Universidad de las Ciencias Informáticas
Facultad 7

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

Título: Desarrollo de la Capa de Mensajería del Servicio de Terminologías Comunes

Autora:

Karín Pérez Salas

Tutor:

MSc. Reinier Alonso González

Co-tutor:

Ing. Alberto Varona Carmenate

Ciudad La Habana

Junio 2011

“Año 53 de la Revolución”

DATOS DE CONTACTO

MSc. Reinier Alonso González: Graduado de Ingeniería en Ciencias Informáticas en la Universidad de Ciencias Informáticas (UCI), año 2007 con categoría de Instructor. Ha impartido las asignaturas de Introducción a la programación, Programación I y Arquitectura de Software para la Salud. Graduado de Máster en Gestión de Proyectos en el año 2010. Se ha desempeñado en tutorías y oponencias de tesis en cursos anteriores. Ha presentado varias publicaciones y trabajos en eventos como UCIENCIA, en las diferentes ediciones realizadas, en Informática 2009 y 2011, entre otros.

Correo electrónico: ralonso@uci.cu

Ing. Alberto Varona Carmenate: Trabajador del CESIM en el departamento Tecnologías, Integración y Estándares, TIE, desempeñando el rol de Arquitecto de la Información. Fue tutor en el 2009-2010 de la tesis Módulo Salud Materno Infantil de la Atención Primaria de Salud.

Correo electrónico: avarona@uci.cu

"El trabajo debe dejar de ser una penosa necesidad para volverse un agradable imperativo. Las nuevas relaciones de producción deben servir para acentuar la evolución del hombre hacia el reino de la voluntad".

Ernesto Guevara de la Serna.

AGRADECIMIENTOS

Agradecer es reconocer el esfuerzo y la dedicación de aquellos que nos dan la mano, es reconocer que necesitamos de ellos y valoramos su presencia en nuestras vidas.

Primeramente quiero agradecer a las personas que han contribuido a la realización de este trabajo de diploma. A mi compañero de tesis Alberto que aunque no esté hoy conmigo me ayudó mucho con su esfuerzo y dedicación. A mis tutores, profesores del proyecto y mi oponente por guiarme y orientarme durante la realización del trabajo.

Quiero agradecer además a las personas que durante mi paso por la universidad han ayudado a ser una mejor profesional. A mis padres por ser mi mejor ejemplo, porque todo lo que soy es fruto de su sacrificio, gracias por estar siempre. A mi hermano por ser mi inspiración para ser una mejor persona cada día y ser un buen ejemplo para él. A Yoenny por todo el tiempo que me ha dedicado, por ser mi mayor guía, por estar siempre en todos los momentos, por escucharme y comprenderme. A mis compañeras de estos 5 años Yisel y Angélica. A mi compañero de siempre Maikel por todas las noches de estudio y su dedicación .A todos mis compañeros de grupo y apartamento.

A todos aquellos profesores que han contribuido en mi formación

A la gran obra que es la Revolución Cubana y a Fidel, por haberme dado la oportunidad de estudiar en la Universidad de Excelencia.

DEDICATORIA

Dedico este trabajo de diploma a mis padres, los que me hacen ser mejor persona cada día, por todos los sacrificios que han hecho para verme realizada como profesional y a mi hermano, que espero que le sirva de guía para convertirse el día de mañana en un buen profesional.

RESUMEN

En el Centro de Informática Médica se desarrollan aplicaciones especializadas en las diferentes áreas de atención sanitaria, estas manejan varias terminologías médicas y gestionan grandes volúmenes de información. Estos sistemas tienen dificultades para interoperar, interpretar y traducir la información que circula entre ellos. Por lo que se hace necesario implementar un Servicio de Terminologías Comunes que funcione de puente de comunicación entre los sistemas garantizando un buen entendimiento entre ellos y para esto es necesario la construcción y ejecución de la mensajería de manera estándar que facilite la interpretación, interoperabilidad y traducción de la información.

En el desarrollo del sistema se utilizó el Proceso Unificado de Desarrollo y se basa en tecnologías libres, multiplataforma. Utiliza Java como lenguaje de programación. Como Sistema de Gestión de Bases de Datos se utiliza PostgreSQL. Como herramienta CASE el Enterprise Architect que utiliza el lenguaje de modelado UML y para la implementación del sistema el Eclipse, un Entorno Integrado de Desarrollo.

Entre los beneficios que aporta el servicio se encuentran mejorar la gestión de la información entre los sistemas de salud, garantizando homologar las terminologías médicas que utilizan y un factible intercambio e interpretación de información entre los mismos.

Palabras clave:

Servicios de Terminologías Comunes, Capa de Mensajería, Interoperabilidad, Interpretación, Terminologías Médicas.

ÍNDICE

ÍNDICE

INTRODUCCIÓN

El vertiginoso desarrollo de la informática y las comunicaciones ha impactado de forma positiva en casi todos los sectores de la sociedad moderna. El uso de la informática en la medicina es una de las aplicaciones más comunes e importantes desde hace varias décadas. Esto ha permitido al sector de la salud, no sólo contar con métodos novedosos, sencillos y eficaces de gestión administrativa en consultas, hospitales y centros de investigación biomédica; sino también disponer de complejas aplicaciones que reducen la posibilidad de error en el diagnóstico de las enfermedades, y que aceleran su formulación. (1)

Cuba se inserta y avanza en el mundo de la informática, trazándose como una de sus metas más ambiciosas, dar una alta prioridad a la salud de todo el pueblo. Para ese efecto se dispone de cuantos recursos y facilidades fueran necesarias, por lo cual se están diseñando y organizando estrategias que permiten convertir los conocimientos, las tecnologías de la información y las comunicaciones en instrumentos a disposición del avance de las profundas transformaciones de la sociedad.

Como parte de este proceso surge en el sector de la salud cubana, la informatización del Sistema Nacional de Salud (SNS), dirigida al manejo de la información en esta área, incrementa la efectividad y eficiencia en el entorno de la salud asistencial, tiene como eslabón principal, la población. De ahí que las distintas formas de atención a la salud aparecen relacionadas con los sistemas de información.

En la actualidad debido a la informatización de los procesos clínicos asistenciales, más la integración de redes asistenciales se migró del concepto de Sistemas de Información Hospitalarios a Sistemas de Información de Salud (HIS = Healthcare Information Systems). (2) Una de las premisas de estos nuevos sistemas fue respetar los procesos asistenciales, coloca al acto médico como eje central de su modelo de información. Desde entonces la descentralización de la atención médica en redes asistenciales generó una nueva necesidad, la de conectar múltiples sistemas. Ya no basta con mantener el registro único en los sistemas internos de una organización, sino que es necesario ir más allá de los muros de una institución y permitir una fluida comunicación de la información clínica.

La Universidad de las Ciencias Informáticas es una casa de altos estudios que además de formar profesionales altamente capacitados en el área de la informática, tiene como misión potenciar el desarrollo de software en el país. Como parte de la estrategia para dar cumplimiento a sus objetivos se crearon

varios centros especializados para el desarrollo de software en diferentes líneas de trabajo. La actividad fundamental del Centro de Informática Médica ha sido enfocada a la informatización del Sistema Nacional de Salud Pública de Cuba.

Situación problémica

En el estudio realizado para conocer los Sistemas de Salud, se encontraron varias dificultades que limitan el intercambio e interpretación de información entre diferentes aplicaciones de salud. Entre estas dificultades se tiene que la información se encuentra fragmentada entre los diferentes sistemas, hay una falta de coordinación entre los niveles de atención y los lugares donde se ofrecen servicios que conducen a una atención fragmentada e inoportuna, a la duplicación de la información y los servicios médicos que se brindan como: estudios de laboratorio, estudios por imágenes y consultas médicas; existen además deficiencias en el rápido acceso a los datos y la terminología médica que contienen estos sistemas; otra dificultad que se evidencia es el almacenamiento del texto en forma de lenguaje natural porque es poco útil al momento de recuperar y analizar la información que es recopilada.

Además existe la dificultad de entender los mensajes de comunicación de manera correcta, hay deficiencias en el manejo de vocabularios clínicos y codificadores de terminología médica, los sistemas desarrollados en el CESIM manejan terminologías médicas conceptualmente iguales pero lo hacen de forma diferente, por ejemplo el alasHIS para clasificar a una persona en femenino o masculino utiliza el dominio de vocabulario género mientras que el alasRIS utiliza el dominio de vocabulario sexo, esta ambigüedad está presente en la mayoría de los sistemas de salud e impide que las terminologías médicas utilizadas por estos sistemas sean homólogas, esto es una gran limitante para la correcta integración, interoperabilidad e interpretación de la información entre los diferentes sistemas de información sanitaria.

Dada la situación anterior el **problema a resolver** radica en: ¿Cómo facilitar la interpretación de mensajes entre aplicaciones de información clínica y el servicio de terminologías comunes especificado en el estándar HL7 (Health Level Seven)?

Se enmarca como **objeto de estudio** el estándar de comunicación entre aplicaciones médicas HL7. El **campo de acción** se centra en la Capa de Mensajería del Servicio de Terminologías Comunes (CTS, por sus siglas en inglés).

INTRODUCCIÓN

Se propone **como objetivo general** implementar una Capa de Mensajería que garantice la navegación y ejecución de los mensajes HL7 que contienen la información que circula entre los distintos sistemas de información sanitaria.

Como **objetivo específico** se plantea:

- Implementar las funcionalidades de navegación (Message Browser).

Para dar cumplimiento al objetivo definido se trazaron las siguientes **tareas de la investigación:**

- Realizar un análisis crítico y valorativo de las tendencias, técnicas, tecnologías, plataformas, bibliotecas, metodologías y herramientas usadas en la actualidad.
- Analizar los procesos de negocio asociados al intercambio de mensajes relacionado con la Capa de Mensajería de CTS, logrando un modelo único como guía para la implementación de la misma.
- Generar los artefactos correspondientes a los Flujos de Trabajo propuestos por la Metodología RUP, sirviendo de base a los desarrolladores.
- Implementar la Capa de Mensajería siguiendo lo establecido en la Especificación de Requisitos de Software.

Entre los métodos científicos de investigación se utilizaron los métodos teóricos que a continuación se argumentan:

Analítico-Sintético: este método permite estudiar los sistemas de salud y buscar información para encontrar una posible solución a la situación problémica planteada, así como identificar herramientas y métodos a utilizar relacionados con el objeto de estudio.

Inductivo-Deductivo: este método permite llegar a conclusiones lógicas a partir de los conocimientos adquiridos en la investigación, y así poder plantear la estrategia de desarrollo para lograr una correcta interoperabilidad entre aplicaciones de salud.

Análisis Histórico-Lógico: este método permite estudiar cómo ha evolucionado el intercambio de información entre aplicaciones de salud y la interpretación de los vocabularios clínicos en los diferentes sistemas de salud.

Como **resultados esperados** una vez concluidas las tareas de la investigación se plantean: Obtener un servicio web implementado con las funcionalidades de navegación de la Capa de Mensajería del proyecto Servicios de Terminologías Comunes y la integración de estas funcionalidades con las demás capas de CTS.

La estructura del presente documento se compone de tres capítulos en los que se recoge la información obtenida desde el punto de vista investigativo y lo referente al negocio, diseño e implementación de la Capa de Mensajería.

CAPÍTULO 1: Fundamentación Teórica: En este capítulo se abordan los temas relacionados al proceso de informatización en el sector de la salud, incluyendo conceptos fundamentales relacionados con el dominio del problema. Se realiza un estudio de las soluciones existentes y un resumen descriptivo de las tendencias, tecnologías y herramientas actuales más usadas que son consideradas para la elaboración de la capa.

CAPÍTULO 2: Características del Sistema: Ubica al lector en un marco conceptual asociado a la información que será manipulada por la Capa de Mensajería, se centra en la modelación detallada y la construcción de la estructura del servicio.

CAPÍTULO 3: Análisis de Resultados: Se presenta la propuesta de solución para lograr el correcto intercambio de mensajes entre las aplicaciones de salud y se analizan los resultados obtenidos con la implementación de la Capa de Mensajería.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se brinda la base teórica y conceptual de la capa a desarrollar. Como aspectos esenciales se abordan las principales definiciones relacionados con el dominio del problema y el objeto de estudio, los antecedentes existentes de sistemas o subsistemas similares al campo de acción de la investigación. Así como, un análisis de las tecnologías, metodologías y herramientas a utilizar.

1.1 Sistema Nacional de Salud.

La garantía de atención médica gratuita a toda la población cubana se convirtió desde los primeros momentos del triunfo de la Revolución en uno de los paradigmas sociales fundamentales. Se comenzó a trabajar por la creación del Sistema Nacional de Salud que llevó la acción del trabajador de la salud a los lugares más apartados.

El Ministerio de Salud Pública es el organismo rector del SNS (Sistema Nacional de Salud). Encargado de dirigir, ejecutar y controlar la aplicación de la política del Estado y del Gobierno en cuanto a la Salud Pública, el desarrollo de las Ciencias Médicas y la Industria Médico Farmacéutica. El SNS es un sistema único, integral y descentralizado para la atención de salud de la población y su pilar fundamental lo constituye la atención primaria de salud. (3)

1.2 Informatización del Sistema Nacional de Salud.

La principal meta del proceso de informatización de la salud cubana está encaminada a elevar la calidad en los servicios de salud que se prestan a la población. Se están dando los primeros pasos para llevar a cabo la implementación de los Servicios Integrales de Salud (SISalud), con el objetivo de lograr la incorporación progresiva y sistemática de las Nuevas Tecnologías de la Información y las Comunicaciones (NTICs), en función de desarrollar la informática en el sector para la adquisición, procesamiento e interpretación de datos de pacientes, con ayuda de la investigación científica. (4)

En la concepción y desarrollo para lograr la integración e implementación de sistemas informáticos y servicios encaminados a informatizar la salud cubana participan diferentes empresas del MIC como: DESOFT, derivada de la entidad legal Empresa Nacional de Software, SOFTEL, la UCI, el Centro de Desarrollo Informático para la salud Pública (CEDISAP) y las Direcciones Nacionales del Ministerio de Salud Pública, implicadas directamente en la informatización en este sector, teniendo como principal objetivo elevar la eficiencia de los servicios de salud con sistemas informáticos y tecnologías de avanzada.

1.3 Sistema de Información en los Sistemas de Salud.

Un sistema de información es aquel que está constituido por un conjunto de instrucciones organizadas, sistematizadas y lógicas que se relacionan entre sí por medio de un lenguaje informático. Persigue el fin de obtener información, analizarla, relacionarla y generar nueva información para satisfacer las necesidades de las áreas administrativas, operativas de una organización en general. Estos proveen comunicación entre los miembros del equipo de salud y dan soporte organizacional a la gestión de la información para realizar operaciones, planeamiento, atención del paciente y el registro de sus actos. (5)

Los sistemas de información clínicos están compuestos por múltiples componentes, estos no solo están conformados por piezas de software sino también por recursos humanos y tecnológicos. Deben ser considerados como subsistemas en el sistema de información en organizaciones de salud. Cada uno de ellos tiene funciones específicas. (6)

Los componentes de los Sistemas de Información en Salud son:

- Componente Computacional.
- Componente de Sistemas Administrativos.
- Componente de Interoperabilidad.
- Componente de Servicios Terminológicos.
- Componente de Registro Clínico Electrónico.
- Componente de Registro Personal de Salud.
- Componente de Seguridad.
- Componente de Soporte para la Toma de Decisiones.
- Componente de Agregación de la Información.
- Componente Organizacional.

Componente Computacional: este componente envuelve al resto, es la matriz que posibilita el funcionamiento de los otros componentes. Representa todo el soporte tecnológico por medio del cual el

sistema de información puede llevar a cabo sus funciones. Éste componente puede subdividirse en tres grandes áreas: la tecnología de comunicaciones utilizada para dar conectividad a los sistemas, el hardware (computadoras) y el software (aplicaciones) que se utilizan para dar soporte al procesamiento de la información institucional. (7)

Componente de Sistemas Administrativos: los sistemas de información en salud centraron su desarrollo en el soporte administrativo de los procesos asistenciales, con énfasis en la logística y facturación de los actos médicos. Actualmente la mayoría de las instituciones poseen sistemas informáticos que brindan esta funcionalidad. En esta capa administrativa se encuentran sistemas dedicados a la gestión de diferentes ámbitos tales como: gestión de pacientes, gestión de recursos, gestión de productos, gestión de registros médicos y gestión contable y financiera. (8)

Componente de Interoperabilidad: cuando el sistema de información está construido por un sólo tipo de desarrollo informático y no requiere la integración de otros sistemas, el componente de interoperabilidad no es necesario (salvo si desea comunicarse externamente). Pero si el sistema está compuesto por componentes o sistemas de múltiples desarrollos informáticos, existe la imperiosa necesidad de integrarlos para lograr la interoperabilidad, que no es más que la capacidad de dos o más sistemas de intercambiar y utilizar información entre ellos. (9)

De esta definición se desprenden dos tipos de interoperabilidades:

Interoperabilidad sintáctica (operativa o funcional): se encarga de comunicar diferentes sistemas preexistentes o nuevos (generalmente creados con disímiles herramientas de desarrollo y sobre múltiples bases de datos) en el sistema de información. Una de las posibles soluciones es crear una interfaz que haga que las aplicaciones se comuniquen entre sí, pero no asegura que los mismos puedan utilizar la información que intercambian. Es una solución muy utilizada cuando se tienen pocos sistemas, pero carece de escalabilidad ya que la cantidad de interfaces a generar crece exponencialmente al aumentar el número de sistemas a integrar. (10)

Una solución que da escalabilidad es utilizar una interfaz de comunicación estándar, este es el caso por ejemplo del estándar Health Level Seven (HL7) que utiliza mensajería electrónica para integrar múltiples sistemas. Esta solución se encarga de comunicar los sistemas desde el punto de vista operativo

o funcional, sin embargo la verdadera interoperabilidad se obtiene sumando el componente de integración semántica.

Interoperabilidad semántica: es el encargado de almacenar, administrar, integrar y unificar los diccionarios o vocabularios comunes que utilizan los sistemas que deben ser integrados. El diccionario más importante es el denominado Maestro o Padrón Único de Pacientes, porque es el identificador por el cual se almacena toda la información en el repositorio de datos clínicos.

Componente de Servicios Terminológicos: los miembros del equipo de salud registran su actividad asistencial de forma narrativa, la cual mantiene gran cantidad de información contextual necesaria para la comunicación con sus pares pero la misma puede ser ambigua. Varios conceptos pueden estar representados por un mismo término. De ese texto narrativo se presenta como una de las soluciones la codificación secundaria. Otra solución para disminuir la ambigüedad, es obligar el ingreso estructurado de la información, codificación primaria, lo que permite una rápida utilización de la información cargada por los sistemas de información y la agregación de los datos para su posterior análisis. Una opción de avanzada es lograr la autocodificación de textos, (11) permitiendo la entrada de textos libres o narrativos y la interacción dinámica en la incorporación de los mismos por parte del sistema de información.

El componente de terminología clínica se encarga de dar servicios terminológicos que permitan lograr un adecuado equilibrio entre la libertad de los textos narrativos y los beneficios de la introducción estructurado de datos. Debido a la complejidad de las terminologías de referencia ya no es posible la codificación manual en estos entornos, por lo cual es necesario desarrollar un software denominado servidor de terminología clínica. (12)

Brinda servicios al resto de los componentes permitiendo la incorporación de textos narrativos en las aplicaciones, que luego son autocodificados por el servidor. Esto facilita el refinamiento crítico e interactivo de los textos ingresados por el usuario y permite mejorar la calidad de registro, almacenando códigos controlados como texto narrativo en el repositorio de datos clínicos. El adecuado almacenamiento de la información en el repositorio de datos clínicos es vital, entre otros beneficios, para dar sustrato a los sistemas de soporte para la toma de decisiones, razón de ser de todo aplicativo en capa clínica.

1.4 La especificación Servicios de Terminologías Comunes.

La especificación Servicios de Terminologías Comunes (CTS), fue desarrollada como una alternativa a una estructura de datos común. En lugar de especificar lo que una terminología externa debe parecer, HL7 ha elegido para identificar las características funcionales comunes que una terminología externa debe ser capaz de proporcionar. A modo de ejemplo, un servicio de terminología HL7 compatible, tendrá que ser capaz de determinar si un concepto de código es válido en el recurso en particular, en lugar de describir una tabla con llave con el identificador de recursos y el código de concepto, la especificación CTS describe una Interfaz para Programación de Aplicación (API) que toma un identificador de recurso y código, de tipo conceptual, como entrada y devuelve un valor verdadero / falso. Cada desarrollador de terminología es libre de aplicar esta llamada a la API de manera que sea más apropiado para ellos. (13)

Para el desarrollo de la especificación CTS de HL7 se utilizaron los siguientes principios de diseño:

- Será fácil desarrollar sistemas que utilizan HL7 CTS.
- Se especificaron solamente los servicios básicos que proporciona CTS.
- El diseño de la especificación es formal y conciso, puede adaptarse fácilmente a las necesidades de los desarrolladores que deseen utilizar la especificación.
- La tecnología inicial puesta en práctica para el desarrollo de HL7 CTS incluye XML (Lenguaje de Marcas Extensible).
- Debe ser compatible con la nomenclatura, el modelo y el enfoque expresado en el documento de vocabulario HL7, la versión 3 de RIM (Reference Information Model) y sus estructuras derivadas.
- Siempre que sea posible, el HL7 CTS seguirá siendo un subconjunto coherente del (OMG) (Grupo de administración de objetos), Servicios de Consultas de Terminologías (TQS), siempre que el modelo de TQS no entre en conflicto con otros principios de diseño HL7 CTS.

1.4.1 Las Capas Mensajería y Vocabulario.

Existen dos capas diferentes entre las aplicaciones de procesamiento de mensajes HL7 y las de vocabularios: la Capa de Mensajería (Message API) es la superior, esta se comunica con el software de mensajería y lo hace en términos de dominios de vocabulario, contexto, conjuntos de valores, atributos codificados y otros artefactos del modelo de mensajes HL7; la capa inferior Vocabulario (Vocabulary API)

se comunica con el software de almacenamiento de terminología utilizando un modelo conceptual propio que habilita el manejo de sistemas de código, códigos de concepto, definiciones, relaciones y otras entidades específicas de la terminología. (14)

Las Capas Mensajería y Vocabulario tienen funcionalidades específicas de tiempo de ejecución (Message Runtime) y navegación (Message Browser):

	Runtime	Browser
Capa de Mensajería	Message Runtime	Message Browser
Capa de Vocabulario	Vocabulary Runtime	Vocabulary Browser

Tabla1.1 *Especificación de componentes.*

Esta especificación debe estar escrita de manera que sea posible que cada una de estas capas puedan ser desarrolladas de forma independiente y aun así interoperar.

La Capa de Vocabulario es genérica, esta permite a las aplicaciones hacer diferentes consultas a terminologías de manera coherente y bien definida. La Capa de Mensajería utiliza la Capa de Vocabulario. Por ejemplo, una aplicación solicita que el servicio de Mensajería en Tiempo de Ejecución llene los detalles de un atributo. El servicio, a su vez, hace varias llamadas al servicio Capa de Vocabulario con el fin de obtener las designaciones concepto de código, los nombres de código del sistema, las versiones de lanzamiento, etc. (15)

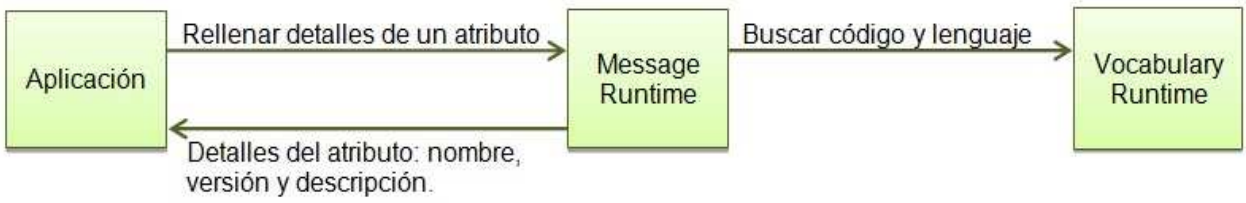

Figura 1.1 *Ejemplo de interacción entre las capas Mensaje y Vocabulario.*

1.5 Conceptos relacionados con el campo de acción.

HL7 (Health Level Seven): organización que desarrolla especificaciones de estándares para el intercambio electrónico de información médica.(16) Estas especificaciones pueden ser usadas para solucionar problemas de integración entre distintos sistemas.

Servidor de Terminologías: es un servidor que almacena todos los vocabularios de los sistemas médicos informatizados, en vocabularios controlados. Facilita el manejo e intercambio de información del sistema. (17)

CTS (Common Terminology Services): define una serie de Interfaces de Aplicación (API) y un modelo de datos que permiten integrar y acceder de manera uniforme a diferentes terminologías. Ofrece un modelo conceptual y de datos para la gestión y almacenamiento de terminologías. CTS identifica las características funcionales comunes que una terminología debe proveer.

API (Application Programming Interface): es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Las APIs de CTS describen la funcionalidad básica necesaria para el acceso al contenido terminológico desde sistemas de información. (18)

1.6 Estudio del Estado del Arte.

En la actualidad no se ha desarrollado un estándar para los servicios de terminologías, sin embargo, hay varias especificaciones de ellos implementadas. A continuación se abordan las principales especificaciones encontradas que son desarrolladas para servicios terminológicos.

1.6.1 TQS/LQS

Servicios de Consultas de Terminologías (TQS), formalmente conocido como Servicios de Consultas Léxicas (LQS), es un completo servicio de terminología publicado por el Grupo de Gestión de Objetos (OMG, por sus siglas en inglés). OMG es una organización internacional fundada en 1989 que cuenta con más de 800 miembros incluyendo proveedores de sistemas, desarrolladores de software y usuarios. Sus principales metas son la reusabilidad, la portabilidad y la interoperabilidad de softwares basados en objetos en ambientes distribuidos y heterogéneos. (19) (20)

LQS se encuentra en su versión 1.0, está compuesto por 13 módulos y especifica un conjunto de interfaces, Interface Description Language (IDL, por sus siglas en inglés) que se utilizarán en la consulta y el acceso a servicios informatizados de terminología médica. Esta especificación define cómo los métodos deberían ser invocados desde una amplia variedad de lenguajes de programación y describe el comportamiento esperado de estos métodos. Aborda un amplio espectro de terminologías médicas representadas como una simple tabla con los atributos código-valor o como una terminología sofisticada,

ejemplo, SNOMED-CT. No es aplicado de manera generalizada, el soporte del proveedor es mínimo y es muy compleja. (21)

1.6.2 OWL

El Lenguaje de Ontología Web (OWL) es un lenguaje de marcado semántico creado por el OWL Working Group para publicar y compartir ontologías en la World Wide Web. Actualmente se encuentra en su versión 2. Destinado a ser utilizado cuando la información contenida en los documentos necesita ser procesadas por las aplicaciones. Puede ser utilizado para representar explícitamente el significado de los términos en los vocabularios y las relaciones entre esos términos. Esta representación de términos y sus interrelaciones es una ontología. (22)

OWL tiene más facilidades para expresar el significado y la semántica que Extensible Markup Language (XML), Resource Description Framework (RDF, por sus siglas en inglés) y Resource Description Framework - Schema (RDF-S, por sus siglas en inglés), y por lo tanto va más allá de estos lenguajes en su capacidad para representar en la Web, el contenido interpretable por la máquina. Es una revisión del lenguaje de ontologías web DARPA Agent Markup Language (DAML) + Ontology Inference Layer (OIL), incorporando las lecciones aprendidas en el diseño y aplicación de DAML + OIL. (23).

Fue propuesto OWL en vez de WOL por ser un acrónimo de fácil pronunciación en inglés, facilita buenos logos y se relaciona con el prestigioso proyecto de representación del conocimiento de los años setenta de Bill Martin One World Language. (24)

Actualmente, OWL tiene tres variantes:

- OWL Lite.
- OWL DL.
- OWL Full.

Estas variantes incorporan diferentes funcionalidades, y en general, OWL Lite es más sencillo que OWL DL, y OWL DL es más sencillo que OWL Full. OWL Lite está construido de tal forma que toda sentencia pueda ser resuelta en tiempo finito, la versión más completa de OWL DL puede contener 'bucles' infinitos.

OWL no es estrictamente una especificación de servidor de terminología pero contiene elementos de la representación de aspectos ontológicos que son relevantes para algunas de las terminologías a gran

escala, tales como SNOMED, NHS Clinical Terms Version 3, y GALEN. Esta propuesta basada en la Web no facilita su implementación generalizada. (25)

1.6.3 CliniClue Xplorer

Fue desarrollado por Clinical Information Consultancy Ltd y es uno de las buscadores gratuitos descargables vía web más conocidos. Requiere registro en línea, pero también ofrece una versión de prueba de 10 días sin registro. El programa obtiene la terminología SNOMED mediante descarga desde un servidor de actualizaciones propio, y permite que el usuario seleccione la distribución que le resulte más conveniente. No sólo permite búsquedas, también permite la navegación de SNOMED mediante un visor en el que se puede, interactivamente, saltar a través de sus términos siguiendo las relaciones que aplican a cada uno de los términos. (26)

Ofrece también servicios web de búsqueda sobre su servidor de terminología. Además de las búsquedas por palabras, que ya ofrecen los buscadores en red, permite también filtrar los resultados por jerarquías. Tiene como desventajas que no permiten el uso de otras terminologías, ni el acceso de herramientas externas a sus funciones. (27)

1.6.4 Bioportal

Bioportal es una herramienta del Centro Nacional de Ontologías Biomédicas (NCBO, por sus siglas en inglés) de los Estados Unidos de América que permite tanto la carga de terminologías y ontologías por red a una base de datos como el acceso posterior a las mismas mediante una página web, o mediante funciones de servicio web. Esta característica hace que puedan programarse herramientas que utilicen estos servicios, de forma gratuita, aunque para poder añadir aportes propios a la base de datos se precisa crear una cuenta en la comunidad NCBO.

Todas las terminologías y nuevas ontologías cargadas son validadas antes de permitir su almacenamiento y posterior acceso. Aunque conveniente, esta característica podría limitar el uso de cualquier herramienta implementada que use Bioportal para acceder a los datos, ya que su funcionamiento dependerá del estado de un servidor ajeno. Por otro lado, las funciones de servicio web son genéricas para permitir el acceso a distintas terminologías. Para crear una herramienta basándose en Bioportal habría que modificar las funciones que se implementarían para hacer búsquedas específicas en

SNOMED a través de un servicio web, que a su vez accedería al servicio web de Bioportal, sobre el que no se tendría control alguno. (28)

1.6.5 UMLS

El Sistema de Lenguajes Médicos Unificados (UMLS, por sus siglas en inglés) es un proyecto creado por la Biblioteca Nacional de Medicina (NLM, por sus siglas en inglés) de los Estados Unidos de América para unificar las diferentes terminologías médicas existentes. Sus tres componentes principales son:

- El MetaThesaurus.
- Una taxonomía o red semántica de tamaño reducido (poco más de un centenar de clases).
- Un conjunto de herramientas léxicas sólo disponibles en inglés.

El MetaThesaurus es un gran compendio de diccionarios donde las palabras son términos clínicos y los diferentes idiomas/diccionarios son las diferentes terminologías que las contienen y que UMLS abarca (SNOMED CT, ICD-9-CM, MeSh, entre otras). La propia NLM se encarga de integrar las terminologías más relevantes en el MetaThesaurus (unas 150 en la última versión del 2009), así como de su actualización y mantenimiento.

Cada terminología es independiente y establece unas relaciones entre sus términos y una codificación propia de los mismos. Pero además, dentro del MetaThesaurus se establecen relaciones de equivalencia entre términos de diferentes terminologías que aluden al mismo elemento (concepto o universal), como si se tratara de una traducción entre idiomas. Para hacer esto posible, cada terminología es sometida a un proceso de mapeo de sus términos con conceptos de UMLS ya existentes, y a cada uno se le asigna un Common Unique Identifier (CUI) del concepto que representa. Si el concepto no existe todavía en UMLS, se crea uno nuevo, al que se asigna un nuevo CUI, tras lo cual se le asocia el término importado normalmente. De esta forma no se pierden los términos originales y se permite la coexistencia de distintas terminologías aunque subconjuntos de sus términos se solapen unos con otros.

El objetivo del NLM al crear UMLS era facilitar el acceso homogéneo a terminologías y, además, facilitar la creación de herramientas semánticas para aprovechar estos recursos. Por ello, además del identificador de concepto, a los términos importados se les asigna otro atributo importante, el Tipo Semántico (o clase semántica), que los enlaza con la Red Semántica de UMLS. Esta asociación permite diferenciar entre significados de un mismo término según el contexto en el que aparece.

La herramienta de instalación del MetaThesaurus de UMLS, MetamorphoSys, permite elegir las terminologías que se van a instalar generando unos archivos en un formato Rich Release Format (RRF), además de facilitar programas script para la inserción de los datos en una base de datos relacional, por ejemplo MySQL.

Aunque UMLS incluye un buscador propio para realizar consultas sobre las terminologías instaladas, no ofrece ni interfaces de programación (API, por sus siglas en inglés) ni servicios para la programación de buscadores, ya sean accediendo a una distribución en local o en remoto mediante servicio web. Se puede emplear UMLS para obtener una base de datos de terminología como SNOMED-CT pero para acceder a ella se necesitarían otras herramientas. (29)

Luego de haber realizado un profundo análisis sobre las especificaciones y herramientas encontradas desarrolladas para la salud, como parte de los resultados de la investigación, se puede concluir que algunas no son generalizadas para adaptarlas a otros desarrolladores, no facilitan sus implementaciones y otras son específicas para las necesidades de una terminología en particular por lo que no brindan una solución al problema a resolver.

En Cuba no hay experiencias registradas de uso o implementaciones de este tipo de servicios, actualmente se trabaja en el desarrollo de la plataforma de aplicaciones SISalud, la cual permite incorporar nuevos módulos compatibles entre sí, aunque esta presenta problemas y limitaciones para lograr la integración e interoperabilidad entre los sistemas de información de salud. Sin embargo en el Centro de Informática Médica de la Universidad de las Ciencias Informáticas se desarrollan sistemas especializados que utilizan estándares y codificadores internacionales para el manejo de terminologías médicas como es el CIE 9 y el CIE 10 para la Clasificación Internacional de Enfermedades y el CIAP para la Clasificación de Enfermedades en la Atención Primaria, estos sistemas carecen de una estructura de datos común entre las terminologías médicas que utilizan, lo que se convierte en un gran problema para lograr integrar las soluciones que se desarrollan en el Centro.

1.7 Estilos arquitectónicos estudiados.

Para realizar el diseño de la capa se hace necesario estudiar y analizar algunos patrones de diseño con el objetivo de lograr una mejor estructuración de los elementos del diseño e implementación correspondientes Los patrones GRASP los cuales tienen como objetivo la descripción de los principios fundamentales de diseño de objetos para la asignación de responsabilidades y dentro de estos, los

patrones Experto, Creador, Alta cohesión y Bajo acoplamiento. Mediante la asignación a cada clase de las tareas o responsabilidades que estas pueden realizar en dependencia de la información que contienen se evidencia el uso de los patrones Experto y Creador. Estos conservan el encapsulamiento y definen quién será el responsable de crear una instancia de una clase respectivamente. Al utilizar los patrones Alta cohesión y Bajo acoplamiento se permite la colaboración entre clases o elementos del diseño sin que se afecte su reutilización y entendimiento cuando se encuentren aislados.

No se debe dejar de mencionar SOA (Arquitectura Orientada a Servicios), establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios. La forma más habitual de implementarla es mediante Servicios Web, una tecnología basada en estándares e independiente de la plataforma, con la que SOA puede descomponer aplicaciones monolíticas en un conjunto de servicios e implementar esta funcionalidad en forma modular. (30)

La Arquitectura Orientada a Servicios (SOA) es una filosofía de diseño que permite un mejor alineamiento de las Tecnologías de Información (IT) con las necesidades de negocio, permitiendo a empleados, clientes y socios comerciales responder de forma más rápida y adaptarse adecuadamente a las presiones del mercado. (31)

Entre los beneficios de SOA se destacan:

- La reducción de costos y tiempo en el desarrollo de aplicaciones ya que SOA permite reutilizar los módulos de aplicaciones existentes y el código nuevo para generar nuevas aplicaciones. Como consecuencia también se reducen los costos mantenimiento.
- Las metodologías que aterrizan el concepto de SOA facilitan la integración entre aplicaciones nuevas así como con los sistemas existentes.
- Desarrollo de aplicaciones más productivas, flexibles, más seguras y manejables para gestionar procesos de negocio críticos a medida que evolucionan o cambian las necesidades del negocio.
- Fortalecimiento y consolidación de los procesos de negocio a través de aplicaciones que comparten servicios comunes.

1.8 Tendencias y tecnologías actuales a considerar.

1.8.1 Lenguaje de Marcado de Hipertexto Extensible (XHTML)

XHTML es una versión más estricta y limpia del Lenguaje de Marcado de Hipertexto (HTML), que nace precisamente con el objetivo de reemplazarlo ante su limitación de uso con las cada vez más abundantes herramientas basadas en el Lenguaje de Marcado Extensible (XML), un formato de datos universal altamente extensible, que define una manera estándar de estructurar el marcado de documentos.

XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos. Es un lenguaje cuyo etiquetado, más estricto que HTML, va a permitir una correcta interpretación de la información independientemente del dispositivo desde el que se accede a ella.

1.8.2 JBoss Seam

JBoss Seam 2.0 (Seam) es un framework para el desarrollo de aplicaciones web en Java, que define un modelo de componentes uniforme para toda la lógica de negocio de las aplicaciones que sean desarrolladas mediante su utilización. Revitaliza el estándar Java EE poniéndole fin a sus divergencias y unificando sus componentes, haciéndolos más accesibles y funcionales.

Integra fácilmente tecnologías estándares como Java Server Faces (JSF), modelo de componentes para la capa de presentación; Enterprise JavaBeans (EJB3), modelo de componentes para la lógica de negocio y persistencia del lado del servidor; Java Persistence API (JPA), y de Business Process Management (BPM). Integra además bibliotecas de controles de código abierto basadas en JSF como RichFaces e ICEFaces. (32)

Realiza nuevos aportes en el campo de la administración de estado. Mientras que en los frameworks tradicionales todo el estado es administrado básicamente en la sesión HTTP, Seam provee una mayor granularidad de contextos de estado y aporta un nuevo concepto, la administración de espacios de trabajo. Esta permite al usuario tener en varias pestañas o ventanas del navegador actividades del negocio con contextos completamente aislados. (33)

1.8.3 Java Persistence API (JPA)

Es la Interfaz de Programación de Aplicaciones (API) estándar para la persistencia y el mapeo objeto/relacional para la plataforma Java EE, y permite utilizar un modelo de dominio Java para administrar bases de datos relacionales. (34)

1.8.4 Enterprise JavaBeans (EJB3)

Arquitectura componente del lado del servidor para la plataforma Java. Permite realizar la administración automática de transacciones, seguridad, escalabilidad, concurrencia, distribución, acceso a ambientes portables y persistencia de datos. Incorpora el estándar JPA como el principal API de persistencia para aplicaciones EJB3.

1.8.5 Hibernate

Framework que provee herramientas de mapeo objeto/relacional y permite reducir significantemente el tiempo de desarrollo. Con su API nativa es el servicio base para la persistencia de datos. Posee un lenguaje de consultas llamado HQL (bastante parecido al lenguaje de consultas SQL). Sus herramientas soportan distintos tipos de base de datos lo que confiere cierto nivel de portabilidad a las aplicaciones que lo utilizan. A través de la implementación del estándar JPA que provee Hibernate 3.3, se puede realizar el acceso a datos. (35)

1.9 Tecnologías y metodologías horizontales.

1.9.1 PostgreSQL Versión 8.4

Es un Sistema Gestor de Base de Datos (SGBD, por sus siglas en inglés) relacional de código abierto, muy poderoso, con una arquitectura probada. Puede ser ejecutado sobre la mayoría de los sistemas operativos que existen hoy en día. Posee protección de transacciones u operaciones de Atomicidad, Consistencia, Aislamiento, Durabilidad (ACID).

Es un gestor de base de datos empresarial, que posee características sofisticadas como Control de Concurrencia Multi - Versión (MVCC, por sus siglas en inglés), replicación asíncrona, transacciones anidadas, realización de respaldo de datos en línea, optimizador o planificador de consultas, soporta internacionalización. Es altamente escalable en cuanto a la cantidad de información que puede manejar y al número de usuarios concurrentes que puede alojar. (36)

El tamaño máximo para las bases de datos PostgreSQL es ilimitado. Es totalmente compatible con el estándar ANSI-SQL 92/99. Permite realizar relaciones de herencia entre tablas, definir sistemas de reglas y eventos. PostgreSQL ejecuta procedimientos de almacenados en más de una docena de lenguajes de programación tales como Java, Perl, Python, Ruby, Tcl, C/C++, y su propio lenguaje PL/pgSQL.

De esto se deriva que existan muchas bibliotecas para realizar la conexión a PostgreSQL desde varios lenguajes, compilados o interpretados, entre ellos Java, Perl, Python, Ruby, C, C++, PHP, Lisp, Scheme, para nombrar algunos. Desde el año 1999 hasta la fecha ha ganado un gran número de premios que ratifican a este SGBD como el más completo dentro del mundo del código abierto. (37)

1.9.2 Java

Lenguaje de programación que fue desarrollado por la antigua compañía Sun Microsystems, que utiliza el paradigma de la Programación Orientada a Objetos (POO). Es un lenguaje que no permite el manejo directo del hardware ni de la memoria. Dentro de sus principales ventajas se encuentra la de ser multiplataforma.

En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Con Java se pueden programar aplicaciones web dinámicas, con acceso a bases de datos, utilizando XML, con cualquier tipo de conexión de red entre cualquier sistema. Este lenguaje es utilizado de manera horizontal en el desarrollo del sistema, pues puede estar presente en las diferentes capas de la aplicación. (38)

1.9.3 JBoss Server

Servidor de aplicaciones Java más utilizado actualmente en el mercado. Cientos de profesionales y desarrolladores de código abierto han contribuido a su creación y desarrollo. Es una plataforma certificada que cumple con las especificaciones de J2EE para el despliegue y desarrollo de aplicaciones empresariales Java, aplicaciones web, y portales. Provee servicios extendidos almacenamiento de datos en memoria y de manera persistente. Permite la integración de todas las tecnologías y herramientas utilizadas por JBoss Seam. Es actualizado e integrado constantemente con lo último del estado del arte de las aplicaciones web. (39) (40)

1.9.4 Lenguaje Unificado de Modelado

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un plano del sistema, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. UML, es un lenguaje no propietario de tercera generación para modelado y definición. Proporciona una forma sencilla, mediante diagramas e iconografías, de definir y trasmitir ideas complejas. (41)

1.9.5 Proceso Unificado de Desarrollo (RUP)

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes.

Es un proceso que en su modelación define como sus principales elementos: los trabajadores, las actividades, los artefactos y el flujo de actividades. Entre algunas de sus características se pueden citar las siguientes:

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.
- Unifica los mejores elementos de metodologías anteriores.
- Preparado para desarrollar grandes y complejos proyectos.
- Orientado a Objetos.
- Utiliza el UML como lenguaje de representación visual.
- Guiado por Casos de Uso.
- Iterativo e Incremental.
- Centrado en la Arquitectura.

RUP divide el proceso de desarrollo en ciclos, teniendo un producto al culminar de cada ciclo, estos se dividen en fases que finalizan con un hito donde se debe tomar una decisión importante: (42)

- **Inicio:** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema**.**

- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.
- **Construcción:** Se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.
- **Transición:** Se implementa el producto en el cliente y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

1.10 Herramientas.

Analizadas las metodologías y tecnologías a utilizar se pueden definir las herramientas que conforman el ambiente de desarrollo de la Capa de Mensajería.

1.10.1 Enterprise Architect Versión 7.0

Es una herramienta de uso muy sencillo, que aborda el diseño y análisis UML y cubre el desarrollo de software desde la captura de requerimientos a lo largo de las etapas de análisis, diseño, pruebas y mantenimiento. Enterprise Architect es una herramienta multi-usuario, diseñada para ayudar a construir software robusto y fácil de mantener. Además, permite generar documentación e informes flexibles y de alta calidad. (43)

Las bases de Enterprise Architect están sustentadas en la especificación de UML 2.1.Usa Perfiles UML para extender el dominio de modelado, mientras que la validación del modelo asegura la integridad del proyecto. Combina los procesos de negocio, información y flujos de trabajo en un modelo usando las extensiones gratuitas para BPMN (Notación para el Modelado de Procesos de Negocio). Proporciona trazabilidad completa desde el análisis de requerimientos y los artefactos de diseño, hasta la implementación y el despliegue. En combinación con la asignación de recursos y tareas que incorpora, los equipos de Gestión de Proyectos y Calidad están dotados con toda la información necesaria para ayudarles a controlar los proyectos y sus entregas. (44)

1.10.2 PGAdmin III

Para administrar la base de datos se escoge el pgAdmin III, debido a que es un ambiente gratuito, relacional, muy poderoso y puede ser ejecutado sobre la mayoría de los sistemas operativos que existen hoy en día.

1.10.3 Eclipse Versión Galileo

Para la implementación del sistema se escoge el Eclipse, un Entorno Integrado de Desarrollo (IDE, por sus siglas en inglés) que cuenta con la posibilidad de añadir nuevas funcionalidades al editor, a través de nuevos módulos, lo que es beneficioso para la utilización de JBoss Tools, conjunto de herramientas que posibilitan el desarrollo desde el IDE Eclipse con RichFaces, Seam e Hibernate, además de realizar la administración y configuración del servidor JBoss AS.

En este capítulo se valoraron los principales conceptos relacionados con el campo de acción de la investigación, lo que permitió una mejor comprensión de los mismos. Con el estudio realizado se pudo concluir que las especificaciones y herramientas analizadas en el estudio del estado del arte no brindan una solución al problema a resolver. Se escogieron las tecnologías, metodologías y herramientas adecuadas para el desarrollo de la capa evaluándose cada una de ellos por sus características y ventajas que aportan facilidad para el desarrollo de la misma.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

En el presente capítulo se exponen elementos importantes de la capa a construir. Se definen los conceptos y sus relaciones, agrupados en el Modelo de Dominio debido a la poca definición de los procesos de negocio. Se presentan los requisitos funcionales y no funcionales, la propuesta de solución del sistema y se diseña la arquitectura del mismo.

2.1 Propuesta del Sistema.

Un Servicio de Terminologías Comunes (CTS), es una pieza clave para procesar información médica, ya que permite asignar un código único a los textos ingresados en forma de lenguaje natural. Con lo cual, además de poder mantener el texto de la misma manera en que fue ingresado, el código permitirá realizar todo tipo de análisis automático posterior. Facilitando así la integración e interoperabilidad entre diferentes sistemas de información médica.

CTS cuenta con tres capas, mensajería, vocabulario y mapeo, además de un sistema de administración que es el encargado de permitir a los usuarios realizar diferentes operaciones sobre la fuente de información con que cuenta el Servicio de Terminologías Comunes, permitiendo que pueda ser utilizado en un largo período de tiempo ya que brinda la facilidad de realizarle actualizaciones según estas vayan surgiendo.

Por su parte la Capa de Mapeo tiene como objetivo facilitar la traducción de mensajes entre aplicaciones de información clínica, donde traduce un sistema de código que contiene de manera general la relación entre los conceptos, sus descripciones y el código que los representa. Cumpliendo con las funcionalidades especificadas por la capa Vocabulario. Esta permite relacionar terminologías médicas no homólogas con el estándar de comunicación HL7.

La Capa de Vocabulario se encarga de permitir a las aplicaciones realizar consultas a diferentes terminologías de manera coherente y bien definida. Está diseñada para ser genérica y fácilmente utilizable en diferentes entornos.

Por último la Capa de Mensajería tiene como objetivo permitir a una variedad de aplicaciones de procesado de mensajes, crear, validar y traducir datos clínicos de manera consistente. Esta permite a las aplicaciones de procesado de mensajes abstraerse de la complejidad de las funciones específicas del

vocabulario. Este nivel sirve de puente entre estas aplicaciones y la capa de funciones específicas de la terminología.

Por lo que se propone implementar un conjunto de funcionalidades de la Capa de Mensajería para integrarla a la Capa de Vocabulario del Servicio de Terminologías Comunes y publicarlas como servicios web para que sean consumidas por sistemas externos, permitiendo la comunicación de estos con la terminología médica.

Una vez terminada la implementación de todas las capas de CTS se realizará la integración de las mismas para que pueda ser utilizado el servicio, quedarán integradas de la siguiente forma: un sistema externo realiza solicitudes a las funcionalidades de la Capa de Mensajería, esta solicita a la Capa de Vocabulario los datos necesarios para responder la petición realizada por el usuario, vocabulario hace llamadas a la Capa de Mapeo para satisfacer la petición, luego de obtener toda la información solicitada, la Capa de Mensajería envía un mensaje al sistema externo con la respuesta a su solicitud.

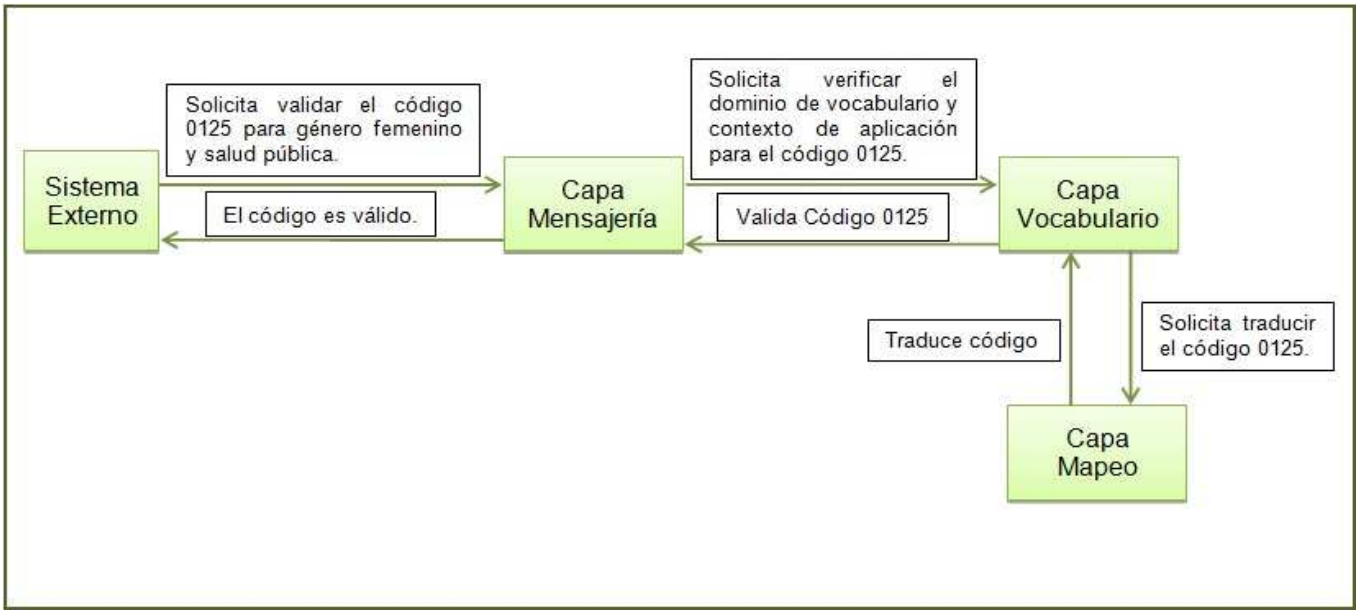

Figura 2.1 *Integración de las capas de CTS.*

2.2 Modelo de Dominio.

Un modelo de dominio es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, en la tarea de construcción del modelo de dominio. Es una representación

visual estática del entorno real objeto del proyecto. Este modelo es utilizado por el analista como medio para comprender el sistema que se va a realizar. (45)

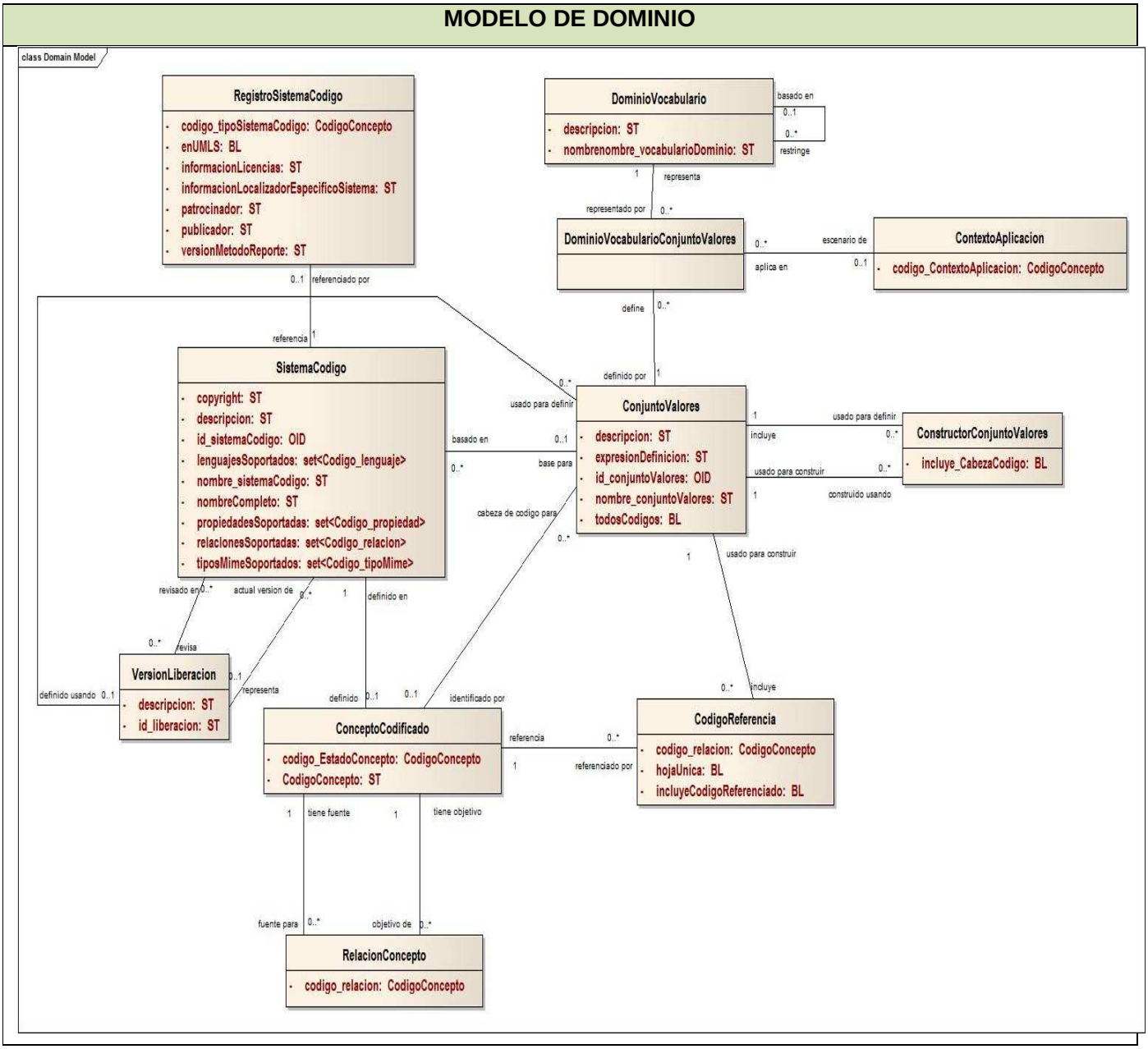

Figura 2.2 *Modelo de Dominio.*

2.2.1 Conceptos Fundamentales.

DominioVocabulario: sirve de enlace entre un atributo HL7 codificado del conjunto de códigos válidos para ese atributo. Representa un espacio conceptual abstracto tal como "países del mundo", "el género de una persona usada con fines administrativos". Describe un espacio conceptual del cual los valores de un atributo pueden ser elaborados antes que un atributo pueda ser usado en un mensaje, la lista real de código de conceptos debe ser definida. Puede estar representado por cero o más **ConjuntoValores**. Mientras, es posible que atributos abstractos RIM (Reference Information Model) y DIM (Domain Information Model) no estén representados por ningún **ConjuntoValores**. Los **Dominios de Vocabulario** que describen los posibles valores de atributos codificados utilizados en mensajes reales, deben ser representados por al menos un **ConjuntoValores**.

DominioVocabularioConjuntoValores: representa una asociación entre exactamente un **DominioVocabulario** y un **ConjuntoValores.**

ContextoAplicación: nombra una específica entidad geopolítica, por ejemplo: Cuba, Venezuela, Canadá y prácticas como: medicina veterinaria y salud pública.

ConjuntoValores: puede incluir una lista de cero o más **ConceptoCodificado** derivado de un único **SistemaCódigo.** Un **ConjuntoValores** puede ser representado por:

- Todos los **ConceptoCodificado** definidos en exactamente un **SistemaCódigo.**
- Una lista específica de **ConceptoCodificado** que están definidas en exactamente un **SistemaCódigo.**
- El conjunto de **ConceptoCodificado** representado por otro **ConjuntoValores.**

SistemaCódigo: puede ir desde una simple tabla de géneros a sistemas de clasificación como CIE (Clasificación Internacional de Enfermedades) 9 o SNOMED CT. Representa las características comunes para todos los sistemas de códigos usados en el ambiente de HL7.

CódigoReferencia: relaciona a un **ConceptoCodificado** con un **ConjuntoValores.** Puede incluir o excluir el referenciado **ConceptoCodificado,** además de todos los códigos de destino o solo los nodos hoja.

CostructorConjuntoValores: tiene dos usos:

- Permitir a los **ConjuntoValores** ser incluidos por referencia, esto significa que los cambios en el conjunto contenido son automáticamente contenidos en el conjunto contenedor.
- Hacen posible incluir **ConceptoCodificado** desde más de un **SistemaCódigo** en un único **ConjuntoValores.**

RegistroSistemaCódigo: el proceso de registro asigna un Identificador de Sistema de Código (OID) para un **SistemaCódigo** si no existe ya.

2.3 Especificación de requerimientos de software.

La especificación de los requerimientos de software constituye un elemento de vital importancia para la elaboración de un software de calidad superior. El Glosario Estándar de Terminologías de Ingeniería de Software de la IEEE define los requerimientos de software como condiciones o capacidades que deben estar presentes en un sistema o componentes de este, para satisfacer un contrato, estándar, especificación u otro documento formal. Es necesario hacer énfasis en la precisión con que se debe realizar esta tarea por cumplir un papel primordial en el proceso de producción de software, pues se enfoca en un área fundamental: la definición de lo que se desea producir, mediante una descripción más clara del comportamiento del sistema, minimizando los problemas derivados de su desarrollo. (46).

2.3.1 Requerimientos funcionales.

Los requerimientos funcionales especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física, de manera que especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto. A partir de los procesos de negocio estudiados y las actividades a automatizar identificadas se pueden definir los siguientes requisitos funcionales. (47)

Requerimientos Funcionales	
RF1: Devolver el nombre del servicio (Get Service Name).	RF13: Determinar equivalencia (Are Equivalent).
RF2: Devolver la versión del servicio (Get Service Version).	RF14: Expandir conjunto de valores (Lookup Value Set Expansion).

RF3: Devolver la descripción del servicio (Get Service Description).	RF15: Ampliar conjunto de valores expandido (Expand Value Set Expansion Context).
RF4: Devolver la versión de liberación de HL7 (Get HL7 Release Version).	RF16: Devolver los atributos soportados (Get Supported Attributes).
RF5: Devolver la versión de CTS (Get CTS Version).	RF17: Devolver conjunto de valores (Get Supported Value Sets).
RF6: Devolver los algoritmos soportados (Get Supported Match Algorithms).	RF18: Devolver los Sistemas de códigos soportados (Get Supported Code Systems).
RF7: Devolver los dominios de vocabulario soportados (Get Supported Vocabulary Domains).	RF19: Buscar dominios de vocabulario (Lookup Vocabulary Domain).
RF8: Validar código (Validate Code).	RF20: Buscar conjunto de valores (Lookup Value Set).
RF9: Validar traducción (Validate Translation).	RF21: Buscar sistemas de código (Lookup Code System).
RF10: Traducir código (Translate Code).	RF22: Buscar conjunto de valores para un dominio (Lookup Value Set For Domain).
RF11: Completar detalles (Fill In Details).	RF23: Determinar si un código es válido (Is Code In Value Set).
RF12: Determinar correspondencia (Subsumes).	

Tabla 2.1 *Requerimientos Funcionales*

2.3.2 Requerimientos no funcionales.

Los requerimientos no funcionales son cualidades o propiedades que el producto debe tener, es decir, restricciones en el producto que está siendo desarrollado. Estos no describen lo que el software

hará, sino cómo lo hará. Son importantes debido a que permiten a los clientes valorar las características no funcionales del producto como: usabilidad, rendimiento y portabilidad, que junto a las funcionalidades esperadas del software ayudarán a marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. Es preciso tener en cuenta que la definición de los requerimientos no funcionales cobra un valor adicional porque los errores que puedan existir en ellos son difíciles y caros de resolver. (48)

Requerimientos de usabilidad.

El servicio debe cumplir con los requisitos de usabilidad establecidos por los estándares con los que se trabaja.

Requerimientos de fiabilidad.

- El servicio web debe estar disponible 24 horas todos los días para que las aplicaciones de salud puedan interactuar con él cuando requiera de sus funciones.
- Se debe garantizar que la información que se brinda sea íntegra.

Requerimientos de eficiencia.

- La eficiencia dependerá en gran medida del buen aprovechamiento de los recursos disponibles.
- Las consultas a la base de datos deben ejecutarse rápida y eficientemente.
- El flujo de mensajes de petición y retorno debe ser estable y eficaz.
- Garantizar que no se pierda la comunicación con la base de datos y la información que se brinde sea la solicitada.

Requerimientos de soporte.

- Al concluir la implementación de la capa se debe confeccionar un plan de mantenimiento para garantizar que no deje de funcionar y evitar que ocurran fallas.
- Se debe documentar bien todo el servicio para en caso de la ocurrencia de fallas pueda recuperarse y volver a la normalidad en un corto período.

Requerimientos de interfaz.

Interfaz de Software: el servicio debe correr sobre el sistema operativo Linux, como sistema gestor de base de datos debe contar con el PostgreSQL 8.4.

Interfaces de Comunicación: debe contar con una tarjeta de red para la conectividad de los servidores.

Interfaces Hardware: se debe disponer de una PC que funcione como servidor de aplicaciones, el cual hospedará la solución integrada y garantizará la disponibilidad de la información que será almacenada en otra PC que será el servidor de Base de Datos. Estas PCs servidores deben poseer los siguientes requerimientos de hardware:

- Tipo de procesador: Intel Pentium IV
- Velocidad del procesador: 2.00 GHz
- Memoria RAM: 1 GB o superior.
- Disco Duro: 40 GB para servidor de aplicación y el de base de datos.
- Se requiere tarjeta de red.

2.4 Modelo de casos de uso del sistema.

El Modelo de Casos de Uso del Sistema es un artefacto de Ingeniería de Software que describe, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. De esta forma permitiendo del establecimiento de un acuerdo entre clientes y desarrolladores sobre las condiciones y requerimientos que debe cumplir el sistema. Este modelo está formado por actores, casos de usos y las relaciones que se establecen entre estos. Es decir representa gráficamente a los procesos y su interacción con los actores y constituye una entrada de gran valor para las siguientes fases de construcción de un software. (49)

2.4.1 Definición de actores.

Un actor del sistema es una persona o la abstracción de un software que interactúa de alguna manera con el sistema: puede intercambiar información con él, ser un recipiente pasivo de información, representar el rol que juegan una o varias personas, un equipo o un sistema automatizado. A continuación se definen los actores del sistema en desarrollo: (50)

Actor	Justificación
Sistema Externo	Se encarga de realizar solicitudes al servicio web para obtener datos clínicos, información de los dominios de vocabularios, conjuntos de

	códigos, contexto de aplicación y atributos codificados, etc. Es el que está autorizado a acceder a la información que brinda el sistema.

Tabla 2.2 *Justificación de actores del sistema.*

2.4.2 Diagrama de Casos de Uso del Sistema.

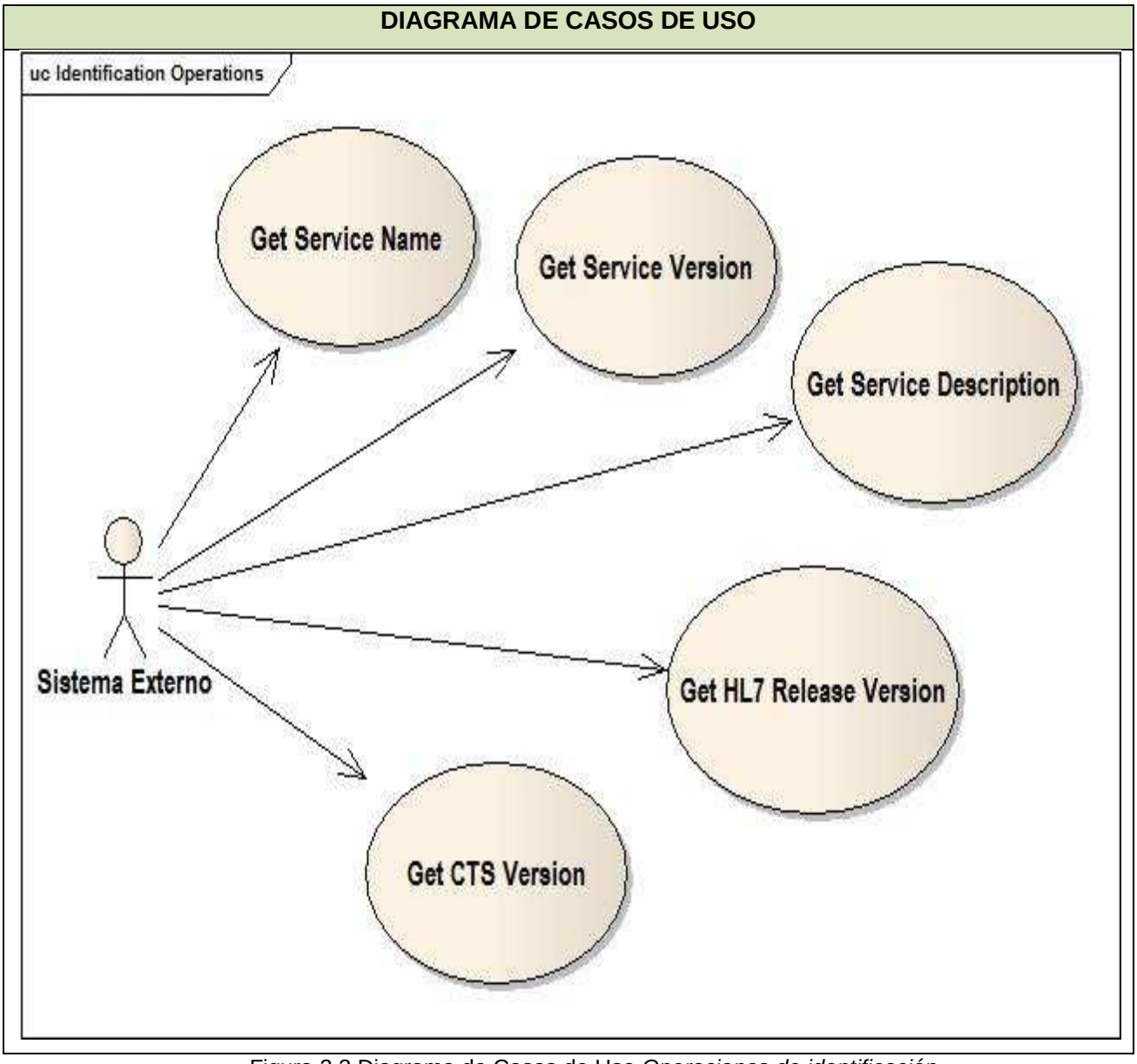

Figura 2.3 Diagrama de Casos de Uso *Operaciones de identificación.*

DIAGRAMA DE CASOS DE USO

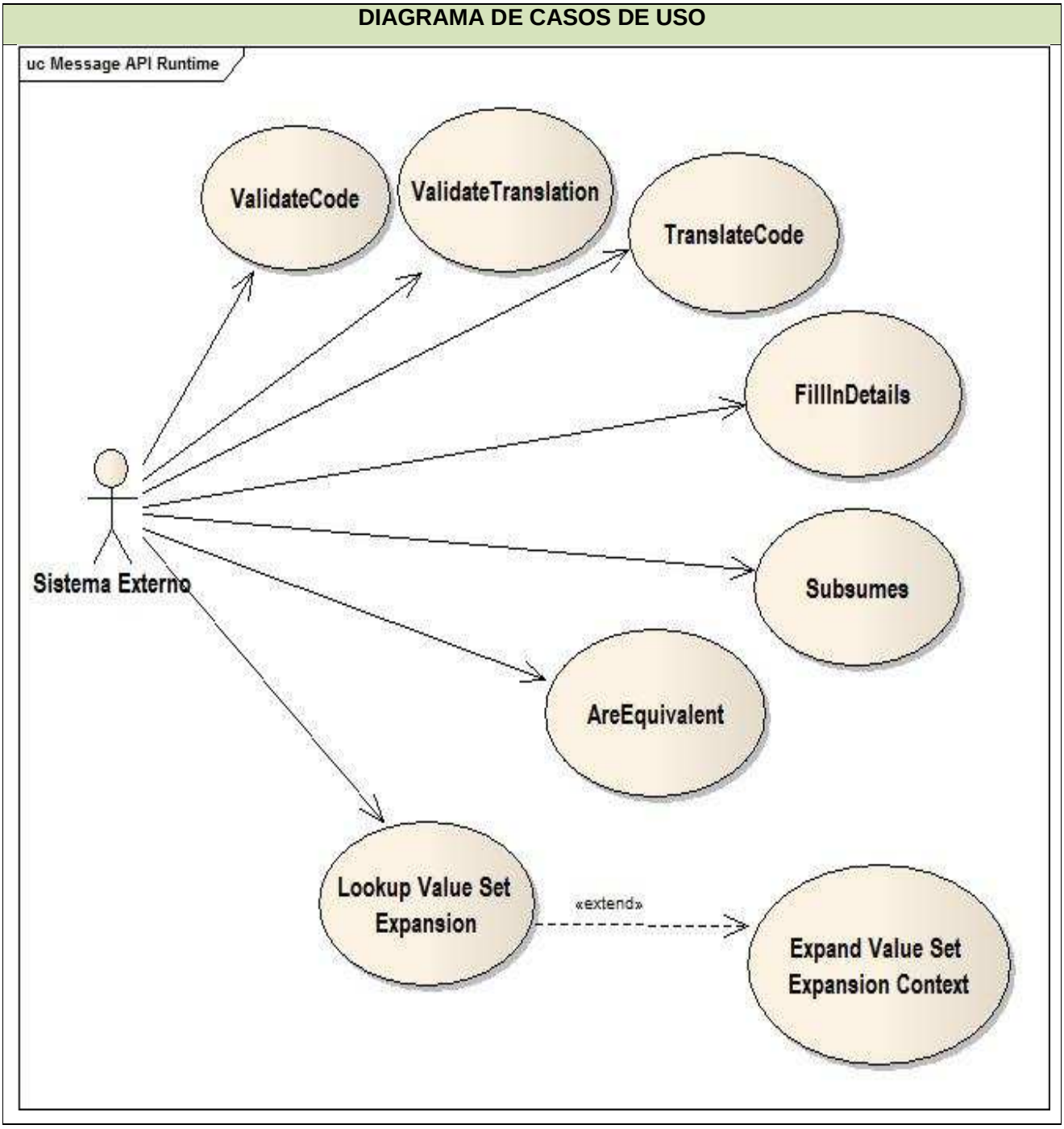

Figura 2.4 Diagrama de Casos de Uso *Message Runtime.*

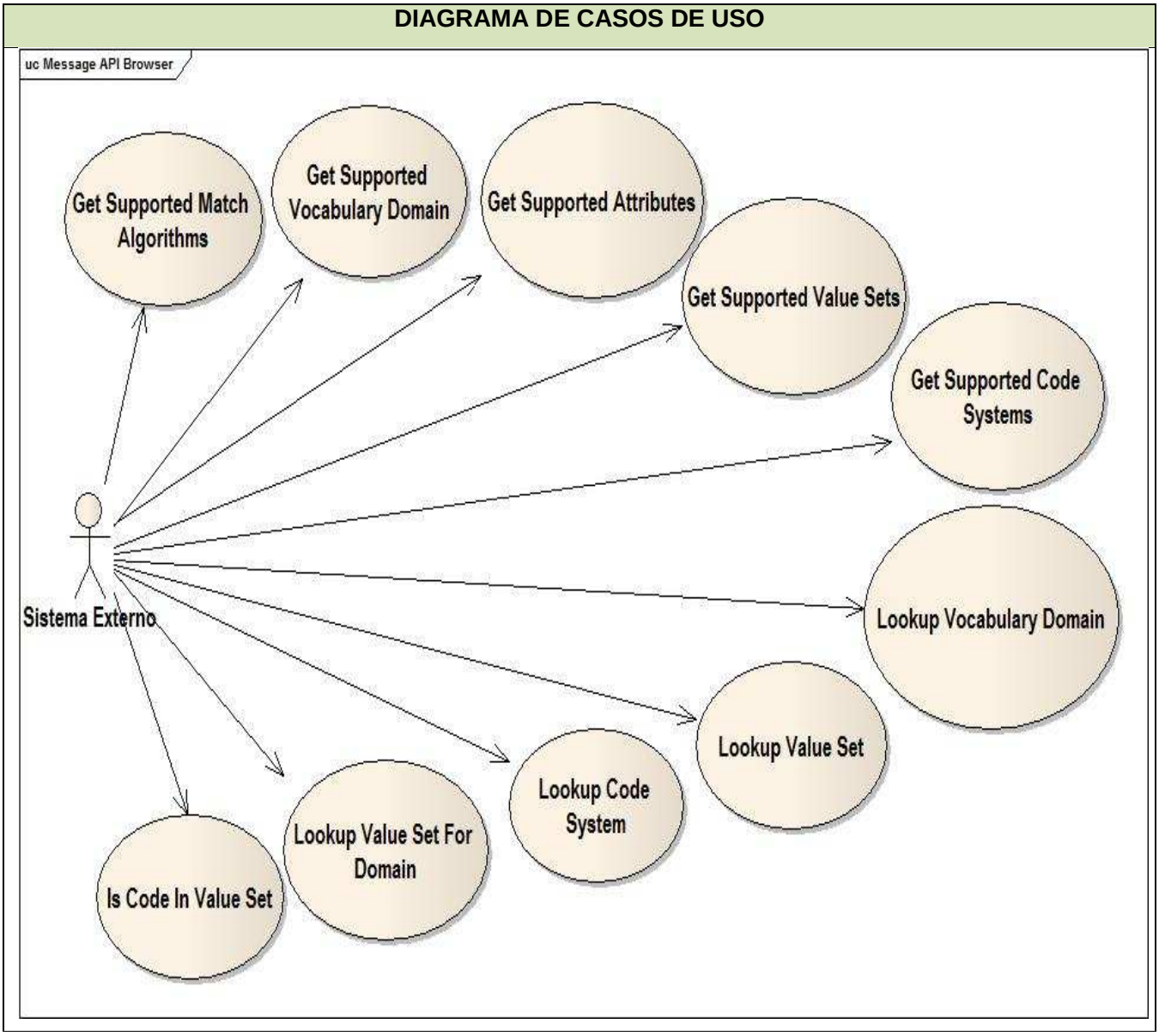

Figura 2.5 Diagrama de Casos de Uso Message Browser.

2.4.3 Descripción Textual de los Casos de Uso.

CU-1	Get Supported Match Algorithms.
Actor	Sistema Externo.
Descripción	El CU inicia cuando el usuario solicita los algoritmos implementados por el servicio. El sistema devuelve una lista de algoritmos conocidos

	soportados por el servicio.
Referencia	RF6

Tabla 2.3 Descripción textual *CUS_ Get Supported Match Algorithms.*

CU-2	Get Supported Vocabulary Domains.
Actor	Sistema Externo.
Descripción	El CU inicia cuando el usuario solicita los dominios de vocabulario que conoce el servicio. El sistema devuelve una lista de dominios de vocabulario que contienen el texto que es reconocido por el servicio.
Referencia	RF7

Tabla 2.4 Descripción textual *CUS_ Get Supported Vocabulary Domains.*

CU-4	Get Supported Value Sets
Actor	Sistema Externo.
Descripción	El CU inicia cuando el usuario solicita los conjuntos de valores soportados por el servicio. El sistema devuelve una lista de conjunto de valores cuyo nombre coincide con el texto suministrado conocido por el navegador.
Referencia	RF17

Tabla 2.5 Descripción textual *CUS_ Get Supported Value Sets.*

CU-5	Get Supported Code Systems
Actor	Sistema Externo.
Descripción	El CU inicia cuando el usuario solicita los sistemas de códigos soportados por el servicio. El sistema devuelve una lista de los sistemas de código cuyo nombre coincide con el texto suministrado conocido por el navegador.
Referencia	RF18

Tabla 2.6 Descripción textual *CUS_ Get Supported Code Systems.*

CU-7	Lookup Vocabulary Domain.
Actor	Sistema Externo.
Descripción	El CU inicia cuando el actor desea obtener información de un dominio de vocabulario. El actor introduce el dominio de vocabulario que quiere analizar. El sistema busca toda la información conocida sobre el dominio de vocabulario suministrado y devuelve una descripción detallada del mismo.
Referencia	RF19

Tabla 2.7 Descripción textual *CUS_Lookup Vocabulary Domain.*

CU-6	Lookup Value Set.
Actor	Sistema Externo.
Descripción	El CU inicia cuando el actor desea obtener información de un conjunto de valores. El actor introduce el conjunto de valores que quiere analizar. El sistema busca información conocida del conjunto de valores suministrado y devuelve una descripción detallada del mismo (incluyendo dominios de vocabularios, constructores, etc.).
Referencia	RF20

Tabla 2.8 Descripción textual *CUS_Lookup Value Set.*

2.5 Descripción de la arquitectura.

La arquitectura de software propone los elementos necesarios que sirven como guía para lograr el desarrollo exitoso del software. La misma abarca decisiones importantes sobre: la organización del sistema software; los elementos que compondrán el sistema y sus interfaces, junto con sus comportamientos, tal y como se especifican en las colaboraciones entre estos elementos; la composición de 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.

2.5.1 Fundamentación del uso de patrones.

Los patrones de diseño que se utilizan en el desarrollo de un sistema son los GRASP, que son los patrones generales para la asignación de responsabilidades. En la propuesta de la solución se destacan por su utilización:

- **Experto:** es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo.
- **Creador:** el patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase.
- **Alta cohesión:** Expresa que la información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible relacionada con la clase.
- **Bajo acoplamiento:** Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

La aplicación de estos patrones se pone de manifiesto en la capa con la asignación a cada clase las funcionalidades que pueden realizar según la información que las mismas contienen para no sobrecargarlas y que cada una realice las operaciones que les corresponde. Además se le da la responsabilidad de crear instancias de otras clases solamente a aquellas que contengan a las mismas. Se modelan también las clases controladoras, las cuales son las encargadas de realizar las operaciones de la capa y el diseño permite que las distintas clases interactúen entre sí, sin afectar esto la reutilización de las mismas o el correcto funcionamiento estas por separado.

2.6 Valoración Crítica.

A pesar de las ventajas que ofrece el diseño de la arquitectura de la capa, existen inconvenientes con su aplicación para el desarrollo de la solución propuesta. Uno de los principales problemas con los que cuenta el diseño actual de la arquitectura es el nivel al que se maneja la integración de las diferentes capas del Servicio de Terminologías Comunes.

Durante el proceso de desarrollo de software, son comunes los cambios en la base de datos del servicio que se implementa. Dichos cambios pueden ocurrir por varias causas, entre ellas por variación en los requerimientos funcionales del sistema o por decisión de los analistas e implementadores. Normalmente un cambio en la base de datos de un sistema en desarrollo produce alteraciones en el mismo, sin embargo éstas deben ser manejables de modo tal que el impacto en su implementación se reduzca al máximo.

La arquitectura propuesta para el desarrollo de la capa, tiene una alta dependencia a nivel de base de datos. El intercambio de información se realiza mediante consultas directas a los esquemas de la base tanto de la Capa de Mensajería en particular como a los de la Capa de Vocabulario. Esta relación tan estrecha a este nivel provoca que, cuando se produce un cambio en alguno de los esquemas que son utilizados por otras capas del servicio, los mismos se vean afectados directamente en sus funcionalidades. La capa de acceso a datos encapsula la lógica de almacenamiento, independizando al resto del sistema del mecanismo de persistencia. Esta capa es la encargada de persistir las entidades que se manejan en el negocio, el acceso a los datos almacenados, la actualización, etc.

En la arquitectura que se utilizará para el desarrollo de la solución propuesta, la capa de acceso a datos solo se limita a representar, a través de Hibernate las tablas de la base de datos mediante las clases entidades. Esto provoca que las consultas que se realizan para obtener los datos, ya sea al esquema perteneciente a la capa o a otro esquema del cual se necesite información, se realicen directamente desde las clases controladoras lo cual le agrega al programador la tarea de preocuparse por la forma en que están representados de los datos.

2.7 Modelo de Diseño.

El Modelo de Diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema. Además describe las clases más importantes, su organización en paquetes y subsistemas, sirviendo como abstracción de la implementación y utilizándose como entrada fundamental de las actividades de implementación. Es representado por un sistema de diseño que denota el subsistema de nivel más alto del modelo. La utilización de otro subsistema es una forma de organización de este artefacto, en porciones más manejables. Los casos de uso son realizados por las clases del diseño y sus objetos mediante colaboraciones en el Modelo de Diseño, denotando la realización de casos de uso del diseño. (51)

Para la conformación del Modelo de Diseño se hace necesario su organización en piezas más manejables: subsistemas o paquetes, que contengan sus interfaces y las dependencias entre cada una de ellas, conformando de esta manera la estructura principal del sistema.

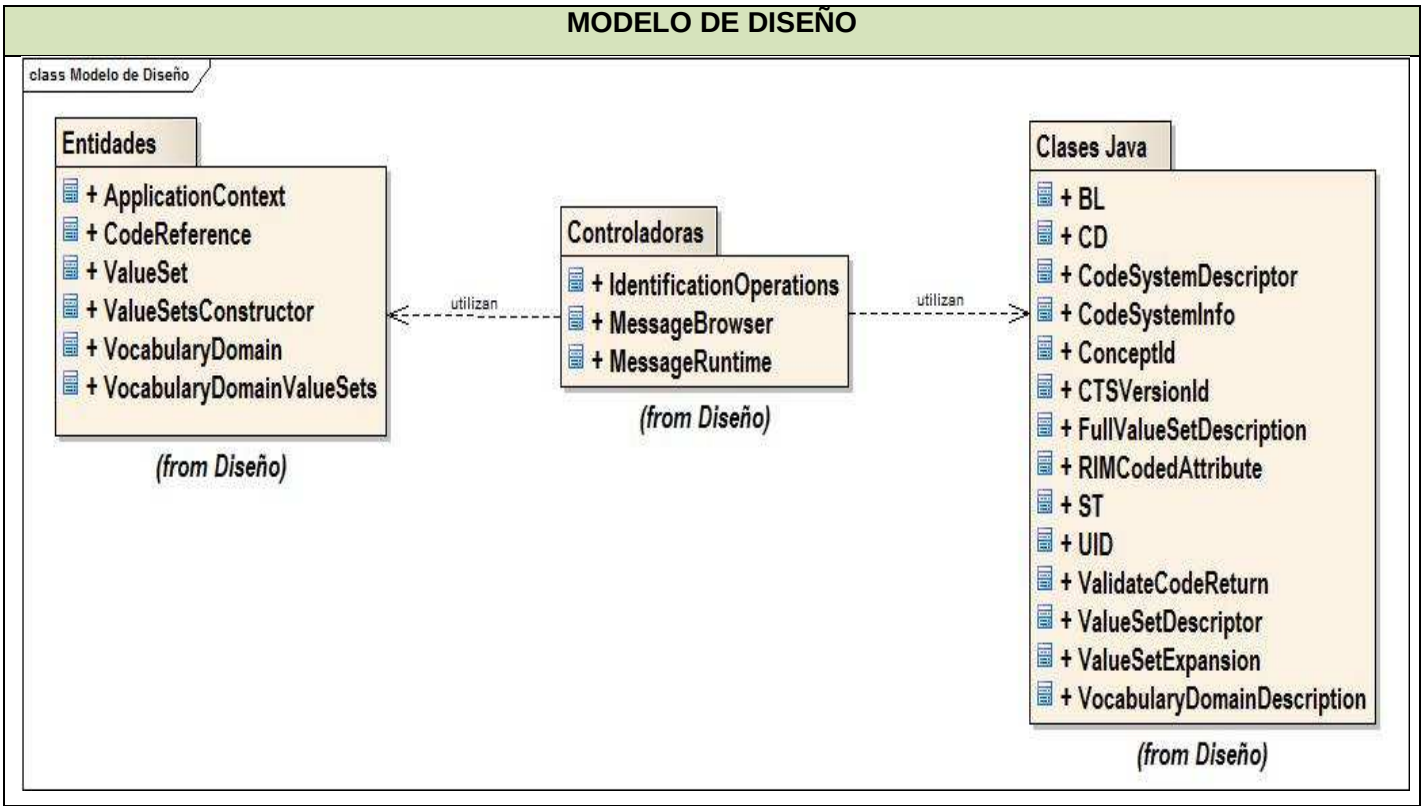

Figura 2.6 *Modelo de Diseño.*

Para posibilitar una mejor comprensión del Modelo Diseño estructurado en paquetes, se describe a continuación el contenido de cada uno de los paquetes contenidos en él.

Entidades: las entidades que posee la base de datos que son utilizadas por las funcionalidades de la capa.

Clases Java: las clases definidas en la especificación de CTS que implementan los tipos de datos que posee el servicio.

Controladoras: son las clases que poseen todas las funcionalidades de la capa agrupadas según la especificación CTS.

2.8 Diagramas de Clases del Diseño.

El diagrama de clases es el principal del diseño para un sistema, elaborado para satisfacer los detalles concretos de la implementación. En él se muestra la estructura estática del sistema, es donde se pueden ver las relaciones entre todas las clases, además de los atributos y operaciones de cada una de las mismas. Los diagramas de clases del diseño describen gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contienen la información de: las clases, asociaciones y atributos, interfaces, con sus operaciones y constantes, métodos, información sobre los tipos de los atributos, navegabilidad y dependencia. (52)

DIAGRAMA DE CLASES

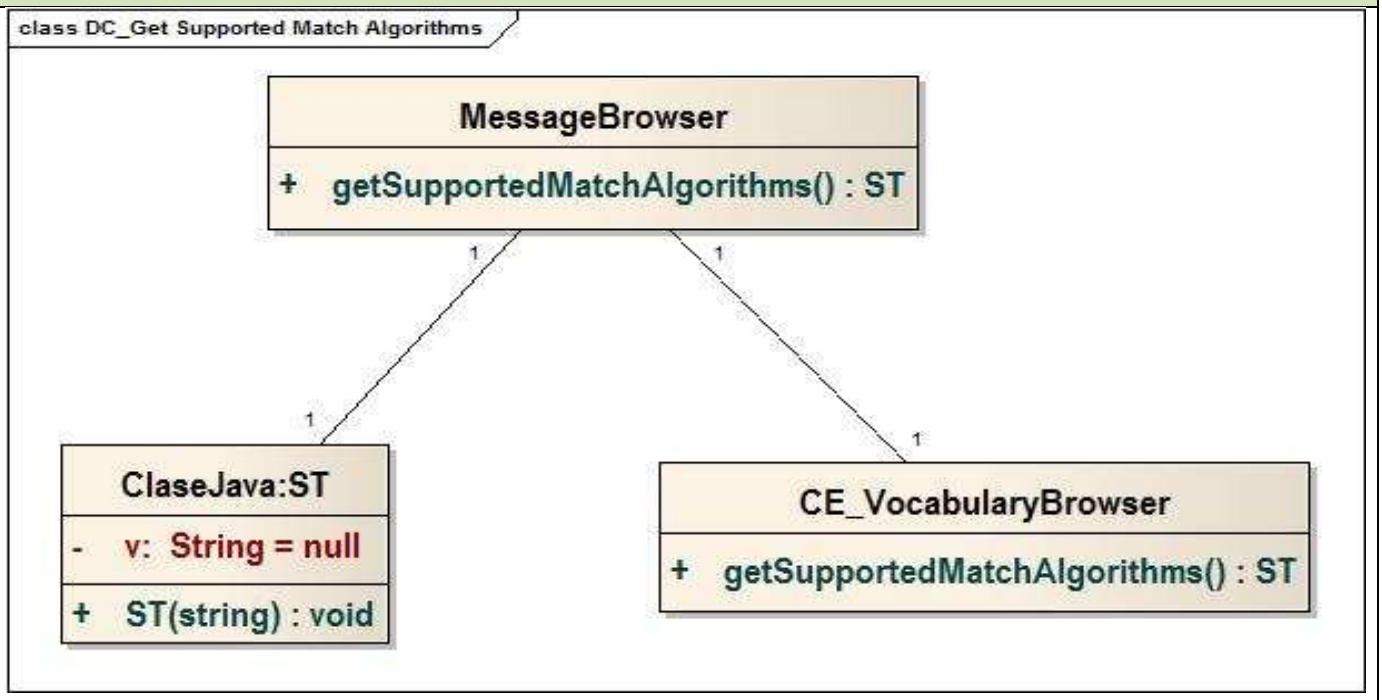

Figura 2.7 Diagrama de Clases del Diseño *CUS_ Get Supported Match Algorithms*.

DIAGRAMA DE CLASES

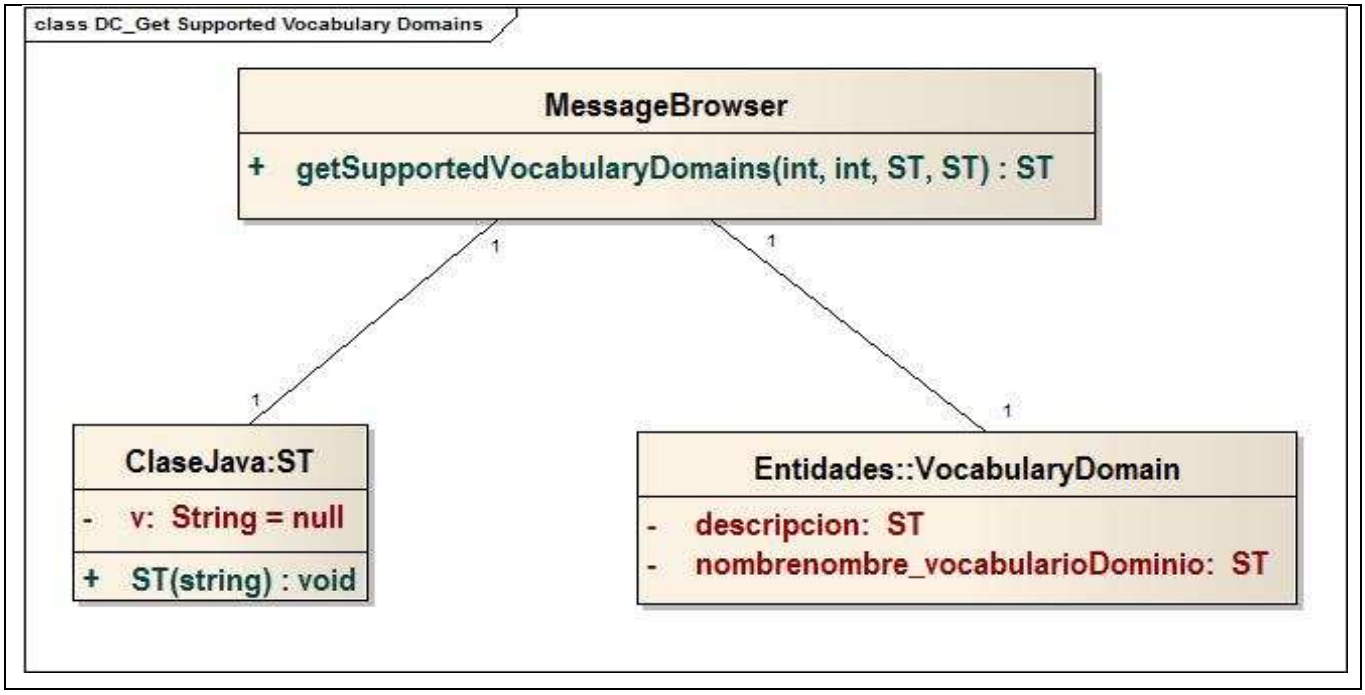

Figura 2.8 Diagrama de Clases del Diseño *CUS_ Get Supported Vocabulary Domains.*

DIAGRAMA DE CLASES

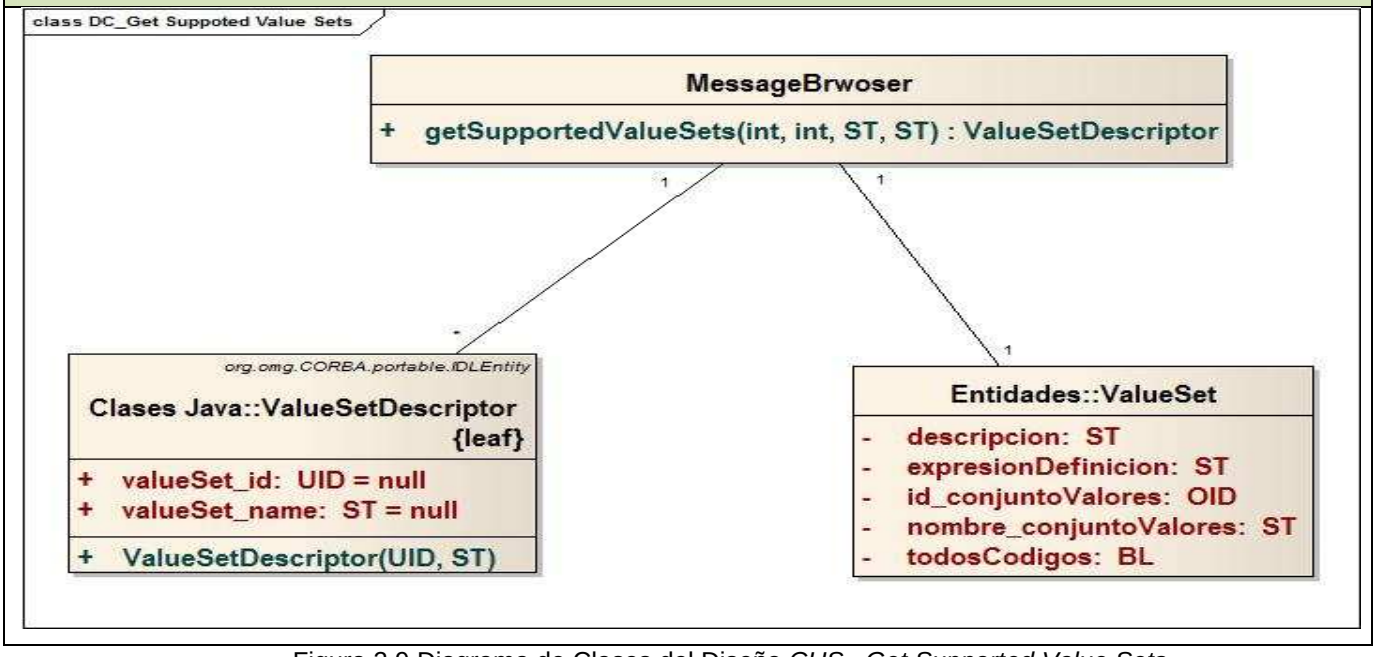

Figura 2.9 Diagrama de Clases del Diseño *CUS_ Get Supported Value Sets.*

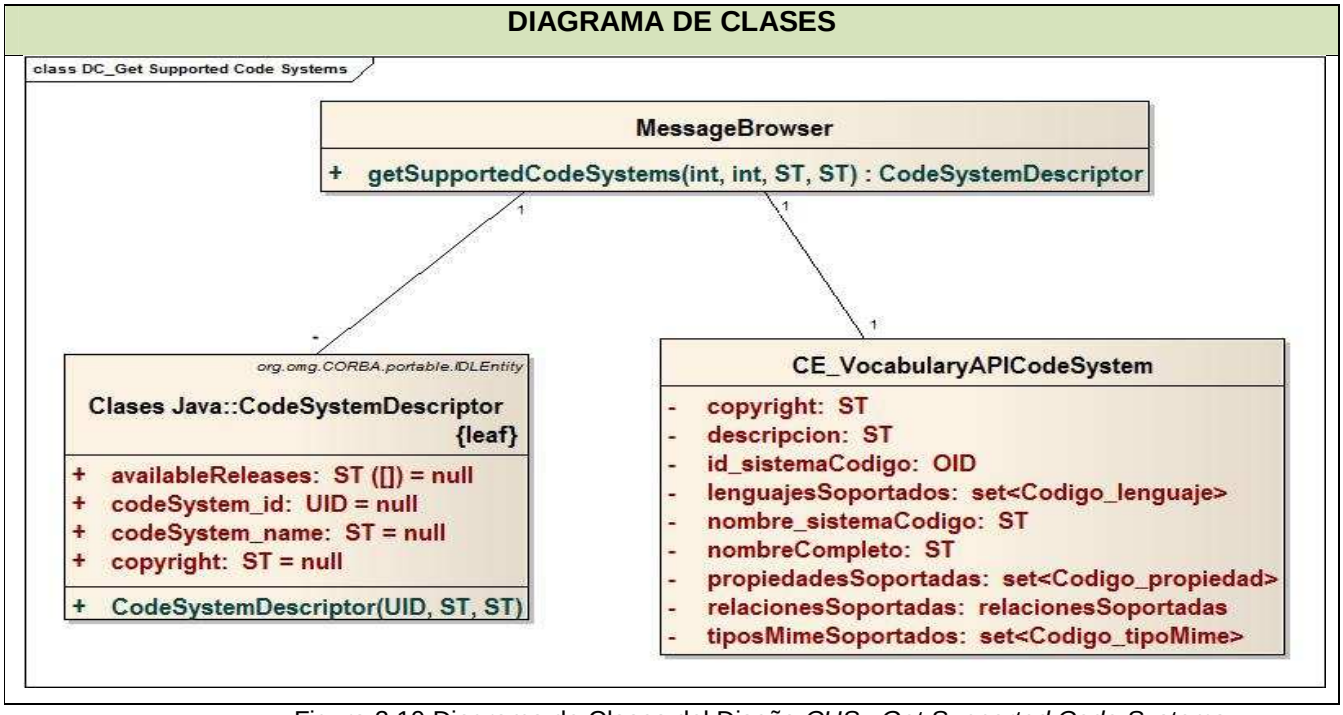

Figura 2.10 Diagrama de Clases del Diseño *CUS_ Get Supported Code Systems.*

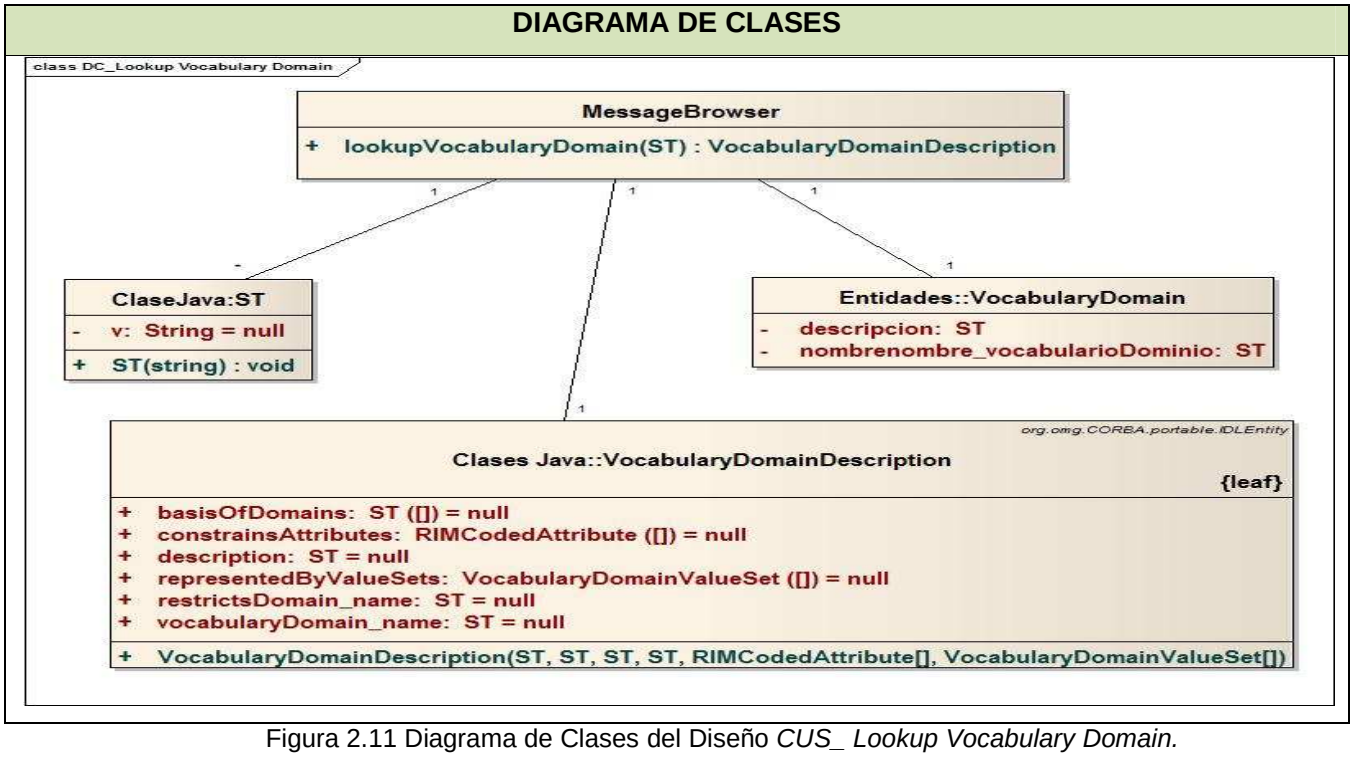

Figura 2.11 Diagrama de Clases del Diseño *CUS_ Lookup Vocabulary Domain.*

DIAGRAMA DE CLASES

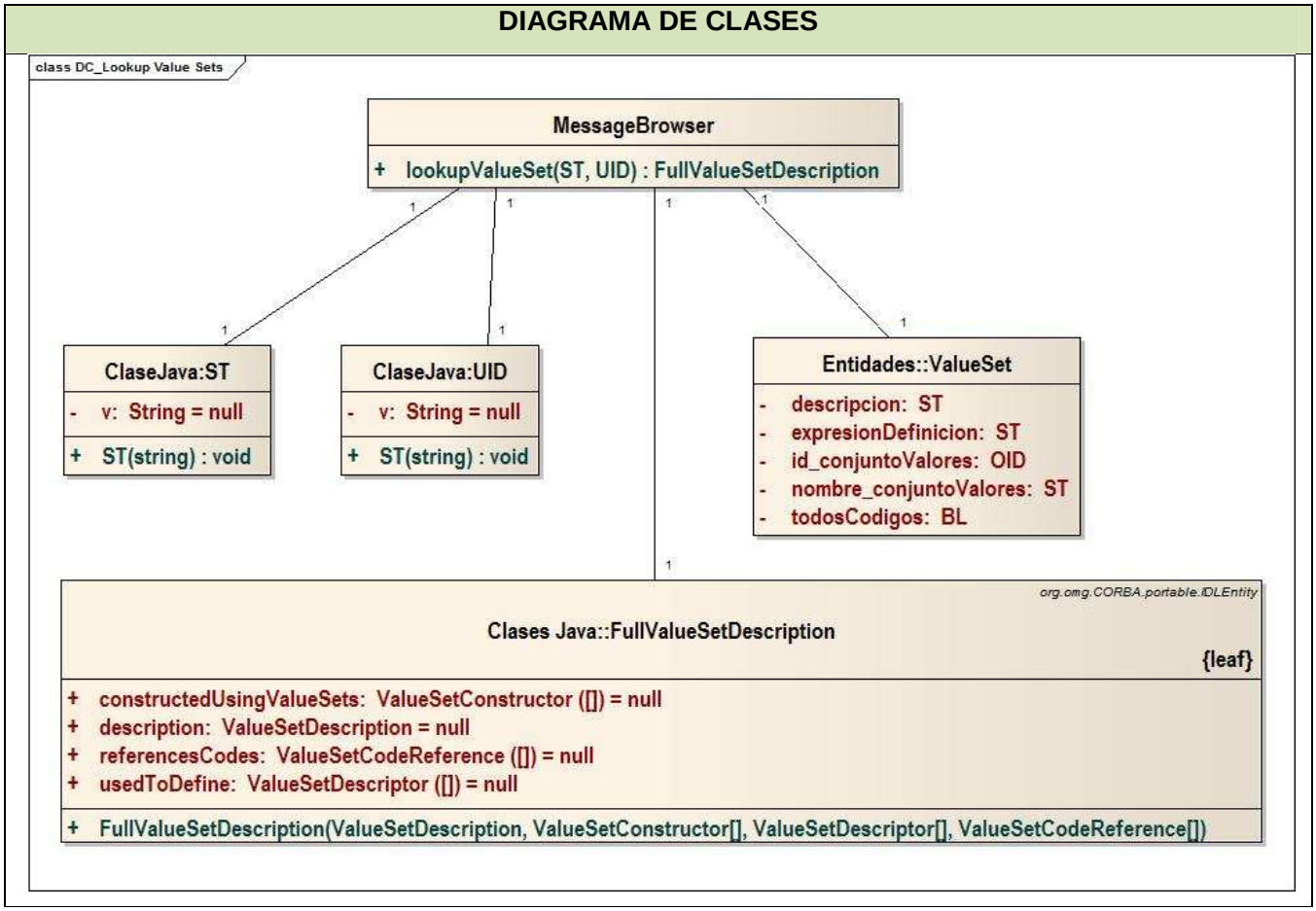

Figura 2.12 Diagrama de Clases del Diseño *CUS_Lookup Value Sets.*

2.9 Descripción de las clases y sus atributos.

Nombre: Message Runtime

Controladoras::MessageRuntime
- vbo: VocabBrowser = new VocabBrowser() - vro: VocabRuntime = new VocabRuntime()
+ areEquivalent(CD, CD) : BL + expandValueSetExpansionContext(byte[]) : ValueSetExpansion[] + fillInDetails(CD, ST) : CD + lookupValueSetExpansion(ST, ST, ST, BL, int, int) : ValueSetExpansion[] + subsumes(CD, CD) : BL + translateCode(ST, CD, UID, ST) : CD + validateCode(ST, CD, ST, BL, BL) : ValidateCodeReturn + validateTranslation(ST, CD, ST, BL, BL) : ValidateCodeReturn

Tipo de clase: Controladora.	
Para cada responsabilidad:	
Nombre:	Are Equivalent(CD code 1,CD code 2):BL.
Descripción:	Determina si dos atributos codificados son equivalentes.
Nombre:	Expand Value Set Expansion Context (byte []): ValueSetExpansion [].
Descripción:	Devuelve una mayor expansión de contextos expandidos para el dominio de vocabulario.
Nombre:	Fill In Details (CD codeToFillIn, ST displayLanguage_code): CD.
Descripción:	Completa las partes opcionales del atributo codificado como el nombre para mostrar concepto, el nombre de código del sistema y la versión del sistema de código.
Nombre:	Get Supported Match Algorithms (): ST [].
Descripción:	Devuelve una lista de algoritmos conocidos implementados por este servicio.
Nombre:	Get Supported Vocabulary Domains (ST matchText, ST matchAlgorithm_code, int timeout, int sizeLimit): ST [].
Descripción:	Devuelve una lista de dominios de vocabulario que contiene el texto que es reconocido por este servicio.
Nombre:	Lookup Value Set Expansion (ST vocabularyDomain_name, ST applicationContext_code, ST language_code, BL expandAll): ValueSetExpansion [].
Descripción:	Devuelve una lista jerárquica de los conceptos seleccionados para el dominio de vocabulario y contexto suministrados.
Nombre:	Subsumes (CD parentcode, CD childcode): BL.
Descripción:	Determina si el atributo codificado padre implica el hijo.
Nombre:	Translate Code (ST vocabularyDomain_name, CD fromcode, UID to_codesystemid, ST toapplicationContext_code): CD.
Descripción:	Traduce los atributos codificados suministrados en un formato que utiliza el sistema de códigos de destino o usa cualquier sistema de código que es apropiado para el contexto suministrado.
Nombre:	Validate Code (ST vocabularyDomain_name, CD codetovalidate, ST

	applicationContext_code, ST toapplicationContext_code): ValidateCodeReturn [].
Descripción:	Valida el atributo codificado para el dominio de vocabulario y contexto suministrado
Nombre:	Validate Translation (ST vocabularyDomain_name, CD codetovalidate, ST applicationContext_code, ST toapplicationContext_code): ValidateCodeReturn [].
Descripción:	Valida las traducciones para el dominio de vocabulario y contexto suministrado.

Tabla 2.9 *Descripción de la clase controladora Message Runtime.*

Nombre: Message Browser	
	
Tipo de clase: Controladora.	
Para cada responsabilidad:	
Nombre:	Get Supported Attributes (ST matchText, ST matchAlgorithm_code, int timeout, int sizeLimit): RIMCodeAttribute [].
Descripción:	Devuelve una lista de atributos RIM cuyo nombre coincide con el texto suministrado que es conocido por el navegador.
Nombre:	Get Supported Code Systems (ST matchText, ST matchAlgorithm_code, int timeout, int sizeLimit): CodeSystemDescriptor [].
Descripción:	Devuelve una lista de los sistemas de código cuyo nombre coincide con el texto suministrado conocido por el navegador.

Nombre:	Get Supported Match Algorithms(): ST.
Descripción:	Devuelve una lista de algoritmos implementados por el servicio conocidos por el navegador.
Nombre:	Get Supported Value Set (ST matchText, ST matchAlgorithm_code, int timeout, int sizeLimit): ValueSetDescriptor [].
Descripción:	Devuelve una lista de conjunto de valores cuyo nombre coincide con el texto suministrado conocido por el navegador.
Nombre:	Get Supported Vocabulary Domains (ST matchText, ST matchAlgorithm_code, int timeout, int sizeLimit): ST [].
Descripción:	Devuelve una lista de dominios de vocabulario cuyo nombre coincide con el texto suministrado conocido el navegador.
Nombre:	Is Code In Value Set (UID valueSet_id, ST valueSet_name, BL includeHeadCode, Conceptid codeToValidate): BL.
Descripción:	Determina si el código de concepto suministrado es un valor válido en el conjunto de valores suministrado.
Nombre:	Lookup Code System (UID codeSystem_id, ST codeSystem_name): CodeSystemInfo.
Descripción:	Busca información de un sistema de código.
Nombre:	Lookup Value Set (UID valueSet_id, ST valueSet_name): FullValueSetDescription.
Descripción:	Busca información detallada de un conjunto de valores (incluyendo dominios de vocabularios, constructores, etc.).
Nombre:	Lookup Value Set For Domain (ST vocabularyDomain_name, ST applicationContext_code): ValueSetDescriptor.
Descripción:	Devuelve el identificador del conjunto de valores que se utilizaría para el vocabulario en el contexto suministrado (si existe).
Nombre:	Lookup Vocabulary Domain (ST vocabularyDomain_name): VocabularyDomainDescription
Descripción:	Busca toda la información conocida sobre el dominio de vocabulario suministrado.

Tabla 2.10 *Descripción de la clase controladora Message Browser.*

Nombre: Identification Operations.

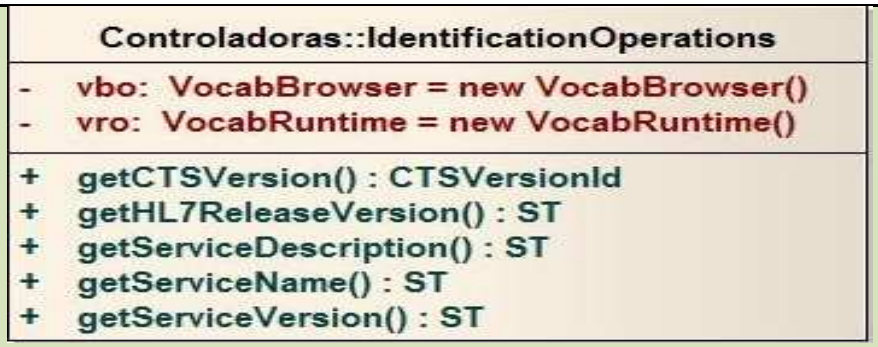	
Tipo de clase: Controladora.	
Para cada responsabilidad:	
Nombre:	Get CTS Version (): CTSVersionId.
Descripción:	Devuelve la versión de CTS que implementa este servicio.
Nombre:	Get HL7 Release Version (): ST [].
Descripción:	Devuelve la versión de HL7 que está soportada por el servicio.
Nombre:	Get Service Description (): ST [].
Descripción:	Devuelve una descripción de la función de servicio, los autores, derecho de autor, etc.
Nombre:	Get Service Name (): ST [].
Descripción:	Devuelve el nombre del servicio que brinda el sistema.
Nombre:	Get Service Version (): ST [].
Descripción:	Devuelve el identificador de la versión actual del software de servicio.

Tabla 2.11 *Descripción de la clase controladora Identification Operations.*

Con el desarrollo de este capítulo se obtuvo la propuesta de solución del sistema a implementar, fue obtenido el Modelo de Diseño y se describió la realización física de los casos de uso. Centrándose en cómo los requerimientos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en la capa a desarrollar, lo que constituyó una abstracción para la implementación y es utilizado como una entrada fundamental en las actividades que se realizan en el Flujo de Trabajo de Implementación.

CAPÍTULO III: ANÁLISIS DE RESULTADOS

Este capítulo describe cómo los elementos del diseño son implementados en términos de componentes, así como su organización física de acuerdo a los nodos específicos en el Diagrama de Despliegue. Este artefacto junto con el Diagrama de Componentes conforma el Modelo de Implementación. Además se aborda sobre el tratamiento de excepciones, la seguridad del sistema, así como las estrategias de codificación, estándares y estilos que se utilizan.

3.1 Propuesta de integración.

Ningún sistema en la actualidad se concibe de forma aislada por lo que se hace necesario pensar en su integración con otros sistemas en funcionamiento o en vías de desarrollo e incluso con posibles sistemas a desarrollar.

La Capa de Mensajería es la capa superior del Servicio de Terminologías Comunes, mantiene una estrecha relación con la Capa de Vocabulario y esta a su vez con la Capa de Mapeo, el Sistema de Administración es el encargado de gestionar todas las terminologías médicas que son utilizadas por los sistemas de salud del CESIM, así como de la seguridad del servicio, entre otras funciones. Para que dicha interacción sea consumada deben estar incluidas en el Sistema de Administración, los esquemas de las bases de datos de todas capas, para permitir a las mismas realizar sus funcionalidades de manera coordinada y organizada, a través de la comunicación de estas con los datos almacenados en la fuente de información del servicio, además deben estar publicados como servicios web todas las funcionalidades de dichas capas para que puedan ser consumidos por los sistemas de salud y de esta forma permitir la realización de consultas al contenido terminológico de CTS.

3.2 Modelo de Datos.

Es el modelo que describe de manera abstracta cómo se representan los datos de una aplicación o sistema de información. Consiste en una descripción de algo conocido como contenedor de datos, así como de los métodos para almacenar y recuperar información de esos contenedores. El Modelo de Datos tiene gran importancia en el ciclo de desarrollo de software, y de manera particular para la fase de implementación, pues define formalmente las estructuras permitidas y las restricciones que se aplican con el fin de representar los datos del dominio de la aplicación. Está compuesto por objetos: entidades que

existen y se manipulan; y atributos: características básicas de dichos objetos y relaciones: forma en que se enlazan los objetos entre sí. (53)

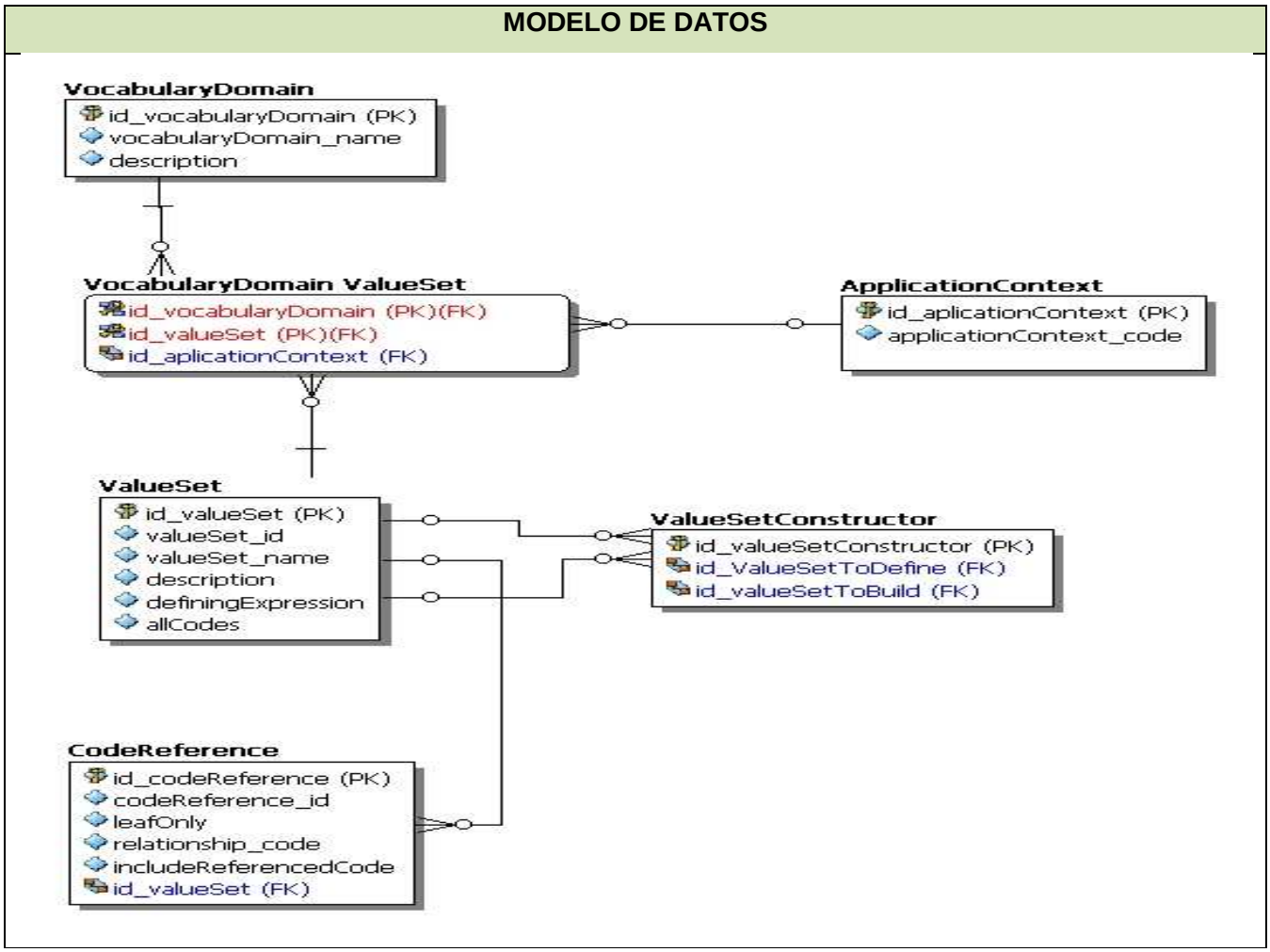

Figura 3.1 *Modelo de Datos.*

3.2.1 Descripción de las tablas de la Base de Datos.

A continuación se realiza la descripción de las tablas de la Base de Datos especificando el nombre de cada atributo, su tipo, si no admite valores nulos y una breve descripción del mismo. En las descripciones se especificará además si el atributo es llave primaria (PK) o llave foránea (FK).

Nombre: VocabularyDomain
Descripción: Almacena todos los dominios de vocabulario que utilizan los sistemas de salud.

Atributo	Tipo	Nulo	Descripción
id_vocabularyDomain	Integer	No	Identificador único de un dominio de vocabulario (PK).
vocabularydomain_name	ST	No	Representa el único nombre de un dominio de vocabulario.
description	ST	No	Descripción de un dominio de vocabulario.

Tabla 3.1 Descripción de la tabla de la Base de Datos: *VocabularyDomain.*

Nombre: VocabularyDomain ValueSet			
Descripción: Almacena todos los dominios de vocabularios asociados a conjuntos de valores.			
Atributo	**Tipo**	**Nulo**	**Descripción**
id_vocabularyDomain	Integer	No	Identificador único para la tabla dominio de vocabulario (PK) (FK).
id_valueSet	Integer	No	Identificador único para la tabla conjunto de valores (PK) (FK).
id_applicationContext	Integer	No	Identificador único para la tabla contexto de aplicación (FK).

Tabla 3.2 Descripción de la tabla de la Base de Datos: *VocabularyDomain ValueSet.*

Nombre: ValueSet			
Descripción: Almacena uno o más conceptos codificados derivados de un único sistema de código.			
Atributo	**Tipo**	**Nulo**	**Descripción**
id_valueSet	Integer	No	Identificador único para la tabla conjunto de valores (PK)
valueSet_id	OID	No	Identificador de los conjunto de valores que contiene HL7.
valueSet_name	ST	No	Nombre único de un conjunto de valores.
description	ST	No	Descripción de un conjunto de valores.
definingExpresion	ST		Una expresión que define el contenido del conjunto de valores.
allCodes	BL		Si es verdadero significa que todos los códigos de un sistema de código están incluidos en un conjunto de valores y el conjunto de valores no referencia ningún código adicional.

Tabla 3.3 Descripción de la tabla de la Base de Datos: *ValueSet.*

Nombre: ApplicationContext			
Descripción: Almacena todos los contextos de aplicación asociados a los dominios de vocabularios existentes.			
Atributo	**Tipo**	**Nulo**	**Descripción**
id_applicationContext	Integer	No	Identificador único para la tabla contexto de aplicación (PK).
applicationContext_code	ST	No	Un código que identifica un contexto que puede ser un dominio geopolítico, una profesión, etc.

Tabla 3.4 Descripción de la tabla de la Base de Datos: *ApplicationContext.*

Nombre: ValueSetConstructor			
Descripción: Almacena los conjuntos de valores incluidos por referencia en el conjunto contenedor.			
Atributo	**Tipo**	**Nulo**	**Descripción**
id_valueSetConstructor	Integer	No	Identificador único para la tabla constructor de conjunto de valores (PK).
id_valueSetToDefine	Integer	No	Identificador de un conjunto de valores a definir (FK).
id_valueSetToBuild	Integer	No	Identificador de un conjunto de valores a construir (FK).

Tabla 3.5 Descripción de la tabla de la Base de Datos: *ValueSetConstructor.*

Nombre: CodeReference			
Descripción: Almacena los códigos de concepto referenciados en un conjunto de valores.			
Atributo	**Tipo**	**Nulo**	**Descripción**
id_codeReference	Integer	No	Identificador único para la tabla código de referencia (PK).
codeReference_id	OID	No	Identificador de los códigos de referencia de HL7.
leafOnly	BL		Si es verdadero sólo incluye los nodos hojas de la relación, si es falso incluye todos los nodos descendientes, excepto los códigos referenciados. Debe ser siempre falso si no hay códigos referenciados.
relationship_code	RelationshipCode		Si un código de concepto está relacionado todos los descendientes de los códigos referenciados son considerados parte de un conjunto de valores.
includeReferenceCode	BL		Si es verdadero el código referenciado forma parte de un conjunto de valores, si es falso solo los descendiente son incluidos. Siempre será verdadero cuando la relación de un código no sea suministrada.
id_valueSet	Integer	No	Identificador de un conjunto de valores (FK).

Tabla 3.6 Descripción de la tabla de la Base de Datos: *CodeReference.*

3.3 Modelo de Implementación.

Está conformado por los Diagramas de Componentes y de Despliegue, describiendo cómo los elementos del Modelo de Diseño se implementan en términos de componentes, ficheros de código fuente y ejecutables. El Modelo de Implementación describe además cómo se organizan los componentes de

acuerdo con los mecanismos de estructuración y modularización, disponibles en el entorno de implementación y en el lenguaje de programación utilizado, y cómo dependen los componentes unos de otros. (54)

3.3.1 Diagrama de Componentes.

Es una representación gráfica que muestra un conjunto de elementos del modelo tales como componentes, subsistemas o paquetes de implementación y sus relaciones. Permite modelar la vista estática del sistema mostrando las dependencias lógicas entre un conjunto de componentes de software. Este diagrama se estructura en paquetes, que son divisiones físicas del sistema. Los paquetes están organizados en una jerarquía de capas donde cada capa tiene una interfaz bien definida. Los componentes pueden ser de código fuente, librerías, binarios o ejecutables y tienen relaciones de traza con los elementos del modelo que implementan. (55) Para la confección del diagrama de componentes de la capa se utilizaron los de código fuente.

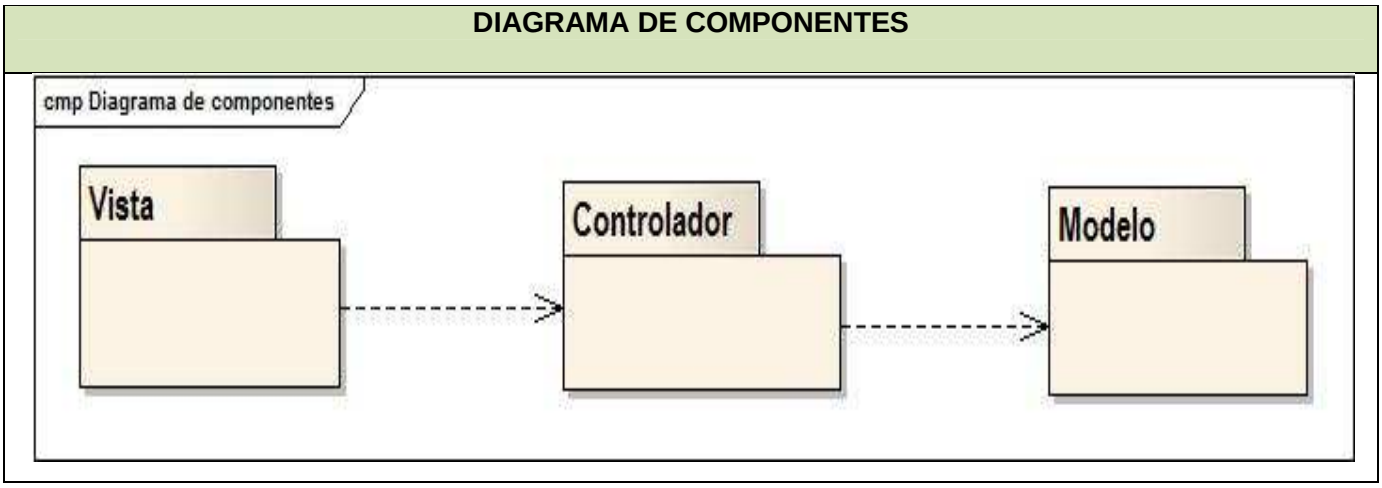

Figura 3.2 *Diagrama de Componentes Capa Mensajería.*

DIAGRAMA DE COMPONENTES

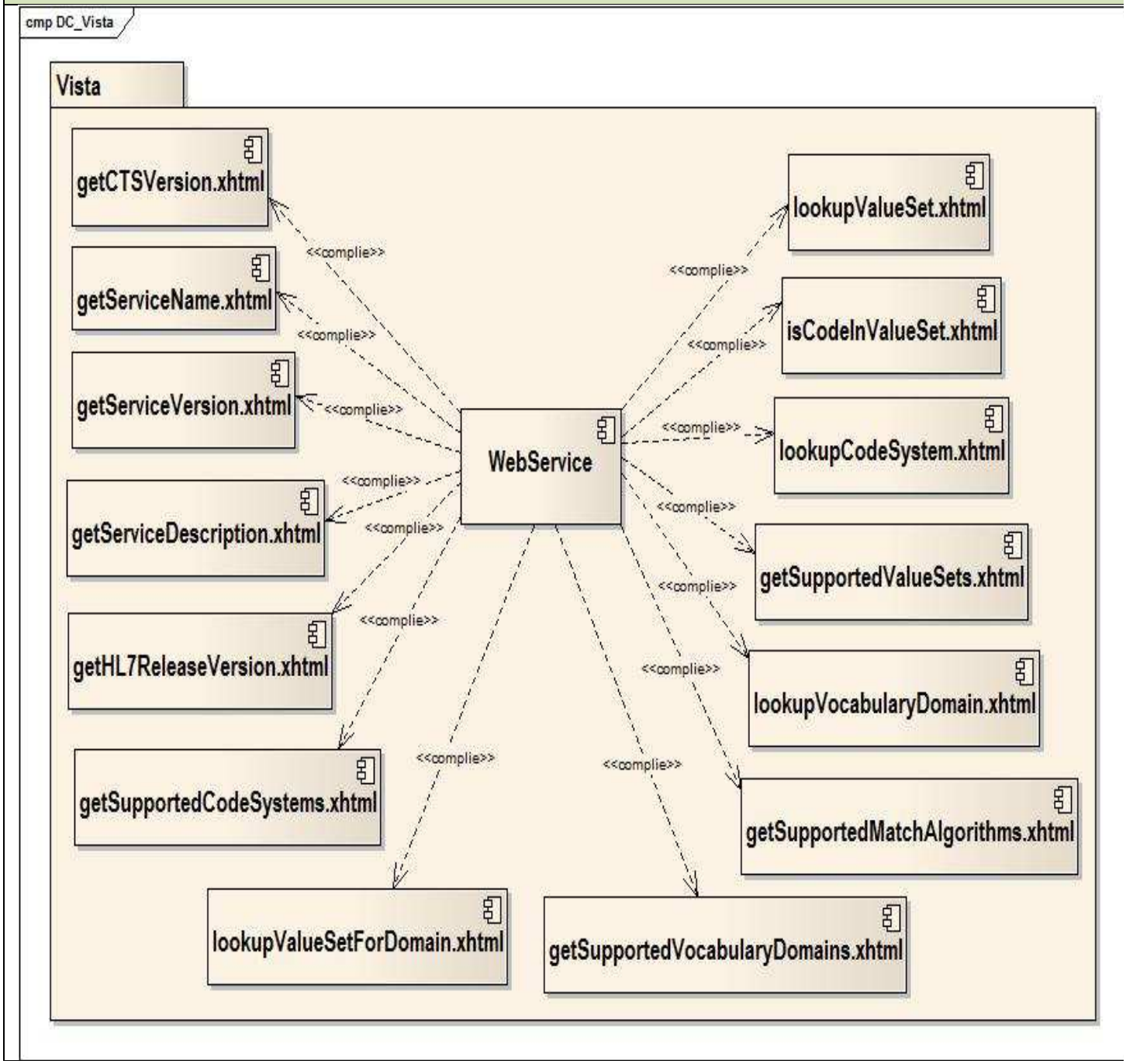

Figura 3.3 *Diagrama de Componentes Vista.*

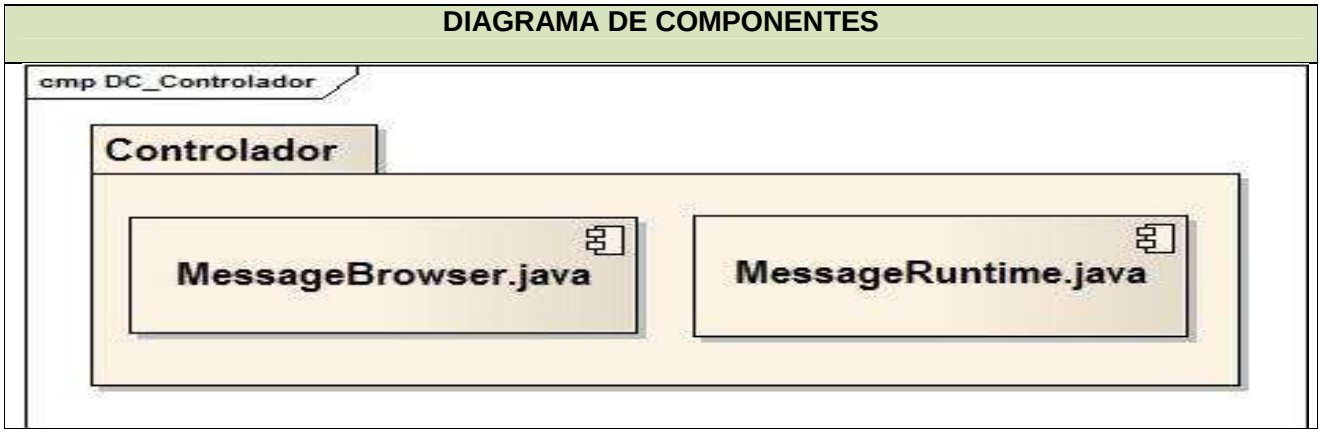

Figura 3.4 *Diagrama de Componentes Controlador.*

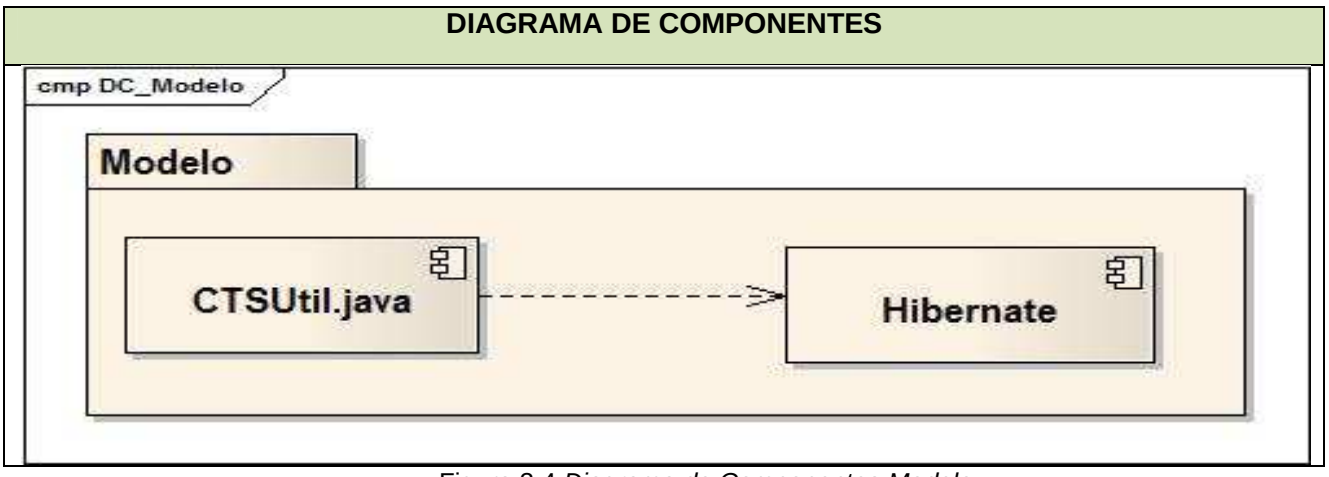

Figura 3.4 *Diagrama de Componentes Modelo.*

3.3.2 Diagrama de Despliegue.

Es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Por esta razón, tal distribución tiene una influencia principal para las actividades de implementación. Cada nodo representa un recurso de cómputo. (56)

La Capa de Mensajería contará con dos servidores, uno de aplicaciones (web) y otro de base de datos, conectados entre sí mediante el Protocolo de Control de Transmisión/ Protocolo de Internet (TCP/IP, por sus siglas en inglés). A continuación se muestra el Diagrama de Despliegue que evidencia las características de software y de hardware de cada uno de los nodos físicos.

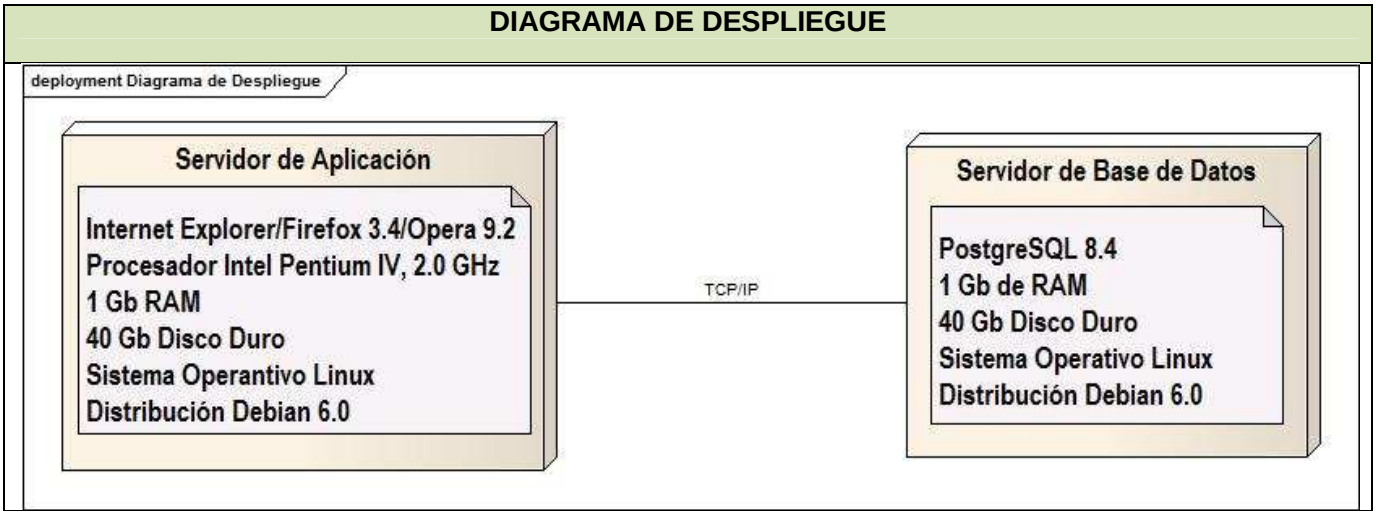

Figura 3.5 *Diagrama de Despliegue.*

3.4 Tratamiento de Excepciones.

Una excepción es un evento que ocurre durante la ejecución de un programa interrumpiendo el flujo normal de las sentencias. Dicho evento puede ser desde serios problemas de hardware, como la avería de un disco duro, hasta los simples errores de programación y pueden ser tratados mediante una estructura de control que poseen los lenguajes de programación de alto nivel, diseñada para manejar condiciones anormales que pueden ser tratadas por el mismo programa que se desarrolla. A esta estructura de control se le conoce como tratamiento de excepciones.

Un buen tratamiento de excepciones garantiza un aumento considerable en la calidad de las aplicaciones que se desarrollen. El lenguaje de programación que se utiliza para el desarrollo de la capa, proporciona una buena solución al problema, lo que permite escribir el flujo principal de su código y tratar los casos excepcionales en otro lugar, como se muestra en la Figura 3.4 Con esta separación se diferencia la conducta a seguir en dependencia del tipo de excepción ocurrida.

EJEMPLO DE TRATAMIENTO DE EXCEPCIONES

```
public static void main(String[] args) {
    try {
        // Se ejecuta el flujo principal del código, que puede producir una
        // excepción.
    } catch (IOException e) {
        // Tratamiento de una excepción de entrada/salida.
    } catch (Exception e) {
        // Tratamiento de una excepción cualquiera.
    } finally {
        // Código a ejecutar haya ocurrido o no una excepción.
    }
}
```

Figura 3.6 *Tratamiento de excepciones con el lenguaje de programación Java.*

En la implementación de la Capa de Mensajería se realiza el tratamiento de excepciones de persistencia de datos, así como de validación y conversión de valores de entrada proporcionados por el usuario. Para ello utiliza controles de conversión y validación que proveen algunas de las tecnologías que se utilizan en el desarrollo; como el validador que proporciona el framework Hibernate, el cual implementa la validación de datos multicapa, donde las restricciones y constantes de validación son expresadas mediante anotaciones en el Modelo (Entidades), además de un conjunto de excepciones especificadas por CTS para las funcionalidades de la capa.

EJEMPLO DE TRATAMIENTO DE EXCEPCIONES

```
public ST getServiceDescription() throws UnexpectedError {
    try {

        return new ST(vro.getServiceDescription());
    } catch (Exception e) {

        UnexpectedError ue = new UnexpectedError();
        ue.setStackTrace(e.getStackTrace());
        ue.possible_cause = new ST(e.getMessage());

        throw ue;
    }
}
```

Figura 3.7 *Tratamiento de excepciones en una de las funcionalidades de la Capa.*

3.5 Seguridad.

La seguridad es un tema de gran importancia para cualquier servicio informático que se brinde, toma mayor relevancia cuando se gestiona información relacionada con la salud de las personas. Para garantizar la seguridad de la misma y que el servicio sea calificado como seguro debe contar con un correcto equilibrio entre las siguientes características:

Integridad: La información sólo puede ser modificada por usuarios autorizados y de manera controlada.

Confidencialidad: La información sólo puede ser accedida por los usuarios autorizados para hacerlo.

Disponibilidad: La información debe estar disponible cuando se necesite.

En correspondencia con dichas características se plantea que el servicio permitirá la recuperación de la información de la base de datos a partir de los respaldos o salvas realizadas.

3.6 Estrategias de codificación. Estándares y estilos.

Las estrategias de codificación son muy importantes para la implementación de cualquier producto de software, en especial para las personas que los desarrollan. Un producto de software puede estar más del 80% de su vida útil en mantenimiento y en la mayoría de los casos, no es llevado a cabo por su autor original. Las estrategias de codificación hacen al software más legible, permitiendo que los desarrolladores puedan entender su código a fondo y más rápido.

Como estrategia de codificación para el desarrollo de la capa se utiliza el estándar de codificación del leguaje de programación Java (Java Code Conventions), definido en las especificaciones del propio lenguaje. Todo el código se escribirá en idioma inglés siguiendo la especificación de HL7 CTS, se utilizará comentarios en español para una mejor comprensión del código fuente. Para el nombre de las clases se utilizará la notación o estilo Pascal Casing, con la cual los nombres quedan compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula. Los identificadores de los atributos y métodos de la clase se escribirán en notación Camel Casing: nombres compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula excepto la primera palabra que debe iniciar con minúscula.

Con el desarrollo de este capítulo se ha concluido el proceso de implementación y se obtuvieron los artefactos que conforman el Modelo de Implementación, Diagrama de Despliegue y Componentes. También se desarrollaron la totalidad de las funcionalidades de la Capa de Mensajería previstas para el

correcto funcionamiento del Servicio de Terminologías Comunes. Esto facilita interoperabilidad interpretación y traducción de la información entre los diferentes sistemas de salud.

CONCLUSIONES

Con el desarrollo de la presente investigación se determinó la necesidad de implementar una Capa de Mensajería para el proyecto Servicios de Terminologías Comunes. El desarrollo de las tareas de la investigación posibilitó dar cumplimiento al objetivo general trazado y arribar a las siguientes conclusiones:

- El proceso de gestión de información entre las aplicaciones es complejo y las especificaciones y herramientas analizadas en el estudio del estado del arte no responden a las necesidades de los sistemas de salud que se desarrollan en el CESIM.

- La guía del Proceso Unificado de Desarrollo es muy importante para la comprensión del negocio, el levantamiento de requerimientos, y la correcta realización del diseño e implementación de cualquier sistema de información.

- La integración de la Capa de Mensajería a las demás capas de CTS, permite obtener una estructura de datos común para las terminologías médicas que se utilizan en los sistemas desarrollados en el CESIM, contribuyendo a garantizar un fácil y rápido acceso a las mismas.

- El desarrollo de la Capa de Mensajería proporciona al Servicio de Terminologías Comunes la interacción de los sistemas externos con las terminologías médicas almacenadas en la base de datos que posee el servicio.

- La implementación de las funcionalidades de navegación permite a los sistemas externos validar y traducir terminologías, facilitando una mejor interpretación de estas, posibilitando además interactuar con las funcionalidades específicas de la Capa de Vocabulario.

RECOMENDACIONES

- Implementar las funcionalidades específicas de tiempo de ejecución agrupadas en el Message Runtime con el fin de completar la implementación de la Capa de Mensajería.

- Una vez publicada la versión 3.0 de CTS se debe revisar la especificación correspondiente con el objetivo de actualizar la implementación de la Capa de Mensajería, si es necesario.

- Implementar un Universal Description, Discovery and Integration (UDDI) que permita obtener una mejor descripción de los servicios implementados de la Capa de Mensajería así como facilitar la interacción con los mismos.

REFERENCIAS BIBLIOGRÁFICAS

1. *Pilares y debilidades del proceso de informatización de la Atención Primaria de Salud (APS).* **Marín, ME.** Ciudad de La Habana .s.n.
2. *From hospital information systems to health.Problems, challenges, perspectives. Methods Inf Med.* **Kuhn, K.A and D.A. Giuse.** 40(4):p. 275-87.
3. **INFOMED.** Portal de Salud de Cuba. . *Sistema de Salud.* [En línea] http://www.sld.cu/sistema_de_salud/aspectos.html.
4. **Dr. Ramírez Márquez, Abelardo, Dr. Castell-Florit Serrate, Pastor,Dr. Mesa, Guillermo.** INFOMED,Portal de Salud de Cuba. *Escuela Nacional de Salud Pública(ENSAP), El Sistema Nacional de Salud de Cuba.* [En línea] http://www.sld.cu/galerias/doc/sitios/infodir/09_el_sistema_nacional_de_salud.doc.
5. *TSMI: a CEN/TC251 standard for time specific problems in healthcare informatics and telematics.Int J Med Inform.* **Ceusters, W.** 46(2): p. 87-101.
6. *Coding in the coming decade.* **Buck, S.L.** 79(10): p. 100-1.
7. *A practical approach to advanced terminology services in health information systems. Stud Health Technol Inform.* **Gambarte, M.L.** 129(Pt 1): p. 621-5.
8. Ídem a la referencia 7.
9. Ídem a la referencia 7.
10. Ídem a la referencia 7.
11. Ídem a la referencia 7.
12. Ídem a la referencia 7.
13. HL7. [En línea] http://10.128.50.168/HL7/welcome/environment/index.htm.
14. Ídem a la referencia 8.
15. Ídem a la referencia 8.
16. HL7. [En línea] http://www.hl7.org/.
17. Ídem a la referencia 9.
18. Ídem a la referencia 9.
19. *OMG Lexicon Query Service Specification:Object Management Group.* Report No.: 00-06-31.pdf.
20. Lexicon Query Services. [En línea] http://www.omg.org/technology/documents/formal/lexicon_query_service.htm.
21. **Harold R Solbrig, James D Buntrock.** *LexGrid: A Framework for Representing, Storing, and Querying Biomedical Terminologies from Simple to Sublime - Jyotishman Pathak,.* 2009.
22. Web Ontology Language. [En línea] http://www.w3.org/2007/OWL/.
23. Guía de la versión 1 de OWL. [En línea] http://www.w3.org/TR/2004/REC-owl-guide-20040210/.
24. Guía de la versión 2 de OWL. [En línea] http://www.w3.org/TR/owl-guide/.
25. *LexGrid: A Framework for Representing, Storing, and Querying Biomedical Terminologies from Simple to Sublime.* **Jyotishman Pathak, Harold R Solbrig, James D Buntrock.** 2009.
26. CliniClue Xplorer. [En línea] http://www.cliniclue.com.
27. **Markwell, David.** CliniClue Xplore Additional Tools. *The Clinical Information Consultancy Ltd.*
28. Bioportal de NCBO. [En línea] http://bioportal.bioontology.org/.
29. Unified Medical Language System (UMLS). [En línea] http://www.nlm.nih.gov/research/umls/.

30. **Angela, Martín M.** ChanelPlanet. *Software: Beneficios de la Arquitectura Orientada a Servicios.* [En línea] http://www.desarrolloweb.com/articulos/392.php.
31. Ídem a la referencia 30.
32. Seam - Contextual Components. Introduction to JBoss Seam. [En línea] http://www.seamframework.org/Documentation.
33. *Seam in action. Manning Early Access Program. Manning Publications Co.p 4, p 5, p 6.* **Dan, Allen.**
34. *Java Code Conventions. Sun Microsystems.* California, Estados Unidos de América.s.n.
35. *Java Persistence with Hibernate.Manning Publications Co.* **Bauer Christian, King Gavin.** p 67.
36. PostgreSQL About. [En línea] http://www.postgresql.org/about.
37. PostgreSQL Awards. [En línea] http://www.postgresql.org/about/awards.
38. *Aprenda Java como si estuviera en primero.Universidad de Navarra. España.Escuela Superior de Ingenieros Industriales* . **García de Jalón Javier, Rodriguez Iñigo Mingo José Ignacio, Alfonso Brazález Aitor Imaz,Larzabal Alberto, Calleja Jesús, García Jon.** España. s.n.
39. **Michael, Dr. Juntao Yuan.** On the road to simplicity. [En línea] http://www.javaworld.com/javaworld/jw-02-2005/jw-0221-jboss4.html .
40. JBoss Server Manager Reference Guide. [En línea] http://www.jboss.org/jbossas/docs/ .
41. **James Rumbaugh, Ivar Jacobson, Grady Booch.** El Lenguaje Unificado de Modelado. Manual de Referencia. s.l. : Pearson Education, 2009.
42. **Canós José H., Letelier Patricio, Penadés Mª Carmen.** Metodologías Ágiles en el Desarrollo de Software.DSIC. Universidad Politécnica de Valencia. [En línea] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.
43. **Carol O'Rourke, Neal Fishman, Warren Selkow.** Enterprise Architecture Using the Zachman Framework. 2010.
44. **Zachman, John A.** The Zachman Framework for Enterprise Architecture - A Primer. 2009.
45. **S, Presman Roger.** Ingeniería de Software, un enfoque práctico. La Habana, Cuba. Editorial Félix Varela.
46. Ídem a la referencia 45.
47. **S, Presman.** Ingeniería de Software, un enfoque práctico. *McGraw-Hill.*
48. Ídem a la referencia 45.
49. Ídem a la referencia 47.
50. Ídem a la referencia 45.
51. Ídem a la referencia 45.
52. Ídem a la referencia 47.
53. Ídem a la referencia 45.
54. Ídem a la referencia 47.
55. Ídem a la referencia 45.
56. Ídem a la referencia 47.

BIBLIOGRAFÍA

1. **Angela, Martín M.** ChanelPlanet. *Software: Beneficios de la Arquitectura Orientada a Servicios.* [En línea] http://www.desarrolloweb.com/articulos/392.php.
2. *A practical approach to advanced terminology services in health information systems. Stud Health Technol Inform.* **Gambarte, M.L.** 129(Pt 1): p. 621-5.
3. *Aprenda Java como si estuviera en primero.Universidad de Navarra. España.Escuela Superior de Ingenieros Industriales* . **García de Jalón Javier, Rodriguez Iñigo Mingo José Ignacio, Alfonso Brazález Aitor Imaz,Larzabal Alberto, Calleja Jesús, García Jon.** España.s.n.
4. Bioportal de NCBO. [En línea] http://bioportal.bioontology.org/.
5. *Coding in the coming decade.* **Buck, S.L.** 79(10): p. 100-1.
6. **Canós José H., Letelier Patricio, Penadés Mª Carmen.** Metodologías Ágiles en el Desarrollo de Software.DSIC. Universidad Politécnica de Valencia. [En línea] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.
7. **Carol O'Rourke, Neal Fishman, Warren Selkow.** Enterprise Architecture Using the Zachman Framework. 2010.
8. CliniClue Xplorer. [En línea] http://www.cliniclue.com.
9. **Dr. Ramírez Márquez, Abelardo, Dr. Castell-Florit Serrate, Pastor,Dr. Mesa, Guillermo.** INFOMED,Portal de Salud de Cuba. *Escuela Nacional de Salud Pública(ENSAP), El Sistema Nacional de Salud de Cuba.* [En línea] http://www.sld.cu/galerias/doc/sitios/infodir/09_el_sistema_nacional_de_salud.doc.
10. *From hospital information systems to health.Problems, challenges, perspectives. Methods Inf Med.* **Kuhn, K.A and D.A. Giuse.** 40(4):p. 275-87.
11. *Fundamentos del Sistema de Salud Pública en Cuba para estudiantes de Informática.* **Marín Díaz, Miguel E.** La Habana. Cuba. s.n.
12. **Fernán., Mandirola Brieux Humberto.** La crisis en el Área Salud en la era de la información. Buenos Aires: BIOCOM. [En línea] http://www.biocom.com/informatica_medica/La%20crisis%20en%20la%20%20era%20de%20la%20informaci%C3%B3n%20en%20el%20Area%20Salud.pdf.
13. Guía de la versión 1 de OWL. [En línea] http://www.w3.org/TR/2004/REC-owl-guide-20040210/.
14. Guía de la versión 2 de OWL. [En línea] http://www.w3.org/TR/owl-guide/.
15. Group, Business Process Modeling Notation Specification. Object Management. [En línea] http://www.omg.org/technology/agreement.htm.
16. HL7. [En línea] http://10.128.50.168/HL7/welcome/environment/index.htm.
17. HL7. [En línea] http://www.hl7.org/.
18. **Harold R Solbrig, James D Buntrock.** *LexGrid: A Framework for Representing, Storing, and Querying Biomedical Terminologies from Simple to Sublime - Jyotishman Pathak,.* 2009.
19. *Hospital information systems-past, present, future.Int J Med Inform.* **Reichertz, P.L.** 75(3-4): p. 282-99.
20. **INFOMED.** Portal de Salud de Cuba. . *Sistema de Salud.* [En línea] http://www.sld.cu/sistema_de_salud/aspectos.html.
21. *Java Code Conventions. Sun Microsystems.* California, Estados Unidos de América. s.n.

BIBLIOGRAFÍA

22. *Java Persistence with Hibernate.Manning Publications Co.* **Bauer Christian, King Gavin.** p 67.
23. **James Rumbaugh, Ivar Jacobson, Grady Booch.** El Lenguaje Unificado de Modelado. Manual de Referencia. s.l. : Pearson Education, 2009.
24. Java Persistence API FAQ. Sun Microsystems. [En línea] http://java.sun.com/javaee/overview/faq/persistence.jsp.
25.Java Persistence API FAQ. . [En línea] Sun Microsystems. http://java.sun.com/javaee/overview/faq/persistence.jsp.
26. JBoss Server Manager Reference Guide. [En línea] http://www.jboss.org/jbossas/docs/.
27. JBoss Server Manager Reference Guide. [En línea] http://www.jboss.org/jbossas/docs/ .
28. Lexicon Query Services. [En línea] http://www.omg.org/technology/documents/formal/lexicon_query_service.htm.
29.LexGrid: A Framework for Representing, Storing, and Querying Biomedical Terminologies from Simple to Sublime. **Jyotishman Pathak, Harold R Solbrig, James D Buntrock.** 2009.
30. **Markwell, David.** CliniClue Xplore Additional Tools. *The Clinical Information Consultancy Ltd.*
31. **Michael, Dr. Juntao Yuan.** On the road to simplicity. [En línea] http://www.javaworld.com/javaworld/jw-02-2005/jw-0221-jboss4.html .
32. **Michael, Dr. Juntao Yuan.** On the road to simplicity. [En línea] http://www.javaworld.com/javaworld/jw-02-2005/jw-0221-jboss4.html .
33. *OMG Lexicon Query Service Specification:Object Management Group.* Report No.: 00-06-31.pdf.
34. **Owen Martin, Raj Jog.** BPMN and Business Process Management. Introduction to the New Business Process Modeling Standard. Popkin Software. [En línea] http://www.bpmn.org/Documents/6AD5D16960.BPMN_and_BPM.pdf.
35. PostgreSQL About. [En línea] http://www.postgresql.org/about.
36. PostgreSQL Awards. [En línea] http://www.postgresql.org/about/awards.
37. *Pilares y debilidades del proceso de informatización de la Atención Primaria de Salud (APS).* **Marín, ME.** Ciudad de La Habana .s.n.
38. Seam - Contextual Components. Introduction to JBoss Seam. [En línea] http://www.seamframework.org/Documentation.
39. *Seam in action. Manning Early Access Program. Manning Publications Co.p 4, p 5, p 6.* **Dan, Allen.**
40.Seam - Contextual Components. *Introduction to JBoss Seam.* [En línea] http://www.seamframework.org/Documentation.
41. Sistema de Información Hospitalaria. *México D.F.: Universidad Autónoma de México. D. R. Facultad de Medicina.* [En línea] http://www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf .
42. **S, Presman Roger.** Ingeniería de Software, un enfoque práctico. La Habana, Cuba.Editorial Félix Varela.
43. **S, Presman.** Ingeniería de Software, un enfoque práctico. *McGraw-Hill.*
44. *TSMI: a CEN/TC251 standard for time specific problems in healthcare informatics and telematics.Int J Med Inform.* **Ceusters, W.** 46(2): p. 87-101.
45. Unified Medical Language System (UMLS). [En línea] http://www.nlm.nih.gov/research/umls/.
46. Web Ontology Language. [En línea] http://www.w3.org/2007/OWL/
47. **Zachman, John A.** The Zachman Framework for Enterprise Architecture - A Primer. 2009.

ANEXOS

1.1 Diagramas de Secuencia del Diseño.

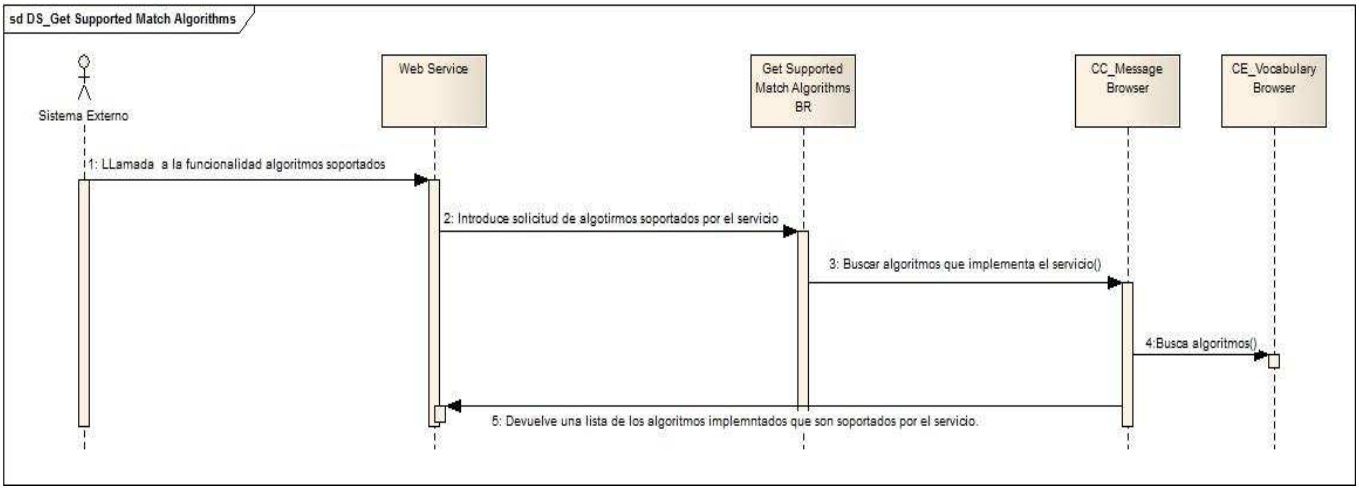

Figura 1 Diagrama de Secuencia *CUS_ Get Supported Match Algorithms*.

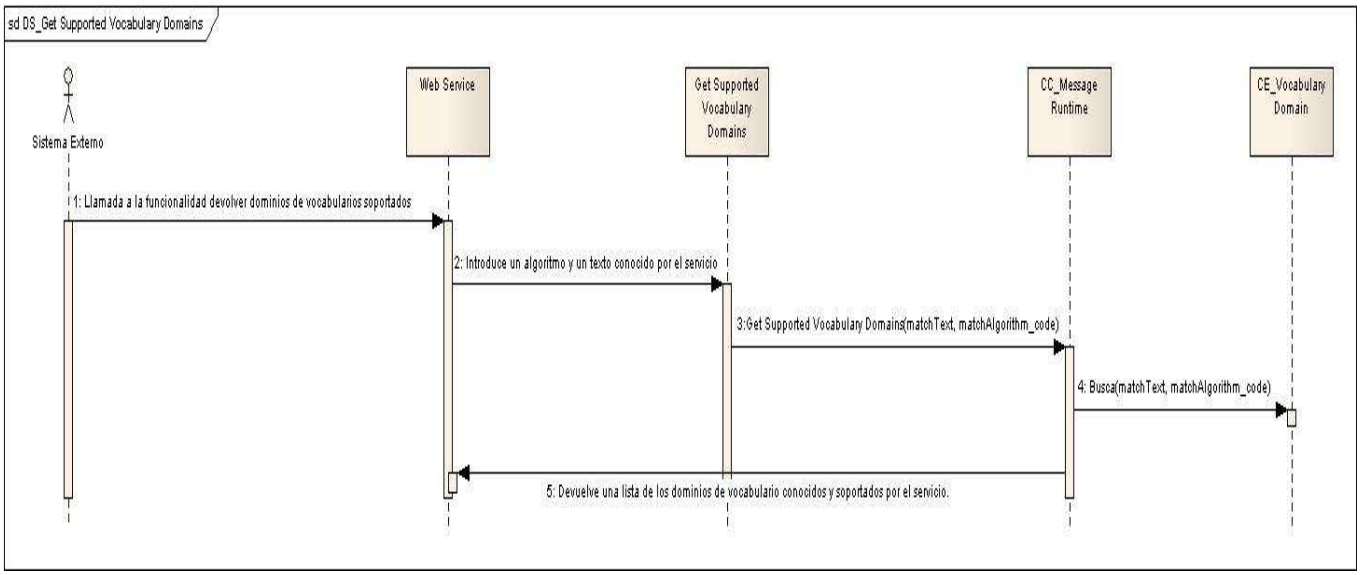

Figura 2 Diagrama de Secuencia *CUS_ Get Supported Vocabulary Domains*.

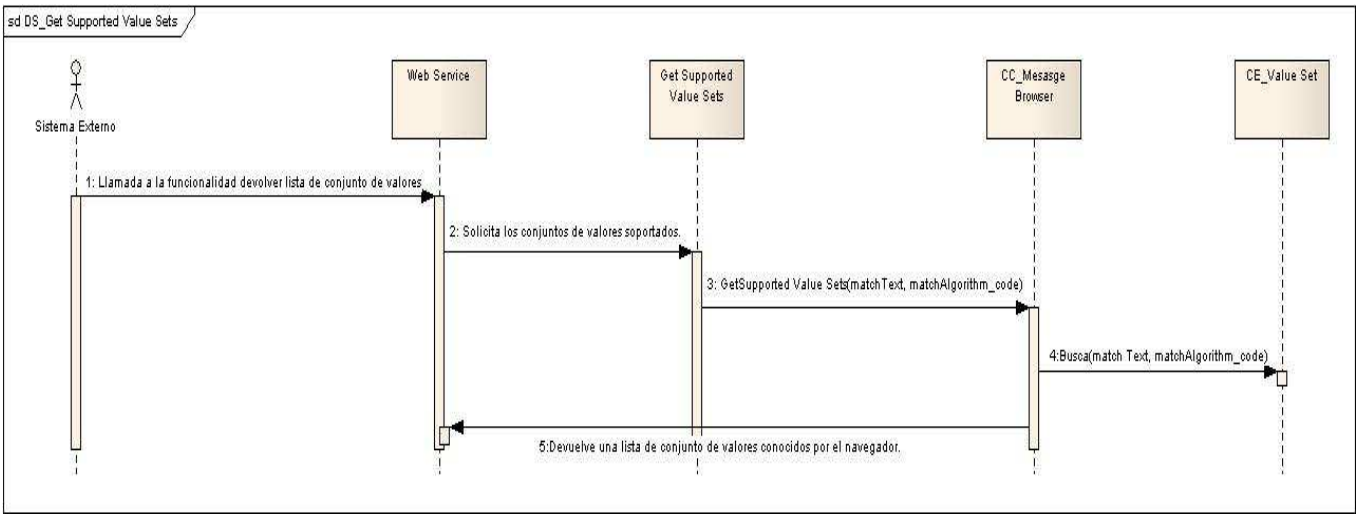

Figura 3 Diagrama de Secuencia *CUS_ Get Supported Value Sets*.

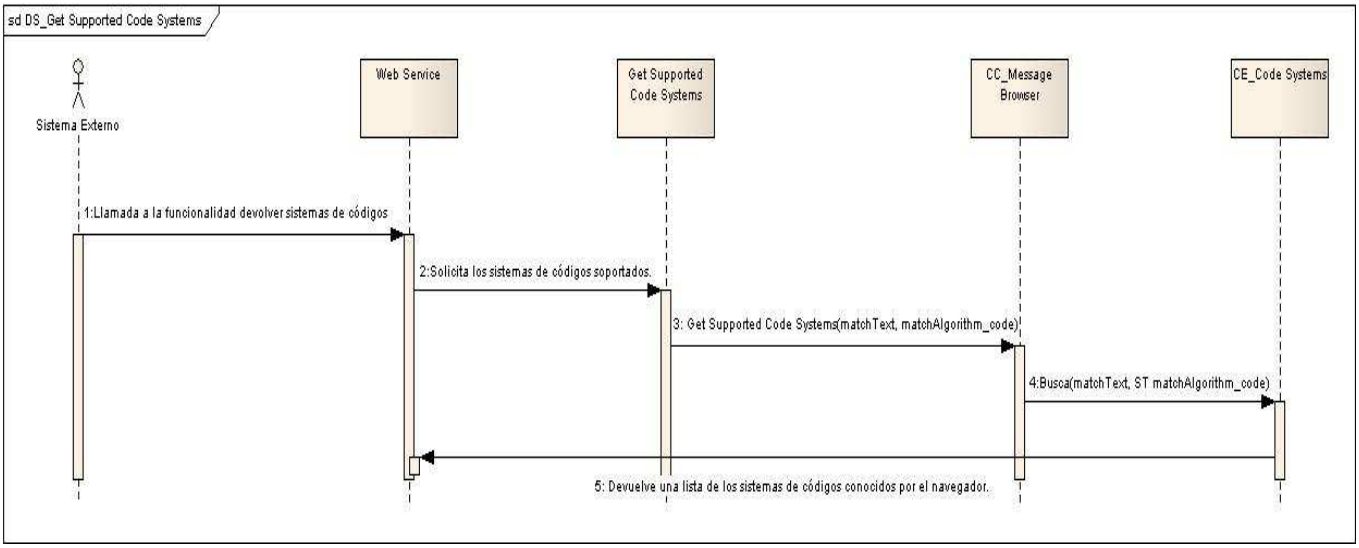

Figura 4 Diagrama de Secuencia *CUS_ Get Supported Code Systems*.

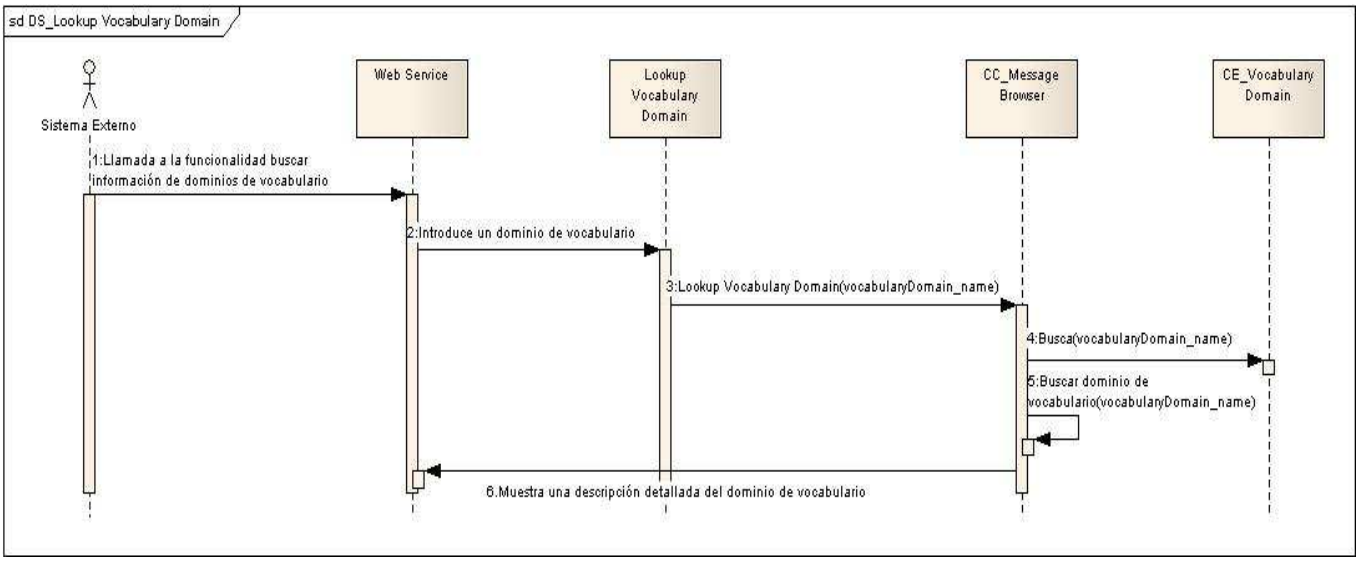

Figura 5 Diagrama de Secuencia *CUS_Lookup Vocabulary Domain*.

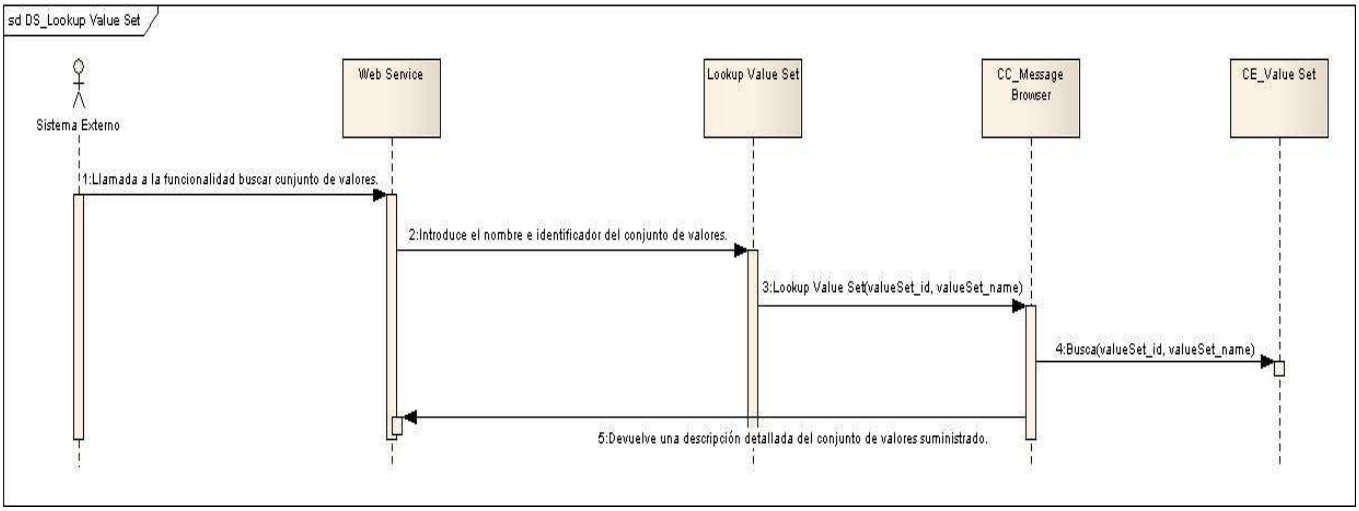

Figura 6 Diagrama de Secuencia *CUS_Lookup Value Sets*.

GLOSARIO

Accesibilidad: grado en el que todas las personas pueden utilizar un objeto, visitar un lugar o acceder a un servicio, independientemente de sus capacidades técnicas, cognitivas o físicas.

API de mensajería: su propósito principal es permitir una amplia variedad de aplicaciones de procesamiento de mensajes para crear, validar y traducir datos del Servicio de Terminologías Comunes.

Componente: se aplica al elemento que forma parte de una cosa o a la parte de una cosa que, junto con otras, la compone.

Codificadores de terminologías médicas: crean códigos específicos para cada paciente, usando su conocimiento de los sistemas de codificación médicos, terminología médica, enfermedades y medicamentos, estos crean y mantienen estos sistemas para garantizar un seguimiento eficiente de los pacientes y de los diagnósticos.

Estándar: es una especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad de un sistema.

Homogenizar: hacer uniforme los diversos elementos y terminologías que poseen los sistemas de salud.

Interpretación: es el hecho de que un contenido dado es comprendido o traducido a una nueva forma de expresión, independiente del intérprete. La interpretación debe ser fiel al contenido original del objeto interpretado.

Interoperabilidad: capacidad de dos o más sistemas de intercambiar y utilizar información entre ellos.

Ontologías: hace referencia a la formulación de un exhaustivo y riguroso esquema conceptual dentro de uno o varios dominios dados; con la finalidad de facilitar la comunicación y el intercambio de información entre diferentes sistemas y entidades.

Protocolo: es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red. Un protocolo es una convención o estándar que controla o permite la conexión, comunicación, y transferencia de datos entre dos puntos finales.

GLOSARIO

Red semántica: es una forma de representación de conocimiento lingüístico en la que los conceptos y sus interrelaciones se representan mediante un grafo. Las redes semánticas son usadas, entre otras cosas, para representar mapas conceptuales y mentales.

Servicios Terminológicos: es una herramienta diseñada específicamente para resolver la interoperabilidad terminológica en un SIS (Sistema de Información en Salud). Permiten lograr un adecuado equilibrio entre la libertad de los textos narrativos y los beneficios del ingreso estructurado de datos.

Servicio de Terminologías Comunes: es un servidor que almacena todos los vocabularios de los sistemas médicos informatizados, en vocabularios controlados, facilita el manejo e intercambio de información del sistema.

Servicios web: es una pieza de software que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet.

Sistemas de información: es un conjunto de elementos orientados al tratamiento y administración de datos e información organizados y listos para su posterior uso, generados para cubrir una necesidad u objetivo.

Terminología médica: es un campo de estudio que se nutre de un conjunto específico de conocimientos conceptualizados diferentes disciplinas. Se utilizan para referenciar tanto a la tarea de recolectar, describir y presentar términos de médicos de manera sistemática.

Vocabulario médico: es el conjunto de palabras que forman parte de un idioma específico, conocidas por una persona o una entidad como un diccionario, está conformado por las palabras más utilizadas en el ámbito de la salud.