

Universidad de las Ciencias Informáticas

Facultad 7



**Título: Desarrollo del Sistema de Administración para
el Servicio de Terminologías Comunes**

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

Autores: Dunia Rojas Martínez

Anadalys González Rodríguez

Tutor: Ing. Karel Fernández Cedeño

Co-Tutor: Ing. Niurka Córdova Osorio

La Habana, junio 2011

“ Año 53 de la Revolución”

*"El único autógrafo digno de un hombre
es el que deja escrito con sus obras."*

José Martí

DATOS DE CONTACTO

Ing. Karel Fernández Cedeño: Graduado en Ingeniería en Ciencias Informáticas en el 2008. Profesor Instructor, ha impartido las Asignaturas Técnicas de Programación, Ingeniería de Software y Práctica Profesional. Actualmente se desempeña como Analista Principal del Centro de Informática Médica. Correo electrónico: kfernandez@uci.cu Teléfono: 8358900

Niurka Córdova Osorio: Graduado en Ingeniería en Ciencias Informáticas en el 2009. Profesor en adiestramiento. Correo electrónico: ncordova@uci.cu Teléfono: 8358852

DEDICATORIA

Dunia Rojas Martínez: Dedico este trabajo de diploma a mi mamá y a mi papá que son lo más grande que tengo y que lo han dado todo por mí a lo largo de mi vida brindándome un buen ejemplo. A mi tío Ricardo que es como otro padre para mí, a mi tío Rafael que es como el hermano que no tuve, a mi tía Coralía que me cuidó como si fuera su hija y a mi abuela Hilda que la quiero mucho.

Anadafys González Rodríguez: A mi mamá por estar siempre a mi lado, apoyarme en mis decisiones, aconsejarme y darme todo su amor. A mi papá por ayudarme en todo. A mi hermana que siempre ha estado ahí para mí. A mi abuela y abuelo que son padres para mí y por darme todo su cariño. A mi tío que lo quiero mucho, que es mi ejemplo a seguir y lo quiero como el hermano mayor que siempre quise tener. A mi novio por confiar en mí y ayudarme todo este tiempo que hemos estado juntos.

RESUMEN

En el Centro de Informática Médica de la universidad de las Ciencias Informáticas se desarrolla un Servicio de Terminologías Comunes (CTS). Este servicio es estático y no permite gestionar las terminologías presentes en la fuente de información. Por lo que surge la necesidad de desarrollar un sistema que permita efectuar las configuraciones requeridas por el CTS. Para ello el presente trabajo de Diploma tiene como objetivo implementar un Sistema de Administración para el Servicio de Terminologías Comunes que permita su mantenimiento y actualización de manera segura.

Para el desarrollo del sistema se utilizó la metodología RUP como guía para construir los procesos necesarios del mismo. Se trabajó con la herramienta Enterprise Architect versión 7.1 para el modelado, haciendo uso del lenguaje de modelado UML (Lenguaje Unificado de Modelado). Como Sistema Gestor de Base de Datos se utilizó PostgreSQL en su versión 8.4. El entorno de desarrollo utilizado fue Eclipse y como lenguaje de programación Java. Se implementó el sistema en plataformas de software libre, manejando el Sistema Operativo Linux en su distribución Ubuntu versión 10.4.

El sistema de Administración brindará la posibilidad de poder insertar nuevos diccionarios terminológicos, poder modificar o insertar sus terminologías. Permitirá poder realizar búsquedas y ver los detalles de los datos. Estas funcionalidades proporcionan al Servicio de Terminología Común la posibilidad de satisfacer las demandas de los sistemas de información por más tiempo y con mayor calidad. Esto propicia la efectividad de los servicios sanitarios que se brinden, basados en información precisa y concreta del paciente.

Palabras clave:

Servicio de Terminologías Comunes, Diccionarios Terminológicos, Configuración, Sistema de Administración.

TABLA DE CONTENIDOS

TABLA DE CONTENIDOS

INTRODUCCIÓN	1
CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA	7
1.1 Conceptos fundamentales.....	7
1.2 Estado del Arte	8
1.3 Softwares creados a nivel internacional	9
1.4 Tecnologías actuales empleadas	12
CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA	20
2.1 Modelo de Dominio	20
2.2 Definición de las clases del modelo del dominio.....	21
2.3 Objeto de Automatización	23
2.4 Propuesta del Sistema	24
2.5 Especificación de los Requisitos del Sistema.....	25
CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA	39
3.1 Patrón de Arquitectura	39
3.2 Modelo de Clases del Diseño.....	39
3.3 Diagramas de clases del diseño.....	48
3.4 Diagramas de Secuencia	48
CAPÍTULO IV: IMPLEMENTACIÓN	49
4.1 Modelo de Datos.....	49
4.2 Implementación.....	52
4.3 Pruebas	59
CONCLUSIONES	60
RECOMENDACIONES	61
REFERENCIAS BIBLIOGRÁFICAS	62
BIBLIOGRAFÍA	64
ANEXOS	67
GLOSARIO DE TERMINOS	70

INTRODUCCIÓN

La introducción acelerada y masiva de la computación en Cuba es vista ya como una realidad que no sorprende ni se percibe en toda su grandeza y amplitud, hasta que no se compara con su situación en otros países subdesarrollados o en vías de desarrollo.

En el informe Central al III Congreso del Partido Comunista de Cuba, celebrado en el mes de febrero del año 1986, el compañero Fidel Castro refirió la necesidad de adecuar el desarrollo científico-técnico a las necesidades de la economía, al fomento de las ciencias y tecnologías de uso pacífico, de la biotecnología, de la electrónica y de las técnicas de computación. Así como de otras ciencias de avanzadas que serán las bases del progreso futuro del país, trayendo consigo transformaciones que interactúan entre la sociedad, propiciando una mejoría acelerada, que garantice una mayor calidad de vida al hombre a tono con las exigencias actuales y futuras.

La informatización ha sido de vital importancia para este desarrollo proclamado por el Comandante en Jefe de la Revolución cubana, así como para la organización de los sistemas de salud a nivel nacional y mundial. Los sistemas de salud a lo largo de la historia han estado íntimamente asociados a la forma en que se ha organizado el modo de producción de la sociedad. Han dependido del contexto social, político y económico de cada país, en los distintos momentos históricos. En Cuba esta ciencia está llamada a desempeñar un importante papel dentro del esfuerzo general que debe desarrollar el sistema de salud, para enfrentar con éxito las tareas que demanda la construcción del socialismo y de las ciencias en particular.

Con este fin, y por indicación de la dirección del Partido y del Gobierno, el Ministerio de Salud Pública (MINSAP) comenzó a desarrollar, programas de introducción de la informática. A pesar de las dificultades económicas que se avizoraban desde el triunfo de la Revolución; no se escatimaron esfuerzos, ni se limitaron recursos para llevar estos medios al servicio de la sociedad. Reafirmando la idea de que un buen sistema de salud mejora la vida cotidiana de las personas de manera tangible.

INTRODUCCIÓN

El MINSAP es el organismo rector del 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. (1) En él se trabaja para informatizar los sistemas de información referentes al paciente atendido en cualquier área de salud o unidad hospitalaria. Estas iniciativas persiguen, ante todo, elevar la calidad del servicio de salud en Cuba, además de buscar la creación de una plataforma que permita incrementar el intercambio entre los especialistas, y que contribuya a potenciar proyectos de investigación-desarrollo y consoliden la posición de vanguardia de la medicina cubana. (2)

La informatización de los sistemas de salud en todos sus ámbitos tomó como eje la información clínica; una de las premisas de estos nuevos sistemas fue respetar los procesos asistenciales poniendo al médico como eje central del modelo de información. Para permitir el desarrollo de estos sistemas, es preciso garantizar el correcto funcionamiento y acoplamiento de los diferentes componentes que requieren los mismos.

Dentro de ellos se destaca el Componente de Interoperabilidad que une al resto de los componentes, se encarga de comunicar diferentes sistemas preexistentes o nuevos, generalmente creados con disímiles herramientas de desarrollo, sobre múltiples plataformas y con diferentes fuentes de datos en el sistema de información. En esto juegan un papel crucial los Servicios Terminológicos que brindan acceso a diferentes terminologías que permiten lograr un adecuado equilibrio entre la libertad de los textos narrativos y los beneficios del ingreso estructurado de datos.

Estos componentes al principio estaban solo en los hospitales y desde allí fueron expandiéndose a otras unidades de salud. En la actualidad varios de ellos se encuentran en casi todos los sistemas de información de salud y otros representan etapas evolutivas avanzadas. (3)

Para disminuir la descentralización de la información en la atención médica en redes asistenciales se generó la necesidad de interoperar entre sus sistemas de información, teniendo en cuenta que la interoperabilidad no solo consiste, en que se envíen informes y documentos entre los sistemas, sino que los datos generados en ellos, puedan ser manejados de manera fiable por los profesionales que operan con las diversas esferas de salud; logrando con esto la conexión de múltiples sistemas. Ya no basta con

INTRODUCCIÓN

mantener el registro único en los sistemas internos de una organización, sino ir más allá de los muros de una institución, permitiendo una fluida comunicación de la información clínica.

Para que esta sea posible es una condición primordial que se establezcan criterios de normalización y equivalencia entre los datos, de manera que estos queden identificados y puedan ser tratados de manera automática por los diferentes sistemas.

Estos sistemas hacen uso de términos o frases que tratan de dar explicación a algún proceso o enfermedad, los cuales a medida que han ido surgiendo se han recopilado en diccionarios terminológicos, o también llamados sistemas de códigos, de acuerdo al área en la que se especializan. Para el trabajo con estos términos se han creado estándares para la salud que permiten el intercambio de los mismos entre los sistemas de las diferentes Unidades de Salud.

En el Centro de Informática Médica (CESIM) de la Universidad de las Ciencias Informáticas (UCI) se han desarrollado varios sistemas, en aras de automatizar y mejorar los procesos en las instituciones sanitarias dentro y fuera del ámbito nacional. Al ir adquiriendo experiencias en este sentido, el centro se ha visto en la necesidad de apegarse al uso de estándares internacionales, que permitan la interoperabilidad de los sistemas entre sus propios productos y con productos de terceros.

Dentro de los estándares con los que se trabaja se encuentra el HL7 (Health Level Seven), compuesto por un conjunto de estándares para el intercambio electrónico de información médica, el buen uso de estos proporciona las bases de la interoperabilidad que se pretende para mejorar la atención en la salud. Una de las especificaciones que posee HL7 es el Servicio de Terminologías Comunes (CTS), el cual permite realizar consultas y acceder a contenido terminológico de un diccionario de terminológicas médicas. Este está compuesto por tres capas: Mensajería, Vocabulario y Mapeo.

A raíz del estudio realizado para el desarrollo de este servicio en el centro se ha identificado como **Situación Problemática**: que una vez implantado un servicio CTS resulta compleja la realización de cambios para su correcto funcionamiento. No se podrán incorporar nuevas terminologías, ni activar o desactivar un sistema de códigos de acuerdo a las necesidades, ni permitirá cambiar el lenguaje o idioma, entre otros.

INTRODUCCIÓN

Hasta el momento, para realizar dichos cambios se necesitaría la utilización de un Sistema Gestor de Base de Datos (SGBD) para actualizar y acceder a la fuente de información del servicio directamente, lo que puede traer dificultades como la introducción de datos incorrectos, sin permitir estar al tanto de quien realice dichos cambios. Por todo lo anterior se puede inferir que el CTS no cuenta con una herramienta capaz de facilitar la ejecución de las acciones previamente mencionadas, ni proporciona la seguridad requerida como parte del proceso de administración de los datos. Lo que influye tanto en el tiempo de durabilidad del servicio como en la usabilidad del mismo, no solo en instituciones nacionales sino también en otras de ámbito internacional.

Al identificar estos inconvenientes a la hora de hacer las configuraciones del CTS, se presenta el siguiente **Problema a Resolver:** ¿Cómo proporcionar al Servicio de Terminologías Comunes opciones de actualización y mantenimiento, sin riesgos para la integridad de la información?

Para el desarrollo de la investigación se establece como **Objeto de Estudio:** El proceso de integración de las aplicaciones desarrolladas en el Centro de Informática Médica. Enmarcando el **Campo de Acción:** En la especificación de HL7 para un Servicio de Terminologías Comunes.

Para la solución del problema se plantea como **Objetivo General:** Implementar un Sistema de Administración para el Servicio de Terminologías Comunes que permita su mantenimiento y actualización de manera segura. Para dar cumplimiento al objetivo planteado se proponen como **Tareas de la investigación:**

1. Realizar un análisis crítico y valorativo de los sistemas informáticos de Servicios Terminológicos existentes a nivel nacional e internacional, estableciendo similitudes con la investigación en curso.
2. Realizar análisis de las tendencias, técnicas, tecnologías, plataformas, librerías, metodologías y herramientas usadas en la actualidad.
3. Analizar los procesos de negocio asociados a la gestión de la información relacionada con el Sistema de Administración de CTS, logrando un modelo único como guía para la implementación del sistema.

4. Generar los artefactos correspondientes a los Flujos de Trabajo propuestos por la Metodología seleccionada, sirviendo de base a los desarrolladores.
5. Implementar el sistema informático siguiendo lo establecido en la Especificación de Requisitos de Software.
6. Realizar pruebas de caja negra al sistema para verificar su correcto funcionamiento.

Esperando obtener como **Resultado:** La implementación del Sistema de Administración para el proyecto Servidor de Diccionarios Terminológicos. Lo que propiciará un mayor tiempo de vida útil al Servicio de Terminología Común y un mayor nivel de efectividad en su accionar.

El sistema brindará entre sus **Beneficios:**

- ❖ Brindará la posibilidad de realizar el mantenimiento y actualización a las terminologías que vayan surgiendo.
- ❖ El sistema permite realizar salvadas de resguardo de la base de datos que proporcionan agilidad de despliegue e instalación en dominios similares y rápidas restauraciones del sistema en caso de fallos.
- ❖ Para el mantenimiento de la seguridad e integridad de los datos el sistema permite efectuar las operaciones solo bajo autenticaciones autorizadas, dejando a la vez registros de cada una de las operaciones realizadas que facilitan la identificación y corrección de errores cometidos.

El presente documento está constituido por cuatro capítulos, que contienen lo referente a la investigación hecha, el diseño e implementación del sistema en que se trabajó.

Capítulo 1. Fundamentación teórica: En este capítulo se hace un estudio del arte sobre los distintos servicios terminológicos que existen en el mundo hasta el momento y cómo fue que se comenzaron a utilizar, se especifican los conceptos importantes, además de exponer las tecnologías y herramientas que han sido usados para el desarrollo de este servicio.

Capítulo 2. Características del sistema: Representa la estructura del sistema, los procesos del dominio, representación del modelo del dominio del Servicio de Terminologías Comunes y se explica sobre cada

INTRODUCCIÓN

clase existente en el dominio. Además de presentar la propuesta del sistema, así como la descripción de algunos de los casos de uso con los que cuenta y la interfaz de los ya realizados.

Capítulo 3. Análisis y diseño del sistema: Para este capítulo se realiza el diseño de la arquitectura, de posibles implementaciones. Se confeccionan y se documentan los diagramas de clases de diseño y de interacción.

Capítulo 4. Implementación: Se exhibe el modelo de datos, el modelo de despliegue, que permitirá ver cuales recurso se necesitan para que funcione el sistema; y el diagrama de componentes que muestra una vista de cómo están distribuidos los componentes del sistema. Asimismo se describe el tipo de prueba que se le realizara al sistema para verificar su correcto funcionamiento.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

En el contenido de este capítulo se expondrán los conceptos afines con el problema a resolver, planteándose los antecedentes que posee un sistema de Administración de un Servicio de Terminologías Comunes y los servicios terminológicos en Cuba y a nivel mundial. Se describe además, las tecnologías requeridas para su desarrollo, así como los lenguajes y técnicas a emplear.

1.1 Conceptos fundamentales

1.1.1 Interoperabilidad: Característica de los ordenadores que les permite su interconexión y funcionamiento conjunto de manera compatible. Esto no siempre es posible, debido a los diferentes sistemas operativos y arquitecturas de cada sistema, pero los esfuerzos de estandarización están permitiendo que cada vez sean más los ordenadores capaces de interoperar entre sí. (4)

1.1.2 Administración: Conjunto de las funciones o procesos básicos que, realizados convenientemente, repercuten de forma positiva en la eficacia y eficiencia de la actividad realizada. (5)

1.1.3 Servidor: En informática, un servidor es un tipo de software que realiza ciertas tareas en nombre de los usuarios. El término servidor ahora también se utiliza para referirse al ordenador físico en el cual funciona ese software, una máquina cuyo propósito es proveer datos de modo que otras máquinas puedan utilizar esos datos. (6)

1.1.4 Terminología: La palabra terminología puede entenderse de diferentes maneras: en primer lugar, la terminología es el conjunto del vocabulario especial de una disciplina o un ámbito de conocimiento (la terminología de la química, del marketing, de la lingüística, etc.); en segundo lugar, la terminología puede entenderse como aquella disciplina, que tiene por objeto la construcción de una teoría de los términos, el estudio de los mismos y su recopilación y sistematización en glosarios especializados. (7)

1.1.5 Código: El término código tiene diferentes usos y significados. Se trata, por ejemplo, de una combinación de signos que, dentro de un sistema establecido, tiene un determinado valor. (8)

1.1.6 Servicio de Terminologías Comunes: El Servicio de Terminología Común descrito por HL7 (HL7 CTS): define una serie de Interfaces de Aplicación (API) y un modelo de datos que permiten integrar y acceder de manera uniforme a diferentes terminologías.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

Las APIs de CTS describen la funcionalidad básica necesaria para el acceso al contenido terminológico desde sistemas de información. Se presenta como una API con el objetivo de dar libertad a los desarrolladores sobre cómo almacenar la información terminológica. (9) Es una interfaz de programación (API) que describe las funcionalidades básicas para realizar consultas y acceder a contenido terminológico, que serán necesarias para la implementación de software que sean basados en HL7 versión 3.

1.2 Estado del Arte

1.2.1 Antecedentes a nivel internacional

Los sistemas de códigos han ido surgiendo en la medida en que ha aumentado la necesidad de tener agrupadas las terminologías con las que trabajan los profesionales, tanto en la salud como en otras ramas. Al ver la utilidad que presentaban estos diccionarios, los estándares que han sido creados, como el HL7, pensaron en la creación de especificaciones que permitieran un mejor manejo de los mismos. Siguiendo esta idea, realizaron la especificación Servicio de Terminologías Comunes versión 1 (CTS-1), que permite acceder a la información de los sistemas de códigos. Pero esta versión no toma en cuenta la realización de un sistema capaz de poder administrar esta información.

Actualmente en el mundo se han desarrollado algunas herramientas que se rigen por esta especificación, aunque no en su totalidad, las que no cuentan con un Sistema de Administración que facilite el manejo de los datos.

1.2.2 Antecedentes en Cuba

En Cuba, se han desarrollado varios sistemas para la salud, que utilizan de acuerdo al área al que pertenecen, las terminologías asociadas a cada una de estas áreas. No se encontraron experiencias anteriores de CTS y menos de administración para un servicio de este tipo. Hasta el momento solo se han tenido ideas de cómo llegar a obtener este tipo de servicio para poder utilizarlo en la medicina y otras ramas.

En el CESIM se desarrollan varios sistemas dirigidos a las diferentes ramas de la salud, los que hacen uso de sistemas de códigos, de acuerdo a su área. Está el caso del Clasificador Internacional de

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

Enfermedades (CIE) en sus versiones 9 y 10. El CIAP es otro sistema de código que contiene todas las terminologías referentes a la Clasificación Internacional de la Atención Primaria, y el ACR que abarca la codificación perteneciente al área de Radiología.

Cada sistema al utilizar los codificadores necesarios para su trabajo, no le facilita mantener una comunicación con los demás. Por lo que le resulta difícil que exista interoperabilidad entre ellos.

1.3 Softwares creados a nivel internacional

Itálica

En el año 1998 el Hospital Italiano de Buenos Aires (HIBA) decidió implementar el proyecto, llamado Itálica, el cual fue gestionado y desarrollado de forma interna y actualmente se utiliza para gestionar toda la información relacionada a la asistencia sanitaria tanto a nivel clínico como administrativo.

Este trabaja con Servidor de Terminología: compuesto por un vocabulario interfaz local (tesauro) asignada a un vocabulario de referencia, SNOMED CT. El tesauro es una lista de términos creados a partir de casi 2 millones de entradas de texto libre en el repositorio de datos clínicos. El objetivo fundamental del Servidor es funcionar en base a un vocabulario de interface que permite a los usuarios ingresar entidades clínicas eligiendo en una lista de términos familiares, pero que se almacenan como códigos de SNOMED CT.

Los usuarios pueden refinar términos, elegir alguna entidad en específico, y proponer nuevos términos o "descripciones" para mejorar la cobertura. El área de "Terminología Clínica" se encarga del mantenimiento de la terminología de interfaz, mapeos así como la expansión de los mismos a otras clasificaciones (CIE-9 CM, CIE-10 CM, ICPC-2). El Servidor de Terminología comenzó a utilizarse en el año 2006 con las Epicrisis de Hospitalización, para luego ser incorporado en todos los software clínicos que utilizan terminologías clínicas.

Este es un servidor que no se basa en un estándar internacional de medicina, por lo que no da la seguridad de que los proyectos se comuniquen de manera eficiente. Además no especifica un Sistema de

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

Administración que permita realizar cambios en su base de datos. Por tales razones no aporta datos suficientes para el desarrollo del sistema.

LOINC® (Logical Observation Identifies Names and Codes)

LOINC es una base de datos con nombres y códigos estandarizados para la identificación de resultados de laboratorio, observaciones clínicas y observaciones de estudios diagnósticos. (10) Este identifica de forma incuestionable una prueba de laboratorio que se haya realizado.

Se inicia en 1994 dando respuesta a la demanda de transferencia electrónica de datos clínicos entre laboratorios, hospitales y administraciones. Su parte más importante corresponde a observaciones de laboratorio, incluyendo además en su base de datos observaciones clínicas. Su propósito, es suministrar el intercambio y la integración de los resultados de la atención clínica, la gestión y la investigación.

HL7 lo escoge en el año 1999 como un código para la transacción de descripciones de pruebas de laboratorio entre los disímiles sistemas sanitarios. En la actualidad su uso se ha extendido por todo el planeta, observando una adopción progresiva por parte de otros países que han decidido manejar este estándar e incluso traducirlo a su idioma para implementarlo en proyectos de salud como la HCE (Historia clínica Electrónica.)

Loinc al ser un diccionario de terminologías de laboratorio, presenta la dificultad de ser solo para resultados de laboratorio, además de no tener una interfaz que permita su configuración y actualización.

Regenstrief LOINC Mapping Assistant (RELMA)

El RELMA fue elaborado por el Instituto Regenstrief como una utilidad de asignación basada en Windows, facilitando con esta las búsquedas a través de la base de datos LOINC y de esta manera ayudar en los esfuerzos de asignación de los códigos locales, de los códigos LOINC. Esta utilidad brinda ayuda a los usuarios asociados o mapa, sus términos locales al código universal LOINC.

Al facilitar solo búsquedas en la base de datos de LOINC no llega a cumplir todas las necesidades que están planteadas para el sistema en cuestión.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

Browser de CliniClue

Es una de las herramientas descargables vía web gratuitas más conocidas. Incluye una potente API que se pueden utilizar para proporcionar acceso mediante programación a SNOMED CT a partir de datos de diversas aplicaciones de Windows. (11) 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 escoja la distribución que le resulte más beneficiosa. Al permitir búsquedas, también permite la navegación de SNOMED mediante un visor en el que se puede, participativamente, saltar a través de sus términos siguiendo las relaciones que aplican a cada uno de los vocablos.

Este browser permite facilidades para el sistema, las que pueden llegar a utilizarse, pero no cumple con los objetivos planteados.

SNOB (SNOMED Browser)

Programa gratuito de búsqueda, solicita cargar la terminología SNOMED desde los archivos de una distribución oficial obtenida anteriormente. Además de tener sus funciones habituales, proporciona editar los conceptos y poder guardar los cambios en el mismo formato de la distribución oficial, convirtiéndolo en una herramienta atrayente para la traducción de términos o simplemente para crear extensiones locales de SNOMED.

El SNOB al solo cargar la terminología SNOMED no cumple los requisitos especificados para el sistema.

Herramienta LexBIG

Es un conjunto de servicios creados para almacenar y acceder a metadatos de terminologías. (9) Este aplica la tecnología diseñada en el modelo LexGRID. Tiene como objetivo, construir servidores de vocabulario que puedan ser accedidos mediante una API bien estructurada, capaz de acceder y distribuir recursos. El servidor se crea a partir de tecnologías estándares bien conocidas. (11)

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

La herramienta LexBig permite, a partir de la distribución de una terminología, generar una base de datos de tipo mySql, PostgreSQL, Oracle, HyperSonicSQL, etc. Cargada con el contenido terminológico y enlazarla para el uso de diferentes APIs cuyas funciones tiene ya implementadas el CTS de HL7 y la propia API de LexBIG. (11) Basado en el modelo de información de LexGRID para representar y compartir recursos terminológicos a gran escala, LexBIG se compone de una serie de programas para cargar, indexar, publicar y editar contenidos para un servidor de terminologías. (11)

LexBIG además de brindar varias funcionalidades, no posee un sistema específico para administrar, y utilizar herramientas asociadas, por tanto, no cumple con los objetivos trazados.

1.4 Tecnologías actuales empleadas

1.4.1 Metodologías de desarrollo

Las metodologías son importantes cuando se va a desarrollar cualquier aplicación pues ofrecen toda la información referida a los procesos, procedimientos y políticas que participan en el desarrollo del software. Brindan una gran utilidad, garantizando la validez con los requisitos puestos y la eficiencia al disminuir cualquier tipo de pérdida en el proceso de creación del software. De igual manera el garantizar un proceso de desarrollo con calidad influye directamente en la calidad del producto final. Es por esto que se tienen en cuenta dos pilares en cuanto a guía de desarrollo de software se refiere:

❖ Metodología: Rational Unified Process (RUP)

Proceso para el desarrollo de un proyecto de un software con lo que se define quién, cómo, cuándo y qué debe hacerse en el proyecto.

La metodología RUP sirve de guía a los equipos de proyecto en cómo administrar el desarrollo iterativo. Define un manajo entero de las actividades y de los artefactos que usted necesita elegir para construir sus propios procesos. Los procesos de RUP estiman tareas y horario del plan midiendo la velocidad de iteraciones concerniente a sus estimaciones originales. La ventaja principal de RUP es que se basa todo en las mejores prácticas que se han intentado y se han probado en el campo. (12)

Se caracteriza por tener 4 fases de desarrollo de software (Inicio, Elaboración, Construcción y Trasmisión), 6 disciplinas o flujos de ingeniería (Ingeniería de Negocios, Requerimientos, Análisis y

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

Diseño, Implementación, Pruebas) y 3 flujos de apoyo (Configuración y administración del cambio, Administración de proyecto y Ambiente).

❖ **Modelo CMMI**

CMMI (Capability Maturity Model Integration), su significado va encaminado a la capacidad que tiene una empresa para desarrollar software. Es un conjunto de modelos elaborados por el SEI, que permiten adquirir un diagnóstico preciso de la madurez de los procesos relacionados con las tecnologías de la información de una organización, y describen las tareas que se tienen que llevar a cabo para mejorar esos procesos. Posee dos enfoques, Representación Escalonada, que cuenta con 5 niveles de madurez en donde están distribuidas 22 áreas de procesos, y otro enfoque, Representación Continua, en donde se establecen 6 niveles de capacidad también con 22 áreas de procesos. Las áreas que define CMMI, en donde se llevan a cabo las prácticas específicas o genéricas, presenta el hecho de que al trabajar con la metodología RUP para el desarrollo de un proyecto, implica que algunas de estas áreas sean alcanzadas y otras no.

1.4.2 Tecnologías de desarrollo

1.4.2.1 Servidor de Aplicaciones

Un servidor de aplicaciones es un software que proporciona aplicaciones a los equipos o dispositivos cliente, por lo general a través de Internet y utilizando el protocolo http. Los servidores de aplicación se distinguen de los servidores web por el uso extensivo del contenido dinámico y por su frecuente integración con bases de datos. (13)

❖ **JBoss Server o JBoss AS (14)**

JBoss es un servidor de aplicaciones J2EE de código abierto implementado solo en Java. Manejo de múltiples tipos de archivos y extensiones como .xml, .war.

Se puede utilizar en cualquier sistema operativo que soporta Java para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java. Es ideal para aplicaciones Java y aplicaciones basadas en la web. Lo cual lo convierte en el servidor de aplicación necesario para el desarrollo del software.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.4.2.2 Framework

FrameWork es un concepto sumamente genérico, se refiere a “ambiente de trabajo, y ejecución”. En general los framework son soluciones completas que contemplan herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución). (15) Facilita el desarrollo de software, dejando que los desarrolladores como los diseñadores se concentren más en identificar los requerimientos del software que con la programación de la aplicación. Trabaja con la programación en tres capas Modelo-Vista-Controlador e intenta manipular herramientas avanzadas.

❖ JBoss Seam

Es un framework que integra y unifica los distintos estándares de la plataforma Java EE 5.0, pudiendo trabajar con todos ellos siguiendo el mismo modelo de programación. (16) Tiene como característica importante, que puedes hacer validaciones en los POJOs (Plain Old Java Objects) como además manejar directamente la lógica de la aplicación y de negocios desde sus sessions beans.

JBoss Seam es un framework que integra la capa de presentación (JSF) con la capa de negocios y persistencia (EJB). Puede integrarse además con otros framework, como MyFaces, Hibernate.

1.4.2.3 Mapeo Objeto-Relacional (Object Relational Mapping)

Es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional. (17)

Al utilizar el ORM, permite rapidez en el desarrollo, permitiendo utilizar los métodos de un objeto de datos desde distintas zonas de la aplicación, incluso desde aplicaciones distintas.

❖ Hibernate

Es una herramienta de mapeo objeto relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML). Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Hibernate ofrece también un lenguaje de consulta de

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

datos llamado HQL (Hibernate Query Language), al mismo tiempo que una API para construir las consultas programáticamente. (18)

1.4.2.4 Máquina virtual

❖ Máquina Virtual Java (Java Virtual Machine: JVM) (19)

La máquina virtual de Java constituye el núcleo del lenguaje de programación, es el entorno donde se ejecutan los programas java. Cuenta con características tales como: la portabilidad, eficiencia y seguridad. Cumple con reservar espacio en memoria para los objetos creados, además libera la memoria no usada, y acude al sistema huésped para ciertas funciones que permiten la ejecución de las instrucciones, y la realización de las demás funcionalidades que brinda.

1.4.2.5 Librerías

Forman todos los procedimientos y funciones agrupados en un archivo con la finalidad de que se empleen por otros programas.

❖ RichFaces (20)

Es una librería de componentes visuales para JSF. Además, posee un framework avanzado para la integración de funcionalidades Ajax en dichos componentes visuales, mediante el soporte de la librería Ajax4JSF. Se integra perfectamente en el ciclo de vida de JSF, contiene un set de componentes visuales, los más comunes para el desarrollo de una aplicación web rica (Rich Internet Application), con un número bastante amplio que cubren casi todas nuestras necesidades. Soporta css themes o skins, y es un proyecto open source, activo y con una comunidad también activa.

1.4.3 Sistema Gestor de Base de Datos (SGBD)

Un Sistema Gestor de base de datos (SGBD) es un conjunto de programas que permiten crear y mantener una Base de datos, asegurando su integridad, confidencialidad y seguridad.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

❖ Sistema Gestor de Base de Datos: PostgreSQL 8.4

Esta versión domina una gran cantidad de mejoras para que la administración, consulta y programación en PostgreSQL sea más fácil que nunca.

Características de PostgreSQL 8.4 (21)

Es un sistema de base de datos relacional perteneciente al ámbito del software libre, se destaca por tener robustez, escalabilidad y cumplimiento de los estándares SQL. Esta versión muestra un enfoque orientado a la adición de características (por ejemplo, la autenticación, el control, la reutilización del espacio), y agrega las capacidades definidas en las normas de SQL posteriores. Concibe el análisis de datos mucho más sencillo a través de funcionalidades avanzadas de ANSI SQL: 2003. Estas estructuras de consulta aumentan sustancialmente la expresividad del dialecto SQL de PostgreSQL, permitiendo a los usuarios hacer preguntas interesantes en una sola consulta.

Presenta entre sus características:

- Funciones 'Windows'.
- Expresiones comunes de tabla.
- Restauración de backups en paralelo.
- Permisos para usuarios a nivel de columnas.
- Configuración de 'locales' per base de dato.
- Asignación automática de espacio para el 'Free Space Map'.
- Uso de certificados SSL para autenticar a usuarios.
- Editado fácil de funciones en psql. (22)

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.4.4 Herramientas

1.4.4.1 Herramienta de Modelado

❖ Enterprise Architect 7.0

Enterprise Architect combina el poder de la última especificación UML 2.1 con alto rendimiento, interfaz intuitiva, para traer modelado avanzado al escritorio, y para el equipo completo de desarrollo e implementación. (23) La misma es comprensible de diseño y análisis UML, abarcando el desarrollo de software desde el paso de los requerimientos a través de las fases del análisis, modelos de diseño, pruebas y mantenimiento. Es multi-usuario, basada en Windows, hecha para ayudar a construir software robusto y fácil de mantener. Ofrece la salida de documentación flexible y de alta calidad.

1.4.4.2 IDE de Desarrollo (Entorno de Desarrollo Integrado)

❖ Eclipse

Eclipse es una plataforma de desarrollo open source basada en Java. Es un desarrollo de IBM cuyo código fuente fue puesto a disposición de los usuarios. En sí mismo Eclipse es un marco y un conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados (plug-in).

Hay plug-ins para el desarrollo de Java (JDT Java Development Tools) así como para el desarrollo en C/C++, COBOL. (24)

1.4.5 Lenguajes

1.4.5.1 Lenguaje de programación

❖ Java

Es una plataforma virtual de software desarrollada por Sun Microsystems, de tal manera que los programas creados en ella puedan ejecutarse sin cambios en diferentes tipos de arquitecturas y dispositivos computacionales. (25)

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

Al contener la característica de ser multiplataforma, le posibilita funcionar en diversos tipos de ordenadores y sistemas operativos así como en otros dispositivos inteligentes. Las aplicaciones que se crean con ella son independientes, es "orientado a objetos", lo que facilita que los programas se construyan a partir de módulos independientes, y que su concepción se aproxime al pensamiento humano.

Características:

❖ Lenguaje Simple:

Se lo conoce como lenguaje simple porque viene de la misma estructura de c y c++; ya que c++ fue un referente para la creación de java por eso utiliza determinadas características de c++ y se han eliminado otras.

❖ Orientado a Objeto:

Toda la programación en java en su mayoría está orientada a objeto, ya que al estar agrupados en estructuras en estructuras encapsuladas es más fácil su manipulación.

❖ Distribuido:

Permite abrir sockets, establecer y aceptar conexiones con los servidores o clientes remotos; facilita la creación de aplicaciones distribuidas ya que proporciona una colección de clases para aplicaciones en red.

❖ Robusto:

Es altamente fiable en comparación con c, se han eliminado muchas características con la aritmética de punteros, proporciona numerosas comprobaciones en compilación y en tiempo de ejecución.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

❖ Seguro:

La seguridad es una característica muy importante en java ya que se han implementado barreras de seguridad en el lenguaje y en el sistema de ejecución de tiempo real.

❖ Alto rendimiento

Java es considerado de alto rendimiento por ser tan veloz en el momento de correr los programas y por ahorrarse muchas líneas de código. (26)

1.4.5.2 Lenguaje de Modelado

❖ Lenguaje UML

El lenguaje UML, al ser un lenguaje proporciona un vocabulario y reglas que permiten la comunicación. Posee funciones como la visualización y la especificación, entre otras. Tiene una notación gráfica muy expresiva que permite representar en mayor o menor medida todas las fases de un proyecto informático: desde el análisis con los casos de uso, el diseño con los diagramas de clases, objetos, etc., hasta la implementación y configuración con los diagramas de despliegue.

UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten. (27) AL usar este lenguaje de modelado se pueda implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos). Es un lenguaje que permite visualizar, construir, especificar y documentar.

En el presente capitulo se realizó un análisis crítico y valorativo de los sistemas informáticos de Servicios Terminológicos existentes a nivel nacional e internacional. Se describieron varios conceptos fundamentales que permiten un mejor entendimiento del problema a resolver a través del uso de diferentes metodologías y tecnologías requeridas para su desarrollo. Así como los lenguajes, técnicas y herramientas usadas en la actualidad.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

En este capítulo se exponen los fundamentales conceptos asociados al dominio actual del sistema a desarrollar, donde se describe cada clase del mismo, dando a conocer porque se utiliza cada una de ellas. Se plantea además, la propuesta del sistema, los requisitos funcionales y no funcionales con los que cuenta este, la especificación de los casos de uso del sistema, en donde se representan los casos de uso más significativos.

2.1 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 construcción del modelo del dominio, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de software sino de la propia realidad física. (28) Este modelo puede utilizarse para capturar y expresar el entendimiento sobre un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo. Es utilizado por el analista como medio para comprender el sector hacia el cual va dirigido el sistema.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

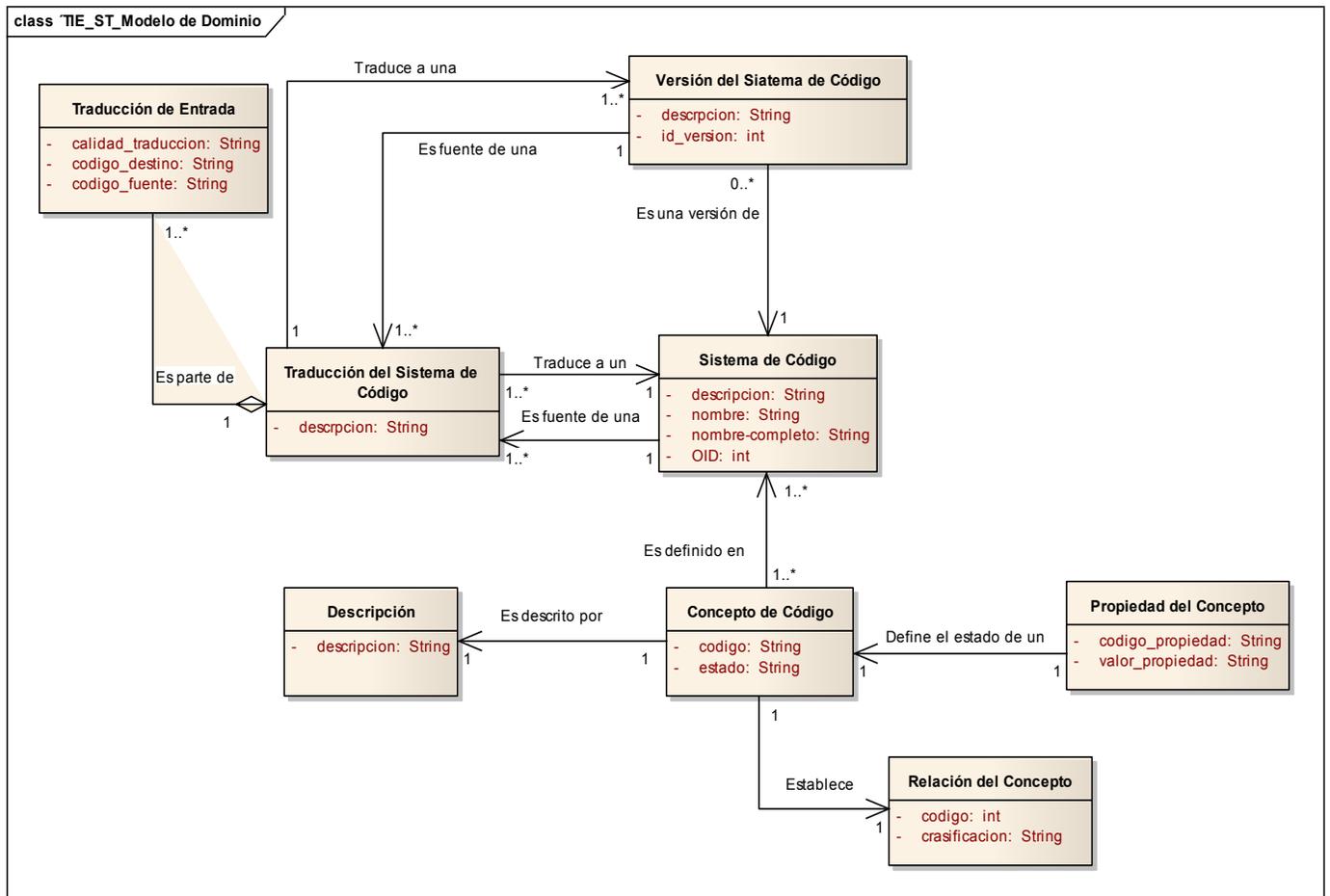


Fig. 1 Modelo de Dominio

2.2 Definición de las clases del modelo del dominio

2.2.1 Descripción de la clase Sistema de Código

La clase define una entidad de Sistema de Código que contiene de manera general la relación entre los conceptos sus descripciones y el código que los representa.

Atributos:

OID: es un identificador único para el sistema de código definido por la ISO.

Nombre: es el nombre asignado al sistema de código.

Nombre-completo: es el nombre completo del sistema de código.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

Descripción: es una descripción del contenido del sistema de código.

2.2.2 Descripción de la clase Concepto de Código

La clase define cada concepto del sistema de código. Los cuales son identificadores numéricos único, asignados a significados clínicos.

Atributos:

Código: nombre del código asignado.

Estado: representa el estado actual del concepto dentro del sistema de código.

2.2.3 Descripción de la clase Versión de Sistema de Código

La clase Versión de Sistema de Código define cada versión que se le realice a un sistema de código.

Atributos:

Descripción: es una descripción de la versión del sistema de código.

Id_versión: es un identificador único para cada versión de un sistema de código.

2.2.4 Descripción de la clase Traducción del Sistema de Código

La clase Traducción del sistema de código es la que define la traducción de un sistema de código a otro.

Atributos:

Descripción: es la descripción definida para la traducción.

2.2.5 Descripción de la clase Traducción de Entrada

La clase Traducción de Entrada representa una traducción del código de tipo conceptual desde el código fuente al código destino.

Atributos:

Calidad_traducción: es la calidad del código fuente al código destino.

Código_destino: es el código que se obtiene de la traducción.

Código_fuente: es el código que se va a traducir.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

2.2.6 Descripción de la clase Propiedad del Concepto

La clase Propiedad del Concepto es un atributo o característica que representa o define el significado de los conceptos.

Atributos:

Código_propiedad: es una combinación del id del sistema de código y el concepto que identifica el tipo de propiedad.

Valor_propiedad: es el valor textual de la propiedad asociada.

2.2.7 Descripción de la clase Relación de Concepto

La clase Relación de Concepto representa las relaciones binarias sobre el conjunto de conceptos definido en un sistema de código.

Atributos:

Código: es el que identifica de qué tipo de relación se trata.

Clasificación: es la clasificación definida para cada relación.

2.2.8 Descripción de la clase Descripción

La clase Descripción es una cadena de texto que proporciona una representación externa de un concepto de código.

Atributos:

Descripción: describe un concepto de código.

2.3 Objeto de Automatización

❖ Descripción del objeto de estudio

El objeto de estudio para la investigación es el proceso de integración de las aplicaciones desarrolladas en el Centro de Informática Médica, el que se describirá a continuación. Las aplicaciones que se desarrollan en el CESIM de la Universidad de las Ciencias Informáticas presentan como características el difícil entendimiento entre ellas, pues trabajan con sistemas de código distintos en su mayoría.

Para que las aplicaciones existentes en el centro y las futuras puedan integrarse favorablemente debe constar con un Servicio de Terminologías Comunes, que cuente con un sistema de Administración capaz

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

de proporcionar a los usuarios la realización de cambios y consultas a la información que posea este servicio.

2.4 Propuesta del Sistema

El objetivo de un Servicio de Terminologías Comunes es mejorar la comunicación y el funcionamiento conjunto de los sistemas de información para la salud, haciendo uso de los términos de salud. Para el cumplimiento de este objetivo el servicio se compone por una capa de Mensajería que permite una variedad de aplicaciones de procesado de mensajes, creación, validación y traducción de datos clínicos de manera consistente. Además sirve de puente entre estas aplicaciones y la capa de funciones específicas de la terminología. Otra de las capas que posee es la de Vocabulario, esta se ha diseñado para ser genérica y fácilmente utilizable en diferentes entornos, trabaja sobre el supuesto de que existe una base de datos que contiene al menos una terminología o sistema de codificación. Por último, cuenta con una capa de Mapeo, la que se encarga de realizar las traducciones de los conceptos de un sistema de código a otro.

Estas tres capas al conformar el CTS, tienen como relación: que la Capa de Mensajería se comunica con los Sistemas que van a consumir este servicio y con la Capa de Vocabulario, a la que le envía el mensaje procesado. Vocabulario se comunica con la Capa de Mapeo enviándole el mensaje después de modificado. Mapeo es el encargado de buscar la información y enviarla hacia atrás, y se produce el proceso a la inversa.

Para que este servicio tenga una mayor durabilidad se propone implementar un Sistema de Administración, que permita configurar la fuente de información del CTS. Tendrá acceso a él dos tipos de usuario, el administrador y el usuario. Este sistema consistirá en una interfaz gráfica de usuario que permitirá incorporar un nuevo sistema de códigos con toda su información, obteniendo la misma desde un fichero. Además, permitirá especificar las traducciones que pueden hacerse entre los sistemas de códigos.

Para insertar nuevas terminologías será necesario especificar a qué sistema de código pertenecen. Los sistemas de código son definidos en más de un idioma, de ellos uno será el preferido; por lo que existirá una opción con la que se podrá cambiar el lenguaje de acuerdo al dominio donde se encuentre.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

Asimismo si no es necesaria la utilización de uno de estos diccionarios, este no será eliminado, sino que se pasará su estado a inactivo, esto es posible pues existe un nomenclador el cual contiene los estados a los que se puede cambiar. Este sistema al manejar información sensible, el usuario no tendrá permisos para hacer alguna modificación en los datos, solo realizará cualquier tipo de consulta. En el caso que no muestre la información esperada por el usuario, el administrador será el encargado de revisar si es un error del sistema o si los datos introducidos no son los correctos.

2.5 Especificación de los Requisitos del Sistema

Los requerimientos son una descripción de las necesidades o deseos de un producto. La meta principal en esta etapa es identificar y documentar lo que en realidad se necesita, en una forma en que pueda fácilmente ser transmitido al cliente y al equipo de desarrollo. (29)

2.5.1 Requisitos Funcionales

A continuación se listan los requerimientos funcionales del sistema propuesto:

RF_1: Adicionar un Code System

RF_2: Modificar Code System

RF_3: Eliminar Code System

RF_4: Listar Code System

RF_5: Mostrar detalles de Code System

RF_6: Buscar CSystem

RF_7: Crear Code System Translation

RF_8: Listar Code System Translation

RF_9: Eliminar Code System Translation

RF_10: Mostrar detalles Code System Translation

RF_11: Buscar CTranslat

RF_12: Modificar Code System Translation

RF_13: Adicionar Concepto

RF_14: Modificar concepto

RF_15: Listar concepto

RF_16: Mostrar detalles del concepto

RF_17: Buscar Concepto

RF_18: Adicionar Concepto de Dominio

RF_19: Modificar concepto de dominio

RF_20: Listar concepto de dominio

RF_21: Buscar CDom

RF_22: Adicionar Language Code

RF_23: Modificar Language Code

RF_24: Eliminar Language Code

RF_25: Mostrar detalles del Language Code

RF_26: Buscar Lcode

RF_27: Adicionar Concept Status

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

RF_28: Modificar Concept Status	RF_56: Listar Usuario
RF_29: Eliminar Concept Status	RF_57: Buscar Usuario
RF_30: Mostrar detalles del Concept Status	RF_58: Adicionar Func
RF_31: Buscar CStatus	RF_59: Modificar Func
RF_32: Adicionar Property Code	RF_60: Eliminar Func
RF_33: Modificar Property Code	RF_61: Listar Func
RF_34: Mostrar detalles de Property Code	RF_62: Exportar la Base de Datos
RF_35: Eliminar Property Code	RF_63: Mostrar el Registro de las Operaciones
RF_36: Buscar Pcode	RF_64: Adicionar Translation Quality
RF_37: Adicionar Relationship Code	RF_65: Modificar Translation Quality
RF_38: Modificar Relationship Code	RF_66: Eliminar Translation Quality
RF_39: Eliminar Relationship Code	RF_67: Mostrar detalles de Translation Quality
RF_40: Mostrar detalles de Relationship Code	RF_68: Buscar Translation Quality
RF_41: Buscar Relac Cod	RF_69: Adicionar Value Set
RF_42: Adicionar Mime Types	RF_70: Modificar_Value_Set
RF_43: Modificar Mime Types	RF_71: Eliminar_Value_Set
RF_44: Eliminar Mime Types	RF_72: Mostrar_Detalles_Value_Set
RF_45: Mostrar detalles Mime Types	RF_73: Buscar_Value_Set
RF_46: Buscar MTypes	RF_74: Adicionar_Aplicacion_Context
RF_47: Adicionar Relationship Quialifiers	RF_75: Modificar_Aplicacion_Context
RF_48: Modificar Relationship Quialifiers	RF_76: Eliminar_Aplicacion_Context
RF_49: Eliminar Relationship Quialifiers	RF_77: Mostrar_Detalles_Aplicacion_Context
RF_50: Mostrar detalles Relationship Quialifiers	RF_78: Buscar_Aplicacion_Context
RF_51: Buscar RQuialifiers	RF_79: Adicionar_Vocabulary_Domain
RF_52: Adicionar Usuario	RF_80: Modificar_Vcabulary_Domain
RF_53: Modificar Usuario	RF_79: Eliminar_Vocabulary_Domain
RF_54: Mostrar detalles de Usuario	RF_81: Mostrar_Detalles_Vocabulary_Domain
RF_55: Eliminar Usuario	RF_82: Buscar_Vocabulary_Domain

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

2.5.2 Requisitos No Funcionales

Se llama requisito no funcional a todas las exigencias de cualidades que se imponen al proyecto: exigencias de usar un cierto lenguaje de programación o plataforma tecnológica, por ejemplo. Un requisito no funcional es una característica ya sea del sistema, del proyecto o del servicio de soporte, que nos es requerida junto con la especificación del sistema pero que como ya dije, no se satisface añadiendo código, sino cumpliendo con esta como si de una restricción se tratara. (30)

Usabilidad

RNF1 Debe garantizar el acceso fácil, garantizando que pueda ser usado por usuarios con pocos conocimientos informáticos.

RNF2 Solo podrá utilizarse por los usuarios definidos para trabajar en el sistema.

Fiabilidad

RNF3 Al concluir el sistema debe permitir que se le realice mantenimiento, que se brinde capacitación al personal que trabajará con él.

Disponibilidad: Cuenta con la autenticación para acceder a los servicios que brinda.

Integridad: La información que se encuentra no será divulgada por personal sin autorización.

Confidencialidad: Los datos que se manejan en la aplicación serán protegidos contra personal no autorizado.

Eficiencia

RNF4 El sistema deberá responder en un tiempo de 10 a 20 seg las solicitudes que hagan los usuarios, para ello, el procesamiento de la información deberá ser en el menor tiempo posible.

La eficiencia dependerá de la rapidez de las consultas que se hagan a la base de datos, y el buen aprovechamiento de los recursos disponibles.

Soporte

RNF5 Se podrá trabajar con él desde cualquier sistema operativo.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

RNF6 El sistema será documentado correctamente de manera que se le pueda dar mantenimiento en el menor tiempo posible.

Restricciones de diseño

RNF7 Utilizar las pautas del diseño creadas por el CESIM para la modelación.

RNF8 Para el correcto funcionamiento del sistema es necesario la utilización de una computadora Pentium III u otra más avanzada, un disco duro de más de 40 GB, la memoria RAM de 512 MB o más y la tarjeta de red de 128 Mbps.

RNF9 Se utilizarán patrones para la implementación descritos en el documento Arquitectura de Software del CESIM.

RNF10 Para el diseño y el análisis del sistema se utilizarán la metodología RUP, usando el UML como lenguaje de modelado, para modelar la herramienta Enterprise Architect.

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

RNF11 Se documentará la información referente al sistema, estará dirigida a los tipos de usuario que trabajarán en ella.

Interfaz

RNF12 Deben ser visibles las opciones en el menú principal.

RNF13 Tendrá colores agradables a la vista del usuario, fácil de manejar y una buena organización en el contenido.

RNF14 Debe permitir una correcta interpretación de la información que se exponga.

RNF15 Los errores deben ser visibles para el usuario, al igual que envíe mensajes o avisos entendibles.

Interfaces de Comunicación

RNF16 Debe contar con una tarjeta de red para la conectividad de la PC cliente con el servidor.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

Interfaz Hardware

El equipo en el que se instale el software servidor web, será el servidor de aplicaciones y atenderá las peticiones de los usuarios. El equipo en el que se instale el software gestor de bases de datos será el servidor de Base de Datos.

RNF17 Deben tener los siguientes requerimientos de hardware:

Tipo de procesador: Intel Pentium III

Velocidad del procesador: 2.00 GHz

Memoria RAM: 512 MB o superior.

Disco Duro: 40 GB para servidor de aplicación y 160 GB para el de base de datos.

Se requiere tarjeta de red.

RNF18 Los ordenadores que serán utilizados por los usuarios del sistema para acceder a la aplicación y operar la misma deben tener los siguientes requerimientos de hardware:

Tipo de procesador: Intel Pentium III o superior.

Velocidad del procesador: 512 MHz o superior.

Memoria RAM: 256 MB o superior.

Disco Duro: 40 GB.

Interfaz Software

RNF19 Se requiere de un navegador para levantar el sistema.

2.6 Especificación de Casos de Uso

Para su mejor entendimiento el diagrama de casos de uso del sistema ha sido separado por paquetes ordenando por dos criterios, primeramente por tipo de actor y por tipo de información, buscando un orden lógico para su creación.

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

2.6.1 Diagrama de Casos de Uso del sistema

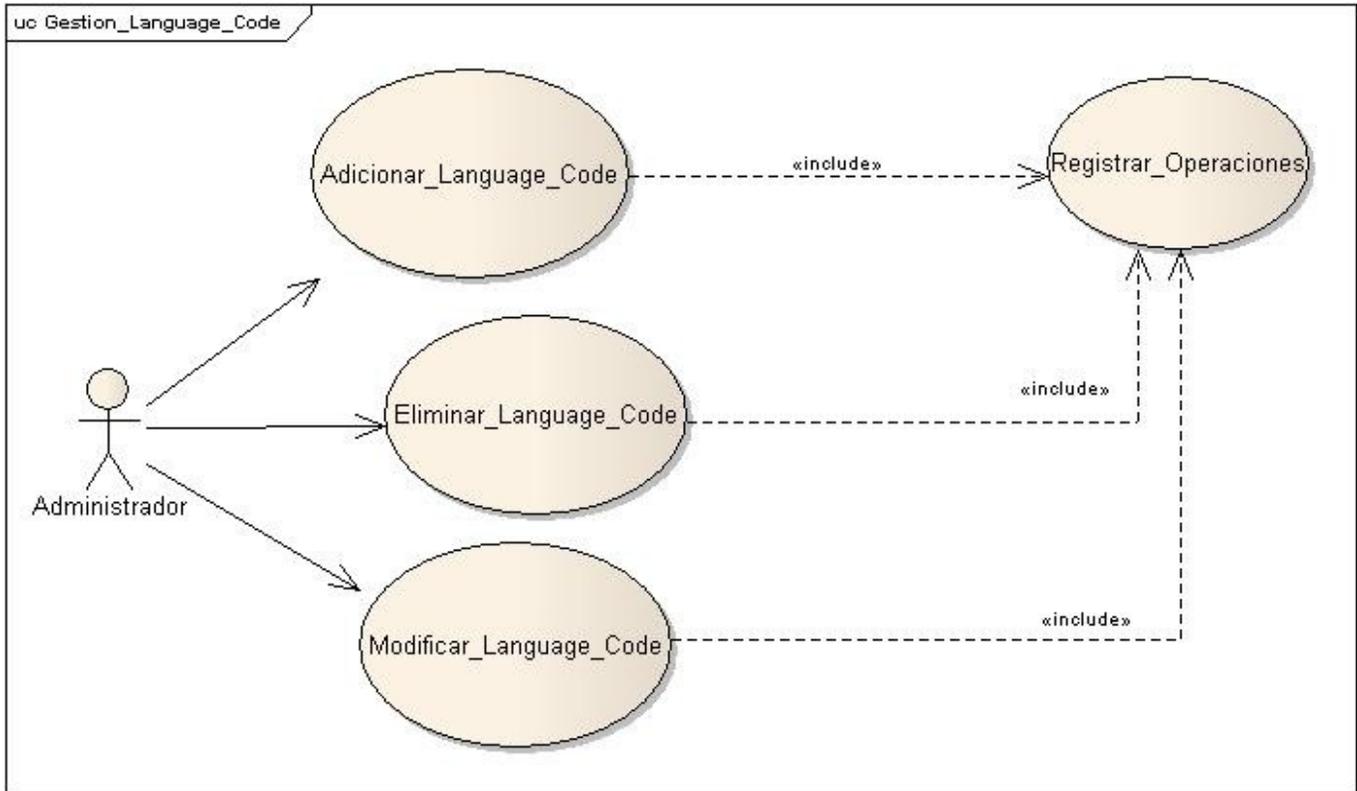


Fig.2 Paquete: Gestion_Language_Code

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

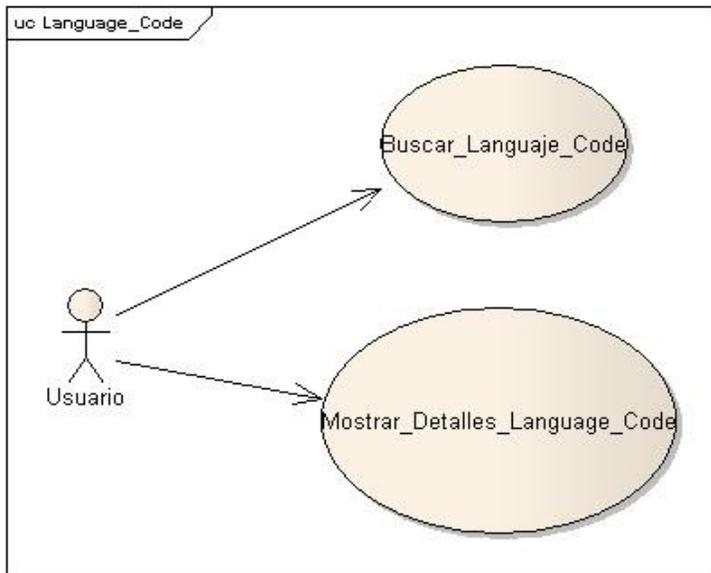


Fig.3 Paquete: Language_Code

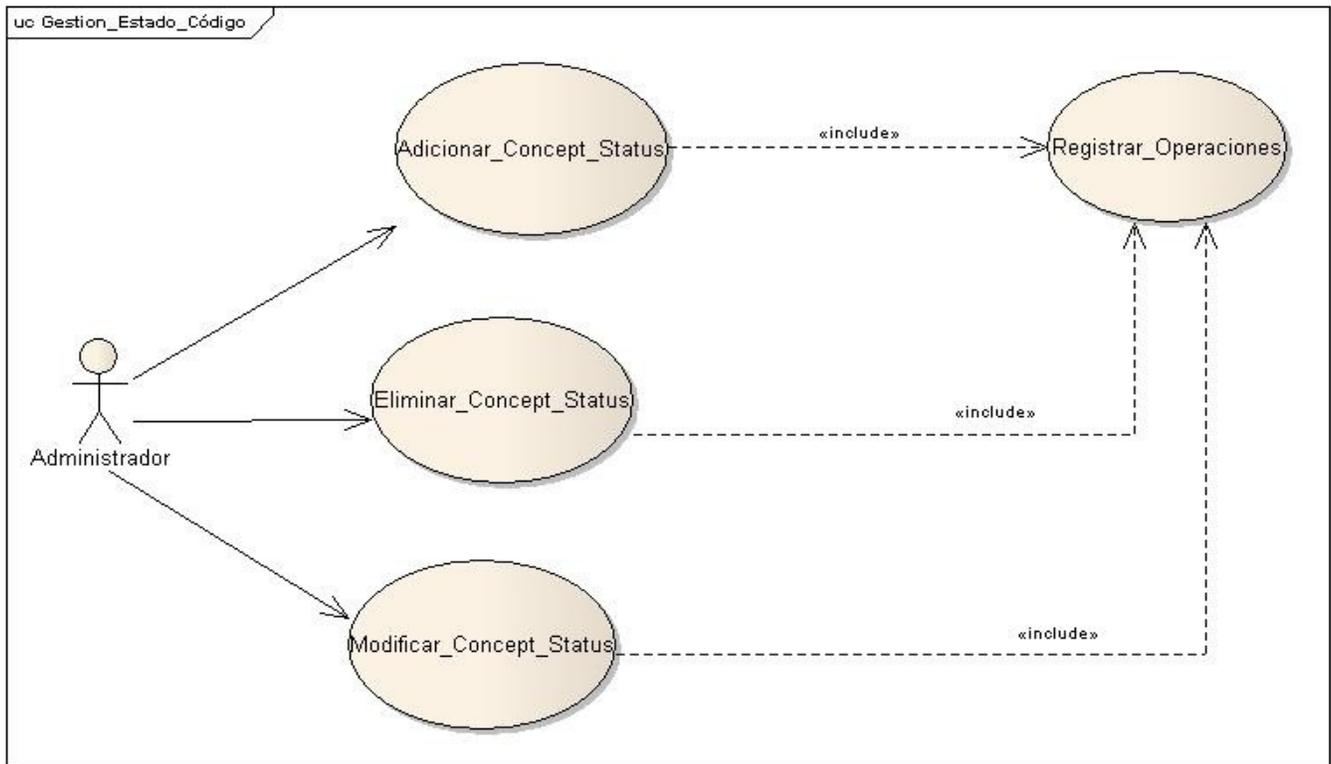


Fig.4 Paquete: Gestion Concept Status

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

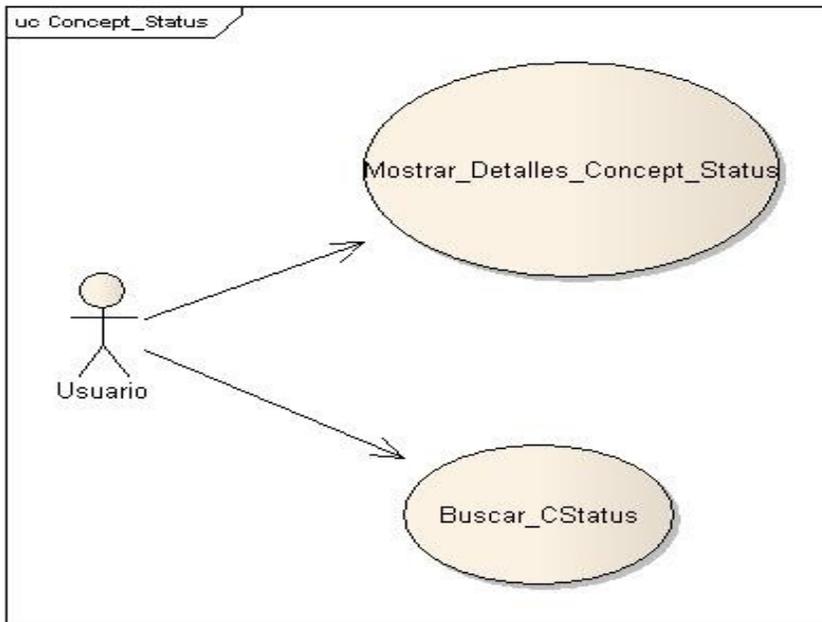


Fig.5 Paquete: Concept Status

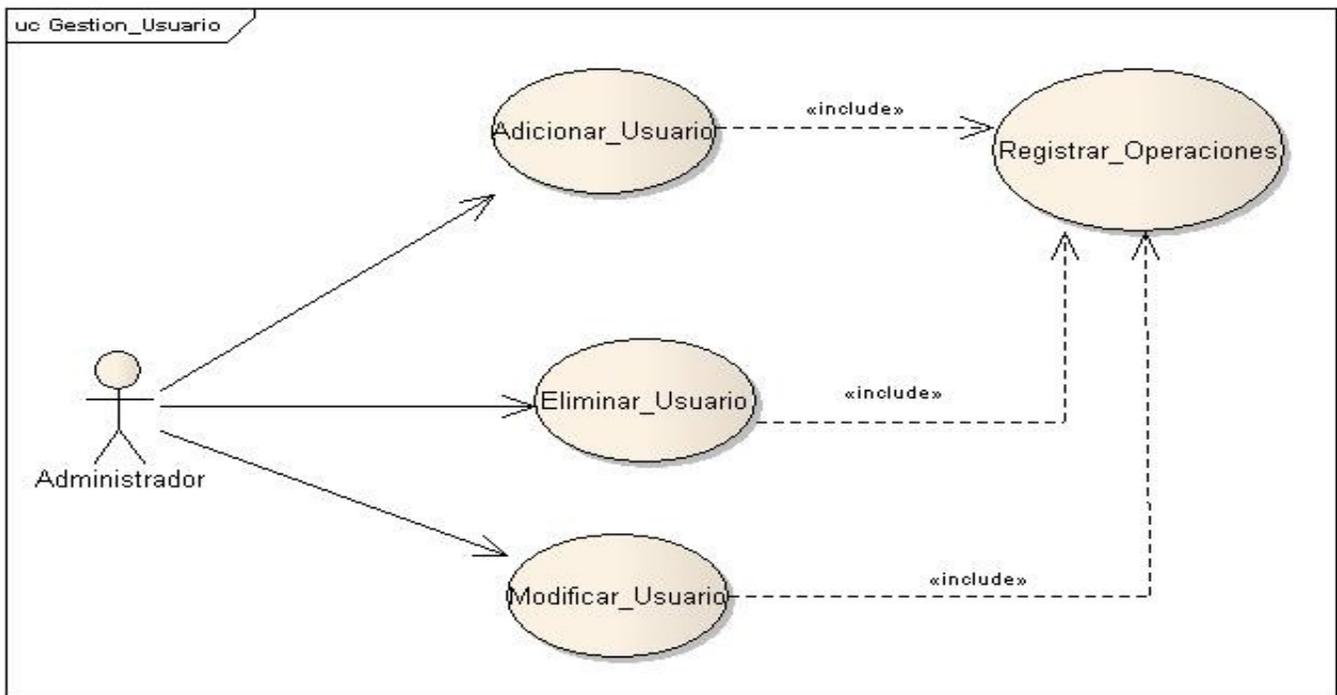


Fig.6 Paquete Gestión_Usuario

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

El diagrama cuenta con otros paquetes dentro de los cuales están los casos de usos no tan significativos, como es el caso de: exportar la base de datos; mostrar registro de las operaciones; adicionar, modificar, eliminar, buscar el estado del código. También existen casos de uso para adicionar, eliminar y modificar el lenguaje de un sistema de código. Asimismo están presentes los casos de uso para la relaciones de los sistemas de código, las propiedades del código, entre otros.

2.6.2 Descripción de Casos de Uso

CU-24	Mostrar_Detalles_ Language_Code
Actor	Usuario, Administrador
Descripción	El CU inicia cuando el actor selecciona la opción de mostrar detalles del lenguaje en el menú principal.
Referencia	RF_24

Tabla 2.1 Descripción del Caso de uso: **Mostrar_Detalles_ Language_Code**



Fig. 7 Mostrar Detalles del Lenguaje

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

CU-24	Eliminar_Language_Code
Actor	Administrador
Descripción	El CU inicia cuando el actor selecciona la opción de eliminar lenguaje del menú principal.
Referencia	RF_24

Tabla 2.2 Descripción del Caso de uso: Eliminar_Language_Code



Fig. 8 Eliminar Lenguaje

CU-26	Adicionar_Concept_Status
Actor	Administrador
Descripción	El CU inicia cuando el actor selecciona la opción de adicionar un estado de concepto del menú principal
Referencia	RF_26

Tabla 2.3 Descripción del Caso de uso: Adicionar_Concept_Status

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA



Fig.9 Adicionar Estado

CU-27	Modificar_Concept_Status
Actor	Administrador
Descripción	El CU inicia cuando el actor selecciona la opción modificar un estado de concepto en el menú principal.
Referencia	RF_27

Tabla 2.4 Descripción del Caso de uso: Modificar_Concept_Status



Fig.10 Modificar Estado

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

CU-51	Adicionar_Usuario
Actor	Administrador
Descripción	El CU inicia cuando el actor selecciona la opción de adicionar un usuario en el menú principal.
Referencia	RF_51

Tabla 2.5 Descripción del Caso de uso: Adicionar_Usuario

The screenshot shows the 'Crear Usuario' (Create User) form in the alasCTS system. The interface is in Spanish and features a green header with the system logo and user information (Ruben Miranda, CESIM, Habana, Cuba). A navigation menu on the left lists various system functions. The main form area is divided into sections: 'Datos de usuario' (User Data) with fields for 'Usuario', 'Contraseña', and 'Repetir contraseña', and a 'Cuenta habilitada' checkbox; 'Foto del usuario' (User Photo) with a 'Foto' field and an 'Examinar...' button; and 'Datos generales' (General Data) with fields for 'Nombre', 'Primer apellido', 'Segundo apellido', 'Carnet identidad', 'Fecha de nacimiento', and 'Dirección particular'. At the bottom right, there are 'Aceptar' and 'Cancelar' buttons. A copyright notice at the bottom center reads '© Universidad de las Ciencias Informáticas, 2011'.

Fig. 11 Adicionar Usuario

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

CU-52	Modificar_Usuario
Actor	Administrador
Descripción	El CU inicia cuando el actor selecciona la opción de modificar usuario del menú principal, para cambiarle los datos que contenga.
Referencia	RF_52

Tabla 2.6 Descripción del Caso de uso: Modificar_Usuario

The screenshot shows a web application interface for 'alascTS' (SERVICIOS DE TERMINOLOGÍAS COMUNES). The user is logged in as 'Ruben Miranda' from 'CESIM, Habana, Cuba'. The main navigation menu includes 'Inicio', 'Administración', 'Ayuda', and 'Salir'. A left sidebar menu lists various management options like 'Gestionar Property Code', 'Gestionar TranslationQuality', and 'Gestionar Usuarios'. The main content area is titled 'Modificar Usuario' and contains several sections: 'Datos de usuario' with fields for 'Usuario' (containing 'k') and 'Cuenta habilitada' (checkbox); 'Foto del usuario' with a 'Foto:' field and an 'Examinar...' button; and 'Datos generales' with fields for 'Nombre' (v), 'Primer apellido' (x), 'Segundo apellido' (x), 'Carnet identidad' (12345678909), 'Fecha de nacimiento' (01/02/2011), and 'Dirección particular' (x). 'Aceptar' and 'Cancelar' buttons are at the bottom right. A copyright notice '© Universidad de las Ciencias Informáticas, 2011.' is at the bottom center.

Fig.12 Modificar Usuario

CAPÍTULO II: CARACTERÍSTICAS DEL SISTEMA

El presente capítulo ha permitido analizar los procesos de negocio asociados a la gestión de la información relacionada con el Sistema de Administración de CTS, logrando un modelo único como guía para la implementación del sistema e iniciar el desarrollo del sistema que se quiere alcanzar. Además se describen las características de los artefactos correspondientes a generar los Flujos de Trabajo propuestos por la Metodología seleccionada, sirviendo de base a los desarrolladores para lograr un resultado final.

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA

En el presente capítulo se presenta cual patrón se utilizara para el diseño del sistema, se expone el modelo de clases del diseño en donde se presentan las clases contenidas en la implementación. Además de los diagramas de clases del diseño y los diagramas de secuencia de algunos de los casos de uso que contiene el sistema.

3.1 Patrón de Arquitectura

❖ Modelo Vista Controlador (MVC)

El patrón MVC es un patrón de diseño que ayuda a darle cierta estructura lógica a nuestras aplicaciones. Su principal objetivo es separar la lógica del negocio de la lógica de presentación ó interfaz. (31) Al separar estas partes del proceso, es posible disponer de varias interfaces gráficas distintas que utilicen el mismo modelo y datos sin formato. Esto significa que se podrá utilizar la aplicación con distintas interfaces de Flash.

Se utiliza para separar la información, la salida y el procesamiento de los datos de la aplicación. La aplicación se divide en tres elementos: el modelo, la vista y el controlador; cada elemento gestiona una parte distinta del proceso. El Modelo, son los objetos de la aplicación, también conocida como lógica de negocio, o lógica de aplicación. La Vista especifica la visualización de los datos, algunas veces conocida como lógica de presentación. El controlador es el coordinador entre estos dos últimos, es decir, define la forma en que la interfaz de usuario reacciona ante la entrada de usuario.

3.2 Modelo de Clases del Diseño

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de contenido. (32) Este se compone por clases las que contienen atributos y métodos, otro elemento por el que se compone son las relaciones existentes entre dichas clases.

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA

(Remitirse a: Expediente.Proyecto\1. ingeniería\1.2 arquitectura y diseño, al documento TIE_ST_0121_MDI)

3.2.1 Descripción de las clases del diseño

- Clases Controladoras

Nombre: Gestionar Code Systems	
Tipo de clase: Controladora	
<pre> classDiagram class ModeloDeClasesDiseño { class GestionarCodeSystems { + Adicionar Code Systems() : void + Buscar Code Systems() : List<Code System> + Listar Code System() : List<Code System> + Modificar Code Systems() : void + Mostrar Detalles Code Systems() : Code System } } </pre>	
Para cada responsabilidad:	
Nombre:	Adicionar Code Systems() : void
Descripción:	Se utiliza para adicionar un sistema de código en la base de datos.
Nombre:	Buscar Code Systems() : List<Code System>
Descripción:	Se utiliza para buscar los sistemas de código existentes en la base de datos.
Nombre:	Listar Code System() : List<Code System>
Descripción:	Se utiliza para listar los sistemas de código que se tienen en la base de datos.
Nombre:	Modificar Code System() : void
Descripción:	Se utiliza para modificar algún dato de los sistemas de códigos.
Nombre:	Mostrar Detalles Code System() : Code System
Descripción:	Se utiliza para mostrar los detalles de un sistema de códigos.

Tabla 3.1 Descripción de la clase de diseño: Gestionar Code Systems

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA

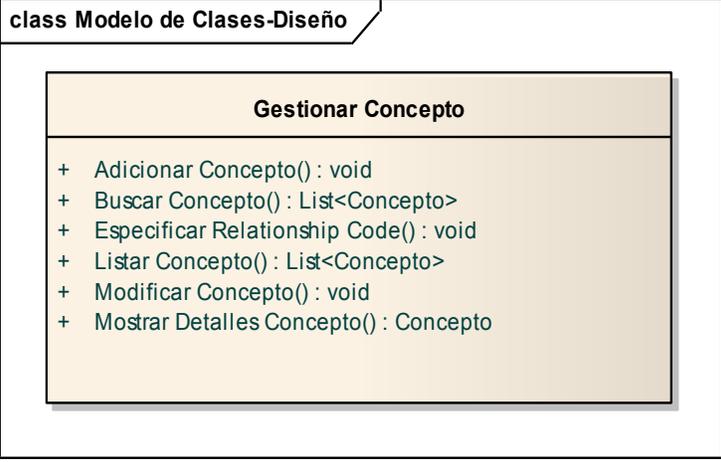
Nombre: Gestionar Concepto	
Tipo de clase: Controladora	
 <pre> classDiagram class GestionarConcepto { +AdicionarConcepto(): void +BuscarConcepto(): List<Concepto> +EspecificarRelationshipCode(): void +ListarConcepto(): List<Concepto> +ModificarConcepto(): void +MostrarDetallesConcepto(): Concepto } </pre> <p>The diagram shows a class named 'Gestionar Concepto' within a 'Modelo de Clases-Diseño' package. The class has six public methods: Adicionar Concepto() returning void, Buscar Concepto() returning List<Concepto>, Especificar Relationship Code() returning void, Listar Concepto() returning List<Concepto>, Modificar Concepto() returning void, and Mostrar Detalles Concepto() returning Concepto.</p>	
Para cada responsabilidad:	
Nombre:	Adicionar Concepto() : void
Descripción:	Se utiliza para adicionar un nuevo concepto a un sistema de código.
Nombre:	Buscar Concepto() : List<Concepto>
Descripción:	Se utiliza para buscar los conceptos existentes.
Nombre:	Especificar Relationship Code() : void
Descripción:	Se utiliza para especificar la relación que tiene el concepto.
Nombre:	Listar Concepto() : List<Concepto>
Descripción:	Se utiliza para listar los conceptos que se tienen en la base de datos.
Nombre:	Modificar Concepto() : void
Descripción:	Se utiliza para modificar algún dato del concepto.
Nombre:	Mostrar Detalles Concepto() : void
Descripción:	Se utiliza mostrar los detalles de algún concepto.

Tabla 3.2 Descripción de la clase de diseño: Gestionar Concepto

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA

Nombre: Gestionar Usuario	
Tipo de clase: Controladora	
<div style="border: 1px solid black; padding: 10px; margin: 10px auto; width: fit-content;"> <p>class Modelo de Clases-Diseño</p> <div style="border: 1px solid black; padding: 5px; margin: 5px auto; width: 80%;"> <p style="text-align: center;">Gestionar Usuario</p> <ul style="list-style-type: none"> + Adicionar Usuario() : void + Buscar Usuario() : List<Usuario> + Eliminar Usuario() : void + Modificar Usuario() : void + Mostrar Detalles Usuario() : Usuario </div> </div>	
Para cada responsabilidad:	
Nombre:	Adicionar Usuario() : void
Descripción:	Se utiliza para adicionar un nuevo usuario en el sistema.
Nombre:	Buscar Usuario() : List<Usuario>
Descripción:	Se utiliza para buscar los usuarios que existen registrados en el sistema.
Nombre:	Eliminar Usuario() : void
Descripción:	Se utiliza para eliminar un usuario.
Nombre:	Modificar Usuario() : void
Descripción:	Se utiliza para modificar los datos de un usuario.
Nombre:	Mostrar Detalles Usuario() : void
Descripción:	Se utiliza mostrar los detalles de un usuario.

Tabla 3.3 Descripción de la clase de diseño: Gestionar Usuario

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA

- Clases Entidad

Nombre: Concept Designations	
Tipo de clase: Entidad	
<div style="border: 1px solid black; padding: 10px;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> class Modelo de Clases-Diseño </div> <div style="border: 1px solid black; padding: 5px;"> <div style="text-align: right; font-size: small;"><i>java.io.Serializable</i></div> <p style="text-align: center;">ConceptDesignations</p> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;"> <ul style="list-style-type: none"> - codedConceptses: Set<CodedConcepts> = new HashSet<Cod... - designation: String - id: int - languages: Languages - preferredForLanguage: boolean - serialVersionUID: long = 629112983893501567L {readOnly} </div> <div style="border: 1px solid black; padding: 5px;"> <ul style="list-style-type: none"> + ConceptDesignations() + ConceptDesignations(int, Languages, String, boolean) + ConceptDesignations(int, Languages, String, boolean, Set<CodedConcepts>) + getCodedConceptses() : Set<CodedConcepts> + getDesignation() : String + getId() : int + getLanguages() : Languages + isPreferredForLanguage() : boolean + setCodedConceptses(Set<CodedConcepts>) : void + setDesignation(String) : void + setId(int) : void + setLanguages(Languages) : void + setPreferredForLanguage(boolean) : void </div> </div> </div>	
Atributo	Tipo
id	int
designation	String
languages	Languages
preferredForLanguages	boolean
Para cada responsabilidad:	
Nombre:	getId() : integer
Descripción:	Retorna el id de la descripción del concepto.
Nombre:	getDesignation() : String
Descripción:	Retorna la descripción del concepto.

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA

Nombre:	getLanguages () : Languages
Descripción:	Retorna el lenguaje de la descripción del concepto.
Nombre:	isPreferredForLanguages () : boolean
Descripción:	Dice si es preferido o no para ese language.

Tabla 3.4 Descripción de la clase de diseño: Concept Designation

Nombre: ConceptRelationships	
Tipo de clase: Entidad	
<div style="border: 1px solid black; padding: 10px;"> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">class Modelo de Clases-Diseño</div> <div style="border: 1px solid black; padding: 10px; margin-bottom: 5px;"> <div style="text-align: right; font-size: small;"><i>java.io.Serializable</i></div> <div style="text-align: center; font-weight: bold; margin-bottom: 10px;">ConceptRelationships</div> <ul style="list-style-type: none"> - codedConceptsBySourceConceptId: CodedConcepts - codedConceptsByTargetConceptId: CodedConcepts - directRelation: boolean - id: int - relationshipCodes: RelationshipCodes - relationshipQualifiers: Set<RelationshipQualifiers> = new HashSet<Rel... - serialVersionUID: long = -44695997039327... {readOnly} <ul style="list-style-type: none"> + ConceptRelationships() + ConceptRelationships(int, CodedConcepts, CodedConcepts, RelationshipCodes, boolean) + ConceptRelationships(int, CodedConcepts, CodedConcepts, RelationshipCodes, boolean, Set<RelationshipQualifiers>) + getCodedConceptsBySourceConceptId() : CodedConcepts + getCodedConceptsByTargetConceptId() : CodedConcepts + getId() : int + getRelationshipCodes() : RelationshipCodes + getRelationshipQualifiers() : Set<RelationshipQualifiers> + isDirectRelation() : boolean + setCodedConceptsBySourceConceptId(CodedConcepts) : void + setCodedConceptsByTargetConceptId(CodedConcepts) : void + setDirectRelation(boolean) : void + setId(int) : void + setRelationshipCodes(RelationshipCodes) : void + setRelationshipQualifiers(Set<RelationshipQualifiers>) : void </div> </div>	
Atributo	Tipo
codedConceptsBySourceConceptId	CodedConcepts

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA

codedConceptsByTargetConceptId	CodedConcepts
Id	int
relationshipCodes	RelationshipCodes
Para cada responsabilidad:	
Nombre:	getCodedConceptsBySourceConceptId(): CodedConcepts
Descripción:	Retorna el id del código de concepto fuente.
Nombre:	getCodedConceptsByTargetConceptId () : CodedConcepts
Descripción:	Retorna el id del código de concepto objetivo.
Nombre:	getId () : int
Descripción:	Retorna el id de la relación del concepto.
Nombre:	getRelationshipCodes () : RelationshipCodes
Descripción:	Retorna la relación de código.

Tabla 3.5 Descripción de la clase de diseño: Concept Relationships

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA

Nombre: Usuario	
Tipo de clase: Entidad	
<i>java.io.Serializable</i>	
Usuario	
<pre> - carnelDentidad: String - direccionParticular: String - eliminado: Boolean - fechaNacimiento: Date - id: int - nombre: String - password: String - primerApellido: String - segundoApellido: String - serialVersionUID: long = -79077587006742... {readOnly} - username: String + getCarneDentidad() : String + getDireccionParticular() : String + getEliminado() : Boolean + getFechaNacimiento() : Date + getId() : int + getNombre() : String + getPassword() : String + getPrimerApellido() : String + getSegundoApellido() : String + getUsername() : String + setCarneDentidad(String) : void + setDireccionParticular(String) : void + setEliminado(Boolean) : void + setFechaNacimiento(Date) : void + setId(int) : void + setNombre(String) : void + setPassword(String) : void + setPrimerApellido(String) : void + setSegundoApellido(String) : void + setUsername(String) : void + Usuario() + Usuario(int, String, String, String) + Usuario(int, String, String, String, String, String, Date, String, String, Boolean) </pre>	
Atributo	Tipo
carnelDentidad	String
direccionParticular	String
eliminado	boolean
fechaNacimiento	Date
Id	int

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA

nombre	String
password	String
primeroApellido	String
segundoApellido	String
username	String
Para cada responsabilidad:	
Nombre:	getCarnelIdentidad():String
Descripción:	Retorna el Carné de Identidad del usuario.
Nombre:	getDireccionParticular () : String
Descripción:	Retorna la Dirección Particular del usuario.
Nombre:	getEliminado () : boolean
Descripción:	Retorna si el usuario está eliminado o no.
Nombre:	getFechaNacimiento () : Date
Descripción:	Retorna la Fecha de Nacimiento del usuario.
Nombre:	getId () : int
Descripción:	Retorna el id del usuario.
Nombre:	getNombre () : String
Descripción:	Retorna el nombre del usuario.
Nombre:	getPassword () : String
Descripción:	Retorna la contraseña del usuario.
Nombre:	getPrimeroApellido () : String
Descripción:	Retorna el primer apellido del usuario.
Nombre:	getSegundoApellido () : String
Descripción:	Retorna el segundo apellido del usuario.
Nombre:	getUsername () : String
Descripción:	Retorna el user del usuario.

Tabla 3.6 Descripción de la clase de diseño: Usuario

CAPÍTULO III: ANÁLISIS Y DISEÑO DEL SISTEMA

3.3 Diagramas de clases del diseño

El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contiene la siguiente información:

- clases, asociaciones y atributos, interfaces, con sus operaciones y constantes, métodos, información sobre los tipos de los atributos, navegabilidad y dependencias. (33)

(Remitirse a: Expediente.Proyecto\1. ingeniería\1.2 arquitectura y diseño, al documento TIE_ST_0121_MDI)

3.4 Diagramas de Secuencia

Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican con qué otros objetos y qué mensajes disparan esas comunicaciones. Los diagramas de secuencia no están pensados para mostrar lógicas de procedimientos complejos. (34)

Estos muestran de una manera gráfica el comportamiento de los actores con las operaciones que relaciona. Parten de la formulación anteriormente hecha de los casos de uso.

(Remitirse a: Expediente.Proyecto\1. ingeniería\1.2 arquitectura y diseño, al documento TIE_ST_0121_MDI)

El presente capítulo ha permitido apreciar cómo, a través de un patrón de diseño el diseño de cualquier trabajo debe regirse, para poder obtener un buen resultado. El diseño representara cada parte por la que se conforma el sistema, no solo en la implementación, sino también en cuales pasos se deben seguir para llegar a realizar cada proceso con los que cuenta el sistema. Los diagramas de clases del diseño, así como los de secuencia representan todos los casos de uso que se implementen.

CAPÍTULO IV: IMPLEMENTACIÓN

En el capítulo que se describe a continuación se presenta lo relacionado con la implementación del sistema. Se modelan los diagramas de componentes y de despliegue, quedando así conformado el modelo de implementación. Además, se brinda una explicación como se realiza el tratamiento de errores, la seguridad y los estándares y estilos utilizados.

4.1 Modelo de Datos

Un modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general, permite describir las estructuras de datos de la base (el tipo de los datos que incluye la base y la forma en que se relacionan), las restricciones de integridad (las condiciones que deben cumplir los datos para reflejar correctamente la realidad que se espera obtener) y las operaciones de manipulación de los datos (agregado, borrado, modificación y recuperación de los datos). (35)

❖ Modelo de Datos

Para la creación del modelo de datos del sistema a desarrollar, se tomaron las clases persistentes, para poder definir la persistencia de los datos y que mantengan su valor. Todas las clases de conforman este modelo de datos están representadas en el diagrama Modelo de Datos que se encuentra en el expediente de proyecto.

(Remitirse a: Expediente.Proyecto\1. ingeniería\1.2 arquitectura y diseño, al documento TIE_ST_0120_ArqSWv1.0)

4.1.1 Descripción de las tablas del Modelo de datos

Nombre: CodeSystem		
Descripción: Tabla que recopilará los datos de cada sistema de código entrado en la base de datos.		
Atributo	Tipo	Descripción
id	integer	Llave primaria, es el id incremental del sistema de código.

CAPÍTULO IV: IMPLEMENTACIÓN

Code_system_id	String	Representa el id del code_system
code_system_name	character varying	Representa el nombre del sistema de código
copyright	character varying	Representa el copyright del sistema de código
fullname	character varying	Representa el nombre completo del sistema de código
code_system_description	character varying	Representa la descripción del sistema de código
vers	character varying	Representa la versión del sistema de código

Tabla 4.1 Descripción de la tabla: CodeSystems

Nombre: CodeConcept		
Descripción: Tabla que recopilará los datos de los códigos.		
Atributo	Tipo	Descripción
id	integer	Llave primaria, es el id incremental del código.
code_system_id	integer	Representa el id del sistema de código al que pertenece el código
concept_code	character varying	Representa el valor del código
concept_status_code_id	integer	Representa el estado del código

Tabla 4.2 Descripción de la tabla: CodeConcept

Nombre: ConceptStatusCode		
Descripción: Tabla que recopilará los datos de los estados en la base de datos		
Atributo	Tipo	Descripción
id	integer	Llave primaria, es el id incremental del sistema de código.

CAPÍTULO IV: IMPLEMENTACIÓN

value	character varying	Representa el valor del estado
-------	-------------------	--------------------------------

Tabla 4.3 Descripción de la tabla: ConceptStatusCode

Nombre: RelationshipCode		
Descripción: Tabla que recopilará los datos de los tipos de relaciones en la base de datos		
Atributo	Tipo	Descripción
id	integer	Llave primaria, es el id incremental del sistema de código.
value	character varying	Representa el valor de la relación
trans	bit	Representa la transitividad de la relación
"symmetric"	bit	Representa si es o no simétrico
reflexive	bit	Representa si es o no reflexivo
inverse	integer	Representa la inversa de la relación

Tabla 4.4 Descripción de la tabla: RelationshipCode

Nombre: ConceptDesignations		
Descripción: Tabla que recopilará los datos de la descripción correspondiente a los códigos		
Atributo	Tipo	Descripción
id	integer	Llave primaria, es el id incremental las descripciones
designation	character varying	Representa el valor de las descripciones
preferred_for_language	bit	Representa si el lenguaje del sistema de código es el preferido o no
language_code_id	integer	Representa el lenguaje del sistema de código

Tabla 4.5 Descripción de la tabla: ConceptDesignation

CAPÍTULO IV: IMPLEMENTACIÓN

Nombre: Language		
Descripción: Tabla que recopilará los datos de los lenguajes existentes en la base de datos		
Atributo	Tipo	Descripción
id	integer	Llave primaria, es el id incremental del sistema de código.
value	character varying	Representa el valor del lenguaje

Tabla 4.6 Descripción de la tabla: Language

4.2 Implementación

La implementación comienza una vez que se tiene terminado o casi terminado la parte del diseño del sistema, por lo que, cada parte del modelo del diseño se implementa en componentes. Los diagramas de despliegue y componentes que se representan en este epígrafe, son artefactos que se generan en este flujo de trabajo, conformando el modelo de implementación.

4.2.1 Diagrama de Despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos. (36) El diagrama representa un grafo, donde sus nodos se une por conexiones de comunicación.

Nodo PC_Cliente: Representa la computadora donde el usuario interactuará con el sistema.

Nodo Servidor de Aplicaciones (Jboss Server): Representa el servidor donde estará el sistema de Administración. Este tendrá comunicación con el nodo donde estará la el servidor de base de Datos.

Nodo Servidor de Base de Datos (PostgreSQL 8.4): Representa el servidor donde estará montada la base de datos del sistema.

CAPÍTULO IV: IMPLEMENTACIÓN

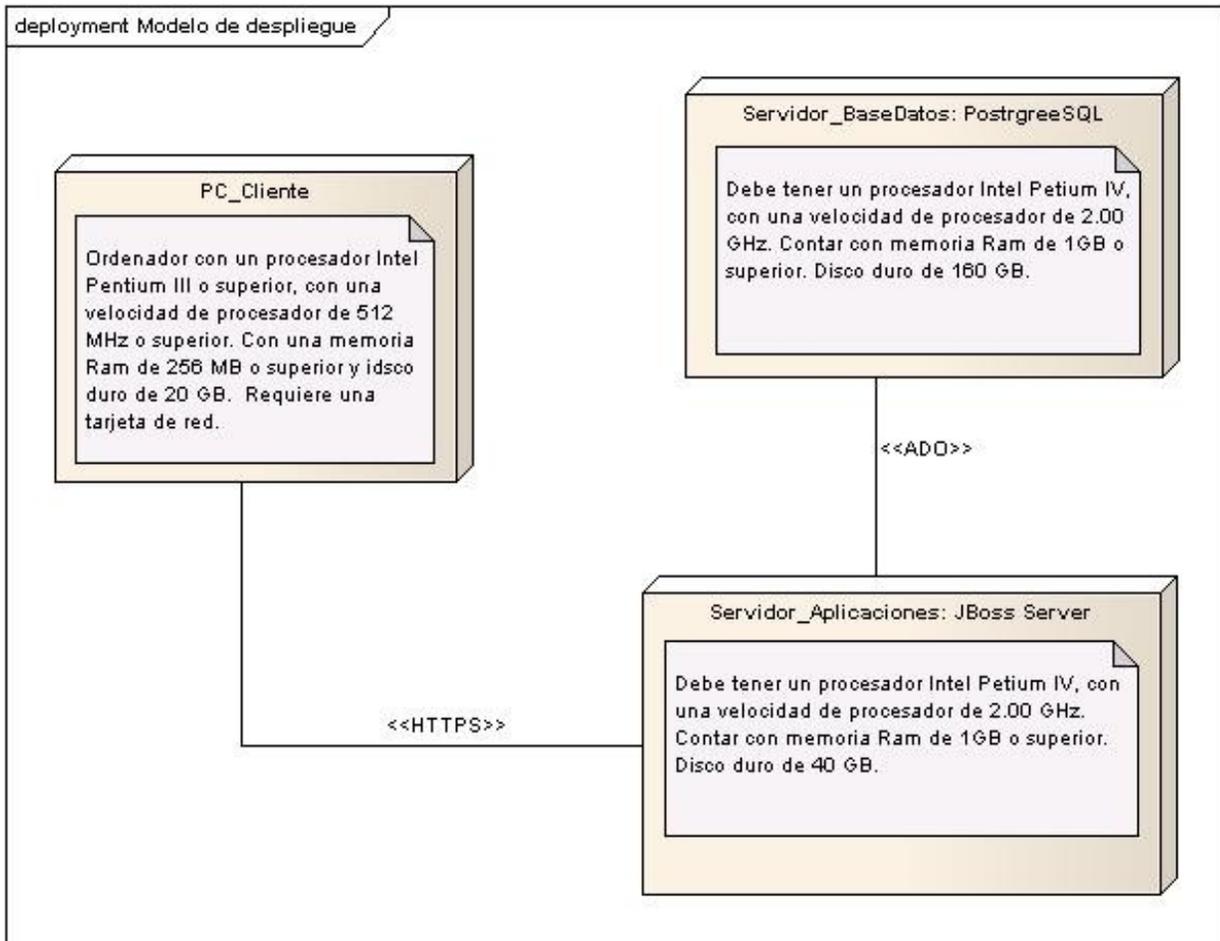


Fig. 13 Diagrama de Despliegue

4.2.2 Diagrama de Componentes

El modelo de componentes ilustra los componentes de software que se usarán para construir el sistema. (37) Además de mostrar la relación entre componentes de software, sus dependencias, su comunicación su ubicación y otras condiciones. Modelando el diagrama de componentes brinda una ayuda a los desarrolladores a visualizar el camino de la implementación. Permite tomar decisiones respecto a las tareas de implementación.

CAPÍTULO IV: IMPLEMENTACIÓN

El lenguaje UML define para este diagrama cinco estereotipos estándar los que son:

Executable: Componente que se puede ejecutar en un nodo.

Library: Define una biblioteca de objetos estática o dinámica.

Table: Describe un componente que representa una tabla de una base de datos.

File: Componente que representa un documento que contiene código fuente o datos.

Document: Componente que representa un documento. (38)

Para un mayor entendimiento del diagrama de componentes se creó por paquetes debido al tamaño de cada uno de ellos. En cada paquete se encuentran los componentes pertenecientes a cada uno de ellos, representados en sus diagramas correspondientes.

(Remitirse a: Expediente.Proyecto\1. ingeniería\1.2 arquitectura y diseño, al documento TIE_ST_0120_ArqSWv1.0)

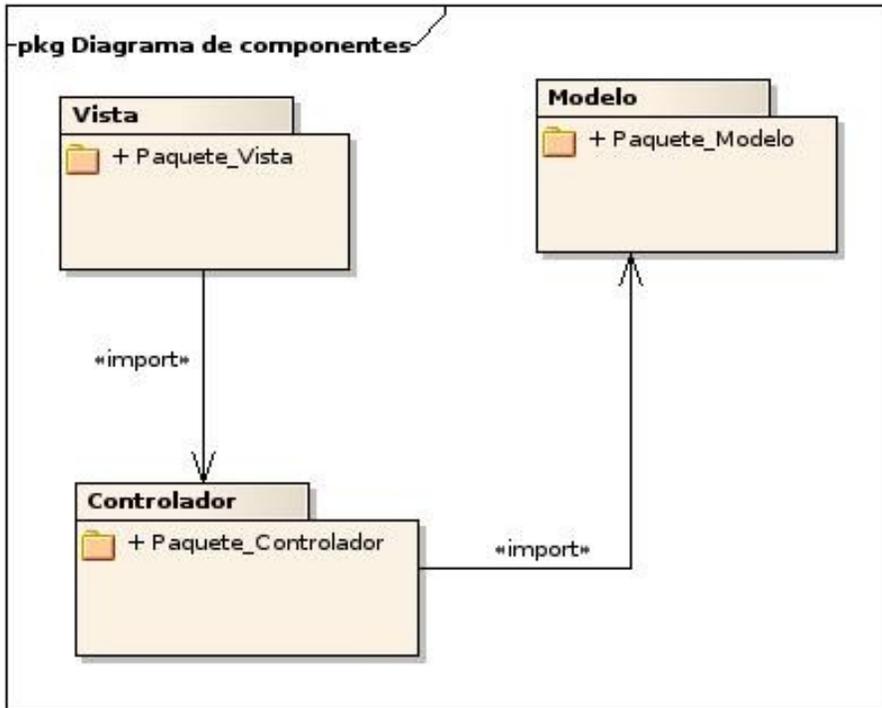


Fig. 14 Diagrama de Componentes (Paquetes)

4.2.3 Estrategias de codificación. Estándares y estilos a utilizar.

4.2.3.1 Estrategias de diseño

Para el diseño de las vistas del sistema de Administración se tomaron las pautas del Diseño de Aplicaciones Web del CESIM, permitiendo igualdad en todos los productos del centro. A continuación se muestran algunas de las pautas por las que se ha regido el diseño del sistema.

General

- El área de trabajo será la zona principal de la aplicación, donde se mostrarán los contenidos o formularios con los que se encuentre trabajando el usuario.



Fig. 15 Encabezado

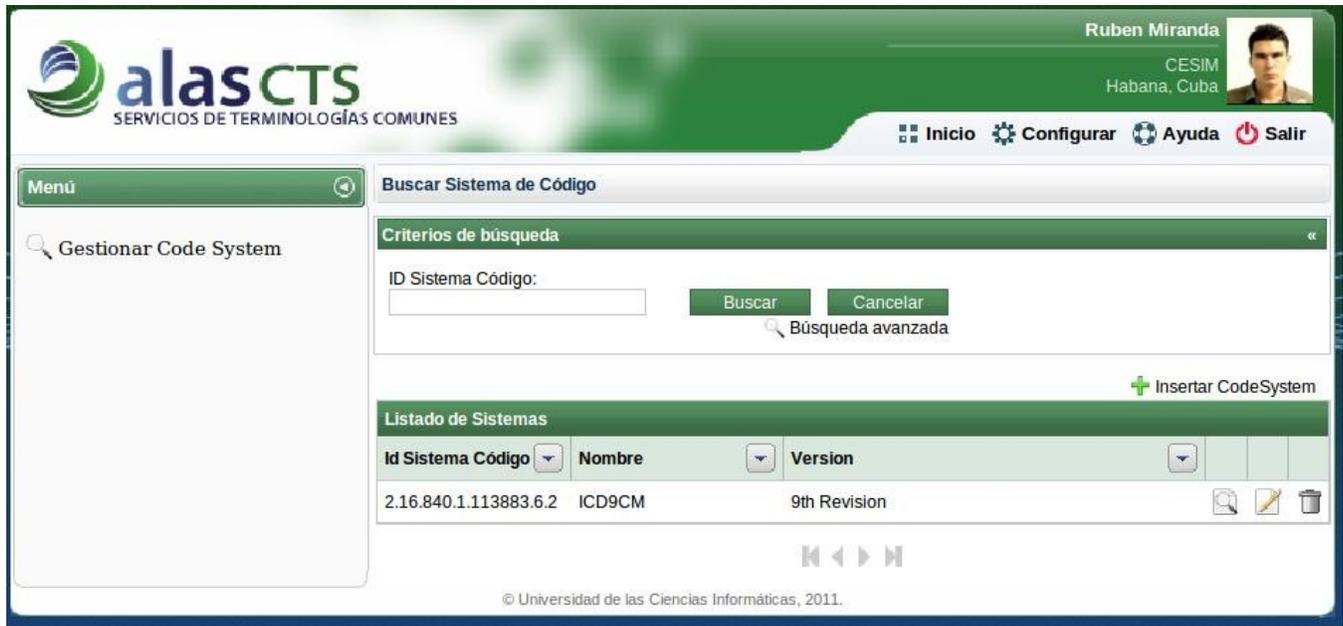


Fig.16 Área de Trabajo

Etiquetas

- Las etiquetas estáticas que aparezcan irán en negro.
- El contenido de las etiquetas no debe contener preposiciones innecesarias, por ejemplo la etiqueta Gestionar un concepto, debe ser Gestionar concepto.

Iconos para funcionalidades

- Las etiquetas del sistema se utilizarán para encabezar o titular los bloques de información. Hacen el papel de títulos o subtítulos, su significado está condicionado por el contexto. (39)

Nombre	Acción
Adicionar	Crea y guarda un nuevo elemento 
Eliminar	Elimina un elemento. 
Buscar	Realiza una búsqueda. 

CAPÍTULO IV: IMPLEMENTACIÓN

Modificar	Muestra los controles editables de un elemento para ser modificado. 
Ver	Muestra los detalles 
Advertencia	Aparece en la interfaz que muestra un mensaje de advertencia 
Información	Aparece en la interfaz que muestra una información 
Error	No se puede ejecutar la acción por cualquier otro tipo de error. 
Estado	Representa el estado de espera a una respuesta de una acción 
Cerrar	Permite cerrar una ventana 

Tabla 4.7 Iconos para funcionalidades

Mensajes

- El contenido de los mensajes se mostrará alineado a la izquierda, terminarán con punto final.
- Los botones de los mensajes deben cumplir las mismas reglas que la sección de Iconos.
- En las ventanas de Información y Error se mostrará solo un botón de “Aceptar”, centrado en la parte inferior.
- Las ventanas de Advertencia utilizarán dos botones colocados en la parte inferior y tendrán los textos: “Sí” y “No”. El botón “Sí” estará a la izquierda y el “No” a la derecha.

4.2.3.2 Estándares de codificación

Cuando se utiliza un estándar para programar, este define cuál vocabulario se utilizará para las variables, métodos, objetos; además de saber con legibilidad el código que se escribe. Se han trabajado con los estándares de codificación que se encuentran en el documento de Arquitectura de Software, con la versión 2.0. Dentro de ellos existen estándares para el idioma a utilizar, los comentarios, como iniciar y terminar un bloque, donde se ponen las líneas en blanco. Asimismo se estandariza que las clases y objetos comienzan con mayúscula, los atributos deben comenzar con minúsculas, entre otros.

CAPÍTULO IV: IMPLEMENTACIÓN

4.2.4 Tratamiento de errores

Un buen indicador de la calidad de un sistema software es comprobar cómo responde cuando se producen fallos, ya sean esperados y controlados (excepciones) o inesperados e incontrolados (errores). (40) En el caso de Java se utilizan bloques try - catch - finally, throw y throws, y las subclases de java.lang.Exception. Un método lanzarán (mediante un throw) una excepción (una clase que hereda java.lang.Exception) provocando que se abandone el flujo de ejecución normal y cediendo el control al bloque en el que se capturará (en un bloque try - catch - finally). En la declaración del método se incluye throws para avisar de que puede lanzar una excepción. (41) En terminología java, crear un objeto exception y manejarlo por el sistema de ejecución se llama lanzar una excepción. (42)

El tratamiento de errores facilita el correcto trabajo de la aplicación, dándole una mejor apariencia ante los usuarios. En esta ocasión, en el sistema se trataron los errores de forma tal que la realización de inserción, eliminación, modificación y otros en la base de datos, se cumplan de manera correcta. Para alcanzar esto se crearon formas de validación que comprobaran el correcto manejo de los datos. Cuando el usuario introduce datos después de haber sido autenticado, son validadas dichas entradas por funciones desarrolladas para validar, pues de existir algún error, mostrarán mensajes que ilustran el mal manejo de los datos, la inserción, o la modificación de los mismos, o en el caso que el usuario no esté registrado en el sistema, este no tendrá acceso.

4.2.5 Seguridad

Para garantizar un sistema confiable, se le establecen normas de seguridad por las que se regirá. En su conjunto permiten que la información no corra el riesgo de ser manipulada por personal ajeno, asegura el acceso del personal autorizado solo cuando sea necesario. De ahí, depende la integridad, autenticidad y confiabilidad de la información. La seguridad no solo dependerá del sistema de Administración, sino del CTS en general. Para alcanzar estos objetivos se agregaron normas de seguridad que permiten una mayor fiabilidad del sistema, como:

- Cuenta con la autenticación de los usuarios, los que tienen distintos roles, que le permiten acceder a los diferentes servicios que brinda el sistema.
- La información que se muestre no será divulgada por personal sin autorización.

CAPÍTULO IV: IMPLEMENTACIÓN

- Los datos que se manejen en la aplicación serán protegidos contra personal no autorizado.
- El código estará validado para tratar las excepciones que se presenten.
- Los datos se guardarán en el lugar correspondiente para cada uno de ellos en la base datos, permitiendo poder hacer una copia de seguridad cuando se requiera.
- Los lugares donde se guarden las copias de seguridad deben garantizar la confidencialidad e integridad de los datos.

4.3 Pruebas

Para probar el correcto funcionamiento del sistema que se encuentra en desarrollo, y poder erradicar los errores que pueda contener, se realizó un tipo de prueba llamado pruebas de caja negra. Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro. (43)

Para realizar las pruebas de caja negra se hicieron diseños de casos de prueba. Estos diseños de casos de prueba permiten poder ver cual resultado debe mostrar el sistema ante cada una de las pruebas descritas en los mismos.

Al efectuar las pruebas pertenecientes a la primera iteración, descritas en los diseños de casos de prueba, se llegó a que el sistema no muestra errores a la hora de insertar datos, ni de hacer cualquier tipo de consulta por parte del usuario o del administrador.

En el capítulo anteriormente presentado se concreta la implementación del sistema informático siguiendo lo establecido en la Especificación de Requisitos de Software, definiendo los patrones de diseño del CESIM, además se modelan los diagramas de componentes y de despliegue, que permiten representar como estará estructurado el sistema de administración. De esta forma queda conformado el modelo de implementación con algunas medidas de seguridad, que admitan que la información con la que se trabaje sea fiable y permita realizar pruebas de caja negra al sistema para verificar su correcto funcionamiento.

CONCLUSIONES

- ❖ Se realizó un estudio de los sistemas asociados al tema en el ámbito nacional e internacional que si bien no arrojó como resultado ninguna experiencia capaz de satisfacer las necesidades del centro permitió que se identificaran experiencias significativas para el desarrollo de la investigación.
- ❖ Se valoraron las tendencias actuales a nivel nacional e internacional comparándolas con las normas del desarrollo en el Centro de Informática Médica. Se decidió asimilar la arquitectura y el marco de trabajo propios del centro, por lo cual se utilizaron como metodología de desarrollo Rational Unified Process, como herramienta de modelado Enterprise Architect versión 7.0, que permite la representación de los artefactos en el Lenguaje de Modelado Unificado en su versión 2.1. Como gestor de base de datos se utilizó PostgreSQL 8.4, como lenguaje de programación Java y como entorno integrado de desarrollo Eclipse Galileo.
- ❖ Se implementó un sistema capaz de satisfacer las necesidades primarias de configuración para poner en marcha el servicio de terminologías comunes, permitiendo la incorporación de nuevas terminologías independientes y de nuevas versiones de sistemas de códigos. Además de realizar comprobaciones visuales que validen el resultado de la instalación.
- ❖ Como parte del proceso de desarrollo se realizaron pruebas al sistema, que permitieron disminuir el posible número de errores en las pruebas de liberación oficial del producto.

RECOMENDACIONES

- ❖ Continuar con el desarrollo del sistema implementando inicialmente requisitos que ya fueron identificados y documentados que no constituyeron prioridad en esta primera iteración.
- ❖ Incorporar al desarrollo funcionalidades que complementan el sistema actual de Administración, que se encuentran definidas en el Servicio de Terminologías Comunes v2 (CTS-2), tales como:
 - Register for Notification
 - Update Notification Registration
- ❖ Optimizar la incorporación de Sistemas de Códigos ampliando el marco no solo a las versiones taxonómicas ya conocidas, si no a cualquiera que sea necesaria.
- ❖ Incorporar técnicas de programación concurrente para lograr mayor rapidez y estabilidad ante funcionalidades que demandan mayores recursos de procesamiento como la incorporación de sistemas de código completos al servicio.
- ❖ Incorporar al sistema opciones de internacionalización que amplíen sus posibilidades de uso sin limitaciones que representan el empleo de un solo idioma.

REFERENCIAS BIBLIOGRÁFICAS

REFERENCIAS BIBLIOGRÁFICAS

1. Informed. *Informed*. [En línea] [Citado el: 7 de 10 de 2010.] http://www.sld.cu/sistema_de_salud/aspectos.html.
2. *Curso Universitario Sistemas de Información en los Sistemas de Salud*. Buenos Aires : Instituto Universitario Hospital Italiano de Buenos Aires.
3. [En línea] 17 de 2 de 2007. [Citado el: 7 de 10 de 2010.] <http://salud-cubana.blogspot.com/>.
4. *mastermagazine*. *mastermagazine*. [En línea] 12 de 2 de 2005. [Citado el: 8 de 10 de 2010.] <http://www.mastermagazine.info/termino/5416.php>.
5. [En línea] [Citado el: 8 de 10 de 2010.] <http://www.promonegocios.net/administracion/definicion-administracion.html>.
6. *masadelante.com*. *masadelante.com*. [En línea] [Citado el: 8 de 10 de 2010.] <http://www.masadelante.com/faqs/servidor>.
7. *educ.ar*. *educ.ar*. [En línea] [Citado el: 8 de 10 de 2010.] <http://aportes.educ.ar/lengua/nucleo-teorico/influencia-de-las-tic/tecnologias-de-la-informacion-y-la-comunicacion-tic-y-lingueistica/terminologia.php>.
8. *Definicion.de*. *Definicion.de*. [En línea] [Citado el: 8 de 10 de 2010.] <http://definicion.de/codigo/>.
9. **Sánchez, M^aJesús Portolés**. *Búsqueda Semántica en Repositorios de Conceptos Biomedicos Estandarizados: C.T.Hunter*. s.l. : Universidad Politecnica de Valencia, 2010
10. **Gallego, Carlos. Mas Mota, Toni. Rodriguez, Mireia**. La terminologia en laboratorio. [En línea] [Citado el: 25 de 11 de 2010.] <http://www.gencat.cat/salut/ticsalut/html/ca/dir3480/loinc%20snomed%20ct.pdf>.
11. The Clinical information Consultancy. *The Clinical information Consultancy*. [En línea] [Citado el: 23 de 11 de 2010.] http://www1.clininfo.co.uk/training_cliniclue_api.
12. [En línea] [Citado el: 8 de 10 de 2010.] [http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP vs. XP.pdf](http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP_vs_XP.pdf).
13. [En línea] [Citado el: 8 de 10 de 2010.] <http://www.editum.org/Que-Es-Un-Servidor-De-Aplicaciones-p-473.html>.
14. JBoss Community. *JBoss Community*. [En línea] [Citado el: 26 de 1 de 2011.] <http://jboss.org/jbossas/>.
15. SOA agenda. *SOA agenda*. [En línea] [Citado el: 26 de 1 de 2011.] <http://soaagenda.com/journal/articulos/que-son-los-frameworks/>.
16. [En línea] [Citado el: 5 de 12 de 2010.] <http://es.debugmodeon.com/articulo/que-es-jboss-seam>.
17. [En línea] [Citado el: 5 de 12 de 2010.] http://www.dosideas.com/wiki/Mapeo_Objeto-Relacional.
18. *Hackelare*. *Hackelare*. [En línea] [Citado el: 5 de 12 de 2010.] <http://hackelare.wordpress.com/2010/06/17/hibernate-para-nonos-introduccion/>.
19. [En línea] [Citado el: 5 de 12 de 2010.] <http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/introduccion/virtual.htm>.
20. *adictosaltrabajo*. *adictosaltrabajo*. [En línea] [Citado el: 5 de 12 de 2010.] <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro>.

REFERENCIAS BIBLIOGRÁFICAS

21. PostgreSQL. *PostgreSQL*. [En línea] [Citado el: 5 de 12 de 2010.] <http://archives.postgresql.org/pgsql-es-fomento/2009-07/msg00000.php>.
22. [En línea] [Citado el: 6 de 12 de 2010.] <http://www.mateamargonerds.com/download/52-programas/703-postgresql-84-disponible.html>.
23. Sparxsystems. *Sparxsystems*. [Online] [Cited: 12 6, 2010.] <http://www.sparxsystems.com.ar/products/ea.html>.
24. [En línea] [Citado el: 6 de 12 de 2010.] http://www.uv.es/~jgutier/MySQL_Java/TutorialEclipse.pdf.
25. WebTaller.com. *WebTaller.com*. [En línea] [Citado el: 6 de 12 de 2010.] <http://www.webtaller.com/construccion/lenguajes/java/lecciones/que-es-java.php>.
26. [En línea] [Citado el: 27 de 1 de 2011.] <http://sheyla88.blogspot.es/>.
27. [En línea] [Citado el: 6 de 12 de 2010.] <http://www.disca.upv.es/enheror/pdf/ActaUML.PDF>.
28. [En línea] [Citado el: 8 de 3 de 2011.] <http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/>.
29. [En línea] [Citado el: 8 de 3 de 2011.] http://www.dlsiis.fi.upm.es/docto_lsiis/Trabajos20052006/Diez.pdf.
30. Tecnologia y Synergix. [En línea] [Citado el: 26 de 5 de 2011.] <http://synergix.wordpress.com/2008/07/07/requisito-funcional-y-no-funcional/>.
31. [En línea] [Citado el: 8 de 3 de 2011.] <http://es.debugmodeon.com/articulo/el-patron-mvc>.
32. [En línea] [Citado el: 10 de 3 de 2011.] <http://www.dcc.uchile.cl/~psalinas/uml/modelo.html>.
33. [En línea] [Citado el: 5 de 5 de 2011.] <http://www.inf.utfsm.cl/~visconti/ili236/Documentos/10-DisenoOO.pdf>.
34. [En línea] [Citado el: 10 de 3 de 2011.] <http://www.scribd.com/doc/15493687/DIAGRAMAS-DE-SECUENCIA>.
35. Definicion.de. *Definicion.de*. [En línea] [Citado el: 10 de 5 de 2011.] <http://definicion.de/modelo-de-datos>.
36. Sparx systems. *Sparx systems*. [En línea] [Citado el: 11 de 4 de 2011.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.
37. Sparx systems. *Sparx systems*. [En línea] [Citado el: 9 de 5 de 2011.] http://www.sparxsystems.com.ar/resources/tutorial/component_model.html.
38. [En línea] [Citado el: 9 de 5 de 2011.] <http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf>.
39. **Cordova Osorio, Niurka**. *Pautas_Diseño_AplicacionesWEB_CESIM*. s.l. : CESIM, 2010.
40. debug_mode=ON. *debug_mode=ON*. [En línea] [Citado el: 23 de 5 de 2011.] <http://es.debugmodeon.com/articulo/manejo-de-excepciones-y-errores-en-una-arquitectura-java>.
41. Ídem a la referencia 43.
42. programacion en castellano. *programacion en castellano*. [En línea] [Citado el: 10 de 5 de 2011.] http://www.programacion.com/articulo/manejo_de_errores_usando_excepciones_java_98/2.
43. [En línea] [Citado el: 10 de 5 de 2011.] <http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing>.

BIBLIOGRAFÍA

1. Informed. *Informed*. [En línea] [Citado el: 7 de 10 de 2010.] http://www.sld.cu/sistema_de_salud/aspectos.html.
2. *Curso Universitario Sistemas de Información en los Sistemas de Salud*. Buenos Aires : Instituto Universitario Hospital Italiano de Buenos Aires.
3. [En línea] 17 de 2 de 2007. [Citado el: 7 de 10 de 2010.] <http://salud-cubana.blogspot.com/>.
4. *mastermagazine*. *mastermagazine*. [En línea] 12 de 2 de 2005. [Citado el: 8 de 10 de 2010.] <http://www.mastermagazine.info/termino/5416.php>.
5. [En línea] [Citado el: 8 de 10 de 2010.] <http://www.promonegocios.net/administracion/definicion-administracion.html>.
6. *masadelante.com*. *masadelante.com*. [En línea] [Citado el: 8 de 10 de 2010.] <http://www.masadelante.com/faqs/servidor>.
7. *educ.ar*. *educ.ar*. [En línea] [Citado el: 8 de 10 de 2010.] <http://aportes.educ.ar/lengua/nucleo-teorico/influencia-de-las-tic/tecnologias-de-la-informacion-y-la-comunicacion-tic-y-lingueistica/terminologia.php>.
8. *Definicion.de*. *Definicion.de*. [En línea] [Citado el: 8 de 10 de 2010.] <http://definicion.de/codigo/>.
9. **Sánchez Portolés, M^a Jesús**. *Búsqueda Semántica en Repositorios de Conceptos Biomedicos Estandarizados: C.T.Hunter*. s.l. : Universidad Politecnica de Valencia, 2010.
10. **Gallego, Carlos. Mas Mota, Toni. Rodriguez, Mireia**. La terminología en laboratorio. [En línea] [Citado el: 25 de 11 de 2010.] <http://www.gencat.cat/salut/ticsalut/html/ca/dir3480/loinc%20snomed%20ct.pdf>.
11. The Clinical information Consultancy. *The Clinical information Consultancy*. [En línea] [Citado el: 23 de 11 de 2010.] http://www1.clininfo.co.uk/training_cliniclue_api.
12. [En línea] [Citado el: 8 de 10 de 2010.] [http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP vs. XP.pdf](http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP_vs_XP.pdf).
13. *Que-Es-Un-Servidor-De-Aplicaciones* [En línea] [Citado el: 8 de 10 de 2010.] <http://www.editum.org/Que-Es-Un-Servidor-De-Aplicaciones-p-473.html>.
14. JBoss Community. *JBoss Community*. [En línea] [Citado el: 26 de 1 de 2011.] <http://jboss.org/jbossas/>.
15. SOA agenda. *SOA agenda*. [En línea] [Citado el: 26 de 1 de 2011.] <http://soaagenda.com/journal/articulos/que-son-los-frameworks/>.
16. *que-es-jboss-seam* [En línea] [Citado el: 5 de 12 de 2010.] <http://es.debugmodeon.com/articulo/que-es-jboss-seam>.
17. *Mapeo_Objeto-Relacional* [En línea] [Citado el: 5 de 12 de 2010.] http://www.dosideas.com/wiki/Mapeo_Objeto-Relacional.
18. *Hackelare*. *Hackelare*. [En línea] [Citado el: 5 de 12 de 2010.] <http://hackelare.wordpress.com/2010/06/17/hibernate-para-nonos-introduccion/>.
19. [En línea] [Citado el: 5 de 12 de 2010.] <http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/introduccion/virtual.htm>.

20. adictosaltrabajo. *adictosaltrabajo*. [En línea] [Citado el: 5 de 12 de 2010.] <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=richFacesJsfIntro>.
21. PostgreSQL. *PostgreSQL*. [En línea] [Citado el: 5 de 12 de 2010.] <http://archives.postgresql.org/pgsql-es-fomento/2009-07/msg00000.php>.
22. [En línea] [Citado el: 6 de 12 de 2010.] <http://www.mateamargonerds.com/download/52-programas/703-postgresql-84-disponible.html>.
23. Sparxsystems. *Sparxsystems*. [Online] [Cited: 12 6, 2010.] <http://www.sparxsystems.com.ar/products/ea.html>.
24. [En línea] [Citado el: 6 de 12 de 2010.] http://www.uv.es/~jgutierrez/MySQL_Java/TutorialEclipse.pdf.
25. WebTaller.com. *WebTaller.com*. [En línea] [Citado el: 6 de 12 de 2010.] <http://www.webtaller.com/construccion/lenguajes/java/lecciones/que-es-java.php>.
26. [En línea] [Citado el: 27 de 1 de 2011.] <http://sheyla88.blogspot.es/>.
27. [En línea] [Citado el: 6 de 12 de 2010.] <http://www.disca.upv.es/enheror/pdf/ActaUML.PDF>.
28. [En línea] [Citado el: 8 de 3 de 2011.] <http://synergix.wordpress.com/2008/07/10/modelo-de-dominio/>.
29. [En línea] [Citado el: 8 de 3 de 2011.] http://www.dlsiis.fi.upm.es/docto_lsiis/Trabajos20052006/Diez.pdf.
30. Tecnología y Synergix. [En línea] [Citado el: 26 de 5 de 2011.] <http://synergix.wordpress.com/2008/07/07/requisito-funcional-y-no-funcional/>.
31. [En línea] [Citado el: 8 de 3 de 2011.] <http://es.debugmodeon.com/articulo/el-patron-mvc>.
32. [En línea] [Citado el: 10 de 3 de 2011.] <http://www.dcc.uchile.cl/~psalinas/uml/modelo.html>.
33. [En línea] [Citado el: 5 de 5 de 2011.] <http://www.inf.utfsm.cl/~visconti/ili236/Documentos/10-DisenoOO.pdf>.
34. [En línea] [Citado el: 10 de 3 de 2011.] <http://www.scribd.com/doc/15493687/DIAGRAMAS-DE-SECUENCIA>.
35. Definicion.de. *Definicion.de*. [En línea] [Citado el: 10 de 5 de 2011.] <http://definicion.de/modelo-de-datos>.
36. Sparx systems. *Sparx systems*. [En línea] [Citado el: 11 de 4 de 2011.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.
37. Sparx systems. *Sparx systems*. [En línea] [Citado el: 9 de 5 de 2011.] http://www.sparxsystems.com.ar/resources/tutorial/component_model.html.
38. [En línea] [Citado el: 9 de 5 de 2011.] <http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf>.
39. **Cordova Osorio, Niurka**. *Pautas_Diseño_AplicacionesWEB_CESIM*. s.l. : CESIM, 2010.
40. debug_mode=ON. *debug_mode=ON*. [En línea] [Citado el: 23 de 5 de 2011.] <http://es.debugmodeon.com/articulo/manejo-de-excepciones-y-errores-en-una-arquitectura-java>.
41. programacion en castellano. *programacion en castellano*. [En línea] [Citado el: 10 de 5 de 2011.] http://www.programacion.com/articulo/manejo_de_errores_usando_excepciones_java_98/2.
42. [En línea] [Citado el: 10 de 5 de 2011.] <http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing>.
43. [Online] [Cited: 10 8, 2010.] <http://www.rational.com.ar/herramientas/rup.html>.
44. **Geoffrey, Sparks**. Modelo de Componentes. Argentina : Sparx Systems.

45. [Online] [Cited: 4 22, 2011.]
<http://ingenieriasoftwaredos.wikispaces.com/Diagrama+de+componentes+y+objetos>.
46. **Martínez Almaguer, Ing. Norge.** *Documento_de_Arquitectura_de_Software*. s.l. : CESIM, 2011.
47. [Online] [Cited: 3 22, 2011.] <http://eva.uci.cu>.
48. [Online] [Cited: 3 21, 2011.] <http://calisoft.uci.cu>.
49. Modulo2 Tema 12: Modulo de Implementación.
50. [Online] [Cited: 4 23, 2011.]
<http://gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node27.html>.
51. FreeSpaceMap [Online] [Cited: 5 23, 2011.] <http://www.network-theory.co.uk/docs/postgresql/vol3/FreeSpaceMap.html>.
52. achisa.org [Online] [Cited: 11 18, 2010.] <http://www.achisa.org/wiki/index.php?title=It%C3%A1lica>.
53. loinc.org [Online] [Cited: 11 18, 2010.] <http://loinc.org/downloads/relma>.
54. Portoles Sanchez, M Jesús[Online] [Cited: 11 18, 2010.]
<http://riunet.upv.es/bitstream/handle/10251/8614/Memoria%20Proyecto%20MJesus%20Portoles%20Sanchez.pdf?sequence=1>.
55. HL7 [Online] [Cited: 9 15, 2010.] <http://10.128.50.168/HL7/welcome/environment/index.htm>.
56. hl7spain [Online] [Cited: 4 27, 2011.]
http://www.hl7spain.org/Ficheros/0/Documentos/hl7spain_Present_Valencia241008.pdf

ANEXOS

1.1 Casos de Uso del Sistema

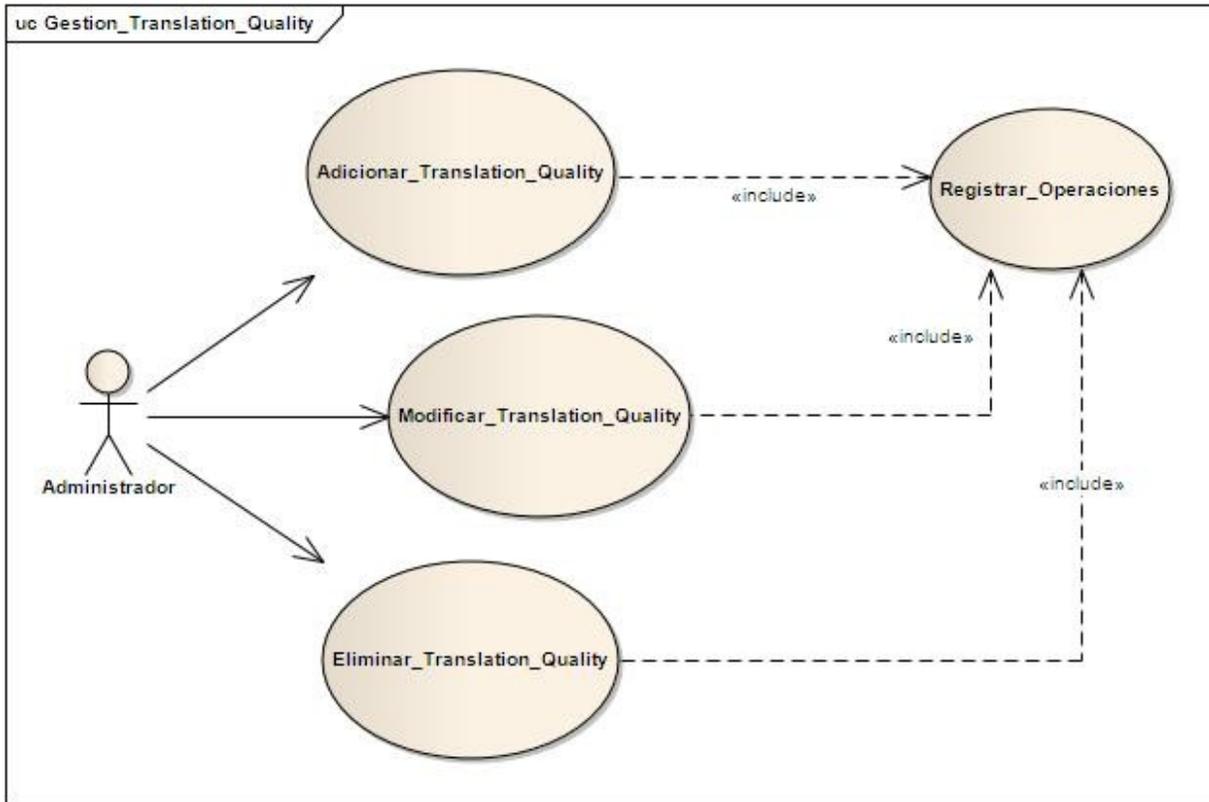


Fig. 1 Gestion de Translation Quality

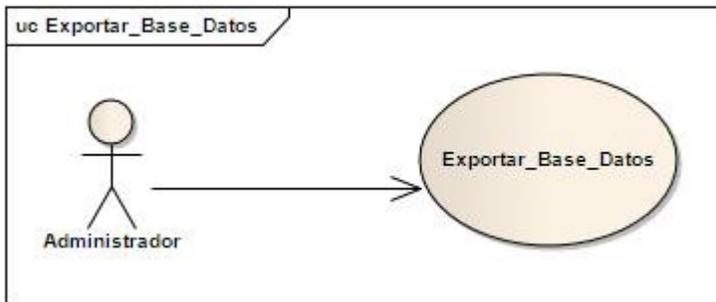


Fig.2 Exportar Base de Datos

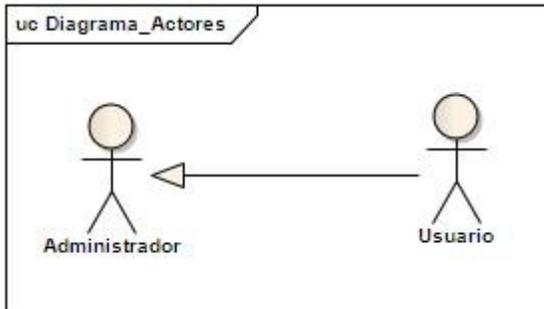


Fig. 3 Generalización de Actores

1.2 Páginas del sistema



Fig.4 Página para autenticarse



Fig.5 Página de configuración de funcionalidades

GLOSARIO DE TERMINOS

Ajax: Acrónimo para Asynchronous JavaScript + XML, se basa en cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en background, los datos que son usados para actualizar la página solo renderizando la página y mostrando u ocultando porciones de la misma.

ANSI: El Instituto Nacional de Normalización Estadounidense (ANSI por su sigla en inglés) es una organización privada sin fines lucrativos, que administra y coordina la normalización voluntaria y las actividades relacionadas a la evaluación de conformidad en los Estados Unidos.

API (Application Program Interfase, o interfaz de comunicación entre programas de aplicación): representa un interfaz de comunicación entre componentes software.

CSS Themes: Templates.

Free Space Map: Estos parámetros, controlan el tamaño de la hoja de espacio libre para compartir, que rastrean la ubicación de espacio no utilizado en la base de datos. Un mapa inferior al espacio libre puede hacer que la base de datos a consumir cada vez más cantidades de espacio en disco con el tiempo, porque el espacio libre que no esté en el mapa no se puede volver a utilizarse, sino PostgreSQL solicitará más espacio en disco cuando se necesita almacenar nuevos datos.

HL7: es una organización internacional, iniciada en los Estados Unidos en 1987, que pretende promover el desarrollo y evolución del estándar HL7 (*Health Level Seven*) para el formato de datos e intercambio de información entre diferentes Sistemas de Información de Salud.

HQL (Hibernate Query Language): Lenguaje de consultas en Hibernate.

IDE (Integrated Development Environment): entorno de desarrollo integrado): un editor de código que sirve para depurar y facilitar las diferentes tareas necesarias en el desarrollo de cualquier tipo de aplicación.

J2EE (Java 2 Enterprise Edition): Son una serie de tecnologías para el desarrollo de aplicaciones Web empresariales.

GLOSARIO DE TERMINOS

JEE: Plataforma de programación para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java.

JSF: aplicaciones basadas en web.

LexBIG: Terminología servidor

LexGRID: Modelo de datos. Proporciona apoyo a una red distribuida de recursos léxicos, como las terminologías y ontologías a través de herramientas basadas en estándares, formatos de almacenamiento y el acceso y mecanismos de actualización.

OMS: es el organismo de la Organización de las Naciones Unidas (ONU) especializado en gestionar políticas de prevención, promoción e intervención en salud a nivel mundial. Organizada por iniciativa del Consejo Económico y Social de la ONU, se redactan los primeros estatutos de la OMS.

POJOs (Plain Old Object Java): Se trata de un objeto Java que no se extiende o implemente algunas clases e interfaces especializadas respectivamente requieren del marco de EJB.

SEI: Instituto de Ingeniería del Software de la Universidad Carneige Mellon.

SGBD (Sistema Gestor de Base de Datos): Es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

Skinnability: cambios de temas.

SNOMED CT: Términos Clínicos

XMI: Es un estándar para definir los nombres y propiedades de los ítems de datos que se intercambian entre el servicio solicitante y el servicio proveedor.

XML: lenguaje de etiquetado extensible.