Universidad de las Ciencias Informáticas.

Facultad 3

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

Título: Módulo de Administración y Configuración para el Repositorio de Esquemas y Metadatos versión 2.0

Autor: Michel Aguilera Martínez.

Tutor: Ing. Reinier Lugo Díaz.

La Habana, Cuba ¨Año 55 de la Revolución¨

Junio del 2013

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Michel Aguilera Martínez. Reinier Lugo Díaz.

_____ _____

(Autor) (Tutor)

"Se puede adquirir conocimientos y conciencia a lo largo de toda la vida, pero jamás en ninguna otra época de su existencia una persona volverá a tener la pureza y el desinterés con que, siendo joven, se enfrenta a la vida."

Agradecimientos

A toda mi familia, en especial a mi mamá y a mi papá, que me han dado todo su apoyo a lo largo de mi carrera. A mi mamá por todo el amor y a mi papá por su ejemplo de sacrificio. A mi hermano, que ha sido hermano y amigo y una guía en más de una ocasión, estoy muy orgulloso de que estén aquí. A mi cuñada y mi sobrina. Gracias por todo.

A mis amigos y compañeros que me ayudaron de muchas maneras dándome su confianza para cumplir con esta meta. A los profesores que me impartieron clases en estos años.

A Nany que ha estado conmigo en las buenas y en las malas durante el transcurso de la carrera.

Los compañeros de aula que durante 5 años hemos estado pasando noches sin dormir cuando teníamos prueba, a Félix, Robin, Juan Carlos, Sandy, Luis Ángel, Osvey, Daniel, Lázaro, Ángel Eduardo, y a todos los que se me quedan muchas gracias.

A mi grupo de baile Los MoonWalkers, por pasar tantas horas de ensayo conmigo, me enseñaron que para que las cosas salgan bien llevan sacrificio, responsabilidad y compromiso. Heiler, Alexis por su paciencia, Osvaldo, Félix, Eidel, Ricnelis, la gente de Infodanz, Vel por su ayuda, Pedro, Ricnelys y todos los demás.

A la revolución por darme la oportunidad de estudiar en esta escuela.

A todas las personas que de una forma u otra me ayudaron en la realización de este trabajo. Gracias.

Dedicatoria

A mi mamá por su preocupación, su cariño y todo su amor, le agradezco las horas que ha dedicado en mi formación. La mejor madre del mundo.

A mi papá, mi guía y ejemplo, sin él no podría ser la persona que soy, padre y amigo, quien me ha enseñado que con sacrificio se logra todo en la vida, por lo responsable y humano que es, nunca voy a olvidar lo que significa la familia para ti.

A mi hermano, quien tuvo la responsabilidad de guiarme en mi formación, ser mi ejemplo a seguir, estoy orgulloso de tener un hermano como él.

A toda mi familia por su apoyo y preocupación para que todo me saliera bien, a mis abuelos y mis tíos.

A mis verdaderos amigos por estar ahí en el momento que los necesitaba, a Nany, Alexis, Manolo, Heiler, Tribilin, siempre los tendré presentes donde quiera que esté, gracias por ser mis amigos.

Resumen

El intercambio de datos e información en un escenario de eGobierno permite la interoperabilidad entre los diferentes factores que componen cualquier plataforma de esta naturaleza. El Órgano de Justicia MININT en Cuba se encarga de conducir la toma de decisiones en Política Penal y Penitenciaria, el cual desarrolló una plataforma de interoperabilidad para este intercambio de información, la cual utiliza esquemas XML para definir el formato de intercambio de datos y mediante un Repositorio de Esquemas y Metadatos (REM) los diferentes sistemas puede acceder a los esquemas, registrarlos y validarlos de forma centralizada.

El presente trabajo describe el módulo de administración y configuración correspondiente al REM de la Plataforma de Interoperabilidad. Este módulo está basado en la gestión de la administración y la configuración y ofrece funcionalidades de este tipo. El estudio basó su fundamentación teórica en el análisis valorativo de la administración y configuración, lo cual permitió adquirir conocimientos necesarios para proponer un nuevo diseño e implementación del módulo para el REM en su versión 2.0.

Como resultado del presente trabajo se obtuvieron los artefactos: definición de procesos, Lista de Reserva del Producto, Historias de Usuarios, tarjetas de Clases, Responsabilidades y Colaboraciones y él modelo de datos físico. La validación de la propuesta de solución incluyó la aplicación de métricas a la Lista de Reserva del Producto para valorar su calidad permitiendo detectar y corregir ágilmente los errores u omisiones encontradas. Finalmente se validó la propuesta a través de pruebas que aseguraron su confiabilidad.

Palabras clave: esquemas XML, eGobierno, interoperabilidad, repositorio de esquemas y metadatos.

Índice

Introducción 1
Capítulo 1: Fundamentación Teórica 6
1.1. Conceptos Fundamentales 6
1.1.1. Interoperabilidad 6
1.1.2. Gobernanza de Interoperabilidad. 6
1.1.3. Interoperabilidad en el entorno del eGobierno 7
1.1.4. Repositorio de Esquemas y Metadatos 7
1.2. Servicios de intercambio/exposición de datos (servicios web): 9
1.3. Sistemas Informáticos Existentes Vinculados al Campo de Acción: 9
1.3.1. Administrador de Esquemas y Metadatos de Chile. 9
1.3.2. Portal del lenguaje común de intercambio de información de Colombia. 10
1.3.3. Sistemas Informáticos en Cuba 11
1.3.4. Valoración crítica de los sistemas analizados. 12
1.4. Administración y Configuración del Repositorio de Esquemas y Metadatos 13
1.5. Proceso de Desarrollo 14
1.5.1. Metodología adoptada para el desarrollo. Extreme Programming (XP) 14
1.6. Tecnologías, lenguajes y herramientas 15
1.6.1. Herramientas para el modelado 15
Visual Paradigm 8.0 16
1.6.2. Entorno de Desarrollo Integrado (IDE) 16
1.6.3. Lenguajes utilizados 17
Java 17
XML 17
UML 2.3 18
1.6.4. Sistema Gestor de Base de Datos 19
PostgreSQL 9.2 19
1.6.5. Marco de Trabajo (Framework) 20
RichFaces Framework 3.3.2 20
Hibernate 4.0 20
SEAM Framework 2.1 21
Servidor de aplicaciones web JBoss AG 22
Conclusiones del Capítulo 22
Capítulo 2: Características del Módulo 23

2.1 Fase de Planificación ... 23
2.1.1 Historias de Usuario (HU) ... 23
2.1.2 Tareas de Ingeniería (Task Card) ... 26
2.1.3 Lista de reserva del producto ... 27
2.1.4 Requisitos funcionales ... 33
2.1.5 Requisitos no funcionales del sistema ... 34
2.2 Validación de Requisitos ... 36
2.2.1 Métricas para la Validación de los Requisitos ... 36
2.2.2 Planificación de entregas ... 38
2.3 Fase de Diseño ... 38
2.3.1 Tarjetas Clase, Responsabilidades y Colaboración (C.R.C) ... 39
2.3.2 Validación del Diseño ... 40
2.3.3 Métricas para la validación del diseño ... 40
Resultados de la métrica TOC ... 41
Resultados de la métrica RC ... 43
2.4 Arquitectura del Software ... 43
2.4.1 Arquitectura de seguridad ... 45
2.4.2 Modelo de datos (entidades persistentes) ... 47
2.4.3 Patrones de diseño empleados ... 52
Aplicación de los patrones GRASP ... 53
Aplicación de patrones GoF. ... 54
2.4.4 Clases de Diseño ... 55
Conclusiones del capítulo ... 56
Capítulo 3 Implementación y Pruebas ... 57
3.1 Fase de Implementación ... 57
3.2 Tareas programadas por iteraciones ... 57
3.3 Estándares de codificación ... 59
3.4 Diagramas de Despliegue ... 60
3.5 Fase de Prueba ... 61
3.5.1 Pruebas de caja blanca ... 61
3.5.2 Pruebas de caja negra ... 64
3.6 Lista de Chequeo ... 66
Conclusiones del Capítulo ... 66
Conclusiones Generales ... 67
Recomendaciones ... 68

Bibliografía 69

Índice de Figuras

Figura 1: Implementación de la Arquitectura en capas del Framework SEAM 44
Figura 2: Entidades relacionadas con la HU Administrar Usuario 48
Figura 3: Entidades Relacionadas a la HU Administrar Órgano 48
Figura 4: Entidades relacionadas con la HU Administrar Rol 49
Figura 5 Entidades Relacionadas a la HU Registrar y Listar Trazas 50
Figura 6 Entidades Relacionadas con la HU Administrar Configuración del Sistema 51
Figura 7 Entidades Relacionadas con la HU Administrar Servicios Web 52
Figura 8 Entidades Relacionadas a la HU Administrar Contraseña 52
Figura 9 Diagrama de Clases del Diseño Buscar Usuario 55
Figura 10: Diagrama de Despliegue 61
Figura 11 Grafo de Flujo asociado a la funcionalidad Mostrar Detalles de Usuario 62
Figura 12 Resultados de las pruebas funcionales en 3 iteraciones. 64

Índice de Tablas

Tabla 1 Historias de Usuario agrupadas 24
Tabla 2 Operaciones asociadas por Historias de Usuarios 25
Tabla 3 Historias de Usuario Registrar y Listar Traza 26
Tabla 4 Administrar configuración del sistema 26
Tabla 5 Tarea de Ingeniería Administrar Contraseña (TC14) 27
Tabla 6 Especificación del requisito funcional Registrar trazas del sistema 33
Tabla 7 Especificación del requisito funcional Registrar Servicio Web. 34
Tabla 8 Tarjeta CRC UsuarioDetailsManager 39
Tabla 9 Tarjeta CRC UsuarioDeleteManager 40
Tabla 10 Clasificación de las clases según el umbral. 41
Tabla 11 Rango de valores para medir la afectación de los atributos de calidad (TOC). 41
Tabla 12 Rango de valores para medir la afectación de los atributos de calidad (RC). 43
Tabla 13 Estimación y Tiempo Real de Implementación por cada Historia de Usuario 57
Tabla 14 Tareas de Programación por cada Historia de Usuario. 58
Tabla 15 Tiempo estimado y real de implementación por cada historia de usuario 58
Tabla 16 Tareas de programación por cada historia de usuario 59
Tabla 17 Caso de Prueba Autenticar Usuario 65
Tabla 18 Caso de Prueba Administrar Usuario 65

Introducción

La gestión de la información hoy en día se encuentra en constante desarrollo, las tecnologías de la información y la comunicación (TIC) han evidenciado que permiten automatizar de manera sencilla diferentes servicios en cualquier país, ahorrándole así una gran cantidad de dinero y permitiéndole a los ciudadanos y entidades acceder fácilmente a estos servicios. Cuba, en su esfuerzo por reducir la brecha digital, se ha mantenido dando pasos en pos de informatizar todos los sectores de la sociedad, una muestra de ello es el desarrollo de una plataforma de interoperabilidad para el intercambio de datos entre los sistemas que comprenden el Grupo de Coordinación de la Política Penal y Penitenciaria (Órgano de Justicia MININT[1]). Para lograrlo, se precisó la homogenización del formato de intercambio de datos, utilizándose varios estándares internacionales de interoperabilidad, como son XML, XSD (XML Schema Definition) (1), Servicios Web, entre otros.

El Órgano de Justicia MININT es una institución del estado que está constituido por el Ministerio de Justicia, la Fiscalía General de la República de Cuba (FGR), el Tribunal Supremo Popular (TSP) y el Ministerio del Interior. Un aspecto fundamental para este órgano es la toma de decisiones a nivel de estado en materia política penal y penitenciaria, sobre los aspectos más significativos del Sistema de Justicia, a partir de una serie de indicadores y datos que proveen las diferentes instituciones adscritas.

Un componente muy importante para el desarrollo de la plataforma de interoperabilidad del Órgano de Justicia MININT es el Repositorio de Esquemas y Metadatos (REM) que no es más que un sistema que brinda la posibilidad de registrar esquemas XSD utilizados para la estandarización del intercambio de datos entre los sistemas informáticos que intervienen en procesos jurídicos. El REM actualmente cuenta con un sistema de administración basado en la web, que permite el registro, gestión y publicación de esquemas y metadatos, a partir del cual los sistemas informáticos pueden acceder a los esquemas XML que definen el formato de intercambio de datos.

Un importante requerimiento de seguridad que se incumple en la versión anterior específicamente asociado a la administración del sistema, es el referido a la capacidad

[1] Ministerio del Interior

que tenga el mismo de proveer un mecanismo de dualidad[2] en la administración. La primera se refiere al mantenimiento técnico del sistema y el manejo de las operaciones de seguridad asociadas a la auditoria, definición de roles, administración de usuarios, configuración y administración de los servidores. La segunda está más vinculada al manejo de aquellas operaciones que determinan la correspondencia del funcionamiento del sistema con el funcionamiento de la organización, o sea el órgano en cuestión.

Después de haber realizado un análisis de la versión actual del REM se ha evidenciado un conjunto de inconvenientes que dificultan un correcto funcionamiento del mismo:

- ✓ Actualmente el módulo de Administración y Configuración posee un modelo de datos que no utiliza un estándar de nomenclatura para nombrar sus tablas, objetos y atributos.
- ✓ Está desarrollado en lenguaje PHP, y uno de los requisitos pedido por el cliente es que fuese desarrollado en Java.
- ✓ El acceso se realiza de manera no segura usando solo el protocolo http.
- ✓ No implementa el acceso a los esquemas mediante el uso de servicios web.
- ✓ No implementa funcionalidades básicas de administración y configuración como son:
    - Gestión de usuarios y roles.
    - Gestión de las funcionalidades del sistema.
    - Gestión de los permisos de acceso de los usuarios y roles a las diferentes funcionalidades del sistema.
    - Configurar notificaciones a los usuarios
    - Configurar el servicio de correo electrónico.

Atendiendo a estas necesidades, el Proyecto Interoperabilidad tiene como prioridad el desarrollo de una nueva versión del módulo de administración y configuración para el Repositorio de Esquemas y Metadatos.

[2] Administración técnica y administración funcional.

Teniendo en cuenta lo antes mencionado queda definido el siguiente **problema a resolver**: ¿Cómo mejorar el módulo de administración y configuración para un correcto manejo de la administración y la configuración de los datos que se intercambian en el REM?

El **Objeto de estudio** está enmarcado en**:** La interoperabilidad entre sistemas.
El **campo de acción** se centra en la interoperabilidad entre los sistemas basados en el estándar XML.

Para ayudar a resolver el Problema de la Investigación enunciado con anterioridad se plantea como **Objetivo General:** Desarrollar una 2da versión del módulo de administración y configuración que considere en su concepción el intercambio de consultas entre sistemas a través de una interfaz de servicios web y elementos relativos a la visualización y estandarización de los esquemas, para el correcto manejo de la administración y configuración a partir de los datos que se intercambian en el REM.

Los **Objetivos Específicos** de este trabajo son los siguientes:

- ✓ Fundamentar teóricamente la propuesta de solución para el desarrollo del módulo de administración y configuración del REM versión 2.0.
- ✓ Identificar mecanismos de interoperabilidad basados en el uso XML para la estandarización a partir del estudio de marcos normativos y experiencias en el ámbito internacional.
- ✓ Diseñar e Implementar el módulo de Administración y Configuración del REM de acuerdo con los requisitos obtenidos.
- ✓ Validar la solución propuesta.

Se define como **Idea a Defender** Considerando el trabajo de las consultas entre sistemas a través de interfaces de servicios web y elementos relativos a la visualización y estandarización de los esquemas en el módulo de administración y configuración se obtendrá una versión mejorada del módulo, que permita un correcto manejo de los datos que se intercambian en el REM.

Los **Resultados** esperados al término de esta investigación son:
El módulo de administración y configuración del Repositorio Esquemas y Metadatos v2.0 de la plataforma de interoperabilidad del Órgano de Justicia MININT.

Con el fin de darle solución a los objetivos específicos se definieron como **Tareas de la investigación** las siguientes:

1. Elaboración del marco teórico del tema de investigación.
2. Caracterización de las herramientas y tecnologías para el desarrollo del módulo de administración y configuración del repositorio de esquemas y metadatos.
3. Identificación de los requisitos funcionales y no funcionales.
4. Validación de los requisitos funcionales y no funcionales.
5. Realización de las historias de usuarios y tarjetas CRC
6. Realización del diseño de la base de datos del módulo de administración y configuración.
7. Realización de los diagramas de componentes del módulo de administración y configuración.
8. Realización del diagrama de despliegue del módulo de administración y configuración.
9. Implementación del módulo de administración y configuración del repositorio de esquemas y metadatos.
10. Realización de las pruebas de validación del producto.
11. Redacción del informe final de la investigación.

Para el correcto cumplimiento de las tareas anteriormente enunciadas se tuvieron en cuenta los siguientes métodos científicos de investigación:

Métodos teóricos

- **Analítico Sintético:** este método permitió realizar el estudio teórico de la investigación, haciendo posible el análisis de documentos y la selección de los elementos más importantes en el proceso de desarrollo del módulo.

- **Histórico-Lógico**: permitió consultar la bibliografía necesaria para dar cumplimiento a las tareas de la investigación y resumir los aspectos fundamentales de cada una de ellas.

- **Modelación:** se utilizó con el objetivo de reproducir la interacción de los objetos en la vida real mediante la creación de prototipos funcionales y la representación de los requisitos del producto.

Métodos empíricos

- **Entrevista:** se emplea para definir las necesidades reales de los clientes, para la definición de los requisitos del sistema y obtención de información necesaria para el desarrollo de la investigación.
- **Medición:** permitió medir los atributos de calidad seguridad (informática), precisión, tolerancia ante fallos, operabilidad y atracción del nuevo módulo para determinar en qué medida se mejoraron los problemas que presentaba el anterior.

El presente trabajo de diploma quedará estructurado en tres capítulos:
Capítulo 1: Fundamentación teórica:

En este capítulo se realiza una revisión y análisis de los elementos generales de los sistemas de apoyo a la toma de decisiones, particularizando en los escenarios de administración y configuración. Además en este capítulo se describen la metodología y tecnologías empleadas para el desarrollo del módulo. Se describen los principales conceptos relacionados con el tema. Se realiza un estudio sobre los elementos teóricos y las soluciones similares en otros países, un estudio del software, tecnologías y herramientas utilizadas, analizando sus características, ventajas y desventajas.

Capítulo 2: Características del Módulo:

En este capítulo se describen los requerimientos funcionales y no funcionales asociados al módulo de Administración y Configuración del REM. Además se describe el comportamiento general del sistema a partir de las Historias de Usuario, como artefacto[3] generado con este fin por la metodología empleada; y se presentan los planes de iteraciones y de entrega del producto.

Capítulo 3: Implementación y pruebas:

En este capítulo se evalúa el cumplimiento de los objetivos planteados a partir del diseño y realización de pruebas funcionales al sistema Así como las validaciones a la solución propuesta a través del método de prueba de caja negra y los casos de pruebas correspondientes.

[3] Artefacto: diagrama o documento obtenido durante el desarrollo.

Capítulo 1: Fundamentación Teórica

Este capítulo versa sobre los conceptos fundamentales que se relacionan con el tema y que argumentan el desarrollo del trabajo. También incluye la conceptualización del REM. Se realiza una breve descripción de las características de la administración y configuración del mismo. También se realiza una descripción de las herramientas y las tecnologías que se emplearon en el desarrollo del módulo de administración y configuración y a la vez se describen las características de la metodología utilizada en el proceso de desarrollo del software que fueron empleadas para garantizar el desarrollo del mismo.

1.1. Conceptos Fundamentales

1.1.1. Interoperabilidad

Según el EIFv2 (European Interoperability Framework versión 2), “la interoperabilidad en el contexto de los sistemas informáticos es la habilidad de los sistemas TIC, y de los procesos de negocios que ellas soportan, de intercambiar datos y posibilitar el compartimiento de información y conocimientos.” (2) Un framework de interoperabilidad puede ser definido como un conjunto de estándares y directrices que describen la forma por la cual organizaciones han acordado, o pueden acordar, interactuar unas con las otras.

La interoperabilidad según la norma ISO 19119, es la capacidad para comunicar, ejecutar programas, o transferir datos entre varias unidades funcionales sin necesitar que el usuario tenga conocimiento de las características de esas unidades.

La interoperabilidad es la capacidad que tiene un producto o un sistema, cuyas interfaces son totalmente conocidas, para funcionar con otros productos o sistemas existentes o futuros y eso sin restricción de acceso o de implementación. (3)

1.1.2. Gobernanza de Interoperabilidad.

La gobernanza de interoperabilidad, según la Comisión Europea y la European Public Administration Network (EUPAN), “se ocupa de las condiciones políticas, legales y estructurales que sean relevantes para el desarrollo y utilización de aplicaciones interoperables”, así como “de la coordinación y alineación de los procesos de negocios y

arquitecturas de información que traspasan los límites intra y extra organizacionales. Su propósito es identificar y enderezar/remover cualquier posible barrera, incluyendo las legislativas, culturales y otras, buscando añadir servicios y compartir información". (4)

1.1.3. Interoperabilidad en el entorno del eGobierno

El intercambio de datos e información en un escenario de Gobierno Electrónico (eGobierno) es uno de los principales requerimientos de interoperabilidad de cualquier plataforma de esta naturaleza. El Grupo de Coordinación de la Política Penal y Penitenciaria en Cuba se encarga de conducir la toma de decisiones en Política Penal y Penitenciaria. Los datos que la Comisión Nacional de este órgano necesita para el proceso de toma de decisiones, son tributados por los diferentes sistemas de información de cada uno de los organismos que participan en la toma de decisiones de los procesos penales y penitenciarios. Los sistemas que intercambian información deberán ofrecer solo los indicadores que se necesitan para dar solución a determinada situación, y los datos se deberán consultar haciendo uso de servicios web proporcionados por cada uno de los involucrados en el proceso de toma de decisiones, cada uno de los indicadores que emiten los organismos miembros del Órgano Justicia MININT son de conocimiento restringido, al igual que los informes que se definen a partir de los indicadores. Por tal motivo, la definición de los permisos asociados a quienes pueden manipular dichos indicadores e informes, es un elemento a tener en cuenta por parte de los sistemas informáticos que intervienen en este proceso (5).

1.1.4. Repositorio de Esquemas y Metadatos

El Repositorio de Esquemas y Metadatos es un sistema informático que permite definir un registro (o repositorio) de todos los archivos XSD a partir de los cuales se estandariza el formato de intercambio de datos (en una primera fase), documentos e información (en fases posteriores). Es un sistema basado en la web, e implementa un procedimiento de registro de archivos XSD que establece un conjunto de pasos y reglas para realizar este importante proceso. Su principal función es brindar un mecanismo centralizado y público para que todos los sistemas que necesiten intercambiar datos puedan acceder, mediante peticiones HTTP, a los formatos en que dichos datos deberán intercambiarse. (6)

Entre los principales conceptos del REM se encuentran:

Esquemas del formato de intercambio (XML Schema o esquemas XSD):

XML Schema es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. Se consigue así una percepción del tipo de documento con un nivel alto de abstracción. Fue desarrollado por el World Wide Web Consortium (W3C) y alcanzó el nivel de recomendación en mayo de 2001. Se refiere a archivos XSD que definen el formato en que ciertos indicadores serán proveídos por los diferentes sistemas informáticos. No obstante, en un sentido más general permitirán definir el formato no solo de datos, sino de otras informaciones y documentos que deban ser intercambiados entre sistemas informáticos para realizar los procesos de cada organismo (u organización en sentido general). (7)

Estándar abierto

Un estándar abierto es un estándar que se encuentra disponible públicamente y que tiene varios derechos de uso asociados, y puede incluir propiedades sobre cómo fue diseñado. Los estándares abiertos proveen apertura tanto en el proceso de adopción como en el acceso a las especificaciones. Estos elementos marcan la diferencia con estándares propietarios. Bruce Perens (8) define las siguientes características:

1. **Disponibilidad:** Los estándares abiertos se encuentran disponibles públicamente para la implementación y lectura.
2. **Maximizar la libertad de elección del usuario final:** Crean un mercado justo y competitivo para las implementaciones del estándar. No atan al usuario a un desarrollador o grupo específico.
3. **Sin realeza:** Los estándares abiertos son libres para la implementación sin realeza o costo. La certificación de cumplimiento de la organización que provee el estándar puede incluir un cobro.
4. **Sin discriminación:** Los estándares abiertos y las organizaciones que los administran no favorecen a una implementación sobre otra por cualquier razón que no sea el cumplimiento de los estándares técnicos por el desarrollador. Las organizaciones de certificación deben proveer vías para que las implementaciones

de bajo o ningún costo sean validadas, pero también pueden proveer servicios de certificación avanzados.

5. **Extensión o subconjunto:** Las implementaciones de estándares abiertos pueden ser extendidas u ofrecidas como subconjunto. Sin embargo, las organizaciones certificantes pueden declinar certificar subconjuntos de los estándares, y pueden establecer requerimientos sobre las extensiones (ver prácticas predatorias).
6. **Prácticas predatorias:** Los estándares abiertos pueden emplear términos de licencia que protegen contra la desnaturalización del estándar por tácticas de adoptar-y-extender. Las licencias adjuntas al estándar pueden requerir la publicación de información de referencia para las extensiones, y una licencia para el resto para crear, distribuir y vender software compatible con las extensiones.

1.2. Servicios de intercambio/exposición de datos (servicios web):

Un servicio web (en inglés, Web Service) es una pieza de software que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. Cada uno de los sistemas que tributan datos (indicadores) deben implementar servicios web que permitan exponer dichos datos para que otros sistemas, según las disposiciones de seguridad establecidas, puedan acceder a ellos sin la necesidad de acceder a las fuentes de datos originales y cumpliendo de este modo con el principio de no intrusión. (9)

1.3. Sistemas Informáticos Existentes Vinculados al Campo de Acción:

1.3.1. Administrador de Esquemas y Metadatos de Chile.

Uno de los grandes sistemas en el mundo Administrador de Esquemas y Metadatos se encuentra en Chile. Es el organismo público encargado de administrar y mantener operativo, el procedimiento de inscripción de esquemas basales y documentales por parte de los Órganos de la Administración del Estado, su evaluación técnica y posterior

publicación en el Repositorio de Acceso Público de Esquemas de Gobierno, de acuerdo con lo señalado en el Decreto Supremo Nº 271 de 2008.

El Administrador gestiona:

- ✓ Un sitio Web de dominio público.
- ✓ Una sección del sitio Web donde los usuarios de los órganos del Estado entran con nombre de usuario y contraseña.
- ✓ Un repositorio de esquemas.
- ✓ Un foro para comentar esquemas.
- ✓ Guía de Desarrollo y uso de esquemas de Gobierno.
- ✓ Otros documentos de apoyo.

Constituye la solución técnica al proceso de estandarización para la gestión de documentos electrónicos en los ámbitos público y privado. Este sistema permite el registro, administración y publicación de todos los esquemas XML que definen la estructura de aquellos documentos que se intercambian entre sistemas informáticos en la plataforma de Gobierno Electrónico de Chile. El 13 de enero del 2009 se publica en el Diario Oficial, el Decreto Supremo N° 271 de 2008, en el que se aprueba el reglamento sobre la Inscripción de Esquemas Documentales en el Repositorio del Repositorio de Esquemas y Metadatos para los Órganos de la Administración del Estado. (10)

El Repositorio de Esquemas y Metadatos tiene por objeto regular el procedimiento de inscripción de esquemas XML necesarios para la definición de estándares de intercambio de datos entre Organismos Públicos. Los esquemas inscritos estarán sujetos a una evaluación técnica y posteriormente serán publicados en este mismo sitio. Es un sistema sencillo desde el punto de vista de la implementación. Su interfaz es simple y fácil de usar. Introduciendo criterios de búsqueda se puede acceder a los esquemas registrados así como a la información asociada al mismo. En definitiva cumple su propósito satisfactoriamente.

1.3.2. Portal del lenguaje común de intercambio de información de Colombia.

Otro de los grandes sistemas es el estándar definido por el Estado Colombiano para intercambiar información entre organizaciones, facilitando el entendimiento de los involucrados en los procesos de intercambio de información, el mismo contempla:

- ✓ Definición estructural y funcional, de un organismo responsable de la administración y gestión del estándar.
- ✓ Definición de una arquitectura de datos, que soporte funcional y conceptualmente la definición, documentación, adopción y uso de los diferentes conceptos susceptibles de ser intercambiados.
- ✓ Definición de un proceso de mantenimiento y evolución del estándar que abarque tanto la especificación funcional del mismo, como la técnica.

Para organizar los elementos de datos, el Lenguaje común de intercambio de información cuenta con una arquitectura de datos, basada en capas. Cada una de las capas incluye los elementos de datos que tienen el ámbito de aplicación de la capa.

Los ámbitos de aplicación de las capas de la arquitectura son:

- ✓ **Uso Común:** conceptos que tienen la misma interpretación o significado, independiente de la localización geográfica o proceso de intercambio de información en el que se utilice.
- ✓ **Uso Local:** conceptos que tienen la misma interpretación o significado, independiente de la entidad, trámite, servicio o proceso de intercambio de información en el que se utilice, a nivel nacional.
- ✓ **Uso Sectorial:** conceptos que tienen una interpretación particular, o son de uso específico, en un sector.
- ✓ **Uso Proyectos:** conceptos que son de uso específico en un determinado proyecto de una organización.
- ✓ **Uso Servicios de la Plataforma de Interoperabilidad – PDI:** conceptos que son de uso específico de la Plataforma de Interoperabilidad.
- ✓ **Uso Internacional:** Concepto que tiene elementos de datos que hacen parte o se encuentran definidos en estándares internacionales y que cubren las necesidades del Lenguaje común de intercambio de información.

1.3.3. Sistemas Informáticos en Cuba

El desarrollo y crecimiento de las estrategias que forman parte del llamado *Movimiento de Acceso Abierto* (11), representa un beneficio fundamental para la comunicación científica mundial. Los repositorios institucionales, que atesoran la producción científica de los

miembros de una institución científica o académica, constituyen la vía más propicia para elevar la visibilidad y el impacto de la investigación. En cuba desde los años 80 se vienen incorporando estudios acerca del tema de la interoperabilidad y el uso de repositorios de datos, con vistas a la durabilidad de la información.

Scielo es una biblioteca electrónica que incluye una colección seleccionada de revistas científicas cubanas en todas las áreas del conocimiento. Desarrollado por la Red Telemática de Salud en Cuba (INFOMED) en colaboración con el Centro Latinoamericano y del Caribe de Información en Ciencias de la Salud de Brasil (BIREME), forma parte de un proyecto de carácter regional. (12)

También se han dado pasos en vistas a crear repositorios de tesis doctorales para el sector de la salud y en general para los repositorios electrónicos por parte de las empresas vinculadas al Movimiento de Acceso Abierto y a la comunidad científica nacional.

En la UCI se desarrolló un Repositorio de Esquemas y Metadatos con el fin de lograr el intercambio de información entre los sistemas del Órgano de Justicia MININT de Cuba. Esta plataforma utiliza esquemas XML para definir el formato de intercambio de datos. Sin embargo, no existe una forma en que los sistemas puedan acceder a los esquemas, ni de registrar y validar los mismos de forma centralizada. La gestión gubernamental en los últimos años ha estado enfocada a la búsqueda de una solución viable para implementar una plataforma de Gobierno Electrónico (eGobierno) que satisfaga sus necesidades. Esto motivado por las satisfacciones que ofrece una plataforma de este tipo en cuanto a la agilización de los procesos de gobierno.

1.3.4. Valoración crítica de los sistemas analizados.

Habiendo analizado las iniciativas internacionales chilena y colombiana, existen elementos a tener en cuenta para la implementación del sistema. La simplicidad de la interfaz del sistema AEM chileno, la facilidad en el acceso a los esquemas y la ligereza del sitio, se encuentran entre los aspectos más importantes, también posee funcionalidades importantes como un foro para debatir los esquemas, una guía de desarrollo y uso de los esquemas del gobierno y otros documentos de apoyo, no se puede emplear esta solución debido que no se está realizando un sistema a nivel de gobierno, y

no se posee la infraestructura tecnológica capaz de abarcar tales aspectos, por otra parte el Portal del Lenguaje común de intercambio de información de Colombia, el cual ya posee un organismo que es el encargado de la administración y gestión del estándar de intercambio de datos, con lo cual no se cuenta actualmente en nuestro país.

Basándose en algunas funcionalidades que poseen estos sistemas se puede destacar como parte de la solución el uso sectorial que tienen los mismos, conceptos que tiene elementos de datos que hacen parte o se encuentran definidos mediante estándares internacionales y que cubren las necesidades del lenguaje estándar en el intercambio de información.

1.4. Administración y Configuración del Repositorio de Esquemas y Metadatos

Dentro de los requerimientos de seguridad que presenta el REM, se encuentran los asociados a la administración y configuración. Dado a que la seguridad de este sistema debe estar supervisada por dos grupos (administradores técnicos pertenecientes al MININT y administradores funcionales concernientes a la FGR), se define una administración dual. Es decir se establecen diferentes permisos de acceso por cada grupo.

La dualidad en la administración del sistema es una condición establecida por el Grupo de Coordinación. Esta es una forma de distribuir la responsabilidad de aplicar las disposiciones de seguridad, y tiene implícito un carácter burocrático, dado a que la implementación de determinada política de seguridad, estará justificada por algún instrumento administrativo que apruebe la ejecución de la misma. La administración funcional deberá establecer cuáles son los funcionarios que tienen acceso a dichos informes, sin embargo esto dependerá de que dichos funcionarios estén registrados en el sistema, y esto es una responsabilidad de la administración técnica a partir de una indicación de la administración funcional.

La implementación de una política de compartimentación de información está condicionada por los siguientes elementos:

1. Deben definirse cuáles son las fuentes de información (o recursos) a las cuales se les aplicarán las políticas de seguridad para el acceso a ellas.
2. Deben definirse los niveles de acceso a la información, que servirán para implementar la política de seguridad asociada a la compartimentación.

3. Los usuarios a los cuales se les otorgarán los permisos de accesos deben estar definidos.

La configuración en este caso no juega un papel significativo, sin embargo contribuye a complementar los mecanismos de seguridad, como por ejemplo a partir de la configuración de los mecanismos de autenticación. Sin embargo se considera que debe tenerse en cuenta debido a que ayuda a implementar mecanismos que contribuyan a incrementar el nivel de adaptación del sistema.

1.5. Proceso de Desarrollo

1.5.1. Metodología adoptada para el desarrollo. Extreme Programming (XP)

Teniendo en cuenta, las características del equipo desarrollado, los procesos a modelar, el corto plazo de producción y la participación activa y sistemática de los clientes como parte del equipo se decide utilizar la metodología de desarrollo XP. La misma se utiliza para proyectos de corto y mediano alcance, en pequeños equipos y plazos apretados de entrega. La metodología consiste, en un conjunto de prácticas, que se clasifican como de dudosa efectividad cuando se usan por separado, pero que juntas producen un aumento en la productividad y la habilidad del equipo, para enfrentar los cambios del proceso de desarrollo.

XP se traza como objetivo la satisfacción del usuario, esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, se debe responder muy rápido a las necesidades de este, incluso cuando los cambios sean al final del ciclo de la programación. El segundo objetivo es potenciar al máximo el trabajo en grupo, integrado por los jefes de proyecto, los desarrolladores y los clientes, los cuales están involucrados en el desarrollo del software (13). Esta metodología cuenta con un conjunto de artefactos que si bien forman parte del expediente de proyecto, algunos no contribuyen a un mejor entendimiento panorámico del sistema que se desarrolla. Es por eso que se decide utilizar el Diagrama de Despliegue que provee UML para una mejor comprensión sobre cómo estará dispuesto el sistema en un ambiente de producción.

El proceso define una serie de roles: Los roles se distribuyen entre los miembros del proyecto y que definen las tareas de cada uno y el resultado (artefactos) que se espera de ellos, como son Historias de Usuario, Tareas de Ingeniería y Tarjetas Clase-Responsabilidad-Colaboración (CRC). Las historias de usuario representan una breve descripción del comportamiento del sistema, emplea terminología del cliente sin lenguaje técnico, se realiza una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación. Las tarjetas CRC contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores.

1.6. Tecnologías, lenguajes y herramientas

En la actualidad, no es posible imaginarse el desarrollo de software sin antes pasar por una etapa de modelado. La utilización de modelos es una metodología aceptada y recomendada tanto en el ámbito académico como dentro del ámbito profesional privado. Se definieron por parte del equipo de desarrollo una serie de herramientas y lenguajes para llevar a cabo la implementación del sistema, descritas en el documento Proyecto Técnico de Interoperabilidad. Estas herramientas y frameworks están descritos a continuación.

1.6.1. Herramientas para el modelado

Las Herramientas CASE (Computer Aided Software Engineering, del español Ingeniería de Software Asistida por Computadoras) son aplicaciones de apoyo al desarrollo, mantenimiento y documentación automatizados de software. Permiten aplicar la metodología de análisis y diseño orientados a objetos. Cuanto más grande es un proyecto, más importante es utilizar una herramienta CASE, ya que, estas contribuyen a aumentar la productividad en el desarrollo de software reduciendo el coste en términos de tiempo y dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software. Al hacer uso de estas, los analistas pueden representar los requisitos del sistema con un modelo de casos de uso (14).

Visual Paradigm 8.0

Es una herramienta de software libre CASE (Ingeniería de Software Asistida por Computación). La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, y fue concebida para soportar el ciclo de vida completo del proceso de desarrollo del software, a través de la representación de un conjunto amplio de diagramas [9]. Esta herramienta es empleada para visualizar y diseñar elementos de software, para ello utiliza el lenguaje UML. Además, proporciona a los desarrolladores una plataforma que les permite diseñar un producto con calidad de forma rápida. Es multiplataforma, permite la generación de código y realiza ingeniería inversa para diferentes lenguajes de programación como: Java, C++, PHP. Esta herramienta se integra con el IDE Eclipse, con el cual se va a estar desarrollando el sistema, haciendo más fácil el proceso de desarrollo del software. Se utiliza esta versión ya que es estable y cuenta con soporte por parte de sus desarrolladores (15).

1.6.2. Entorno de Desarrollo Integrado (IDE)

Dentro de los posibles IDE a utilizar en el desarrollo del sistema se encuentra: Eclipse, Netbeans y JBoss Studio, este último es un software propietario por lo que si se emplea en el desarrollo del sistema se incurre en gastos financieros adicionales. Aunque Netbeans puede integrarse con JBoss, no se recomienda el uso de este IDE en el desarrollo de aplicaciones web donde se emplee el framework Seam.

Eclipse se ha convertido en uno de los entornos de desarrollo estándar para el desarrollo de aplicaciones en Java. Hay dos razones principales para ello. En primer lugar, es gratis. En segundo lugar, es fácil desarrollar plug-ins para Eclipse, esto ha permitido que JBoss haya creado su propio sistema de plug-ins, facilitando la integración de Seam con este IDE (16). Eclipse trabaja sobre un amplio rango de sistemas operativos y aunque soporta varios lenguajes de programación, es con Java con el que mejor se integra. La característica clave de Eclipse es su extensibilidad, ya que es una gran estructura formada por un núcleo y muchos plug-ins que van conformando la funcionalidad final. La forma en que los plug-ins interactúan es mediante interfaces o puntos de extensión, lo que posibilita que las nuevas aportaciones se integren sin dificultad ni conflictos. Otras de las características de este IDE son:

- ✓ Admite realizar pruebas unitarias con JUnit6[4], estas pruebas permiten detectar cualquier error en el código implementado.
- ✓ Asistentes para la creación de proyectos y clases, agiliza la implementación.
- ✓ Es soportado por sistemas operativos como Windows, Mac OS X y Linux. En caso de tener que emigrar a otro sistema operativo, este IDE puede persistir.

Este IDE es neutral, puesto que se ha utilizado para desarrollar todo tipo de aplicaciones, y ha sido altamente probado en cualquier ambiente, ya sea en la construcción de Servicios Web o aplicaciones de escritorios (17). Puede integrarse con Hibernate y con la herramienta de modelado Visual Paradigm.

1.6.3. Lenguajes utilizados

Java

Java es un lenguaje de programación, su sintaxis y semántica son lo bastante completas para poder abarcar programas de todo tipo. No es necesariamente un lenguaje para Internet, sino que puede ser usado en el desarrollo de un sistema a ejecutarse simplemente en una computadora.

Java puede ser ejecutado en múltiples plataformas. Es uno de los escasos lenguajes cuyos programas pueden ser transportados de sistema operativo, computadora o entorno, sin necesidad de cambiar el código .Esta característica ha sido la que alimentó su auge en Internet. El segundo aspecto a destacar, es que se trata de un lenguaje orientado a objetos. Esa calificación ha sido usada y abusada en muchos productos de computación, pero nunca tan bien aplicada como en Java, debido a que es realmente un lenguaje donde todo es un objeto.

Este lenguaje permite el desarrollo tanto de aplicaciones bajo el esquema de cliente servidor, como de aplicaciones distribuidas, lo que hace factible su empleo en el desarrollo del software en cuestión, siendo capaz de conectar dos o más computadoras u ordenadores, lográndose así la interoperabilidad entre los sistemas y permitiendo ejecutar tareas simultaneas, distribuyendo así el trabajo a realizar (18).

XML

Es un lenguaje extensible, después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.

[4] Framework utilizado para realizar pruebas unitarias

El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan bugs y se acelera el desarrollo de aplicaciones (19).

Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Se pueden comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, se podría tener una aplicación en Linux con una base de datos Postgres y comunicarla con otra aplicación en Windows y Base de Datos MS-SQL Server.

Se transforman datos en información, pues se le añade un significado concreto y se asocian a un contexto, con lo cual se obtiene flexibilidad para estructurar documentos. Se concluye considerando que XML es un medio idóneo para implementar la interoperabilidad de la información de carácter público. Lo cual se hace indispensable para el desarrollo de nuestra solución.

UML 2.3

El Lenguaje de Modelado Unificado (UML: Unified Modeling Language) es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90's. UML es llamado un lenguaje de modelado, no un método. Los métodos consisten de ambos de un lenguaje de modelado y de un proceso. El UML, fusiona los conceptos de la orientación a objetos aportados por Booch, OMT y OOSE (20). UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. Los autores de UML apuntaron también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios.

Actualmente es un lenguaje estándar y pertenece a la OMG (Object Management Group). Es una herramienta o lenguaje de modelamiento unificado que permite a los creadores de sistemas generar diseños que capturen sus ideas de una forma convencional y fácil de comprender y así poder comunicárselas a otras personas.

Se emplea para nuestro desarrollo ya que es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Se empleó para visualizar, especificar,

construir y documentar artefactos del sistema de software. Así como para realizar un adecuado proceso de análisis y diseño orientado a objetos del sistema. El mismo puede aplicar de diversas formas para dar soporte a una metodología de desarrollo de software pero, en sí mismo, no especifica qué metodología o proceso usar. Se utilizó la versión 2.3 ya que ha madurado considerablemente desde la versión 1.1, además de poder realizar diagramas para modelar el negocio, y ser definida esta versión para su uso por el equipo de desarrollo en el documento Proyecto Técnico de Interoperabilidad (21).

1.6.4. Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que administran y gestionan la información contenida en una base de datos. Ayuda a realizar la definición de los datos, el mantenimiento de la integridad de éstos dentro de la base de datos, el control de su seguridad y privacidad; así como la manipulación de los mismos

PostgreSQL 9.2

Se define como Sistema Gestor de Base de Datos (SGBD) PostgreSQL, debido a que es un software no propietario por tanto su utilización no incurre en gastos adicionales en desarrollo de la Plataforma de Interoperabilidad. Presenta una comunidad técnica internacional de la cual Cuba es miembro, además en la UCI existe un centro de desarrollo encargado de la personalización de este SGBD para las particularidades de las empresas cubanas, lo cual es otro elemento favorable.

Dentro de otras ventajas que se perciben a partir de la elección de PostgreSQL como SGBD, es que asegura el correcto funcionamiento y estabilidad del sistema en cualquier circunstancia. Sus últimas versiones son muy competitivas, siendo actualmente una de las alternativas más robustas, potentes en el mercado de los SGBD (22). Este SGBD emplea el modelo cliente/servidor y utiliza tecnología multiproceso en vez de multi-hilos, lo que asegura una mayor estabilidad al sistema, ya que en caso de fallar un proceso no afectará al resto. Funciona muy bien con grandes volúmenes de datos y con una alta concurrencia de usuarios, que acceden simultáneamente al sistema.

Entre sus características más importantes se encuentran que es una base de datos segura, tiene integridad referencial, tiene múltiples métodos de autentificación, posee

acceso encriptado vía SSL, provee una actualización integrada momentánea y una documentación completa. Es importante tener en cuenta que la UCI pertenece a la Comunidad Iberoamericana de PostgreSQL.

1.6.5. Marco de Trabajo (Framework)

Teniendo en cuenta que el REM está basado en la web, se tuvieron en cuenta algunos de los frameworks más utilizados sobre la plataforma JEE para el desarrollo de aplicaciones web, estos son:

- ✓ Java Server Face (JSF), para la presentación.
- ✓ Hibernate, para el acceso a datos.

Independientemente que estos sean los frameworks más empleados en Java para el desarrollo de aplicaciones web, la integración de estos suele ser un elemento preocupante en los desarrolladores. Este, entre otros elementos, es un problema que resuelve el framework Seam, creado por la empresa RedHat para el desarrollo de aplicaciones web en Java. Combina a los frameworks JavaServerFaces (JSF), modelo de componentes para la capa de presentación e Hibernate para la persistencia de datos. (23)

RichFaces Framework 3.3.2

Richfaces es una biblioteca de código abierto basada en Java que permite crear aplicaciones web con Ajax.

Construye sobre el framework de Java Server Faces. Sobre él, implementa unos filtros para permitir peticiones Ajax en la página. La singularidad del planteamiento que ofrece es que la petición Ajax provoca una ejecución en el servidor y finalmente una renderización parcial o total de la página del navegador. Así, el control de lo que sucede está en el servidor.

Java Server Faces es un framework para aplicaciones Java basadas en web, que simplifica el desarrollo de interfaz de usuario mediante componentes reutilizables, en aplicaciones Java EE. Una implementación de este framework la realiza RichFaces el cual está basado en JavaScript y también viene integrado en Seam Framework. El empleo de RichFaces permite la creación de vistas complejas, brindando una serie de componentes y funcionalidades soportados por Ajax que facilita la programación. (24)

Hibernate 4.0

La persistencia de datos de cualquier aplicación de software es uno de los aspectos más importantes que influye directamente en el desempeño y tiempos de desarrollo de cualquier aplicación de software.

Una de las técnicas más utilizadas hoy es la creación de objetos a partir de tablas de un modelo de bases de datos relacional. Las ventajas que ofrece esta estrategia han dado lugar al desarrollo de diversos frameworks. En este sentido Hibernate es uno de los frameworks ampliamente utilizados en el desarrollo de aplicaciones para el acceso a datos. El mismo se especializa en el mapeo de objetos a tablas de un modelo de bases de datos relacional y viceversa. Además, permite desarrollar un modelo de datos basado en un lenguaje orientado a objetos (HQL - Hibernate Query Language), incluyendo asociación, herencia, polimorfismo y composición. (25)

Hibernate está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a una base de datos ya existente. También permite simplificar el proceso de creación de aplicaciones, reduciendo drásticamente el tiempo de desarrollo. Y un elemento fundamental a tener en cuenta es que viene integrado con el framework Seam.

SEAM Framework 2.1

Seam facilita la creación de aplicaciones escribiendo clases simples en Java, con anotaciones, por lo que no es necesario extender cualquier superclases o interfaces especiales. También, permite algunas tareas comunes de programación, proveyendo un set de componentes pre construido, que puede volver a emplear en cualquier momento, reduce la cantidad de código que usted necesita escribir al hacer bases de datos básicas, usando Hibernate. (26)

Teniendo en cuenta lo anteriormente expuesto, se decide emplear como framework de desarrollo Seam, el cual por su característica integradora de varios frameworks de desarrollo (como RichFace e Hibernate), es considerado como una excelente opción para el desarrollo de aplicaciones Java basadas en la web con un enfoque empresarial.

Como servidor de aplicaciones se utilizó JBoss AG, que es un servidor que viene integrado con el framework SEAM, formando parte del mismo como núcleo de la plataforma, dado que en él se montaran todos los sistemas principales para el

funcionamiento de la misma. También proveerá otras tecnologías que en la medida que vaya evolucionando la plataforma irán siendo necesarias.

Servidor de aplicaciones web JBoss AG

Dentro de las tecnologías que provee JBoss, se encuentran: Buses de Mensajería (ESB); registros UDDI[5] para el descubrimiento, orquestación y composición de servicios web; motores de procesos, entre otras que si bien no serán necesarias en una primera fase de desarrollo, se considera una potencialidad el que el servidor JBoss las integre, lo cual es favorable para escalar la Plataforma hacia una solución SOA (27).

JBoss es un servidor de aplicaciones J2EE de código abierto, implementado en Java, por lo que puede ser utilizado en cualquier sistema operativo para el que esté disponible Java. La combinación de una arquitectura orientada a servicios con una licencia de código abierto, le permite a JBoss ser descargado, utilizado, incrustado y distribuido sin restricciones. (26)

Conclusiones del Capítulo

Como resultado del estudio de las definiciones relacionadas, se ha logrado obtener una base conceptual que permite un mejor entendimiento de los temas a tratarse posteriormente en el desarrollo de la solución. Se estudió y analizó la bibliografía consultada referente al tema de investigación. Una vez analizados los sistemas existentes, las condiciones y los problemas que caracterizan las soluciones internacionales y regionales, se determinó la necesidad de una aplicación Web, que cumpla con las necesidades y particularidades del Órgano Justicia MININT, de acuerdo a las características que debe tener este sistema en particular. El estudio de las tecnologías y metodologías permitió definir el marco tecnológico para el desarrollo del módulo, e identificar la metodología XP como la metodología a emplear en este trabajo.

[5] Universal Description, Discovery, and Integration.

Capítulo 2: Características del Módulo

En este capítulo se muestran las características que tendrá el módulo de administración y configuración del REM, se analizan las fases de Planificación y Diseño, en las cuales se detallan las Historias de Usuario, que permitan establecer el orden en que estas serán implementadas atendiendo a su prioridad, la Lista de Reserva de Producto, el plan de iteraciones y se describe la arquitectura utilizada en la implementación del módulo teniendo en cuenta el proceso de desarrollo del software que describe la metodología XP.

2.1 Fase de Planificación

En esta fase se establece la prioridad de cada Historia de Usuario, y se realiza una estimación del esfuerzo de cada una de ellas. Estas asociaciones van a estar relacionadas a su implementación, partiendo al equivalente que no sobrepasen de 3 puntos (un punto equivalente a una semana de trabajo). Se establece también la definición de dos artefactos importantes, la Lista de Reserva del Producto y los Requerimientos Funcionales y No Funcionales. (28)

2.1.1 Historias de Usuario (HU)

Los proyectos que utilizan la metodología XP deben definir las Historias de Usuario con el cliente, las cuales tienen la finalidad de un caso de uso pero con algunas divergencias. Representan una breve descripción del comportamiento del sistema, emplea terminología del cliente sin lenguaje técnico, se realiza una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación. No se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos. (28)

Estas deben proporcionar sólo el detalle suficiente como para poder hacer razonable la estimación de cuánto tiempo requiere la implementación de la historia, difiere de los casos de uso porque son escritos por el cliente, no por los programadores, empleando terminología del cliente. “Las historias de usuario son más “amigables” que los casos de uso formales”.

Las Historias de Usuario tienen tres aspectos:

- ✓ Tarjeta: en ella se almacena suficiente información para identificar y detallar la historia.
- ✓ Conversación: cliente y programadores discuten la historia para ampliar los detalles (verbalmente cuando sea posible, pero documentada cuando se requiera confirmación).
- ✓ Pruebas de Aceptación: permite confirmar que la historia ha sido implementada correctamente.

Las Historias de Usuario son especificadas en un documento Modelo de Historias de Usuario del Negocio del expediente de desarrollo del proyecto. En el que se describe cada una de las HU del sistema. Para el desarrollo del módulo de administración y configuración del REM, se agruparon las HU como se muestra a continuación:

No.	**Historias de Usuario**
1	Administrar Usuario
2	Administrar Rol
3	Administrar Órgano
4	Registrar y listar Trazas
5	Administrar Indicador
6	Administrar Informe
7	Administrar Configuración del sistema
8	Administrar permisos adicionales de usuario
9	Administrar permisos a Indicadores
10	Administrar permisos a Informes
11	Administrar Servicios Web
12	Listar y editar operaciones del sistema
13	Administrar asociaciones parte-contraparte
14	Administrar contraseña
15	Configurar notificaciones a usuarios
16	Configurar servicio de correo electrónico
17	Definición de plazos

Tabla 1 Historias de Usuario agrupadas

En el caso de las HU de la Tabla 2, se agrupan las operaciones de: registrar, modificar, eliminar, listar, ampliar detalles y buscar, asociadas a las entidades correspondientes.

Historias de Usuario	Entidad Negocio.	Entidad BD	Operaciones
Administrar Usuario	Usuario	***usuario***	Registrar, modificar, eliminar, listar y buscar
Administrar Rol	Rol	***rol***	
Administrar Órgano	Órgano	***organo***	
Administrar Indicador	Indicador	***indicador***	
Administrar Informe	Informe	***informe***	
Administrar Servicios Web	Servicio Web	***servicio***	
Listar y editar operaciones del sistema	Operación	***operacion***	Modificar (activar), listar, buscar y mostrar detalles
Administrar Configuración del sistema	Configuración	***configuracion***	Modificar
Administrar permisos adicionales de usuario	Permiso Adicional	***usuario_operacion***	Modificar
Administrar asociaciones parte-contraparte	Asociación Parte-Contraparte	***indicador_indicador***	Modificar
Administrar contraseña	Contraseña	***Contrasenna***	Cambiar con verificación, cambiar sin verificación

Tabla 2 Operaciones asociadas por Historias de Usuarios

A continuación se muestran algunas historias de usuario utilizadas:

Historia de Usuario	
Número: HU4	**Nombre Historia de Usuario:** Registrar y Listar Trazas
Modificación (o extensión) de Historia de Usuario:	
Referencia: Registrar Traza, Listar Traza, Buscar Traza y Mostrar Detalles de Traza	
Programador: -	**Iteración Asignada:** 1
Prioridad: Media	**Puntos Estimados:** 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1

Descripción: Esta HU permite registrar una traza en el sistema, cuando el usuario realiza alguna operación, queda registrada automáticamente en el mismo quedando plasmado su nombre de usuario y tipo de operación realizada. Las trazas son listadas por el sistema de manera automática, pudiendo verse en la opción Listar Trazas Registradas.
Observaciones:

Tabla 3 Historias de Usuario Registrar y Listar Traza

Esta Historia de Usuario define el escenario registrar y listar traza de un determinado usuario, los permisos para realizar esta acción los tiene el administrador del sistema, pudiendo ver los detalles de las trazas y listarlas por prioridad, para ello utiliza las entidades del modelo ***operacion***, ***traza*** y el resto de los escenarios la entidad ***usuario***.

Historia de Usuario	
Número: HU7	**Nombre Historia de Usuario:** Administrar Configuración del sistema
Modificación (o extensión) de Historia de Usuario: HU15	
Referencia: Usuario, Configurar Página Principal, Indicador	
Programador: -	**Iteración Asignada:** 1
Prioridad: Media	**Puntos Estimados:** 1
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Esta HU permite la configuración de la página principal del usuario, permitiéndole configurar los atributos que definen el comportamiento general del sistema en su sesión. El usuario también define que indicadores quiere que se muestren en el sistema.	
Observaciones:	

Tabla 4 Administrar configuración del sistema

Esta Historia de Usuario define el escenario configuración de la página principal del usuario, donde se definen los indicadores que tiene acceso en el sistema, que se mostrarán en su página de inicio, además se registra el listado de indicadores a mostrar en la página principal del usuario. Para ello utiliza las entidades del modelo *usuario, conf_pag_*principal y *conf_pag_principal_indicador.*

2.1.2 Tareas de Ingeniería (Task Card)

Las tareas de ingeniería complementan la descripción de las historias de usuario, desglosando cada HU en tareas ingenieriles para su descripción y especificación. A

continuación se muestra la tarea de ingeniería Administrar Contraseña, de la HU Administrar Usuario.

Tarea de Ingeniería	
Número: TC14	**Historia de Usuario:** HU14 Administrar Contraseña
Nombre Tarea: Administrar Contraseña	
Tipo Tarea: Desarrollo del negocio de las clases que intervienen en el proceso de administración de contraseñas.	**Puntos Estimados: 1**
Fecha Inicio: 29/03/2013	**Fecha Fin:** 5/04/2013
Programador Responsable: -	**Puntos Reales: 1**
Descripción: Esta tarea consiste en proveer los mecanismos del cambio de contraseña del usuario, contando con la verificación de la misma.	
Observaciones:	

Tabla 5 Tarea de Ingeniería Administrar Contraseña (TC14)

2.1.3 Lista de reserva del producto

Los requerimientos funcionales para el módulo de Administración y Configuración, descritos en el documento del expediente de desarrollo Lista de Reserva de Producto, son:

Módulo de Administración y Configuración:

1. **Autenticar usuario:** el sistema debe permitir que los usuarios puedan identificarse introduciendo su nombre de usuario y su contraseña, pudiendo establecerse así los permisos según el rol que desempeña.
2. **Crear usuario:** permite la creación de un usuario en el sistema. Incluye además la asignación de un rol al usuario. Los datos básicos de registro de un usuario deberán ser: nombre y apellidos, correo electrónico, usuario, contraseña.
3. **Eliminar usuario:** permite eliminar un usuario del sistema. La eliminación de un usuario del sistema no implica eliminarlo físicamente de la base de datos, sino que se desactiva su acceso al sistema.
4. **Modificar datos de un usuario:** permite modificar los valores de un determinado usuario, (nunca su nombre de usuario ni su contraseña, esto último se realizará en la

interfaz de cambio de contraseña), tales como su nombre y apellidos, correo electrónico, rol, permisos adicionales, sexo.

5. **Listar usuarios registrados en el sistema:** visualiza todos los usuarios registrados en el sistema, mostrando los datos más relevantes de estos.
6. **Buscar usuario:** permite buscar un usuario según parámetros de búsqueda introducidos. El resultado de la búsqueda deberá ser un listado de usuarios (ordenados alfabéticamente por nombre de usuario) que cumplan con los parámetros de búsqueda.
7. **Mostrar detalles de un usuario:** esta funcionalidad permite ampliar los detalles de un usuario, con otras informaciones como las trazas asociadas según operaciones que ha realizado en el sistema, y los permisos que posee según su rol, además de aquellos que se le han asignado adicionalmente.
8. **Crear rol:** permite la creación de un rol en el sistema, asociándole las operaciones que serán agrupadas en dicho rol.
9. **Eliminar rol:** permite eliminar un rol del sistema, al eliminarse un rol se le desactiva el acceso al sistema a aquellos usuarios que forman parte de este rol, independientemente de que estos usuarios tengan o no permisos adicionales.
10. **Modificar datos de un rol:** permite modificar los valores de un determinado rol, editando la lista de operaciones que agrupa, bien sea adicionando nuevas operaciones o quitando las que han sido asociadas.
11. **Listar roles definidos en el sistema:** permite listar todos los roles definidos en el sistema.
12. **Buscar Rol:** permite buscar un rol según parámetros de búsqueda introducidos. El resultado de la búsqueda deberá ser un listado de roles que cumplan con los parámetros especificados.
13. **Mostrar detalles de un rol**: amplía la información de un rol, mostrando además la lista de todas las operaciones que agrupa, así como un listado de los usuarios que pertenecen a dicho rol.
14. **Registrar nuevo órgano:** permite que se registre un nuevo órgano en el sistema.
15. **Eliminar órgano:** permite eliminar un órgano del sistema. Esto provoca que se inhabiliten automáticamente aquellos usuarios que pertenecen a dicho órgano, impidiendo de este modo su acceso al sistema hasta que un administrador los asigne a un nuevo órgano.

16. **Modificar datos de un órgano:** permite modificar los datos de un determinado órgano como son su nombre, la dirección y tipo de órgano.
17. **Listar órganos:** permite mostrar un listado de todos los órganos registrados en el sistema.
18. **Buscar órgano:** permite buscar un órgano según parámetros de búsqueda introducidos. El resultado de la búsqueda deberá ser un listado de órganos que cumplan con los parámetros especificados.
19. **Mostrar detalles de un órgano:** amplía los datos de un órgano, mostrando además la lista de todos los usuarios que pertenecen al mismo, así como los servicios que provee dicho órgano.
20. **Modificar descripción de una operación:** permite modificar los valores de la descripción de una determinada operación.
21. **Listar operaciones:** muestra un listado de todas las operaciones registradas en el sistema.
22. **Buscar operación:** permite buscar una operación según parámetros de búsqueda introducidos. El resultado de la búsqueda deberá ser un listado de operaciones que cumplan con los parámetros especificados.
23. **Mostrar detalles de una operación:** amplía la información de una operación en particular, mostrando además aquellos roles que la incorporan, así como los usuarios que la realizan, bien sea por su rol o porque se le ha asignado adicionalmente.
24. **Listar trazas del sistema:** permite listar las trazas registradas en el sistema, dando la posibilidad de ordenarlas.
25. **Registrar trazas del sistema:** Esta funcionalidad permite realizar un registro automático de todas las operaciones que un usuario realiza en un sistema.
26. **Cambiar contraseña de usuario:** Permite al usuario creado la opción de cambiar su contraseña introduciendo su usuario, contraseña actual y contraseña nueva. Esta funcionalidad además permite desde la sesión de cualquier usuario, cambiar la contraseña de otro usuario, sin la necesidad de autenticarse en el sistema.
27. **Cambiar contraseña de usuario sin verificación:** Esta funcionalidad permite cambiar la contraseña de cualquier usuario, sin la necesidad de conocer la contraseña anterior. Esta funcionalidad permite (preferentemente a un usuario con permisos de administración) que se registre una nueva contraseña, siguiendo los

mismos principios de la operación de cambio de contraseña. Es una operación muy útil para los casos en que los usuarios hayan olvidado su contraseña anterior, y tengan que necesariamente acudir a un administrador del sistema con permisos de realizar esta operación, para que le actualicen sus credenciales.

28. **Editar permisos adicionales a un usuario:** permite asignar o eliminar los permisos que adicionalmente se le han asignado a un usuario determinado.
29. **Crear informe:** permite definir un nuevo informe en el sistema, a partir de la agrupación de indicadores que ya estén registrados en el sistema.
30. **Eliminar informe:** elimina un informe del sistema, impidiendo de este modo que el sistema lo identifique como un informe válido para ser gestionado o incluido en operaciones de búsqueda.
31. **Modificar datos de informe:** permite editar los datos de un informe en particular, asignando o eliminando indicadores a dicho informe.
32. **Listar informes:** muestra un listado de todos los informes definidos en el sistema.
33. **Buscar informe:** permite buscar un informe según parámetros de búsqueda introducidos. El resultado de la búsqueda deberá ser un listado de informes que cumplan con los parámetros especificados.
34. **Mostrar detalles de un informe:** amplía los datos de un informe en particular, mostrando los indicadores que se le han sido asignado, así como una lista de todos los usuarios que tienen permisos de acceso a dicho informe y el nivel de acceso que tienen, además de una lista de todos los reportes generados de dicho informe ordenados descendentemente por la fecha de realización.
35. **Editar permisos de acceso a un informe:** permite editar los permisos de acceso de un usuario a los informes definidos en el sistema.
36. **Crear indicador:** registra un nuevo indicador en el sistema, definiendo además la lista de servicios web que lo proveen.
37. **Eliminar indicador:** elimina un indicador del sistema, automáticamente se modifican todos los informes que lo han incluido.
38. **Modificar datos de indicador:** edita los datos de un indicador determinado.
39. **Listar indicadores:** lista todos los indicadores registrados en el sistema.

40. **Buscar indicador:** permite buscar un indicador según parámetros de búsqueda introducidos. El resultado de la búsqueda deberá ser un listado de indicadores que cumplan con los parámetros especificados.
41. **Mostrar detalles de un indicador:** amplía la información de un indicador, mostrando además una lista de todos los usuarios que tienen permisos de realizar reportes de este indicador.
42. **Editar permisos de generar reporte de un indicador:** edita los permisos de un usuario de generar reportes de un determinado indicador.
43. **Registrar servicio web:** registra un nuevo servicio web en el sistema, indicando el órgano que lo provee y los indicadores de los que brinda datos.
44. **Eliminar servicio web:** elimina un determinado servicio web registrado en el sistema. Esto provoca que se inactive los indicadores correspondientes siempre y cuando dichos indicadores no tengan otro servicio web que lo provea.
45. **Modificar datos de un servicio web:** edita los datos de un determinado servicio web, tal como el órgano que lo provee, la URL del servicio, así como los indicadores a los cuales responde.
46. **Listar servicios web:** muestra un listado de todos los servicios web registrados en el sistema.
47. **Buscar servicio web:** permite buscar un servicio web según parámetros de búsqueda introducidos. El resultado de la búsqueda deberá ser un listado de servicios que cumplan con los parámetros especificados.
48. **Mostrar detalles de un servicio web**: amplía los detalles de un servicio web, mostrando los indicadores a los que responde y el órgano que lo provee.
49. **Definir asociación parte-contraparte entre indicadores:** Permite definir una asociación entre dos indicadores en la cual un indicador funge como parte y otro como contra parte. Este tipo de asociación es útil para validar coincidencias entre valores de indicadores. Dos indicadores asociados como parte-contraparte deben mostrar los mismos valores en los reportes realizados en el mismo período.
50. **Eliminar asociación parte-contraparte entre indicadores:** Permite eliminar una asociación de parte-contraparte entre indicadores. Haciendo posible que no se verifiquen las coincidencias entre los valores de estos.
51. **Listar asociación parte-contraparte entre indicadores:** Lista todas las relaciones de parte-contraparte entre indicadores.

52. **Mostrar detalles asociación parte-contraparte entre indicadores:** Amplía los detalles de una asociación parte-contraparte. Además muestra el histórico de valores de reportes de consultas y reportes de registro realizados de cada par de indicadores relacionados en una asociación parte-contraparte.
53. **Registrar configuración de parámetros de conexión a servidor de correo electrónico:** permite configurar la conexión a un servidor de correo que empleará el sistema.
54. **Modificar configuración de conexión de correo electrónico:** permite editar los parámetros de conexión a un servidor de correo electrónico definidos.
55. **Verificar conexión:** permite comprobar la conexión a partir de los parámetros definidos.
56. **Cargar configuración de conexión:** permite cargar un archivo de configuración de conexión guardado localmente en el disco duro.
57. **Exportar configuración:** exporta la configuración en un archivo XML.
58. **Eliminar configuración de conexión:** elimina una configuración de conexión a un servidor de correo electrónico registrado en el sistema.
59. **Mostrar detalles de configuración:** permite mostrar los detalles de configuración de conexión a un servidor de correo electrónico.
60. **Registrar configuración global de mecanismo de consulta de indicadores:** permite definir el modo de consumo de los indicadores registrados en el sistema.
61. **Modificar tiempo de caducidad de las contraseñas**: modifica el tiempo de caducidad de las contraseñas.
62. **Registrar configuración global de mecanismo de consulta de indicadores:** permite definir, para cada indicador, el modo de consulta del indicador.
63. **Modificar configuración global de mecanismo de consulta de indicadores:** Permite modificar la configuración global del mecanismo de consumo de indicadores.
64. **Modificar configuración particular de mecanismo de consulta de indicadores:** permite modificar la configuración particular del mecanismo de consumo de un indicador determinado.
65. **Definir indicadores a presentar en página principal:** seleccionar aquellos indicadores que el usuario necesita se muestren en la página principal.

66. **Modificar configuración de indicadores a presentar en página principal:** permite modificar la configuración sobre los indicadores que el usuario necesita se muestren en la página principal.
67. **Definir tiempo de caducidad de las contraseñas en el sistema:** permite establecer el tiempo en que las contraseñas caducarán.

2.1.4 Requisitos funcionales

Los requisitos funcionales indican lo que debe hacer el producto, es la capacidad con la que debe cumplir el mismo. A partir de entrevistas y reuniones con el cliente se obtuvieron en total 67 requisitos funcionales en el módulo. (30)

Seguidamente se muestra la especificación de los requisitos Registrar Trazas del Sistema y Registrar servicio web.

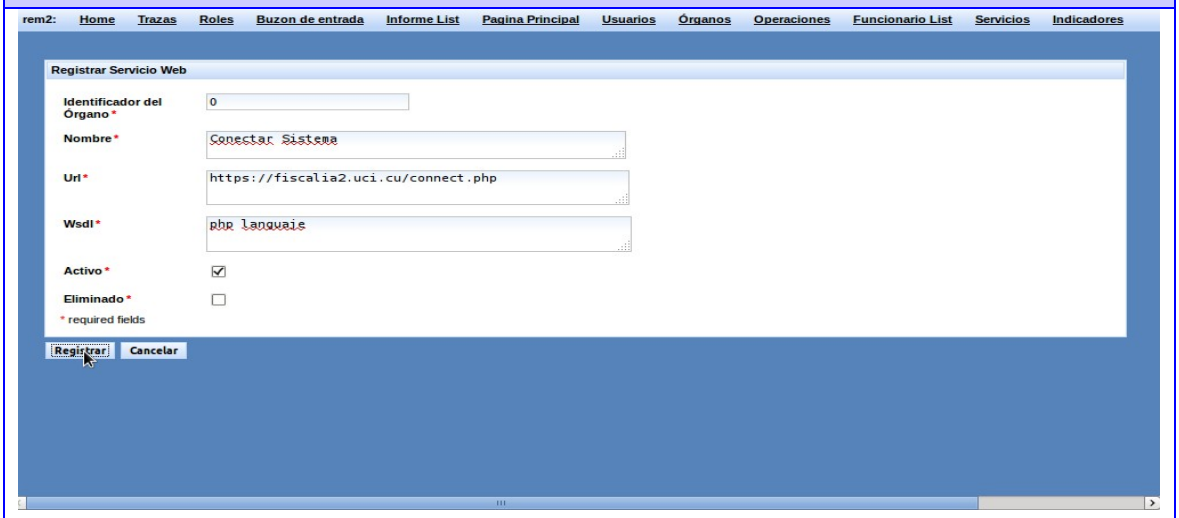
Nombre	**Descripción**	**Complejidad**	**Prioridad para cliente**
Registrar Trazas del Sistema	Las trazas se registran en el sistema de manera automática.	Media	Media
Prototipo			
Esta funcionalidad no lleva prototipo, ya que es una bitácora de las operaciones que se realizan en el sistema de manera automática.			
Campos	**Tipos de Datos**	**Reglas o Restricciones**	
Nombre	Letra	Obligatorio	
Operación	Alfanumérico	Obligatorio	
Usuario	Letra	Obligatorio	
Observaciones			

Tabla 6 Especificación del requisito funcional Registrar trazas del sistema

Nombre	**Descripción**	**Complejidad**	**Prioridad para cliente**
Registrar Servicio Web	Permite registrar un nuevo servicio web en	Alta	Media

	el sistema, indicando el órgano que lo provee y los indicadores de los que brinda datos.		

Prototipo



Campos	Tipos de Datos	Reglas o Restricciones
Nombre	Letra	Obligatorio
Órgano que lo provee	Letra	Obligatorio
URL	Letra	Obligatorio
WSDL	Letra	Obligatorio
Activo	Booleano	Obligatorio
Observaciones		

Tabla 7 Especificación del requisito funcional Registrar Servicio Web.

2.1.5 Requisitos no funcionales del sistema

Los requisitos no funcionales del sistema son cualidades o condiciones que el producto debe de tener, estas lo hacen llamativo al cliente, usable, ligero y confiable.

Para la satisfacción del cliente y lograr calidad del sistema se listan los siguientes requerimientos no funcionales:

Requisito de disponibilidad:

1. Para cumplir con este requerimiento el sistema debe estar disponible las 24 horas del día, todos los días del año.

Requisitos de usabilidad:

1. El sistema debe mostrar mensajes de error al usuario para identificar el tipo de problema.
2. Si el usuario deja algún campo obligatorio en blanco el sistema debe verificar los datos de manera tal que el usuario reciba una notificación identificando donde está el error.
3. El sistema debe validar los datos para verificar si el usuario no ha introducido ningún dato incorrecto, en otro formato o un tipo de dato no válido.
4. El usuario podrá acceder, según sus permisos a todas las interfaces de usuario desde cualquier lugar, para lograr la navegabilidad del sistema.

Requisito de portabilidad:

1. El sistema debe poder instalarse en los sistemas operativos Windows, Linux/Unix.
2. El sistema debe dar la posibilidad de emplear cualquier sistema para la gestión de bases de datos.

Requisito de mantenibilidad:

1. El sistema debe estar complemente documentado, tanto en el código fuente como en los manuales de administración y de usuario.

Requisito de flexibilidad:

1. El sistema deberá permitir la definición dinámica de roles, haciendo posible personalizar la gestión del acceso a los recursos según las necesidades del entorno.

Requisitos de seguridad:

1. El sistema debe recuperarse por completo si ocurre cualquier falla y/o error(es) inesperado(s).
2. El acceso al sistema de la plataforma debe estar restringido por el uso de claves asignadas a cada uno de los usuarios. Solo podrán ingresar aquellos que estén registrados, siendo clasificados en varios tipos de usuarios (o roles) con acceso a las opciones de trabajo definidas para cada rol.
3. El sistema deberá contar con un mecanismo que permita el registro de actividades con la identificación de los usuarios, el momento y el tipo de operaciones que fueron realizadas.
4. El sistema debe contar con pistas de auditoría de las actividades que se realizan con niveles razonables para su reconstrucción e identificación de los hechos.

5. El sistema debe permitir que la contraseña se almacene de manera encriptada en la base de datos.
6. El sistema debe permitir la comprobación de credenciales en la autenticación del usuario en el servidor de aplicaciones y no en el servidor de base de datos, para evitar inyecciones SQL que falseen la autenticación y provoquen la suplantación de identidad.
7. El ingreso de la información en el sistema debe ser transaccional, de manera que se garantice la completitud en la información y la reversión de las operaciones ante alguna falla no prevista.

2.2 Validación de Requisitos

2.2.1 Métricas para la Validación de los Requisitos

Las métricas son la maduración de una disciplina, que según Pressman, van a ayudar a la evaluación de los modelos de análisis y de diseño, en donde proporcionarán una indicación de la complejidad de diseños procedimentales y de código fuente, y ayudarán en el diseño de pruebas más efectivas. Es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. Proporcionan una manera sistemática de valorar la calidad basándose en un conjunto de reglas claramente definidas.

Métricas para la validación de requisitos.

Para llevar a cabo la validación de la calidad de los requisitos se tendrán en cuenta las métricas asociadas a la estabilidad, especificidad y grado de validación de los requisitos, las cuales determinan si el levantamiento de requisitos se efectuó de manera correcta.

Métrica 1. Estabilidad de los requisitos:

El objetivo de esta métrica es medir cuan estables son los requisitos para asegurar su adecuación antes de pasar a la próxima disciplina. Se considera que los requerimientos son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación. La estabilidad de los requerimientos se calcula como:

$$ETR = \left[\frac{RT - RM}{RT} \right] * 100$$

Donde

RT: Total de requisitos definidos.

RM: Cantidad de requisitos modificados.

Esta métrica ofrece valores entre 0 y 100. El mejor valor de ETR es el más cercano a 100 porque mostrará que no se están realizando cambios sobre los requisitos, son estables y, por tanto, es confiable trabajar el análisis y diseño sobre ellos.

Métrica 2. Especificidad de los requisitos:

El objetivo de esta métrica es cuantificar la especificidad o falta de ambigüedad en la definición de los requerimientos. Para calcular esta métrica deben contarse los requerimientos que tuvieron igual interpretación por los revisores y compararlos con el total de requerimientos definidos. La especificidad de los requerimientos se calcula como:

$$Q_1 = \frac{n_{ui}}{n_r}$$

Donde

Q1: Especificidad de los requisitos.

Nr: Total de requisitos definidos.

Nui: Total de requisitos para los que los revisores tuvieron interpretaciones idénticas.

El valor de esta métrica debe estar siempre entre 0 y 1. Mientras más cerca de 1 esté el valor de ER mayor será la consistencia de la interpretación de los revisores para cada requerimiento y menor será la ambigüedad en la especificación de los requerimientos.

Métrica 3. Grado de validación de los requisitos:

Los requerimientos deben ser posibles de validar. La validación de los requerimientos se realiza en consenso del equipo de desarrollo al contrastar lo que desea el cliente con la posibilidad real de implementarlo. El grado de validación de los requerimientos mide la corrección en la definición de los requerimientos. Este valor se calcula como:

$$Q_3 = \frac{n_c}{(n_c + n_{nv})}$$

Donde

Q3: Grado de validación de requisitos.

Nc: Total de requisitos validados correctamente.

Nnv: Total de requisitos no validados.

El resultado de esta métrica está siempre entre 0 y 1. El valor óptimo, es el más cercano a 1 e indica un alto nivel de corrección en la definición de los requerimientos.

Como resultado de las métricas para la validación de los requisitos se tiene en cuenta la estabilidad, especificidad y el grado de validez de los mismos, se definieron 67 requisitos, de ellos se modificaron 3, para un 95,52% de ETR, ya que es un valor cercano a 100 se puede decir que los requisitos son estables, están bien descritos y libres de ambigüedades, lo que asegura un mejor desarrollo en el proceso de implementación del sistema.

2.2.2 Planificación de entregas

Se efectúa una planificación de entregas cuando los desarrolladores y clientes establecen los tiempos de implementación ideales de las Historias de Usuario, determinando cuales y con qué prioridad serán implementadas en cada versión del programa.

Después de una planificación de entregas tienen que estar claros los siguientes factores: los objetivos que se deben cumplir (que son principalmente las historias que se deben desarrollar en cada iteración); el tiempo que tardarán en desarrollarse y publicarse las versiones del programa; el número de personas que trabajarán en el desarrollo; y cómo se evaluará la calidad del trabajo realizado. Durante la planificación de entregas, donde participó el equipo de desarrollo y el cliente, se determinó que para la primera entrega se implementarán las Historias de Usuario pertenecientes al módulo de Administración y Configuración. (29)

El Plan de Iteraciones está dividido en tres períodos, en los que se van a agrupar las Historias de Usuario según el cronograma de entrega del plan, hasta terminar la entrega de todas las funcionalidades del sistema. (*Ver anexo 1 Plan de Iteraciones*).

2.3 Fase de Diseño

Para el diseño de la aplicación, se tuvo presente que la arquitectura y el diseño evolucionan durante el desarrollo, por tanto, se debe diseñar la solución más simple, que pueda funcionar y ser implementada según lo que plantea la metodología XP. Se proponen las tarjetas CRC como técnica de modelado para ayudar a los desarrolladores a crear diseños de clases orientados a responsabilidades, teniendo en cuenta los patrones GRASP (General Responsibility Assignment Software Patterns o Patrones Generales de Software para Asignación de Responsabilidades) (31), logrando elegir las clases adecuadas y decidir cómo estas clases deben interactuar. Además se presenta el modelo de datos físico.

La estrategia es la elaboración de un diseño lo más fácil y sencillo posible, fáciles de implementar y con un buen nivel de entendimiento, que a la larga costará menos tiempo y esfuerzo desarrollar. Un diseño sencillo debe poseer la menor cantidad de clases y métodos. No debe existir código duplicado, cumpliendo con todos los requisitos del software.

La metodología XP asimila dentro de sus flexibilidades las imprecisiones y cambios de los clientes en los requerimientos del software, por lo que rompe con un hábito de la mayoría de los programadores de diseñar anticipándose a los problemas.

Si se parte del hecho que desde su concepción el diseño debe ser simple, progresivamente se le adicionará cuanto sea necesario y se redefinirá. Con ello se evita que el diseño adquiera una complejidad innecesaria, lo cual se traduce en pérdida de tiempo, debido a que, generalmente, lo que se supone nunca ocurre.

2.3.1 Tarjetas Clase, Responsabilidades y Colaboración (C.R.C)

El uso de las tarjetas CRC permite al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de programación. Tienen la finalidad de obtener un diseño simple y no incurrir en la implementación de características que no son necesarias. Estas tarjetas representan objetos, donde la clase a la que pertenece se puede escribir arriba mientras que en una columna a la izquierda se pueden reflejar las responsabilidades u objetivos que debe cumplir el objeto, y a la derecha se describen las clases(los nombres de estas clases no se acentúan y pueden ser compuestos) que colaboran con cada responsabilidad. A continuación se exponen una representación de las tarjetas que se obtuvieron durante el proceso de desarrollo del módulo de Administración y configuración:

Clase: UsuarioDetailsManager	
Responsabilidades	Colaboradores
Esta clase tiene la responsabilidad de mostrar los datos de un usuario, las trazas y las operaciones a las que tiene relación este usuario.	Usuario RolOperacion Operacion UsuarioOperacionList Traza OperacionList

Tabla 8 Tarjeta CRC UsuarioDetailsManager

Clase: UsuarioDeleteManager

Responsabilidades	Colaboradores
Esta clase tiene como objetivo eliminar un usuario conjuntamente con la configuración de la página principal asociada a este.	Usuario ConfPagPrincipal

Tabla 9 Tarjeta CRC UsuarioDeleteManager

2.3.2 Validación del Diseño

2.3.3 Métricas para la validación del diseño.

Las métricas de diseño permiten medir de forma cuantitativa la calidad de los atributos internos del software, permitiendo evaluar la calidad durante el desarrollo del sistema. Se centran en cuantificar tanto la complejidad, como la funcionalidad y eficiencia inmersa en el proceso de desarrollo de software. Sus objetivos están inclinados a mejorar la comprensión de la calidad del producto, a estimar la efectividad del proceso y mejorar la calidad del trabajo. (35)

Métrica 1. Tamaño Operacional de Clase (TOC): Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

- ✓ **Responsabilidad**: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto de la problemática propuesta. Un aumento del TOC implica un aumento de la responsabilidad asignada a una clase determinada. Si existen valores grandes de TOC significa que una clase tiene demasiada responsabilidad, lo cual reduciría la reutilización de la clase y hará complicada la implementación. De forma contraria sucede si los valores de TOC son pequeños.
- ✓ **Complejidad de implementación**: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado. Un aumento del TOC implica un aumento de la complejidad de implementación de una determinada clase.
- ✓ **Reutilización**: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software. Un aumento del TOC implica una disminución del grado de reutilización de una determinada clase.

El tamaño general de una clase se calcula como:

TGC= NA +TO

Siendo **NA** el número de atributos que posee la clase (tanto atributos heredados como atributos privados de la instancia) y **TO** el total de operaciones que posee la clase (tanto operaciones heredadas como operaciones privadas de la instancia).

Luego el resultado es comparado con los valores umbrales que aparecen en la Tabla 17 para determinar el TOC de cada clase.

Clasificación	Valores de Umbrales
Pequeño	<=21
Medio	>21 <=42
Grande	>42

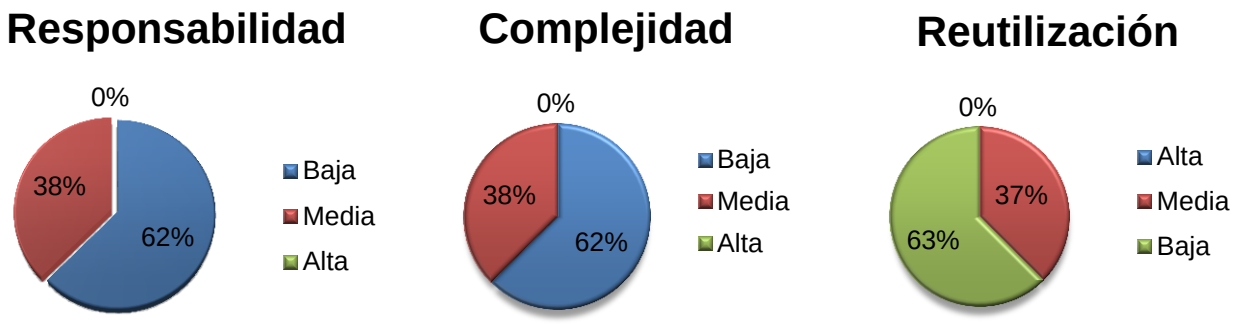
Tabla 10 Clasificación de las clases según el umbral.

Luego se calcula el promedio correspondiente a los umbrales, y por consiguiente la afectación de los parámetros establecidos, según lo que se muestra en la Tabla 3.

Atributos de Calidad	Clasificación	Criterio
Responsabilidad	Baja Media Alta	Umbral<=Promedio Promedio< Umbral < = 2* promedio Umbral > 2* promedio
Complejidad de implementación	Baja Media Alta	Umbral<=Promedio Promedio< Umbral < = 2* promedio Umbral > 2* promedio
Reutilización	Baja Media Alta	Umbral > 2* promedio Promedio< Umbral < = 2* promedio Umbral<=Promedio

Tabla 11 Rango de valores para medir la afectación de los atributos de calidad (TOC).

Resultados de la métrica TOC



Dados los resultados se puede interpretar que un aumento del TOC implica un aumento en la responsabilidad, complejidad y reutilización de las clases, lo cual determina que las clases utilizadas no poseen gran nivel de complejidad de implementación, así como se

hace posible una parte de la reutilización según las funcionalidades y características que se estén utilizando. Trayendo consigo un diseño sencillo y fácil de comprender, mejorando la calidad del trabajo y estimando la efectividad del producto durante el desarrollo.

Métrica 2. Relaciones entre clases (RC): está dada por el número de relaciones de uso de una clase con otra. Para medir el diseño según RC, se evalúan un conjunto de atributos de calidad entre los que se encuentra la Reutilización, definida en la métrica TOC, además los que se relacionan a continuación:

- ✓ **Acoplamiento**: Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.
- ✓ **Complejidad del mantenimiento**: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- ✓ **Cantidad de pruebas**: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

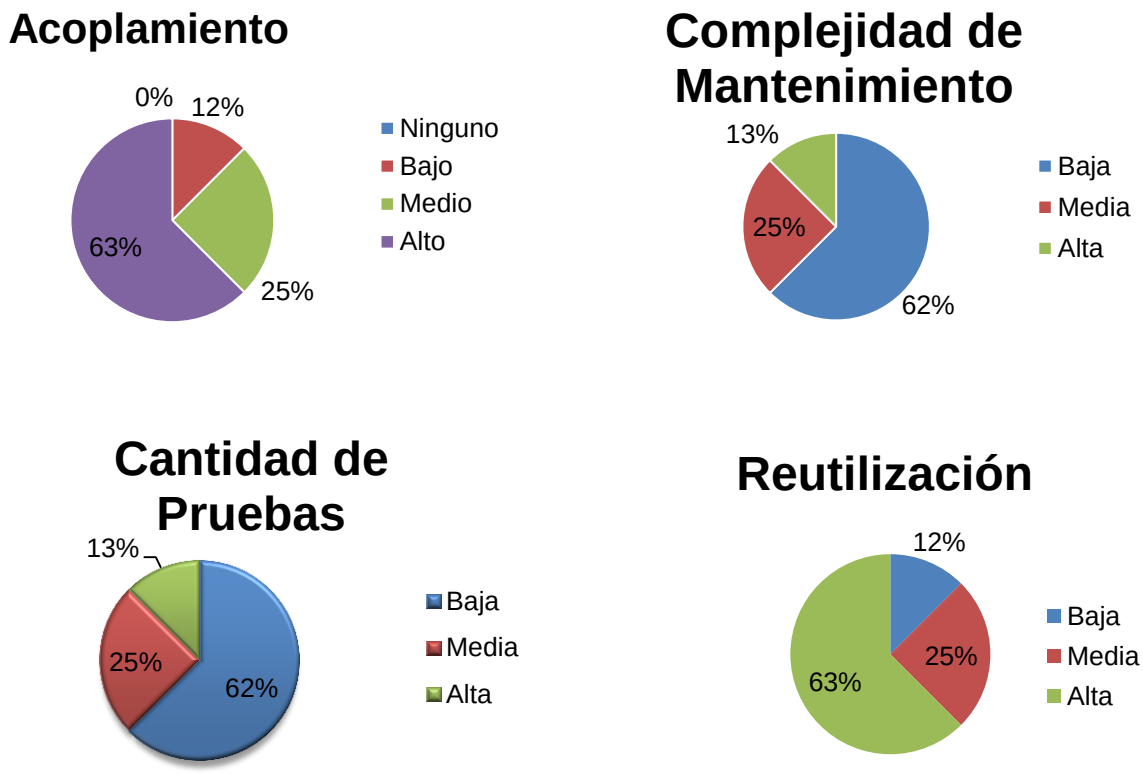
Para determinar el grado de afectación de los atributos de calidad que mide la métrica RC es necesario determinar la cantidad de relaciones de uso (CRU) que posee cada una de las clases a medir. Una vez determinada la CRU, se procede a calcular el promedio de las mismas y teniendo ambos valores según los criterios expuestos en la Tabla 4 se determina la incidencia de los atributos de calidad en cada una de las clases.

Atributos de calidad	Clasificación	Criterio
Responsabilidad	Ninguna Baja Media Alta	CRU=0 CRU=1 CRU=2 CRU> 2
Complejidad de implementación.	Baja Media Alta	CRU<=Promedio Promedio< CRU < = 2* promedio CRU > 2* promedio
Reutilización.	Baja Media Alta	CRU > 2* promedio Promedio< CRU < = 2* promedio CRU<=Promedio

Cantidad de Pruebas	Baja Media Alta	CRU<=Promedio Promedio< CRU < = 2* promedio CRU > 2* promedio

Tabla 12 Rango de valores para medir la afectación de los atributos de calidad (RC).

Resultados de la métrica RC



Siendo los valores de RC elevados implica un aumento del acoplamiento de las clases, así como un aumento de la complejidad de mantenimiento de las mismas. Del resultado se puede inferir que a mayor RC menor grado de reutilización poseen las clases y se necesita mayor cantidad de pruebas unitarias para probarlas. En general, los valores de RC para cada clase deben mantenerse tan bajos como sea razonable.

2.4 Arquitectura del Software

La arquitectura del sistema está basada en J2EE a partir de la definición que establece el framework Seam. La arquitectura está diseñada a partir de un estilo arquitectónico en capas:

- ✓ **Capa de presentación (Vista):** Aquí se definen todas las interfaces (páginas XHTML) con las que interactúa el usuario para realizar las operaciones del sistema.

Cada acción efectuada en la vista es gestionada por un controlador de negocio (clases Java cuya nomenclatura es **Manager.java*), el que evalúa el cumplimiento de las precondiciones, gestiona y valida los datos especificados por el usuario y finalmente realiza la operación, consultando el modelo siempre y cuando lo requiera. También se incluyen archivos XML (cuya nomenclatura es <nombre_vista>*.page.xml*) para la definición de reglas, atributos y configuraciones particulares para cada vista.

- ✓ **Capa de negocio (Manager):** En la capa de Negocio, precisamente donde se implementan los EJB, se definen los controladores, las clases auxiliares para el acceso a datos (clases Java cuya nomenclatura es **List.java*) y otras clases de utilidad para esta capa.
- ✓ **Capa de dominio**: se encuentran las entidades de negocio generadas a partir del *mapping* que realiza el ORM Hibernate y las extensiones de estas entidades y la base de datos.
- ✓ **Capa de dato:** Base de datos en postgreSQL.

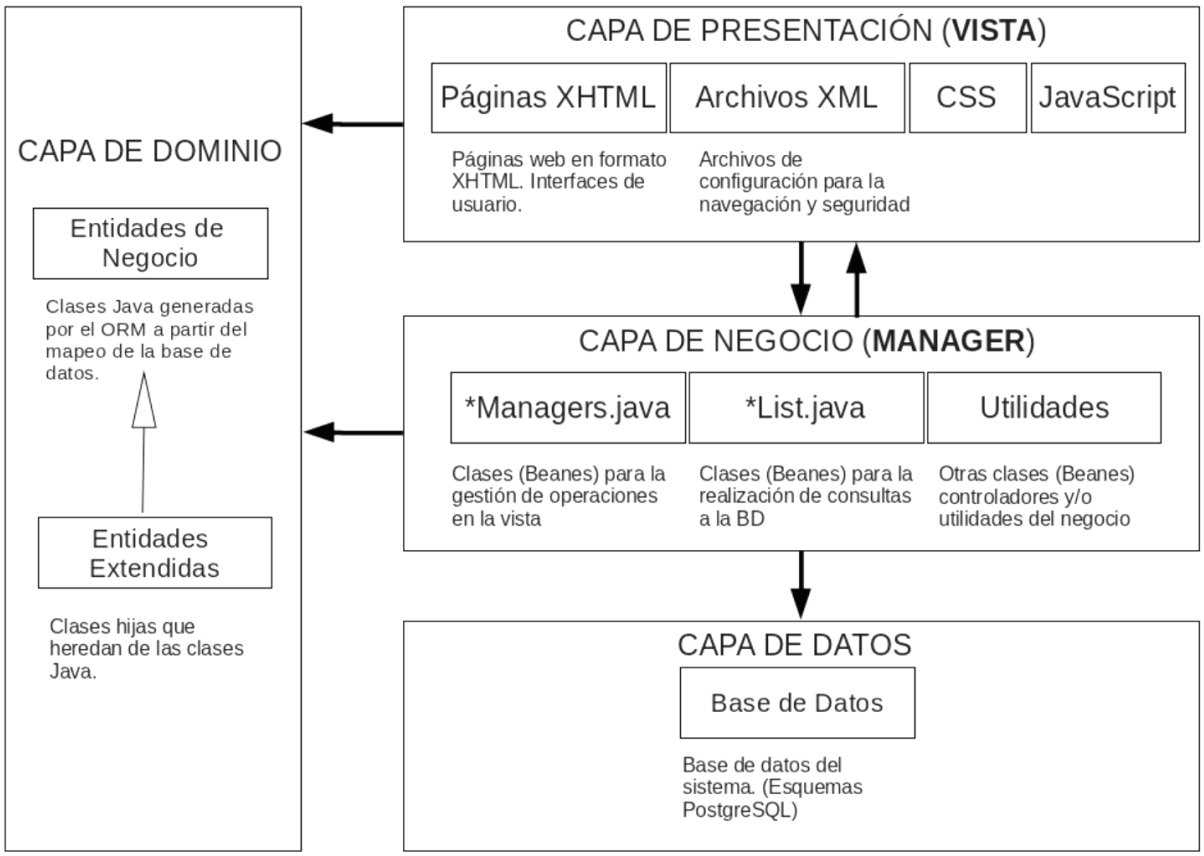


Figura 1: Implementación de la Arquitectura en capas del Framework SEAM

La distribución física en el código fuente de estos componentes de las capas de la arquitectura, está determinada por la forma de empaquetamiento de un proyecto Seam. El framework Seam crea tres tipos de proyectos: el proyecto de la Vista, donde se encuentra el ***WebContent*** (paquete de contenido web), en el cual se definen las páginas XHTML, los archivos de configuración XML, las hojas de estilos CSS y el código JavaScript; un proyecto EAR, en el cual se realiza el empaquetamiento del sistema luego de su compilación; y el proyecto EJB, donde se define toda la capa de negocio. *(Ver anexo 2 Distribucion Fisica de los componentes de la arquitectura).*

2.4.1 Arquitectura de seguridad

En sentido general, la seguridad cuenta con dos mecanismos fundamentales: el mecanismo de autenticación, a partir del cual se identifica al usuario y el mecanismo de autorización, mediante el cual se determina si un usuario tiene los permisos suficientes para acceder o no ha determinado recurso[6] (alguna funcionalidad o método del sistema). Para implementar estos mecanismos se hizo uso de las clases *Identity* y *Credentials* que provee el framework Seam (32), y las que son encargadas, como bien lo reflejan sus nombres, de la identidad del usuario y sus credenciales en el sistema.

El mecanismo de **autenticación** en el sistema está basado en la especificación de usuario y contraseña, a partir de los cuales el usuario se identifica ante el sistema. De ser satisfactoria la autenticación, se cargan de la base de datos todos los permisos que tiene asociado el usuario, para gestionar la autorización a los recursos.

La autenticación es local, contra una base de datos que almacena todos los datos del usuario, sus permisos (operaciones que puede realizar en el sistema) y el registro de las trazas. Las contraseñas son almacenadas en el servidor de manera encriptada y la verificación de estas se realizan en el servidor de aplicaciones.

El mecanismo de control de acceso a los recursos del sistema (**autorización**) se realiza a nivel de funcionalidad, en la cual se verifica que el usuario, actualmente autenticado en el sistema, tiene permisos de realizar la funcionalidad (**operación**) que está solicitando.

[6] Funcionalidad capaz de realizar el sistema(método), o vista(página XHTML), objeto digital(fichero, informe, indicador) , entre otros.

La definición de los permisos a recursos del sistema parte de un esquema dinámico de definición de roles. Es decir, en tiempo de ejecución son definidos los roles del sistema (33), y al momento de registro de un usuario se determina a qué rol pertenecerá el mismo. Para que esto sea posible, el sistema define a priori todas las operaciones y las vistas (interfaces XHTML), estableciendo asociaciones entre ambas a partir de las cuales se puede determinar las vistas que son afectadas por una determinada operación. Por tanto, la definición de un rol no es más que la agrupación de operaciones que realizarán los usuarios que pertenezcan ha dicho rol; y consecuentemente se determinan los recursos de la vista a los que podrán acceder los usuarios asignados al rol. Esto le imprime al sistema un alto grado de flexibilidad, el que hace posible adaptarse a las políticas cambiantes del negocio asociadas a las definiciones de nuevos roles y permisos. Sin embargo recae sobre la organización la eficacia del cumplimiento de las disposiciones de seguridad, dado a que dependerá de un usuario súper-administrador (root) que sean cumplidas a cabalidad.

Ante cada petición del usuario para el acceso a un recurso, es gestionado por los componentes ***PermissionResolverDataLoader*** y ***PermissionViewResolverManager*** el otorgamiento o no al acceso de dicho recurso. El primero tiene la responsabilidad de cargar de la base de datos todos los permisos del usuario y construir una tabla *hash* en la cual las llaves (*keys*) son las URLs de las vistas y los valores (*values*) son el listado de las operaciones que afectan dicha vista. El segundo componente se encarga de realizar la evaluación necesaria para el otorgamiento o no del permiso. Para la realización de este proceso, se realiza un filtro de todas las peticiones HTTP del sistema. Esto se refleja en una regla de navegación en el archivo de configuración global del sistema ***pages.xml***, como muestra el segmento de código a continuación:

```
<page view-id="/*" login-required="true"
action="#{permissionViewResolverManager.chequearPermisosUsuario()}">
<navigation>
        <rule if-outcome="not-authorized">
                <redirect view-id="/error.xhtml">
                        <message>No tiene los permisos para acceder a este
                        recurso.</message>
                </redirect>
          </rule>
        <rule if-outcome="not-exist">
                <redirect view-id="/error.xhtml">
```

```
                <message>Este recurso no existe o no está disponible en
                este momento.
                </message>
            </redirec>
        </rule>
    </navigation>
    </page>
```

2.4.2 Modelo de datos (entidades persistentes)

El modelo de datos asociado a las entidades persistentes del sistema, es uno de los tres componentes del patrón MVC. Este se divide en dos partes: a) modelo lógico, correspondiente a las entidades generadas por el ORM Hibernate a partir de una ingeniería inversa de la base de datos y sobre el cual se realizan todas las consultas utilizando HQL; y b) modelo físico, el cual responde a las tablas de la base de datos del sistema, desplegada en el servidor. (34)

En este epígrafe se presenta el modelo lógico correspondiente a cada una de las Historias de Usuario agrupadas en la Tabla 1. Por cada Historia de Usuario se representa el modelo de datos asociado, y cuál es la relación de las entidades con cada uno de los escenarios de la Historia de Usuario.

1. Modelo de datos de Historia de Usuario Administrar Usuario:

Administrar Usuario involucra varios escenarios (operaciones), como son: a) registrar usuario; b) editar un usuario; c) eliminar usuario; d) buscar usuario; e) mostrar detalles de un usuario y f) listar usuarios.

La operación a) involucra las entidades: ***usuario***, la cual tiene toda la información básica de un usuario; ***organo***, a partir de la cual se extrae la llave del órgano al que pertenece el usuario que se registra; el ***rol*** al cual pertenece el usuario; y en caso de ser funcionario, se almacena en la entidad ***funcionario*** los datos del funcionario. La contraseña se almacena de manera encriptada en la entidad ***contrasena***. El sistema lleva un registro de las contraseñas que el usuario ha empleado, para evitar que este utilice contraseñas que ha empleado con anterioridad. La contraseña en uso es marcada como ***activa*** mientras que las anteriores son inactivadas. En el momento del registro además se crea una configuración de página principal por defecto, y para ello se almacena la información de dicha configuración en la entidad ***conf_pag_principal***.

La operación b) involucra todas las entidades del modelo de la Figura 6, excepto las entidades ***conf_pag_principal*** y ***contrasena***; las que se emplean en otras operaciones como *Administrar configuración de página principal* y *Cambiar Contraseña,* respectivamente.

La eliminación de un usuario representa poner en ***true*** el atributo ***eliminado*** de la entidad ***usuario,*** lo cual a los efectos del sistema el usuario deja de existir. Las operaciones d), e) y f) emplean todas las entidades del modelo (excepto ***contrasena*** y ***conf_pag_principal***) para mostrar los datos de los usuarios.

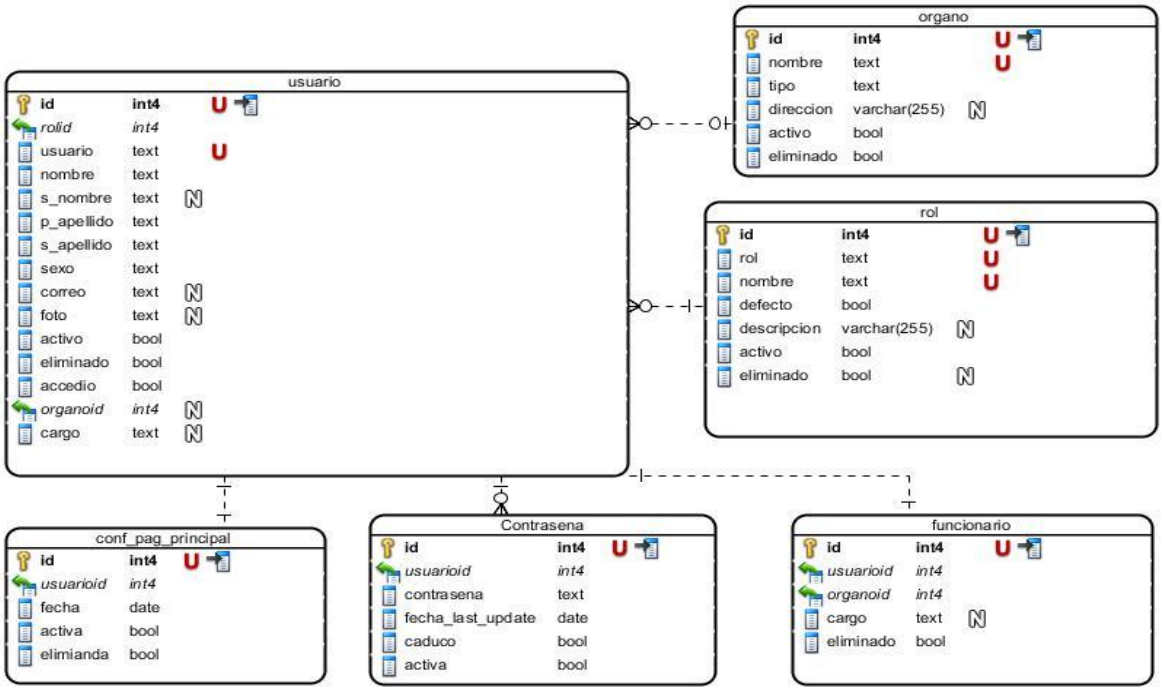


Figura 2: Entidades relacionadas con la HU Administrar Usuario

2. Modelo de datos de Historia de Usuario Administrar Órgano:

Todos los escenarios asociados a la Historia de Usuario *Administrar Órgano* involucran la entidad ***organo*** del modelo (Figura 5).

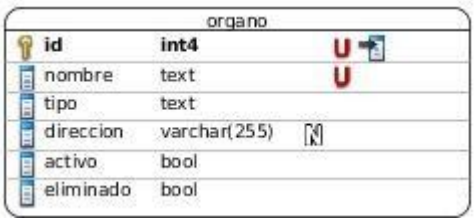


Figura 3: Entidades Relacionadas a la HU Administrar Órgano

3. Modelo de Datos de Historia de Usuario Administrar Rol.

Del mismo modo que la anterior historia de usuario, *Administrar Rol* define los escenarios: a) registrar rol; b) editar un rol; c) eliminar rol; d) buscar rol; e) mostrar detalles de un rol y f) listar roles. Los escenarios a), b) y c) utilizan las entidades del modelo (Figura 4) ***rol*** y ***rol_operacion*** para administrar los datos de un rol; y el resto de los escenarios emplean todas las entidades para mostrar los datos de uno o varios roles. La entidad ***operacion*** es el nomenclador de operaciones del sistema. Para el caso específico del escenario c), la eliminación de un rol provoca la inactivación automática de todos los usuarios que pertenecen a dicho rol.

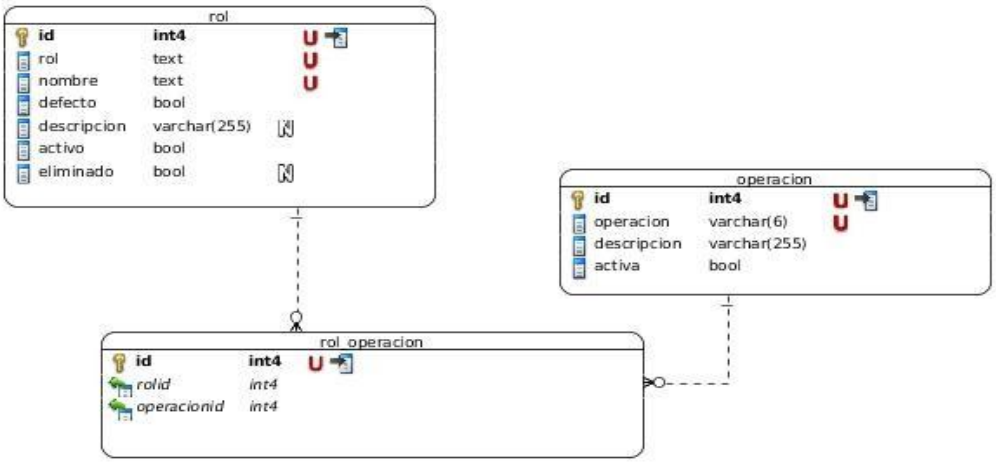


Figura 4: Entidades relacionadas con la HU Administrar Rol

4. Modelo de datos de Historia de Usuario Registrar y listar trazas:

A diferencia de las Historias de Usuarios anteriores, *Registrar y listar Trazas* solo define los escenarios: a) registrar traza; b) listar trazas; c) buscar trazas; y d) mostrar detalles de trazas. El registro de las trazas involucra todas las entidades del modelo (Figura 9), definiendo los detalles de la traza, entre los que se encuentra el usuario que la genera al realizar una determinada operación. El resto de los escenarios emplean de igual modo todas las entidades, para mostrar detalles de las trazas.

Figura 5 Entidades Relacionadas a la HU Registrar y Listar Trazas

5. Modelo de datos de Historia de Usuario *Administrar configuración del sistema.*

La Configuración del sistema se divide en dos partes: a) la configuración por parte del usuario de su página de inicio; b) la configuración de atributos que definen el comportamiento global del sistema. Para el primer caso se definen el escenario editar configuración de página principal; en el cual el usuario define cuáles indicadores a los que tiene acceso en el sistema, se mostrarán en la página de inicio. Este escenario involucra las entidades del modelo (Figura 7) ***usuario***, ***conf_pag_principal*** y ***conf_pag_principal_indicador***, esta última es donde se registra el listado de indicadores a mostrar en la página principal del usuario.

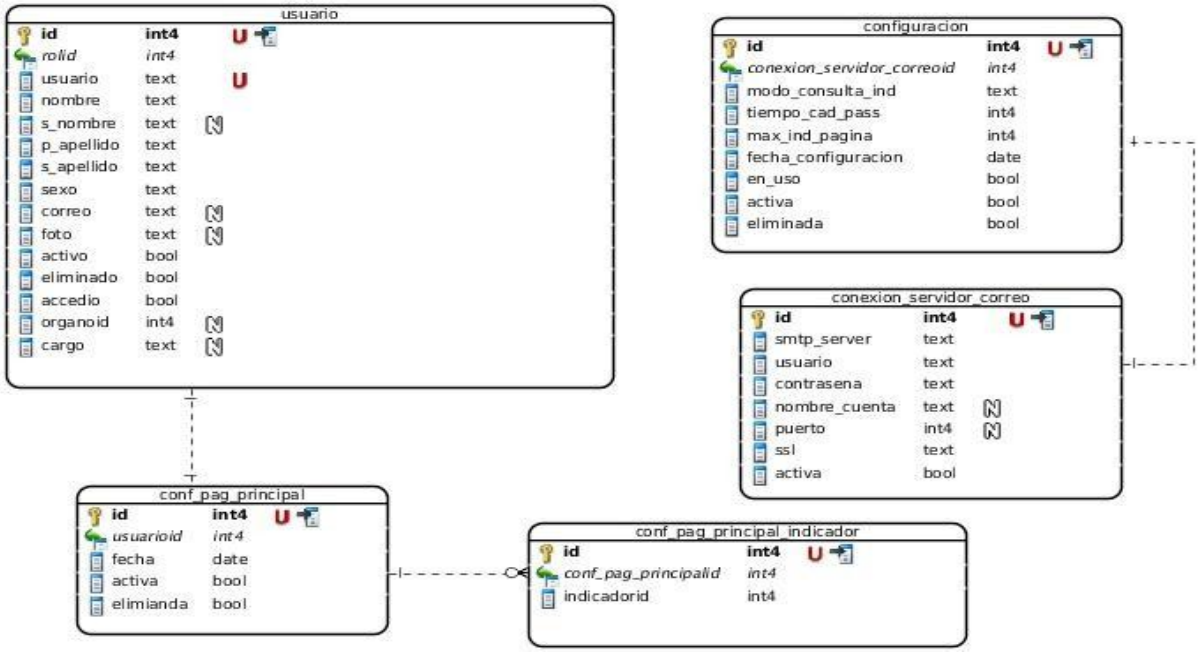


Figura 6 Entidades Relacionadas con la HU Administrar Configuración del Sistema

6. Modelo de datos de Historia de Usuario *Administrar Servicios Web*:

Esta Historia de Usuario define los escenarios: a) registrar servicio web; b) editar servicio web; c) eliminar servicio web; d) buscar servicio web; e) mostrar detalles de servicio web; y f) listar servicios web.

Para el escenario a), se emplean todos las entidades del modelo (Figura 8), dado a que en el registro de un servicio se debe definir cuáles indicadores proveerá dicho servicio (lo cual se registra en la entidad ***indicador_servicio***), además de qué órgano es el responsable de proveer dicho servicio. Los escenarios b) y c) emplean las entidades ***indicador_servicio*** y ***servicio;*** así como el resto de los escenarios utilizan todas las entidades del modelo para mostrar los detalles de los servicios web registrados.

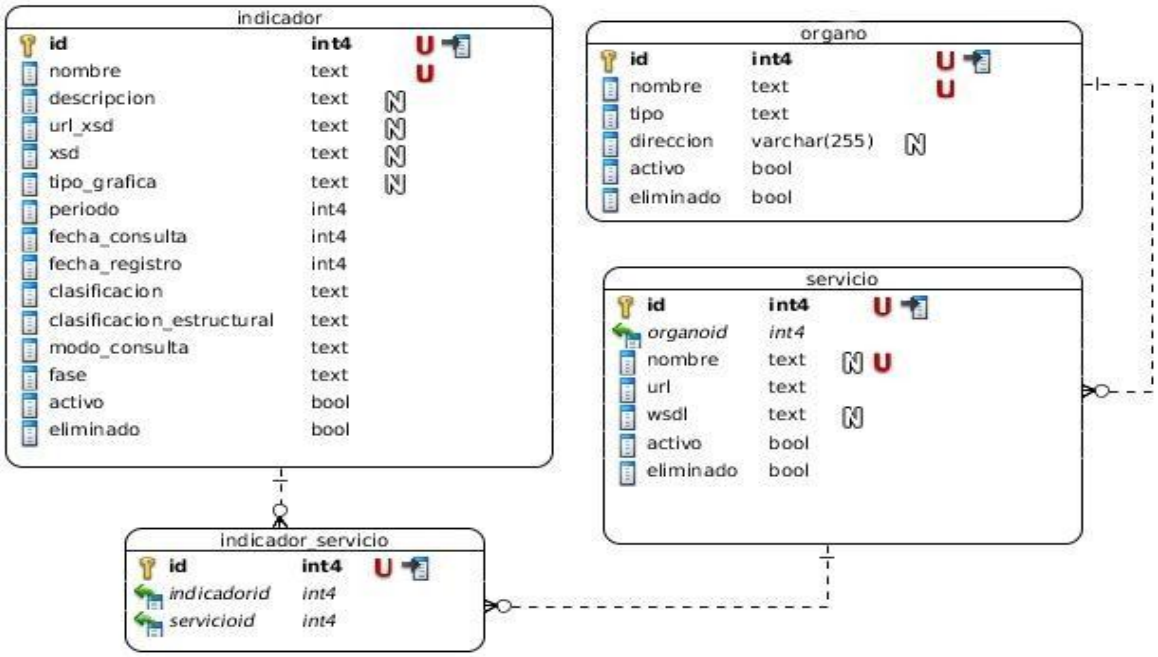


Figura 7 Entidades Relacionadas con la HU Administrar Servicios Web

7. Modelo de datos de Historia de Usuario *Administrar contraseña*

Administrar Contraseña define dos escenarios: a) cambiar contraseña; y b) cambiar contraseña sin verificación. El primero se refiere a la posibilidad de que cualquier usuario, desde cualquier sesión del sistema, pueda cambiar su contraseña a partir de especificar su nombre de usuario, contraseña actual y nueva contraseña. El segundo escenario es menos restrictivo, y se refiere a la posibilidad de actualizar la contraseña de un usuario, sin conocer su actual contraseña. Se hace uso de las entidades del modelo (Figura 9).

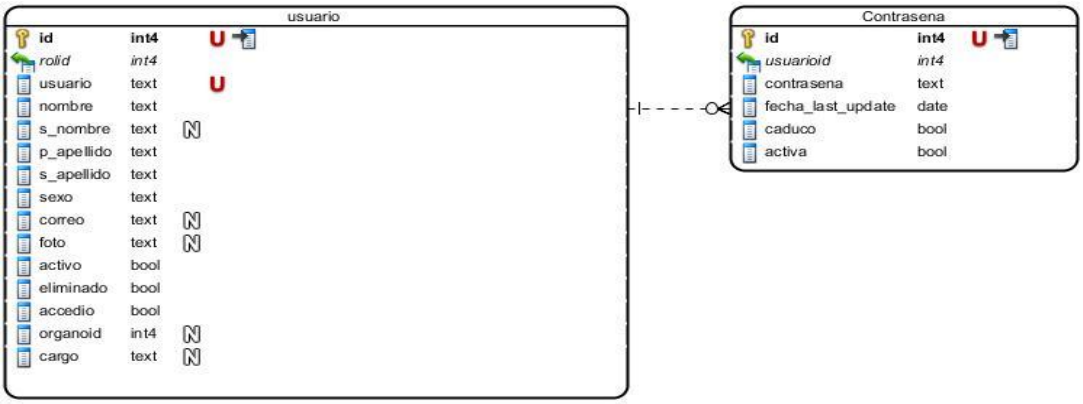


Figura 8 Entidades Relacionadas a la HU Administrar Contraseña

2.4.3 Patrones de diseño empleados

Aplicación de los patrones GRASP

Los patrones GRASP se dedican fundamentalmente a asignar responsabilidades a los objetos. Las primordiales responsabilidades de cualquier objeto son conocer sus atributos y relaciones con otros; así como hacer lo relacionado con las tareas que debe cumplir un objeto. Los patrones GRASP que se tuvieron en cuenta en la implementación del sistema se comentan a continuación:

Experto: consiste en "Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad." (21) Este patrón es aplicado en todas las clases debido a que cada una es experta pues contienen la información necesaria para cumplir con las responsabilidades que le fueron asignadas. Facilitando así el entendimiento, extensión y mantenimiento del sistema. Además de conservar el encapsulamiento y contribuir al bajo acoplamiento.

Controlador (*Controller*): El patrón controlador se utilizó en la definición de la capa de Negocio de la arquitectura, en la implementación de las clases controladoras del sistema. Estas clases tienen la responsabilidad de administrar las peticiones provenientes de las interfaces de usuario. Teniendo en cuenta que el patrón arquitectónico utilizado es el MVC, cada interfaz de usuario cuenta con al menos un controlador en el cual se gestionan las acciones que el usuario realiza en la vista. Un ejemplo en donde se evidencia esto, es en la interfaz para el registro y/o edición de usuarios del sistema ***UsuarioEdit.xhtml.*** El controlador de esta interfaz es ***UsuarioEditManager.java*** el que implementa las operaciones de registrar usuario (método ***save***) y editar usuario (método ***update***). Estos métodos precisamente responden a dos acciones que el usuario puede realizar en la interfaz. Para ello se debe establecer una relación entre las acciones en los componentes de la interfaz, y los métodos que gestionan dichas acciones en los respectivos controladores. A ese proceso se le conoce en el argot de la comunidad técnica que desarrolla sobre la tecnología Seam, como binding. Esta asociación se realiza instanciando el componente a partir del nombre que se le definió en su implementación, utilizando la anotación ***@Name*** del framework Seam.

Creador: Es muy similar el fundamento que justifica el uso de este patrón respecto al aportado para el uso del patrón Experto. Para este caso el patrón Creador permite identificar sobre qué clase recaerá la responsabilidad de la creación de objetos. Por

ejemplo, para el caso de la creación de instancias de objetos para su posterior persistencia en la base de datos, las clases responsables de crear las instancias son los controladores correspondientes, en este caso el controlador ***UsuarioEditManager.java***, el que además emplea directamente estas instancias de objetos Usuario en la implementación de los métodos ***save*** y ***update***.

Bajo Acoplamiento: Este patrón se refiere a la disminución de las dependencias entre clases. Para el caso de este trabajo, el patrón Bajo Acoplamiento se ve favorecido por otro patrón de diseño empleado: el patrón Inyección de Dependencias (DI). Este último garantiza disminuir las dependencia entre clases a partir de mecanismos de inyección de instancias, los cuales se realizan empleando los escenarios de ***injections*** y ***outjections*** que provee el framework Seam a partir de las anotaciones ***@In*** y ***@Out*** respectivamente. Sin embargo en el desarrollo de este trabajo no se implementa este patrón, sino que su utiliza, como otros patrones, a partir de la implementación que provee el framework Seam, no obstante por su utilizad se señala su empleo en esta sección.

Alta Cohesión: se basa en asignar una responsabilidad de modo que la cohesión siga siendo alta, es decir, que cada una de las clases con responsabilidades estrechamente relacionadas no realicen un trabajo enorme (21). El marco de trabajo de Seam favorece la alta cohesión asignado responsabilidades a las clases de manera que estas estén estrechamente relacionadas entre sí y sin llegar a realizar un trabajo enorme o excesivo.

Aplicación de patrones GoF.

Además de los patrones antes mencionados, también se utilizan patrones **GoF** (***Gang of Four***) en el desarrollo, ejemplos de los cuales son:

- ✓ **Patrón *Decorator*:** Este patrón define elementos generales de la vista en los cuales se enmarcan otros más específicos. El archivo llamado ***layout.php*** contiene el layout de la página, por lo cual se le llama plantilla global, y almacena el código HTML y la lógica de presentación común a todas las páginas de la aplicación. El contenido especializado de la vista se integra en este layout mediante un ***template*** o plantilla.
- ✓ **Patrón *Singleton:*** Este es un patrón cuyo propósito es la creación de instancias únicas de objetos, y precisamente por esto actúa en un ámbito de objetos. Su

utilidad radica en garantizar que una clase tenga una única instancia y de esta forma proporcionar un único punto de acceso a los miembros de ella. Se hizo necesario su empleo en este trabajo, en casos en los cuales se gestionaban recursos de manera centralizada, como es el caso de la hora y la fecha, mediante la implementación de la clase ***FechaHora.java*** la cual implementa este patrón.

En Java la mejor manera de implementar este patrón es definiendo el constructor de la clase como privado (***private***) e implementando un método público y estático (un método de clase) en el cual se verifique si está construida la instancia de la clase correspondiente y en cuyo caso se retorna esta instancia, y en caso contrario se retorna una nueva instancia de esta clase. (21)

2.4.4 Clases de Diseño

Una clase de diseño y sus objetos participan en varias realizaciones de casos de uso. También puede suceder que algunas operaciones, atributos y asociaciones sobre una clase específica sean solo relevantes para una sola realización de caso de uso. Para manejar todo esto se utilizan los diagramas de clases conectados a una realización de caso de uso, mostrando así sus clases participantes, subsistemas y sus relaciones [16]. A continuación se expone un ejemplo de diagrama de clases del diseño.

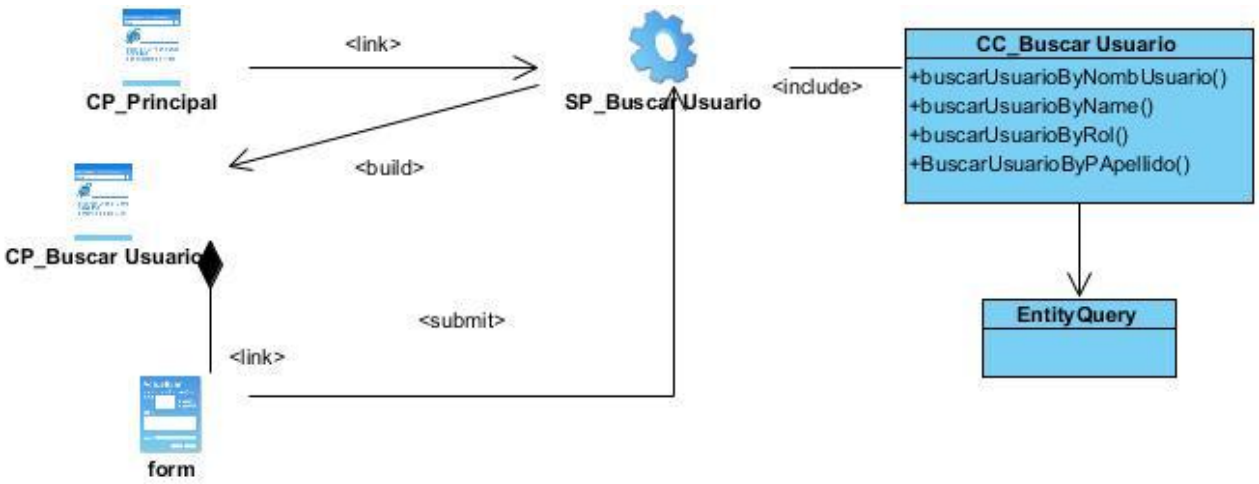


Figura 9 Diagrama de Clases del Diseño Buscar Usuario

Conclusiones del capítulo

El Plan de Iteraciones, dado la prioridad de las Historias de Usuario, determinó el periodo de implementación de estas, las cuales fueron guiadas y orientadas por las tarjetas C.R.C, permitiendo identificar las funcionalidades y características principales del sistema. Al emplear la metodología XP, se generó la documentación necesaria correspondiente a ingeniería de software: LRP y las HU. La elaboración de la LRP facilitó un mayor entendimiento y acuerdo común con los clientes en cuanto a la concepción de las funcionalidades que el sistema debe cumplir. A fin de obtener un sistema lo más estable y seguro posible se define una arquitectura software en capas basada en J2EE y una arquitectura de seguridad la cual se fundamenta en la autenticación del sistema y la autorización de acceso a los diferentes recursos. La aplicación de técnicas de validación de requisitos permitió afirmar que los requisitos obtenidos estaban en correspondencia con las solicitudes del cliente. Por otra parte, la validación del diseño mediante las métricas TOC y RC permitió obtener, de forma general, la creación de un diseño lo más sencillo posible y fácil de implementar.

Capítulo 3 Implementación y Pruebas

En el presente capítulo se detallan las iteraciones llevadas a cabo durante la construcción del módulo de administración y configuración. Se incluyen las tareas generadas por cada Historia de Usuario, así como las pruebas que se le realizaron a las funcionalidades del módulo.

3.1 Fase de Implementación

En esta fase se realiza la implementación de las Historias de Usuario que fueron grupadas en cada iteración. Se comienza con el resultado del diseño y se implementa el sistema en términos de componentes. Al inicio se lleva a cabo un chequeo del plan de iteraciones por si es necesario realizar modificaciones. Como parte de este plan se crean tareas de programación para ayudar a organizar la implementación exitosa de las Historias de Usuario. (13)

3.2 Tareas programadas por iteraciones

Cada Historia de Usuario como funcionalidad de la aplicación está compuesta por una o varias tareas de programación, éstas no son más que pasos lógicos a seguir por el programador para realizar la implementación de una Historia de Usuario. A continuación se detallan para cada una de las iteraciones las tareas a desarrollar por Historia de Usuario.

Iteración 1

Historia de Usuario	**Tiempo de Estimación (Semanas)**	
	Estimación	**Real**
Administrar Usuario	3	3.5
Administrar Órgano	3	2.9
Administrar Rol	3	3.2
Administrar Contraseña	1	1
Administrar Permisos	1	1
Adicionales de Usuario	1	1

Tabla 13 Estimación y Tiempo Real de Implementación por cada Historia de Usuario.

Historias de Usuario	Tareas de programación por Historias de Usuario
Administrar Usuario	Autenticar usuario Crear usuario Eliminar usuario Mostrar detalles de usuario Modificar datos de usuario Listar usuario registrado en el sistema
Administrar Órgano	Registrar nuevo órgano Eliminar órgano Modificar datos de órgano Listar órgano Buscar órgano Mostrar detalles de órgano
Administrar Rol	Crear rol Eliminar rol Listar rol Mostrar detalles de rol Editar rol
Administrar contraseña	Cambiar contraseña Cambiar contraseña sin verificación
Administrar permisos adicionales de usuario	Editar permisos adicionales a un usuario

Tabla 14 Tareas de Programación por cada Historia de Usuario.

Iteración 2

Historias de Usuario	Tiempo de Implementación (Semanas)	
	Estimación	Real
Administrar Informe	3	3.1
Administrar Indicador	3	3
Administrar permisos a Informes	1	1.2
Administrar permisos a Indicadores	1	0.9
Administrar asociaciones parte contraparte	3	2.5
Administrar Configuración del sistema.	3	3.2
Registrar y listar Trazas	3	3.1
Administrar Servicios	3	3.2
Listar y editar operaciones del sistema	3	2.5

Tabla 15 Tiempo estimado y real de implementación por cada historia de usuario

Historias de Usuario	Tareas de programación por Historias de Usuario
Administrar Informe	Crear informe Eliminar informe Listar informe Mostrar detalles de un informe Editar permisos extras de un informe
Administrar Indicador	Crear indicador Eliminar indicador Listar indicador Mostrar detalles de un indicador Editar indicador
Administrar permisos a Informes	Editar permisos de acceso a un informe
Administrar permisos a Indicadores	Editar permisos de generar reporte de un indicador
Administrar asociaciones	Definir asociación parte-contraparte entre indicadores
Administrar Configuración del Sistema	Registrar configuración de conexión al servidor de correo electrónico Definir indicadores a presentar en página principal Definir tiempo de caducidad de las contraseñas en el sistema
Registrar y listar Trazas	Visualizar las trazas Registrar trazas del sistema
Administrar Servicios Web	Crear servicios web Eliminar servicios web Listar servicios web Mostrar detalles de un servicios web Editar servicios web
Listar y editar operaciones del sistema	Visualizar las operaciones Registrar las operaciones del sistema

Tabla 16 Tareas de programación por cada historia de usuario

3.3 Estándares de codificación

Los estándares de codificación son reglas o buenas prácticas que se establecen para dotar el código de una mayor legibilidad y por tanto mayor capacidad de mantenimiento. Mediante la buena aplicación de los estándares de codificación se puede lograr unidad, claridad y limpieza en el código.

Los estilos utilizados son los siguientes:

Notación PascalCasing: En esta notación los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula. El objetivo es que cada nombre aporte la mayor información posible sobre sus características.

Notación lowerCamelCasing: Similar al PascalCasing, pero a diferencia de este, la letra inicial del identificador siempre estará en minúscula.
La forma en que se utilizarán los estilos son:

- ✓ **Para las clases:** se utilizará PascalCasing.
- ✓ **Para las funciones y variables:** se utilizará lowerCamelCasing.

3.4 Diagramas de Despliegue

Este diagrama se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes (nodos), representa la estructura del sistema en ejecución. Muestra la configuración en funcionamiento del sistema. Describen la topología del sistema la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones. Se utilizan para describir la vista de despliegue estática de un sistema. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP, microondas, etc. El servidor de aplicaciones web JBoss es el que despliega todos los sistemas de la plataforma, los que interactúan con la bases de datos PostgreSQL usando el protocolo TCP/IP. Los clientes realizan las peticiones al servidor a través del protocolo HTTP y su implementación segura HTTPS para el caso del intercambio de información sensible. Algunos de estos clientes estarán conectados a dispositivos de salida (impresoras), para realizar operaciones de impresión de informes.

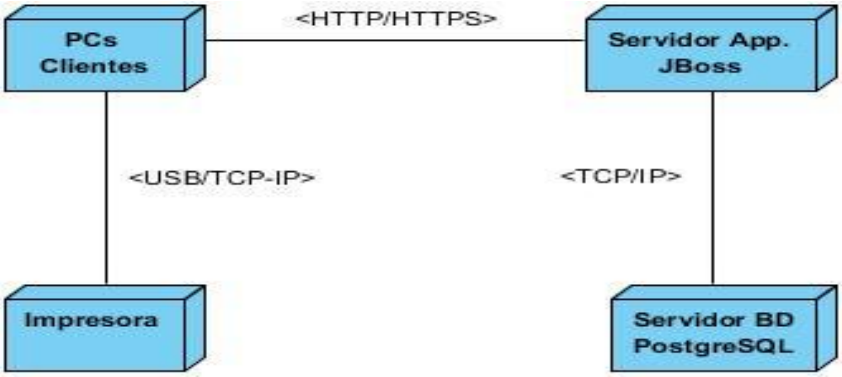


Figura 10: Diagrama de Despliegue

3.5 Fase de Prueba

Uno de los pilares de la metodología XP, es el proceso de pruebas. Esta anima a probar constantemente todo lo que se ha desarrollado, lo que permite aumentar la calidad de los sistemas. Reduce además el número de errores no detectados y disminuye el tiempo transcurrido entre la aparición de un error y su detección. Esta metodología divide las pruebas en dos grupos, estos son: pruebas de caja blanca y pruebas de caja negra. (37)

3.5.1 Pruebas de caja blanca

Durante la implementación del módulo de Administración y Configuración del sistema, se probaron las funcionalidades en la medida que se iban implementando. Para el caso de la realización de las pruebas de caja blanca, se empleó la técnica de Camino Básico. El objetivo de las pruebas de caja blanca es aislar cada parte del programa y mostrar que las partes individuales son correctas.

Las pruebas de Camino Básico consisten en derivar casos de prueba a partir de un conjunto de caminos independientes, por donde puede circular el flujo de control. Aunque estas pruebas no generen artefactos, son de vital importancia para el desarrollo del proyecto.

Los pasos que se siguieron para aplicar la técnica de Camino Básico son:

Paso 1: Construcción del grafo de flujo asociado al segmento de código que se desea probar.

Paso 2: Cálculo de la complejidad ciclomática del grafo (38).

Paso 3: Determinación de un conjunto básico de caminos independientes.

Paso 4: Realización de las pruebas de forma tal que se ejecute cada camino del conjunto básico.

Un ejemplo de la aplicación de este tipo de pruebas se muestra a continuación. Para este caso se utilizó la funcionalidad de ***Mostrar detalles de usuario***, específicamente el escenario en el cual se realiza la carga de los datos de un determinado usuario dado su identificador.

El siguiente segmento de código muestra la implementación del método ***setIdUsuario*** de la clase controladora ***UsuarioDetailsManager***, el cual tiene la responsabilidad de buscar en la base de datos el usuario a partir de su identificador, y a partir de él actualizar los campos de la interfaz ***UsuarioDetails.xhtml***.

```
public void setIdUsuario(int idUsuario) {
    this.idUsuario = idUsuario;   //1
    if(idUsuario != -1){ //2
        usuario = (Usuario) entityManager.find(Usuario.class, idUsuario);//3

        if(funcionarioList.isFuncionario(usuario)) //4
            funcionario = true; //5


        activo = usuario.isActivo(); //6
        cargo = usuario.getCargo();  //6
        correo = usuario.getCorreo();//6
        nombre = usuario.getNombre();//6
        segundoNombre = usuario.getSNombre();//6
        primerApellido = usuario.getPApellido();//6
        segundoApellido = usuario.getSApellido();//6
        nombreOrgano = usuario.getOrgano().getNombre();//6
        rol = usuario.getRol().getRol();//6
        sexo = usuario.getSexo();//6
        userName = usuario.getUsuario();//6

    }

}
```

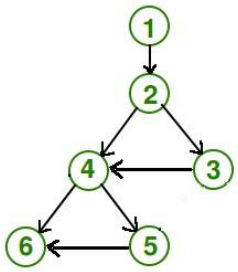


Figura 11 Grafo de Flujo asociado a la funcionalidad Mostrar Detalles de Usuario

Dado el segmento de código, se construye el grafo de flujo (Figura 11) y posteriormente se realiza el cálculo de la complejidad ciclomática a partir de tres variantes (39)

representadas a continuación, las cuales arrojan el mismo resultado, lo cual es una condición necesaria para asegurar que el cálculo de la complejidad es correcto.

Variante 1: Número de nodos y aristas del grafo.

$$V(G) = (E - N) + p;\ A, N \in \mathbb{N}, p = 2, E > 1, N > 1 \quad (1)$$

Donde:

E: Cantidad de aristas

N: Cantidad de nodos

P: Cantidad de componentes conectados en una arista

De manera que:

V (G) = (7-6) + 2

V (G) = 3

Variante 2: Cálculo de la complejidad ciclomática a partir del número de nodos predicados.

$$V(G) = P + 1; P \in \mathbb{N}, P > 0 \quad (2)$$

Donde:

P: Cantidad total de nodos predicados del grafo.

De manera que:

V (G) = 2+ 1

V (G) = 3

Variante 3: Cálculo de la complejidad ciclomática a partir del número de regiones del grafo.

$$V(G) = R \quad (3)$$

Donde:

R: Cantidad total de regiones del grafo.

De manera que:

V (G) = 3

El cálculo efectuado mediante las fórmulas (1), (2) y (3) arrojó un mismo valor, lo que indica que existen 3 posibles caminos por donde el flujo puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

Las representaciones de los caminos básicos por los que puede recorrer el flujo son:

Camino básico #1: 1-2-3-4-6

Camino básico #2: 1-2-4-5-6

Camino básico #3: 1-2-4-6

Se diseñaron tres pruebas para este caso, y se ejecutaron satisfactoriamente, permitiendo concluir que la implementación de la funcionalidad correspondiente al escenario en el cual se realiza la carga de los datos de un determinado usuario dado su identificador, para ser mostrado posteriormente en una interfaz, fue correctamente implementada.

Este mismo proceso para la determinación del número de pruebas, se realizó para cada escenario de las Historias de Usuario del sistema, lo cual permitió constatar que todas las funcionalidades quedaron correctamente implementadas. Sin embargo se considera que la realización de pruebas de caja blanca debió planificarse en varias iteraciones por escenarios y no en una sola iteración, dado a que de esta última forma no se cuenta con un registro de evolución de la implementación.

3.5.2 Pruebas de caja negra

Las pruebas de caja negra se desarrollan sobre la interfaz del software. Estas pruebas se encargan de verificar que las funciones realizadas por el sistema sean satisfactoria o no. Estas pruebas permitieron verificar que las Historias de Usuario son satisfactorias, sin tener en cuenta el funcionamiento interno del sistema. La realización de estas pruebas permitió encontrar:

- ✓ Funciones incorrectas
- ✓ Errores en la interfaz de usuario
- ✓ Errores en las salidas del sistema
- ✓ Errores en el acceso a los datos

Para evaluar el resultado obtenido por la solución, se planificaron 2 iteraciones de pruebas, en las cuales se evaluó el funcionamiento de la aplicación (Figura 12).

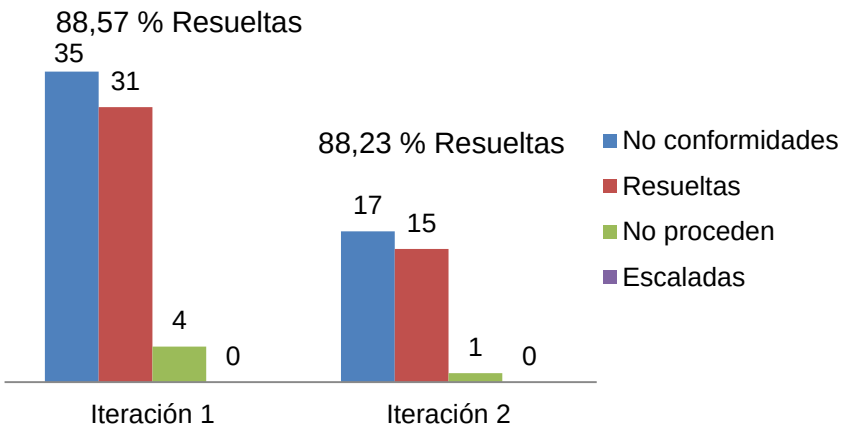


Figura 12 Resultados de las pruebas funcionales en 3 iteraciones.

De manera general, las pruebas permitieron identificar 52 no conformidades, de ellas se resolvieron 46, lo cual representa el 88,46% del total. Estas no conformidades no repercuten en el cumplimiento de los requerimientos de seguridad establecidos por el Grupo de Coordinación, por lo cual no influyen directamente en el cumplimiento del objetivo de este trabajo. Se considera que independientemente de esto, se implementaron satisfactoriamente los requerimientos del sistema de manera general.

A continuación se muestra algunos casos de prueba realizados en la segunda iteración, la cual permitió cerrar el ciclo de pruebas:

Caso de Prueba Funcional	
Código: HU1_P1_I3	Historia de Usuario: No.1 Administrar Usuario
Nombre: Autenticar usuario	
Descripción: Prueba para verificar la funcionalidad autenticar usuario.	
Condiciones de Ejecución: El usuario debe estar registrado en el sistema.	
Entrada / Pasos de ejecución: Es introducido el usuario y la contraseña en los campos requeridos. El usuario seleccionará la opción aceptar.	
Resultado Esperado: El sistema debe iniciar la sección según el rol del usuario, mostrando la página de inicio.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 17 Caso de Prueba Autenticar Usuario.

Caso de Prueba Funcional	
Código: HU1_P2_I3	**Historia de Usuario:** No.1 Administrar Usuario
Nombre: Crear usuario	
Descripción: Prueba para comprobar la creación de un usuario.	
Condiciones de Ejecución: El usuario debe estar registrado en el sistema y tener permiso para crear un usuario.	
Entrada / Pasos de ejecución: Son introducido los valores usuario, nombre, segundo nombre, primer apellido, segundo apellido, contraseña, correo electrónico, órgano, rol, sexo, cargo, funcionario, activo en los campos requeridos. Se selecciona la opción crear.	
Resultado Esperado: El sistema debe registrar el usuario insertado.	
Evaluación de la Prueba: Prueba satisfactoria.	

Tabla 18 Caso de Prueba Administrar Usuario

3.6 Lista de Chequeo

Las listas de chequeo son Propuestas del Instrumento de Medición que están basadas en la identificación de las Técnicas de Prueba para evaluar cada una de las Características de Calidad. Una lista de chequeo es un formulario de preguntas, las cuales dependen del objetivo para el cual son usadas .Estas listas están clasificadas según las etapas del Proceso de Prueba. Para dar respuesta a cada pregunta se considera una escala del 1 al 5, en donde el uno (“1”) siempre es la respuesta menos significativa y cinco (“5”) la más significativa (36). Las listas de chequeo enumeran una serie de puntos (muchos o pocos dependerá de la exhaustividad que se pretenda) que deberían verificarse uno a uno para asegurarnos de lograr el producto final con un nivel de calidad previamente aceptado (36). *(Ver anexo 3 “Lista de Chequeo”.)*

Conclusiones del Capítulo

En el capítulo se observó como la planificación de las tareas a realizar durante el proceso de implementación del sistema, pudo proporcionar una mejor organización del trabajo. El empleo de estándares de codificación, facilitó la comprensión y el mantenimiento del código, manifestándose en el funcionamiento del sistema. Las pruebas realizadas a la solución brindaron un resultado satisfactorio, lo que demuestra que la realización de las mismas juega un papel fundamental en el proceso de desarrollo de software. Se desarrolló el Módulo de Administración y Configuración del Repositorio de Esquemas y Metadatos versión 2.0, lográndose un correcto manejo de la administración y configuración a partir de los datos que se intercambian en el REM.

Conclusiones Generales

- ✓ El estudio de los sistemas informáticos existentes a nivel internacional, permitió identificar elementos a tener en cuenta para elevar la calidad de la solución, y adecuarlas a las necesidades del cliente.
- ✓ Se identificaron mecanismos de interoperabilidad basados en el uso de XML como estándar para el intercambio de datos entre sistemas, haciendo uso de los servicios web.
- ✓ Las pruebas realizadas validaron el correcto funcionamiento de las funcionalidades del sistema, verificando la calidad del producto, por cuanto se cumplió con éxito el principal objetivo del trabajo.
- ✓ Con el desarrollo del sistema se pudo mejorar la administración y configuración del Repositorio de Esquemas y Metadatos, garantizando un correcto manejo de la administración y la configuración de los datos e informaciones que se intercambian en el mismo.

Recomendaciones

Se recomienda:

- ✓ Desarrollar los módulos Informe y Reportes del REM v2.0.
- ✓ Validar los requerimientos con el cliente, a partir de la firma de los mismos.
- ✓ Realizar las pruebas de aceptación del módulo de administración y configuración.
- ✓ Actualizar las funcionalidades necesarias para permitir la interoperabilidad de los módulos existentes con los que posteriormente se integrarán al sistema.
- ✓ Continuar el estudio de las estrategias de eGobierno para mejorar los servicios ofrecidos a los ciudadanos para aumentar la eficiencia y eficacia de la gestión pública e incrementar la participación ciudadana.

Bibliografía

1. ¿Qué es XML? [En línea] 15 de Abril de 2004. [Citado el: 1 de 02 de 2013.] http://mat21.etsii.upm.es/mbs/mechxml/que_es_xml.htm.

2. **Europe., Free Software Foundation.** EIFv2. [En línea] 2011-2013. [Citado el: 12 de 12 de 2012.] http://fsfe.org/activities/os/eifv2.en.html.

3. Ecured. [En línea] 2013. [Citado el: 29 de 5 de 2013.] http://www.ecured.cu/index.php/Interoperabilidad.

4. EUPAN - European Public Administration Network - Knowledge Center. [En línea] 2010-2011. http://www.eupan.eu.

5. **Valdés González, Jorge Luis.** *Modelo de intercambio de datos basado en la homogenización del formato de intercambio.* La Habana : Universidad de las Cencias Informaticas, Departamento de Informática Jurídica, 2012.

6. **Herrera, Esteban Almuna.** Contenidos para Repositorios de Esquemas y Metadatos - Documentos Electrónicos de Servicios Públicos. *Tesis Electrónicas Universidad de Chile.* [En línea] 2012. [Citado el: 20 de 04 de 2013.] http://www.tesis.uchile.cl/tesis/uchile/2007/almuna_e/sources/almuna_e.pdf.

7. **Quin, Liam.** www.w3.org. [En línea] [Citado el: 01 de 02 de 2013.] http://www.w3.org/standars/xml/.

8. **Perens, Bruce.** Perens. [En línea] 2007. http://perens.com/OpenStandards/Definition.html.

9. Guía Breve de Servicios Web. *w3C World Wide Web Consortium.* [En línea] 2013. http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

10. *Administrador de Esquemas y Metadatos AEM.* **Ministerio de Economía Fomento y Reconstrucción.** Chile : Estrategia Digital, 2009.

11. El Movimiento de Acceso Abierto. [En línea] Lic. Nancy Sánchez Tarragó. http://bvs.sld.cu/revistas/aci/vol16_3_07/aci05907.html.

12. SciELO - Scientific Electronic Library Online. [En línea] 2013. [Citado el: 12 de 04 de 2013.] http://scielo.sld.cu/scielo.php.

13. **Wells, Don.** Extreme Programming. [En línea] 28 de 09 de 2009. http://www.extremeprogramming.org.

14. Herramienta CASE. [En línea] EcuRed, 2013. http://www.ecured.cu/index.php/Herramienta_CASE.

15. UML, BPMN and The Enterprise Architecture Tool for Software Development. [En línea] 2013. [Citado el: 13 de 3 de 2013.] http://www.visual-paradigm.com/.

16. **Farley, Jim.** *Practical JBoss® Seam Projects.* New York : Steve Anglin, 2007.

17. **BEATON, W.** Eclipse Platform Technical Overview. [En línea] 2006. http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html.

18. Java.net. [En línea] 2013. [Citado el: 13 de 3 de 2013.] https://home.java.net/.

19. XML. [En línea] 2003. [Citado el: 13 de 3 de 2013.] http://www.hipertexto.info/documentos/xml.htm.

20. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison - Wesley, 2000.

21. *UML y Patrones: Introducción al análisis y diseño orientado a objetos. I y II.* La habana : Felix Varela, 2004.

22. POSTGRESQL. [En línea] 2013. [Citado el: 16 de 1 de 2013.] http://www.postgresql.org.es/sobre_postgresql.

23. Hibernate.org. *hibernate - jboss community.* [En línea] RedHat, 2013. [Citado el: 13 de 3 de 2013.] http://hibernate.org.

24. RichFaces - JBoss Community. [En línea] REDHAT, 2013. [Citado el: 13 de 3 de 2013.] http://www.jboss.org/richfaces.

25. Thye Hibernate Query Language. [En línea] 2004. [Citado el: 13 de 3 de 2013.] http://docs.jboss.org/hibernate/orm/3.3/reference/en/html/queryhql.html.

26. JBoss AS 5 Development. [En línea] Birmingham Packt Publishing Ltd, 2009.

27. UDDI Online Cmmunity for the universal description, discovery and integration. *UDDI xml.org.* [En línea] OASIS, 1993-2013. http://uddi.xml.org.

28. **Andres, Kenth Beck & Cynthia.** *Extreme Programming Explained: Embrace Change, 2nd Edition.* s.l. : Addison-Wesley, 2004. 2.

29. Técnicas para identificar Requisitos Funcionales y No Funcionales. *Metodología Gestión de Requerimientos.* [En línea] 2013. [Citado el: 13 de 3 de 2013.] https://sites.google.com/site/metodologiareq/capitulo-ii/tecnicas-para-identificar-requisitos-funcionales-y-no-funcionales.

30. Planeacion - Metodología XP. [En línea] Google Sites, 2013. [Citado el: 13 de 3 de 2013.] https://sites.google.com/site/xpmetodologia/caso-practico/planeacion.

31. **Astudillo, Marcello Visconti y Hernan.** *Fundamentos de Ingeniería de Software.* Universidad Técnica Federico Santa María : s.n.

32. **Michael Juntao Yuan, Jacob Orshalick, Thomas Heute.** *SEAM Framework. Experience the evolution of Java EE.* Indiana : Pearson Education, 2009.

33. **Autores, Compendio de.** *A Framework for JAVA-SEAM contextual components.* págs. 246-264.

34. Hibernate - Persistencia Relacional para Java idiomático. [En línea] 2004. [Citado el: 13 de 3 de 2013.] http://docs.jboss.org/hibernate/core/3.6/reference/es-ES/html_single/.

35. **Pressman, Roger S.** *Ingeniería de Software, un enfoque práctico.* s.l. : McGraw-Hill Companies, 2002.

36. **Martín Casal, M.** *La Técnica de las Chesklisten.* 1989.

37. Extreme Programming. [En línea] 2009. [Citado el: 13 de 3 de 2013.] http://www.extremeprogramming.org/.

38. **Lopez, Miguel and Habra, Naji.** Relevance of the Cyclomatic Complexity. Pennsylvania State University : s.n., 2010.

39. *A Complexity Measure*. **McCabe, T. J.** s.l. : Engineering, IEEE Transactions on Software, 1976, Vol. 2.

40. **Cobas, Pedro Manuel Nogales.** CyTA Ciencia y Técnica Administrativa. [En línea] [Citado el: 10 de 05 de 2013.] http://www.cyta.com.ar/ta1002/v10n2a3.htm.

41. **Morton, Michael J. Scot.** *DSS are interactive computer-based systems, which help decision makers utilize data and models to solve unstructured problems.* Boston : s.n., 1971.

42. Pruebas Funcionales. *Calidad y Software.* [En línea] 2009. http://www.calidadysoftware.com/testing/pruebas_funcionales.php.

43. Pruebas Caja Blanca. [En línea] ECURED, 2013. http://www.ecured.cu/index.php/Pruebas_de_caja_blanca.

44. Sistema de Soporte de Decisiones. [En línea] 2009. http://www.tecnologias-informacion.com/soportedecisiones.html.