventario del ERP cubano. [En línea]. [Consultado el: 23 de Febrero de 2010.]. Disponible en: http://www.gestiopolis.com/administracion-estrategia/erp-arquitectura-orientada-a-servicios.htm

(14). **CAVSI**. **¿**Qué es HTTPS? [En línea]. [Consultado el: 24 de Abril de 2010]. Disponible en: http://www.cavsi.com/preguntasrespuestas/que-es-https/

(15). **LÓPEZ, G; ECHEVERRÍA, A.; FIERRO, P.; JEDER, I.** Una Propuesta de Modelos de Ciclo de Vida (MCVS) para la Integración de los Procesos de Negocio utilizando Service Oriented Architecture (SOA). [En línea]. [Consultado el: 30 de Abril de 2010]. Disponible en: http://www.ing.unp.edu./arwicc2007trabajosISBD116.pdf.

(16). **Corporation, Copyright (C) IBM.** Ayuda Rational Unified Process. Version 7.0.1. 2006.

(17). **BREY, Gustavo Andrés; ESCOBAR, Gastón; PASSERINI, Nicolas; ARIAS, Juan.** Arquitectura de Proyectos de IT. Evaluación de Arquitecturas. Buenos Aires : Universidad Tecnológica Nacional. Facultad Regional de Buenos Aires - Departamento de Sistemas, 2005.

(18). **GÓMEZ, Omar; GÓMEZ, Salvador**. Evaluando Arquitecturas de Software. Parte 1. Panorama General. 01, México: Brainworx S.A, 2007.

(19). **CAMACHO, Erika; CARDESO, Fabio; NUÑEZ, Gabriel.** Arquitecturas de Software. 2004.

(20). **ISO/IEC 9126-1:2001**. Software Engineering. Product Quality- Part 1: Quality Model. [En línea]. [Consultado el: 25 de Abril de 2010]. Disponible en: http://webstore.iec.ch/preview/info_isoiec9126-1%7Bed1.0%7Den.pdf

(21). **MEAD, Nancy R**. Requirements Prioritization Introduction. [En línea]. [Consultado el: 25 de Abril de 2010]. Disponible en: https://buildsecurityin.us-cert.gov/bsi/articles/best-practices/requirements/545-BSI.html.

BIBLIOGRAFÍA

1. **BARBACCI, Mario R, y otros.** Quality Attribute Workshop Participants Handbook . [En línea] 2000. [Citado el: 22 de Febrero de 2010] http://www.sei.cmu.edu/reports/00sr001.pdf.
2. **BARCO, Antonio.** Artículo Tecnológico: "WSDL: El contrato de un Servicio". [En línea] 12 de Diciembre de 2006. http://arquitecturaorientadaaservicios.blogspot.com/2006/12/artculo-tecnolgico-wsdl-el-contrato-de.html
3. **Catalunya, Universidad Politécnica.** Arquitecturas Empresariales. Orientación a Servicios (SOA) y Gestión de Procesos de Negocio (BMP). [En línea] 2007. http://www.scribd.com/doc/12732986/Arquitecturas-Empresariales-Orientacion-a-Servicios-SOA-y-Gestion-de-Procesos-de-Negocio-BPM.
4. **CLARO, A., y otros.** Sistema Informático de Genética Médica. [En línea] 2009. http://informatica2009.sld.cu/Members/aclaro/sistema-informatico-de-genetica-medica/.
5. **CLEMENTS, Paul C.** Active Reviews for Intermediate Designs(ARID). [En línea] 2000. http://www.sei.cmu.edu/reports/00tn009.pdf.
6. **COLLADO, Cecilia**. SOA: Primeros pasos. [En línea] 2009. http://tecnologiaescrita.wordpress.com/2009/02/17/soa-primeros-pasos/.
7. **DELGADO RAMOS, Ariel, CABRERA HERNÁNDEZ, Mirna y JUNCALS, Virginia.** Registro Informatizado de Salud (RIS). [En línea] 2006. http://www.informatica2007.sld.cu/Members/arieldr/el-registro-informatizado-de-salud-ris-solucion-informatica-integral-para-el-sistema-nacional-de-salud/.
8. **GARTNER.** The Gartner Glossary of Information Technology Acronyms and Terms. [En línea] 2004. http://www.gartner.com/6_help/glossary/Gartner_IT_Glossary.pdf.
9. **GONZALEZ, Benjamin.** UDDI (Universal Description Discovery and Integration). [En línea] 2004 . http://www.desarrolloweb.com/articulos/1589.php.
10. Introducción a la Arquitectura de Software. [En línea] 2004. http://www.willydev.net/descargas/prev/IntroArq.pdf.
11. Principios de la orientación a servicios. [En línea] 2006. http://arquitecturaorientadaaservicios.blogspot.com/2006/06/principios-de-la-orientacin-servicios.html.

12. **QUALITY ATTRIBUTE WORKSHOP(QAW)**. [En línea] http://www.sei.cmu.edu/architecture/tools/qaw/index.cfm.
13. **REINOSO, Carlos Billy.** Arquitectura Orientada a Servicios (SOA). [En línea] http://www.microsoft.com/spanish/arquitectura/roadmap_arq/Arquitectura-de-Software-Arquitecturas-Orientada-a-Servicios-SOA.
14. **REINOSO, Carlos Billy**. MSDN. [En línea]. http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/intro.mspx.
15. **RIVERA, Ignacio y ROBACIO, Luis.** Patrones Arquitectónicos, Layers. [En línea] 2006. [http://dc.exa.unrc.edu.ar/nuevodc/materias/ingenieria/Material/1163109537/1163110516/Layer%201.pdf.
16. **SOFTEL.** Marco Regulatorio GIS. s.l. : GIS 07-01, 2009.
17. **STOERMER, Christoph, BACHMANN, Felix y VERHOEF, Chris.** SACAM: The Software Architecture Comparison Analysis Method. [En línea] 2003. http://www.sei.cmu.edu/reports/03tr006.pdf.
18. **UCI.** Conferencia Ingenieria de Software II. Arquitectura y Patrones de diseño. Ciudad de la Habana : UCI, 2007-2008.
19. **UCI**. Conferencia Ingeniería de Software. Métodos basados en la Arquitectura de Software. Ciudad de la Habana : UCI, 2009.
20. **UNIVERSITY, CARNEGIE MELLON.** Cost Benefit Analysis Method. [En línea] http://www.sei.cmu.edu/architecture/tools/cbam/index.cfm.

ANEXOS

Anexo 1. Cliente solicitando información a un servicio Web utilizando SOAP