

Universidad de las Ciencias Informáticas
Facultad 3



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

Título: Módulos de Administración y Configuración del Portal
Órgano de Justicia-MININT

Autor: Jorge Navarro Martínez
Tutor: Ing. Jorge Luis Valdés González

La Habana, Cuba
Junio 2012

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Jorge Navarro Martínez

Firma del Autor

Jorge Luis Valdés González

Firma del Tutor

Pensamiento:

Siempre he despreciado las posesiones, el éxito aparente, la publicidad y el lujo, creo que lo mejor es que uno lleve una vida sencilla y modesta.

Albert Einstein

Agradecimientos:

Mi agradecimiento más profundo a mis padres que siempre me han apoyado a lo largo de la vida, gracias por su dedicación.

A mi hermano por ser la persona más importante y ser mi amigo, que siempre me apoyó en las decisiones que he tomado y por ser un modelo a seguir en la vida. Gracias por tu amor y dedicación.

A mi esposa Yelene por el amor, dedicación y apoyo que me brinda cada día. A mis amigos de aula que siempre fueron una ayuda preciosa.

A mi tutor Jorge Luis Valdés que sin él no habría podido terminar la tesis.

Dedicatoria

A mi esposa que me dio aliento hasta el último día.

A mis padres que me lo han dado todo, hoy les entrego el fruto de lo que crearon.

A mis suegros que siempre me apoyaron.

Resumen

El Grupo de Coordinación de la Política Penal y Penitenciaria es un órgano de estado encargado de tomar decisiones en materia Penal y Penitenciaria. Durante los procesos de tomar decisiones en este órgano, se ejecutan varios flujos de intercambio de información generalmente confidencial o secreta, producto a la naturaleza de la misma. En consecuencia, y teniendo en cuenta las implicaciones que traería el acceso no autorizado a dicha información, existen disposiciones que establecen cuál es la política de compartimentación de la información en este nivel.

Teniendo en cuenta que actualmente se desarrolla una plataforma de interoperabilidad para el intercambio de datos, que contribuirá a la eficiencia y eficacia con que se realizan estos flujos de información, el Grupo de Coordinación ha establecido un conjunto de requerimientos de seguridad que constituyen pautas a cumplir para garantizar la política de compartimentación de información. La necesidad de que las tecnologías a emplear para informatizar los flujos de información en los órganos del estado, como el Grupo de Coordinación, cumplan con las políticas de seguridad, es de interés para la seguridad nacional.

En este trabajo se desarrollan los módulos de Administración y Configuración del sistema para el apoyo a la toma de decisiones en el Grupo de Coordinación, a partir del cumplimiento de los requerimientos establecidos por el órgano sobre la compartimentación de la información. El resultado de este trabajo es estratégico para el estado, y constituye un importante complemento en el desarrollo de la Plataforma de Interoperabilidad, al mismo tiempo que responde a lineamientos y necesidades de la política del país.

Índice

Introducción.....	10
1.1 Conceptualización.....	16
1.2 Plataforma de interoperabilidad para el intercambio de datos	18
1.3 La web y los sistemas de apoyo a la toma de decisiones	20
1.3.1 Características de un sistema de apoyo a la toma de decisiones	21
1.3.2 Componentes de un DSS.....	22
1.3.3 Clasificación de los DSS	23
1.4 Administración y configuración del portal	25
1.5 Marco tecnológico para el desarrollo	26
1.5.1 Plataforma de desarrollo Java EE	28
1.5.2 Marco de trabajo (Framework)	29
1.5.3 Entorno de Desarrollo Integrado (IDE).....	30
1.5.4 Herramientas para el modelado	31
1.5.5 Sistema Gestor de Base de Datos.....	32
1.5.6 Servidor de aplicaciones web Jboss AG.....	32
1.5.7 Sistema de control de versiones	33
1.6 Metodología de desarrollo	34
1.7 Conclusiones del capítulo	34
2.1 Fase de Planificación.....	35
2.1.1 Historias de Usuario (HU).....	35
2.1.2 Lista de reserva del producto	37
2.1.3 Requisitos no funcionales del Sistema	43
2.1.4 Planificación de entregas (Release planning)	45
2.2 Fase de Diseño.....	46
2.2.1 Tarjetas Clase, Responsabilidades y Colaboración (C.R.C).....	47
2.3 Arquitectura de software.....	48
2.3.1 Arquitectura de seguridad	50
2.2.2 Modelo de datos (entidades persistentes).....	52
2.2.3 Patrones de diseño.....	63
2.2.4 Diagrama de despliegue.....	66
2.2.5 Conclusiones del Capítulo	67
3.1 Fase de Implementación	68

3.1.1	Tareas de programación por iteraciones	68
3.2	Fase de Prueba.....	72
3.2.1	Pruebas de caja blanca.....	72
3.2.2	Pruebas de caja negra	75
3.3	Conclusiones del capítulo	79
	Conclusiones Generales.....	80
	Recomendaciones	81
	Bibliografía	82

Índice de tablas

Tabla 1:	Tabla comparativa para el análisis de selección de tecnologías para el desarrollo	28
Tabla 2:	Historias de Usuario agrupadas	36
Tabla 3:	Operaciones asociadas por Historias de Usuarios.....	36
Tabla 4:	Plan de duración de iteraciones	46
Tabla 5:	Tarjetas C.R.C UsuarioIndicadorEditManager	47
Tabla 6:	Tarjetas C.R.C UsarioDetailsManager	47
Tabla 7:	Tarjetas C.R.C UsuarioDeleteManager	48
Tabla 8:	Tiempo estimado y real de implementación por cada Historia de Usuario.....	68
Tabla 9:	Tareas de programación por Historia de Usuario.....	69
Tabla 10:	Tiempo estimado y real de implementación por Historia de Usuario.....	70
Tabla 11:	Tareas de programación por Historia de Usuario.....	70
Tabla 12:	Tiempo estimado y real de implementación por Historia de Usuario.....	71
Tabla 13:	Tareas de programación por Historia da Usuario.....	71
Tabla 14:	Caso de prueba Autenticar Usuario	78
Tabla 15:	Caso de prueba Crear Usuario	78
Tabla 16:	Caso de prueba Eliminar Usuario	79

Índice de figuras

Figura 1: Arquitectura tecnológica de la Plataforma de Interoperabilidad	18
Figura 2: Escenario de intercambio de datos entre el Portal del Órgano y sistemas externos a la plataforma.	19
Figura 3: Arquitectura en tres capas de un Centro de Datos	27
Figura 4: Implementación de la arquitectura en capas.....	49
Figura 5: Distribución física de componentes de la arquitectura.....	50
Figura 6: Entidades relacionadas con la HU Administrar Usuario	53
Figura 7: Entidades relacionadas a la HU Administrar Rol	54
Figura 8: Entidades relacionadas a la HU Administrar Órgano.....	54
Figura 9: Entidades relacionadas a la HU Registrar y listar trazas	55
Figura 10: Entidades relacionadas con la HU Administrar Indicador	56
Figura 11: Entidades relacionadas a la HU Administrar Informe	57
Figura 12: Entidades relacionadas con la HU Administrar Configuración del sistema.....	58
Figura 13: Entidades relacionadas con la HU Administrar permisos adicionales de usuario	59
Figura 14: Entidades relacionadas a la HU Administrar permisos a indicador	59
Figura 15: Entidades relacionadas a la HU Administrar permisos a informes.....	60
Figura 16: Entidades relacionadas a la HU Administrar Servicios Web	61
Figura 17: Entidades relacionadas con la HU Administrar Operaciones	61
Figura 18: Entidades relacionadas con la HU Administrar asociación parte-contraparte ..	62
Figura 19: Entidades relacionadas con la HU Administrar Contraseña	63
Figura 20: Diagrama de Despliegue	67
Figura 21: Grafo de flujo asociado a la funcionalidad Mostrar detalles de usuario	73
Figura 22: Resultados de las pruebas funcionales en tres iteraciones	76
Figura 23: Resultados generales de las pruebas funcionales	77

Introducción

En Cuba el Grupo de Coordinación de la Política Penal y Penitenciaria¹ es un órgano del estado que al igual que otras organizaciones emplea herramientas informáticas para el apoyo a la toma de decisiones. Este órgano 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 (MININT). Además, cuenta con una Secretaría Ejecutiva, de la cual está a cargo la FGR y un Grupo Técnico integrado por varios funcionarios del Sistema de Justicia en Cuba. Su objetivo fundamental es tomar decisiones a nivel de estado en materia de política penal y penitenciaria, sobre los aspectos más significativos del Sistema de Justicia, a partir de una serie de datos² que le proveen los organismos que lo componen. Está estructurado en grupos de trabajo a nivel provincial y por una Comisión Nacional que sesiona trimestralmente, para la que se confecciona un informe por el MININT, en el cual se evalúa el cumplimiento de la aplicación de la política penal y penitenciaria en el país. La Comisión Nacional emite una serie de indicaciones a partir de las cuales se definen indicadores de trabajo preestablecidos en disposiciones del MININT, la FGR y el TSP.

La Secretaría Técnica del Grupo de Coordinación se encarga de notificar a los organismos correspondientes, cuáles son los indicadores que deberán conformar los informes a discutir en la próxima sesión de la Comisión Nacional. A partir de entonces, los organismos miembros comienzan un proceso de recopilación de datos, apoyado en los sistemas informáticos que facilitan la gestión de los datos primarios para el posterior análisis de los indicadores establecidos.

La conformación de los informes de cada organismo se realiza manualmente, y son entregados al MININT para que finalmente este confeccione el informe general que es entregado a la Secretaría Técnica de la Comisión Nacional. Este proceso demora aproximadamente treinta días, lo que representa una considerable demora en el proceso de toma de decisiones, dado a que los datos no son obtenidos oportunamente. Este elemento adquiere una notable significación, dado a que en un escenario de toma de decisiones la pertinencia de los datos sobre los cuales se realizan los análisis, determina en gran medida la eficacia de las decisiones tomadas. Otro aspecto a resaltar en la

¹ También conocido como Órgano Justicia-MININT. En lo adelante Grupo de Coordinación.

² También conocidos como indicadores del órgano.

confección de estos informes, es la seguridad con que se manipulan, dado a que solo podrán tener acceso a ellos los funcionarios autorizados.

Si hay problemas en la pertinencia con que se obtienen los datos para la toma de decisiones a nivel de estado, esto incide negativamente en la consolidación de la seguridad ciudadana y la efectividad en la prevención y el enfrentamiento contra el delito dentro de la sociedad cubana; lo que evidencia la necesidad social que adquiere la solución del problema sobre el acceso oportuno a los indicadores del Grupo de Coordinación. Al mismo tiempo, la compartimentación en el acceso a los datos mediante los informes, es un aspecto crítico debido fundamentalmente a las siguientes razones: a) los datos a partir de los cuales el Grupo de Coordinación toma decisiones, son confidenciales, algunos de ellos secretos, ya que determinan la política de estado en materia Penal y Penitenciaria; y b) en consecuencia a lo anterior se identifica un riesgo de manipulación de los datos, ya que manipulados por personas no autorizadas, pueden comprometer la política de estado.

El Centro de Gobierno Electrónico (CEGEL) adscrito a la Facultad 3, de la Universidad de las Ciencias Informáticas (UCI), inició el desarrollo de una plataforma de interoperabilidad para el intercambio de datos entre sistemas desarrollados sobre tecnologías heterogéneas. La arquitectura propuesta para esta plataforma (Hogguit, et al., 2011) está definida a partir de un modelo de intercambio de datos basado en la homogenización del formato de intercambio (Valdés González, 2012), utilizando varios estándares internacionales de interoperabilidad, como son XML, XSD, Servicios Web, entre otros. Uno de los componentes de esta arquitectura es el portal del Grupo de Coordinación (denominado Portal Órgano Justicia-MININT), concebido como un sistema de apoyo a la toma de decisiones, mediante el cual los titulares de la Comisión Nacional del órgano accederán a los indicadores que necesitan para desarrollar el proceso de toma de decisiones.

Como en cualquier escenario de toma de decisiones en política de estado, la información que se gestiona y transmite requiere que cumpla los niveles más elementales de seguridad. Para el caso del Grupo de Coordinación, cada uno de los indicadores que emiten los organismos miembros, 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 tanto en la realización manual del proceso como a través de algún sistema informático.

El Portal del Órgano Justicia-MININT como herramienta para el apoyo a la toma de decisiones, tiene como objetivo fundamental proveer mecanismos que permitan a los titulares acceder oportunamente a los indicadores. Sin embargo, estos mecanismos deben cumplir las disposiciones de seguridad en la compartimentación de los datos que establece el Grupo de Coordinación. De igual forma, existen una serie de requerimientos³ que el órgano ha establecido, a partir de los cuales se ha conducido el desarrollo de la plataforma de interoperabilidad, y que no quedan exentos en el desarrollo del portal. Estos requerimientos son:

- Requerimiento de **no intrusión**: el acceso a los datos no podrá realizarse directamente en las fuentes de datos (bases de datos) de los organismos.
- Requerimiento de **dato primario**: los datos deberán ser proveídos por sus fuentes primarias.
- Requerimiento de **no suplantación tecnológica**: la solución brindada deberá evitar sustituir los sistemas de información que actualmente existen.
- Requerimiento de **seguridad en el intercambio de datos**: los datos deberán intercambiarse a partir de mecanismos que garanticen el cumplimiento de indicadores de seguridad establecidos.
- Requerimiento de **compartimentación de la información**: la información deberá accederse por aquellos a los cuales se les han otorgado el autorizo de acceso a ella.

Un importante requerimiento de seguridad, específicamente asociado a la administración de los sistemas de la plataforma de interoperabilidad, es el referido a la capacidad que tengan estos sistemas de proveer un mecanismo de dualidad en la administración. Esto quiere decir, según la terminología que emplea la Secretaría Técnica del órgano, que los sistemas deben permitir una administración técnica y una administración funcional. 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

³ Estos requerimientos están dispuestos en el documento Convenio Colaboración, el cual está en proceso de aprobación por los titulares del Grupo de Coordinación de la Política Penal y Penitenciaria.

del sistema con el funcionamiento de la organización, o sea el órgano en cuestión. Es decir, se refiere a la implementación de las políticas de compartimentación de la información, tales como: definición de informes, otorgamiento de permisos de accesos a los informes y colaboración en el intercambio de documentos de acceso restringido.

A raíz de la situación planteada se define el siguiente **problema a resolver**: *¿cómo implementar los mecanismos de administración y configuración del Portal del Grupo de Coordinación de la Política Penal y Penitenciaria de manera que se garanticen los requerimientos de seguridad establecidos por este órgano?* El **objetivo general** de este trabajo es desarrollar los módulos de Administración y Configuración del portal del Grupo de Coordinación, teniendo en cuenta el cumplimiento de los requerimientos establecidos por este órgano. Por lo que se establece como **objeto de estudio** el Proceso de Desarrollo de Software; y el desarrollo de mecanismos de administración y configuración para el sistema de apoyo a la toma de decisiones del Grupo de Coordinación el **campo de acción**.

A partir de lo antes expuesto se formula la siguiente **hipótesis**: si se desarrollaran los módulos de Administración y Configuración a partir de los requerimientos establecidos por el Grupo de Coordinación, entonces se contribuirá a la seguridad en la compartimentación de la información en este órgano.

Para dar cumplimiento a lo antes expuesto se desglosaron como **objetivos específicos** los siguientes:

1. Elaborar el marco teórico sobre las herramientas para el apoyo a la toma de decisiones, teniendo en cuenta sus particularidades en la compartimentación de la información.
2. Caracterizar los escenarios de administración y configuración del Portal del Órgano, teniendo en cuenta los requerimientos y principios pre-establecidos por esta institución.
3. Desarrollar los módulos de Administración y Configuración siguiendo la Metodología XP para el desarrollo del Portal Grupo de Coordinación de la Política Penal y Penitenciaria.

4. Evaluar la contribución de los módulos de administración y configuración, en la compartimentación de la información asociadas a indicadores e informes del Grupo de Coordinación.

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

1. Conceptualización de la Plataforma de Interoperabilidad.
2. Descripción de la administración y configuración del Portal Grupo de Coordinación de la Política Penal y Penitenciaria.
3. Caracterización de los sistemas de apoyo a la toma de decisiones.
4. Selección de las herramientas y tecnologías para el desarrollo de sistemas web.
5. Descripción y planificación de las Historias de Usuario.
6. Definición de la arquitectura de seguridad y de software.
7. Implementación de los Módulos de Administración y Configuración.
8. Verificación del cumplimiento de los requerimientos funcionales de los módulos de administración y configuración.
9. Evaluación de la contribución en la compartimentación de la información asociada a indicadores e informes del Grupo de Coordinación, por medio de los módulos de administración y configuración implementados.

El documento está estructurado de la siguiente forma:

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 para el caso del Grupo de Coordinación, en el que la seguridad y compartimentación de la información adquieren una alta connotación debido a la naturaleza de los datos que se gestionan. Además en este capítulo se describen la metodología y tecnologías empleadas para el desarrollo de los módulos.

Capítulo 2: Planificación y diseño: en este capítulo se describen los requerimientos funcionales y no funcionales asociados a los módulos de Administración y Configuración del Portal. Además se describe el comportamiento general del sistema a partir de las Historias de Usuario, como artefacto⁴ 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: este es un importante capítulo dado que en el mismo se evalúa el cumplimiento de los objetivos planteados a partir del diseño y realización de pruebas funcionales al sistema.

⁴ Entiéndase como artefacto algún código, objeto o documento obtenido del desarrollo.

Capítulo I Fundamentación teórica

Introducción al capítulo

En el presente capítulo se incluye la conceptualización de la Plataforma de Interoperabilidad. Se realiza una breve descripción de las características de la administración y configuración del Portal Órgano Justicia MININT. Se establece una descripción general del estado de los Sistemas de Apoyo a la toma de Decisiones conocidos también como Decision Support System (DSS) y su relación con la web. Se establece las principales características de los DSS, su clasificación entre los diferentes criterios y los componentes principales por lo que están compuestos. Paralelamente se plasma una descripción de las herramientas y las tecnologías que se emplearon en el desarrollo de los módulos de administración y configuración. También, se describen las características de la metodología utilizada en el proceso de desarrollo del software que fueron empleadas para garantizar el desarrollo del portal.

1.1 Conceptualización

La Plataforma de Interoperabilidad es una solución informática en la que convergen varios sistemas (aplicaciones de administración, sistemas de información, servidores de aplicación, servidores de bases de datos, y otras tecnologías). Esta tiene como objetivo crear, en una primera fase de desarrollo, un mecanismo de intercambio de datos entre los diferentes sistemas informáticos que participan en los procesos penales en cada uno de los organismos involucrados (MININT, FGR y TSP).

Entre los principales conceptos de esta plataforma 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. (Quin) En el contexto de la Plataforma de Interoperabilidad, se refiere a archivos XSD que definen el formato en que ciertos

datos (indicadores del órgano) serán proveídos por los diferentes sistemas informáticos que participan en los procesos penales y penitenciarios. No obstante, en un sentido más general (y en otras fases), 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).

- **Repositorio de Esquemas y Metadatos (REM):** El REM 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) en la plataforma. 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 en la plataforma que necesiten intercambiar datos puedan acceder, mediante peticiones HTTP, a los formatos en que dichos datos deberán intercambiarse.
- **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, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. Precisamente el modelo de intercambio de datos que implementa la Plataforma de Interoperabilidad es sobre la base de la utilización de servicios web que servirán como interfaces de comunicación de cada uno de los sistemas responsables de almacenar los datos primarios que se requieren para presentarlos en forma de indicadores al órgano. Por consiguiente, 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.

1.2 Plataforma de interoperabilidad para el intercambio de datos

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 los procesos penales y penitenciarios. Estos sistemas carecen de la capacidad de poder colaborar en el intercambio de datos. Por consiguiente, se identifica la necesidad (en una primera etapa) de contribuir en la eficiencia y eficacia en el proceso de obtención de datos para la toma de decisiones en la Comisión Nacional y por tanto se inicia el desarrollo de una Plataforma de Interoperabilidad para el intercambio de datos basada en un modelo de homogenización (estandarización) del formato de intercambio (Valdés González, 2012) a través del empleo de tecnologías XML (Esquemas XSD).

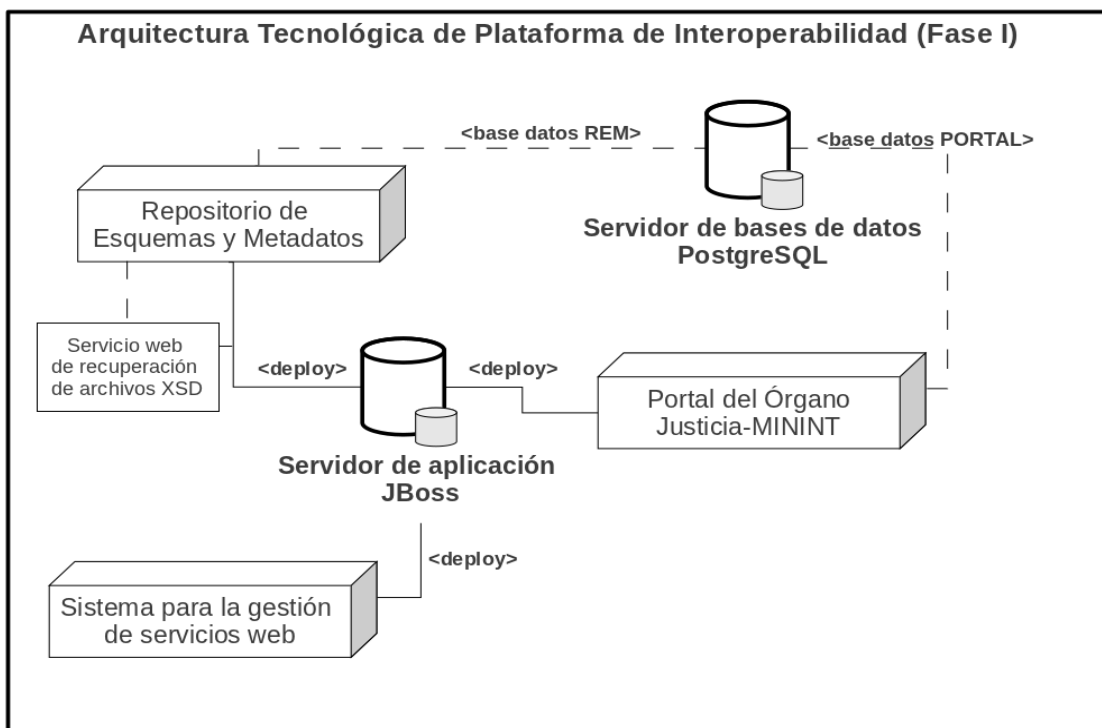


Figura 1: Arquitectura tecnológica de la Plataforma de Interoperabilidad

La Plataforma de Interoperabilidad es una solución informática en la que convergen varios sistemas (aplicaciones de administración, sistemas de información, servidores de

aplicación, servidores de bases de datos, y otras tecnologías) con el objetivo de crear (en una primera fase) mecanismos de intercambio de datos entre los diferentes sistemas informáticos (de información) que participan en los procesos penales, en cada uno de los organismos involucrados (MININT, FGR, TSP).

La plataforma responde a una arquitectura técnica (Figura 1) de seis componentes:

1. Servidor de aplicaciones web (JBoss)
2. Repositorio de Esquemas y Metadatos (REM)
3. Portal del Órgano (PORTAL)
4. Servicios Web para la exposición de los datos
5. Servidor de bases de datos (PostgreSQL)
6. Sistema para la gestión de los servicios web

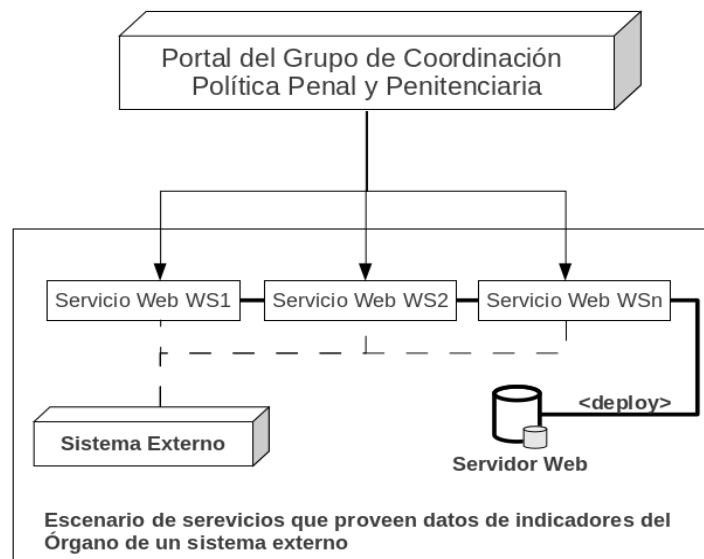


Figura 2: Escenario de intercambio de datos entre el Portal del Órgano y sistemas externos a la plataforma.

La plataforma, en esencia, deviene en intermediaria en los escenarios de intercambio de datos, en los cuales participa un sistema proveedor de datos (generalmente el sistema responsable de almacenar los datos primarios) y otros sistemas consumidores de los datos expuestos. Precisamente el portal del Grupo de Coordinación de la Política Penal y

Penitenciaria, independientemente de ser un componente de la plataforma, es un sistema meramente consumidor de los datos que exponen los sistemas responsables de tributar los indicadores del órgano. En un escenario de intercambio de datos en el cual el portal del órgano funge como consumidor (Figura 2), participan otros sistemas proveedores de datos externos a la plataforma.

El portal del órgano, a partir del objetivo que persigue, constituye una importante herramienta que contribuye directamente al proceso de toma de decisiones de la Comisión Nacional, dado a que está concebido para que permita a los titulares acceder a los indicadores necesarios en la conducción de la política penal y penitenciaria en el país. Con el advenimiento de la web como un medio en el cual se socializan, entre otras cuestiones, datos e información, se contribuye a que los procesos colaborativos adquieran novedosos enfoques. Para el caso particular del Grupo de Coordinación, en el cual el acceso a los datos está determinado por disposiciones de seguridad en la compartimentación de la información, la aceptación del portal como herramienta de apoyo a la toma de decisiones para este órgano del estado, está en dependencia del cumplimiento de estas disposiciones y por tanto de la capacidad que tenga este de administrar y configurar su funcionamiento.

1.3 La web y los sistemas de apoyo a la toma de decisiones

La toma de decisiones ha sido un factor clave en el desarrollo de la sociedad, ya que el hombre siempre ha tenido que escoger entre varias alternativas para llegar a una solución determinada. Es así, como surge una serie de investigaciones dirigidas a perfeccionar el proceso de toma de decisiones, que fueron iniciadas en la década del 1960, donde Scott Morton demostró que los directivos empresariales se beneficiaban del uso de un sistema informático de gestión basado en la decisión (Morton, 1971).

La década del 1970, fue un período de conceptualización y desarrollo tecnológico para los DSS. La atención se centró en el apoyo a las personas responsables de tomar decisiones, que vieron respaldados su accionar por especialistas en computación. Es así, como el uso de los recursos informáticos en la toma de decisiones ha ido creciendo rápidamente, incrementando su uso en el ámbito empresarial e institucional. Debido a esto, se hizo necesario poner a disposición de los directivos estos sistemas, lo cual fue facilitado gracias a la incorporación de los DSS en la web.

La web como herramienta de apoyo a la toma de decisiones, se refiere a un sistema computarizado que ofrece información de apoyo a las decisiones y las herramientas que brindan soporte para la toma de decisiones, siendo analizadas y visualizadas por un navegador web.

En un sentido amplio, se establece que los sistemas de apoyo a la toma de decisión son un conjunto de programas y herramientas que permiten obtener oportunamente la información requerida durante el proceso de la toma de decisiones, en un ambiente de incertidumbre (Power, 2012).

Un DSS no soluciona problemas, ya que solo apoya al proceso de toma de decisiones. La responsabilidad de tomar, adoptar o realizar una decisión es completamente de los usuarios, no del DSS. Estos sistemas pueden emplearse en la obtención de información que revele los elementos claves de los problemas y las relaciones que existen entre ellos.

En la mayoría de los casos, lo que constituye el detonante de una decisión es el tiempo límite o máximo en la que se debe tomar. Así que, en cada decisión que se tome siempre, se podrá pensar que no se tiene toda la información requerida, sin embargo, al llegar al límite de tiempo establecido se deberá llegar a una decisión. Esto implica necesariamente que el verdadero objetivo de un sistema de apoyo a la toma de decisión sea proporcionar la mayor cantidad de información relevante en el menor tiempo posible, con el fin de tomar la decisión más adecuada (Ralph H. Sprague, 1982).

1.3.1 Características de un sistema de apoyo a la toma de decisiones

Los DSS son sistemas desarrollados para ayudar, asistir o soportar la toma de decisiones. Entre las características relevantes de un DSS están las siguientes:

- **Interactividad:** sistema computacional que emite respuestas y brinda la posibilidad de interactuar en forma amigable con el encargado de tomar decisiones.
- **Tipo de decisiones:** apoya el proceso de toma de decisiones en situaciones estructuradas⁵ y no estructuradas.
- **Frecuencia de uso:** tiene un alto índice en la utilización por parte de la administración, en el apoyo y desempeño de sus funciones.

⁵ Estructurada: Información contenida en diferentes archivos como documentos y páginas web.

- **Variedad de usuarios:** puede emplearse por usuarios de diferentes áreas funcionales como: producción, administración, finanzas, ventas y recursos humanos.
- **Desarrollo:** permite que el usuario desarrolle de manera directa modelos de decisión sin la participación operativa de profesionales en informática.
- **Comunicación Inter-Organizacional:** facilita la comunicación de información relevante de los niveles altos a los niveles operativos y viceversa, a través de diferentes componentes visuales como gráficas, estadísticas, notificaciones, etc.
- **Acceso a Base de Datos:** tiene la capacidad de acceder a información de las bases de datos.
- **Simplicidad:** simple, fácil de aprender y cómodo de utilizar por el usuario final.

Un DSS ofrece ayuda en las tres fases del proceso de toma de decisiones, estas son: la de inteligencia; diseño y elección. La inteligencia es la fase de búsqueda de las condiciones que demandan decisiones. Diseño es la fase de inventar, desarrollar y analizar la posible secuencia de las acciones. Elección es la fase de seleccionar una de las alternativas entre las disponibles (Michael J. Scott, 1971).

1.3.2 Componentes de un DSS

Los DSS constan básicamente de tres componentes: componente de dato; componente de interfaz de usuario y componente de administración del modelo. La integración de ellos permite el funcionamiento de los DSS y que los usuarios puedan interactuar con estos.

- **Componente de Dato:** El componente de dato del DSS captura datos para su inclusión en una base de datos (BD), este componente es el encargado de realizar la gestión (añade, elimina, edita y actualiza) y registros de archivos. Accede a los datos de la BD para realizar consultas e informes, proporciona una completa seguridad de los datos mediante la protección contra el acceso no autorizado. Lleva a cabo tareas complejas de manipulación de datos basados en consultas.
- **Componente de Interfaz de Usuario:** El componente de interface de usuario proporciona una interfaz gráfica amigable, permitiendo que los usuarios interactúen con el sistema de forma cómoda y eficiente. Este componente presenta los datos con

una variedad de formatos, ejemplo: gráficos en barras, gráficos en pasteles y gráficos jerárquicos, todos en colores y bidimensionales. También, ofrece a los usuarios que obtengan un alto nivel de conocimiento, esto admite que las rutinas de diagnóstico y sugerencias o cualquier otra, sean fáciles de comprender. El componente de Interfaz de usuario permite la visualización de todas las funcionalidades del sistema.

- **Componente de Administración de Modelo:** El componente de administración de modelo permite la creación rápida y fácil de modelos existentes o preestablecidos (base de modelos). Facilita a los usuarios manipular los modelos para que ellos puedan dirigir sus propios análisis y experimentos. Dentro de los modelos más utilizados por los DSS están: los estadísticos; los de pronósticos de datos y los de simulación.

1.3.3 Clasificación de los DSS

Los DSS se clasifican bajo diferentes criterios estos son: relación entre el usuario y el DSS (Haettenschwiler, 1999); el de modo de asistencia (Power, 2002) y el ámbito (Power).

Respecto a la **relación entre el usuario y el DSS**, se clasifican en:

- **DSS pasivo:** Es un sistema de ayuda para el proceso de toma de decisiones, pero que no puede llevar a cabo decisiones explícitas, sugerencias o soluciones.
- **DSS activo:** Puede llevar a cabo diferentes tipos de decisiones, sugerencias o soluciones.
- **DSS cooperativo:** Permite al encargado de la toma de decisiones (o a sus asesores) modificar, completar o perfeccionar las sugerencias de decisión proporcionadas por el sistema, una vez editada la decisión esta se enviará de vuelta al sistema para su validación. El DSS mejora, completa y precisa las sugerencias del tomador de la decisión y las envía de vuelta al encargado de tomar la decisión para su aprobación.

En caso de que el encargado de la toma de decisión no esté satisfecho puede volver a modificar, completar o perfeccionar las sugerencias de la decisión. Entonces, todo el proceso comienza de nuevo hasta que se genera una solución consolidada.

Sobre el **modo de asistencia**, se clasifican en:

- **DSS dirigidos por modelos:** se hace hincapié en el acceso y manipulación de modelo: estadístico, financiero, de optimización o de simulación. Utiliza datos y parámetros proporcionados por los usuarios, para ayudar a los encargados de adoptar decisiones en el análisis de una situación.
- **DSS dirigidos por comunicación:** disponen de soporte para varias personas que trabajan en una misma tarea compartida.
- **DSS dirigidos por datos u orientados por datos:** enfatizan el acceso y la manipulación de series temporales de datos internos y a veces también de datos externos.
- **DSS dirigidos por documentos:** gestionan, recuperan y manipulan información no estructurada en una variedad de formatos digitales.
- **DSS dirigidos por conocimiento:** proporcionan experiencias acumuladas en forma de hechos, normas, procedimientos, o en estructuras similares especializadas para la resolución de problemas.

De acuerdo al **ámbito** se clasifican en:

- **DSS para la empresa:** El DSS estará enlazado con un almacén de datos de gran tamaño y dará servicio a muchos dirigentes, directores y/o ejecutivos de los organismos, permitiendo su visualización mediante un navegador web o una aplicación de escritorio.
- **DSS de escritorio:** Es un sistema pequeño que puede correr en el ordenador personal.

Según los requerimientos generales, el portal no aportará posibles soluciones a los titulares del órgano, sino que se limitará solamente a la presentación de informes asociados a los indicadores del órgano. Por tanto clasifica como un **DSS pasivo**. Sin embargo los informes que arroje el portal, constituyen el soporte para la presentación de datos que proveerán sistemas externos; en este sentido el portal es considerado un **DSS orientado por datos**. De acuerdo al ámbito e independientemente que su versión inicial no está prevista integrarla a un almacén de datos, es un **DSS para la empresa**.

1.4 Administración y configuración del portal

Dentro de los requerimientos de seguridad que presenta el Portal Grupo de Coordinación, 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. Por ejemplo, la administración funcional se encargará de definir cuáles son los informes que necesita el Grupo de Coordinación. Para realizar esto debe contar con una indicación de la Secretaría Técnica del Grupo de Coordinación, en la cual se refleje la aprobación por parte de los titulares. Al mismo tiempo 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. En el caso del Grupo de Coordinación, estas fuentes son los indicadores y los informes.
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 Marco tecnológico para el desarrollo

El marco tecnológico para el desarrollo está determinado por diferentes criterios de evaluación:

1. La Plataforma de Interoperabilidad, en la cual el Portal del Órgano es un componente, es una solución informática enfocada a un entorno empresarial.
2. La arquitectura base de la Plataforma de Interoperabilidad debe permitir escalar hacia una Arquitectura Orientada a Servicios.
3. La Plataforma de Interoperabilidad debe ser capaz de desplegarse en Centros de Datos con una arquitectura en tres capas: a) Capa de Presentación; b) Capa de Modelo (o Negocio); y c) Capa de Datos. Cada una de ellas desplegadas en un servidor dedicado y separados por corta fuegos (Firewalls) para evitar un efecto dominó ante el comprometimiento de un servidor.

El primer criterio de evaluación se refiere a la capacidad de la solución informática, para desplegarse en un escenario en el cual participan varios sistemas informáticos que automatizan (parcial o totalmente) procesos internos de las organizaciones, y que requieren un alto grado de interoperabilidad (colaboración) y/o integración. El segundo criterio evidencia una intención hacia una solución a largo plazo en la cual precisamente estén creadas las condiciones técnicas para lograr los niveles de interoperabilidad e integración requeridos. Y por último, el tercer criterio responde a una política del Estado Cubano enfocada a establecer una infraestructura tecnológica nacional capaz de alojar sistemas informáticos desde una perspectiva centralizadora.

Sumados a estos tres criterios se definen otros criterios que a consideración del autor deben tenerse en cuenta en la determinación de las tecnologías para el desarrollo de los módulos de Administración y Configuración del portal del órgano. Estos son:

1. Gastos financieros adicionales por conceptos de licenciamiento y soporte de las tecnologías para el desarrollo.
2. Existencia de una comunidad a la cual se pueda tener acceso para la realización de consultas técnicas.
3. Dominio de las tecnologías.
4. Experiencias anteriores en la comunidad universitaria sobre el uso de las tecnologías.

En la Tabla 1 se realiza una comparación entre las principales tecnologías candidatas para asumir el desarrollo de la Plataforma de Interoperabilidad, y por consiguiente cada uno de los componentes que la integran. En esta tabla se tiene en cuenta los criterios antes mencionados y se evidencia que la plataforma base para el desarrollo de los módulos de Administración y Configuración del Portal del Órgano es Java 2 Enterprise Edition (J2EE) incluyendo la utilización del estándar EJB (Enterprise Java Bean) para la implementación del negocio, y la utilización del servidor de aplicaciones JBoss teniendo en cuenta que brinda los principales componentes para una solución SOA, como son: a) Bus de Mensajería Empresarial (ESB); b) Motor de Procesos JBPM; c) integración con motores de reglas de negocio *Drools*, y e) Registro de Servicios Web (UDDI).

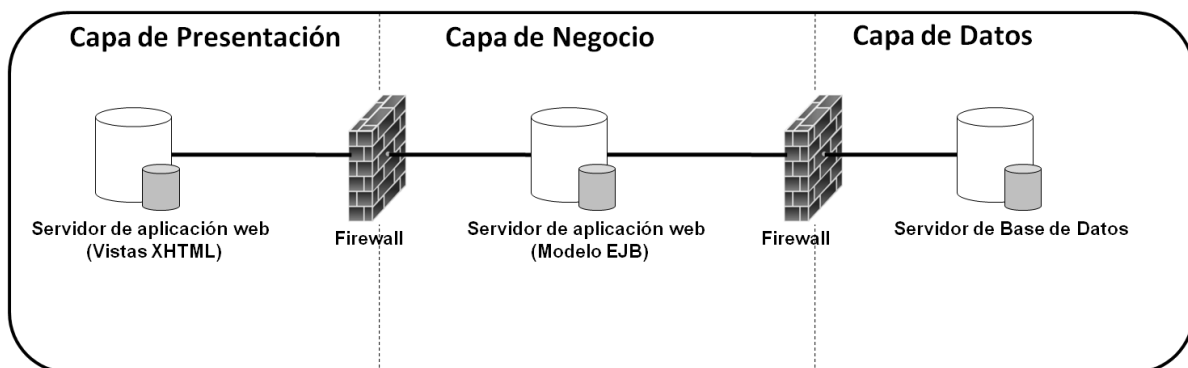


Figura 3: Arquitectura en tres capas de un Centro de Datos

Un criterio determinante en este análisis fue la capacidad de la tecnología a emplear para obtener soluciones que puedan ser desplegadas en Centro de Datos con arquitectura desacoplada en tres capas. Precisamente el empleo de J2EE con EJB hace posible esto,

permitiendo implementar las vistas de los sistemas web desacopladas del negocio, el cual se implementa haciendo uso de los EJB y que pueden ser desplegados en un servidor geográficamente separado del servidor de las vistas. Usando esta tecnología es posible hacer llamadas remotas a los EJB desplegados en el servidor de negocio, logrando el desacoplamiento deseado entre la vista y el modelo (Figura 3).

Tabla 1: Tabla comparativa para el análisis de selección de tecnologías para el desarrollo

	J2EE (EJB, JBoss)	.NET (ASP, IIS)	PHP (Apache)	Phyton
Entorno empresarial	Si	Si	No	No
Escalar hacia SOA	Si	Si	Si	No
Desplegar Centro Datos	Si		No	No
Gastos financieros adicionales	No	Si	No	No
Existencia de una comunidad	Si	Si	Si	Si
Dominio de la tecnología	Medio	Bajo	Ninguno	Ninguno
Experiencias anteriores	Si	Si	Si	Si

1.5.1 Plataforma de desarrollo Java EE

A partir del análisis realizado anteriormente, se determina emplear como plataforma de desarrollo JEE, en la cual se utiliza como lenguaje base 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 (Lopez, 1997).

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 (Lopez, 1997).

La programación en Java permite el desarrollo tanto de aplicaciones bajo el esquema de cliente servidor, como de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras u ordenadores, que permite ejecutar tareas simultáneamente y de esta forma distribuir el trabajo a realizar.

1.5.2 Marco de trabajo (Framework)

Teniendo en cuenta que el Portal 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.
- Spring, para la implementación de los componentes asociados al negocio.
- 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 y Hibernate para la persistencia de datos.

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 base de datos básicas, ganando el acceso a una aplicación de trama, usando Hibernate (Farley, 2007).

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.

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.

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.

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.

1.5.3 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 JBoos, no se recomienda el uso de este IDE en el desarrollo de aplicaciones web donde se emplee el framework Seam (Farley, 2007).

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 (Farley, 2007). 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, 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 (BEATON, 2006). Eclipse permite añadir sistemas de control de versiones como Subversión, puede integrarse con Hibernate y con la herramienta de modelado Visual Paradigm.

1.5.4 Herramientas para el modelado

Visual Paradigm es una herramienta de software libre CASE (Ingeniería de Software Asistida por Computación). La misma propicia un conjunto de ayudas para el desarrollo de

⁶ Framework utilizado para realizar pruebas unitarias.

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 (Pressman, 2002).

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. Se integra con el IDE Eclipse, es multiplataforma, permite la generación de código y realiza ingeniería inversa para diferentes lenguajes de programación como: Java, C++, PHP.

1.5.5 Sistema Gestor de Base de Datos

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 de esta, 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 (PostgreSQL-es, 2010). 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.

1.5.6 Servidor de aplicaciones web Jboss AG

El Servidor de aplicaciones web es el núcleo de la plataforma, dado a que en él se montaran todos los sistemas principales para el funcionamiento de la misma. Sin embargo, el servidor de aplicaciones web JBoss no solo se encargara de desplegar los sistemas web de la plataforma, sino que también permitirá proveer otras tecnologías que en la medida en que vaya evolucionando la plataforma irán siendo necesarias. Dentro de las tecnologías que provee Jboss, se encuentran: Buses de Mensajería (ESB); registros

UDDI⁷ 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.

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 (Marchioni, 2009).

1.5.7 Sistema de control de versiones

Se denomina control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. A lo que se suma, la existencia de sistemas que facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas (Ben Collins-Sussman, 2004).

Un sistema de control de versiones proporciona mecanismos de almacenaje para los ficheros que debe gestionar, ejemplo de esto, son los archivos de texto, imágenes, documentación entre otros. Paralelamente, brinda la posibilidad de realizar cambios sobre los elementos almacenados como modificaciones parciales, añadir, borrar, renombrar o mover elementos.

Subversion Versions System es un sistema de control de versiones libre y de código abierto, que se constituye como uno de los reemplazos más populares de Concurrent Versions System (CVS). Subversion maneja ficheros y directorios a través del tiempo, con un árbol de ficheros en un repositorio central, que admite la recuperación de versiones antiguas de los datos, examinar el historial de cambios de los mismos y permite que cada uno de los usuarios pueda crear una copia local, duplicando el contenido del repositorio para su uso (Ben Collins-Sussman, 2004).

⁷ Universal Description, Discovery, and Integration

1.6 Metodología de desarrollo

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 Extreme Programming (XP).

XP es una de las metodologías ágiles de desarrollo de software más exitosas en la actualidad. 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 cliente, 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 (Wells, 2009).

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.

1.7 Conclusiones del capítulo

A partir del estudio de las características de los DSS, se concluye que el portal del órgano, según su concepción y el objetivo al cual responde, es un **DSS pasivo, orientado por datos y para la empresa**. El estudio de las tecnologías y metodologías permitió definir el marco tecnológico para el desarrollo de los módulos, e identificar la metodología XP como la metodología a emplear en este trabajo.

Capítulo II Planificación y Diseño

Introducción al capítulo

En el capítulo 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 de los módulos.

2.1 Fase de Planificación

En esta fase se establece la prioridad de cada Historia de Usuario, y se realiza una estimación del esfuerzo necesario en cada una de ellas. Las estimaciones son asociadas a la implementación de las historias, partiendo del criterio de que normalmente no sobrepasan de 3 puntos (un punto equivale a una semana de trabajo). Esta fase además incluye la definición de dos artefactos importantes: la Lista de Reserva de Producto, como el artefacto que especifica los requerimientos funcionales del sistema; y los requerimientos no funcionales.

2.1.1 Historias de Usuario (HU)

El primer paso de cualquier proyecto que siga la metodología XP es definir las Historias de Usuario con el cliente. Estas tienen la misma finalidad que los casos de uso pero con algunas diferencias. Constan de tres o cuatro líneas escritas por el cliente en un lenguaje no técnico o natural, sin hacer mucho hincapié en los detalles. No se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos.

Las Historias de Usuario son usadas para estimar los tiempos de desarrollo de la parte de la aplicación que describen, así como en la fase de pruebas con la finalidad de verificar si el programa cumple con lo que especifica cada una de esta.

Para el desarrollo de los módulos de Administración y Configuración del Portal, se agruparon las Historias de Usuario como se muestra en la Tabla 2:

Tabla 2: Historias de Usuario agrupadas

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

Las Historias de Usuario son especificadas en un documento **Modelo de Historias de Usuario del Negocio** del expediente de desarrollo del proyecto. En este se describen cada una de las Historias de Usuario del sistema. Para el caso de las Historias de Usuario de la Tabla 3, agrupan las operaciones de: registrar, modificar, eliminar, listar, ampliar detalles y buscar, asociadas a las entidades correspondientes.

Tabla 3: Operaciones asociadas por Historias de Usuarios

Historias de Usuario	Entidad Neg.	Entidad BD	Operaciones
Administrar Usuario	Usuario	<i>usuario</i>	Registrar, modificar, eliminar, listar y buscar
Administrar Rol	Rol	<i>rol</i>	
Administrar Órgano	Órgano	<i>organo</i>	
Administrar Indicador	Indicador	<i>indicador</i>	
Administrar Informe	Informe	<i>informe</i>	
Administrar Servicios Web	Servicio Web	<i>servicio</i>	
Listar y editar operaciones del	Operación	<i>operacion</i>	Modificar (activar), listar, buscar y mostrar

sistema			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

2.1.2 Lista de reserva del producto

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

Módulo de Administración:

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 descendientemente 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.

Módulo de Configuración:

1. **Registrar configuración de parámetros de conexión a servidor de correo electrónico:** permite configurar la conexión a un servidor de correo electrónico que empleará el sistema para enviar las notificaciones, alertas, avisos y noticias.
2. **Modificar configuración de conexión de correo electrónico:** permite editar los parámetros de conexión a un servidor de correo electrónico definidos.
3. **Verificar conexión:** permite comprobar la conexión a partir de los parámetros definidos.
4. **Cargar configuración de conexión:** permite cargar un archivo de configuración de conexión guardado localmente en el disco duro.
5. **Exportar configuración:** exporta la configuración en un archivo XML.
6. **Eliminar configuración de conexión:** elimina una configuración de conexión a un servidor de correo electrónico registrado en el sistema.

7. **Mostrar detalles de configuración:** permite mostrar los detalles de configuración de conexión a un servidor de correo electrónico.
8. **Registrar configuración global de mecanismo de consulta de indicadores:** permite definir el modo de consumo de los indicadores registrados en el sistema.
9. **Modificar tiempo de caducidad de las contraseñas:** modifica el tiempo de caducidad de las contraseñas.
10. **Registrar configuración global de mecanismo de consulta de indicadores:** permite definir, para cada indicador, el modo de consulta del indicador.
11. **Modificar configuración global de mecanismo de consulta de indicadores:** Permite modificar la configuración global del mecanismo de consumo de indicadores.
12. **Modificar configuración particular de mecanismo de consulta de indicadores:** permite modificar la configuración particular del mecanismo de consumo de un indicador determinado.
13. **Definir indicadores a presentar en página principal:** seleccionar aquellos indicadores que el usuario necesita se muestren en la página principal.
14. **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, haciendo posible definir nuevos indicadores o eliminar los que ya han sido seleccionados.
15. **Definir tiempo de caducidad de las contraseñas en el sistema:** permite establecer el tiempo en que las contraseñas caducarán.

2.1.3 Requisitos no funcionales del Sistema

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener; estas lo hacen atractivo, usable, rápido y confiable.

Para lograr la satisfacción del cliente y una buena calidad en el sistema se listan los siguientes requerimientos no funcionales.

- **Requerimiento de disponibilidad**

1. El sistema deben estar disponibles las 24 horas del día, todos los días de la semana durante todo el año.

- **Requisitos de usabilidad**

1. Los sistemas de la plataforma deben presentar mensajes de error que permitan al usuario identificar el tipo de problema.
2. La verificación de datos requeridos se debe implementar de manera tal que el usuario reciba una notificación cuando ha dejado de especificar algún dato requerido.
3. La validación de datos debe permitir mantener informado al usuario cuando se ha introducido un valor en un formato no esperado o con un tipo de dato no valido. Dicha validación debe hacerse en el servidor y en aquella parte de la web destinada al cliente.
4. La navegabilidad de los sistemas debe permitirle al usuario acceder, según sus permisos, a todas las interfaces de usuario desde cualquier lugar en el que se encuentre.

- **Requisitos de seguridad**

5. El sistema debe recuperarse satisfactoriamente ante la ocurrencia de fallas y/o errores inesperados.
6. 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.
7. 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.
8. 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.

9. El sistema debe permitir que la contraseña se almacene de manera encriptada en la base de datos.
10. 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.
11. 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.

- **Requisito de portabilidad**

12. El sistema debe poder instalarse en los sistemas operativos Windows, Linux/Unix.
13. El sistema debe dar la posibilidad de emplear cualquier sistema gestor de bases de datos.

- **Requisito de mantenibilidad**

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

- **Requisito de flexibilidad**

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

2.1.4 Planificación de entregas (Release planning)

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 a los módulos de Administración y Configuración.

El Plan de Iteraciones está dividido en tres períodos, como se muestra en la Tabla 9:

Tabla 4: Plan de duración de iteraciones

Módulos	Administración	Configuración
Iteración 1 Abril	<ol style="list-style-type: none"> 1 Administrar Usuario 2 Administrar permisos adicionales de usuario 3 Administrar contraseña 4 Administrar Rol 5 Administrar Órgano 	
Iteración 2 Mayo	<ol style="list-style-type: none"> 1. Administrar Indicador 2. Administrar Informe 3. Administrar asociaciones parte contraparte 4. Administrar permisos a Informes 5. Administrar permisos a Indicadores 	
Iteración 3 Junio	<ol style="list-style-type: none"> 1. Administrar Servicios Web 2. Registrar y listar Trazas 3. Listar y editar operaciones del sistema 	<ol style="list-style-type: none"> 1. Administrar Configuración del sistema

2.2 Fase de Diseño

La estrategia durante esta fase, fue la elaboración de un diseño lo más sencillo y simple posible, fáciles de implementar y entendibles, que a la larga costará menos tiempo y

esfuerzo desarrollar. Un diseño sencillo es aquel que posea la menor cantidad de clases y métodos. No debe existir código duplicado, y debe cumplir con todos los requisitos del software.

Una de las flexibilidades de la metodología XP, es que asimila 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.

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

El uso de las tarjetas Clase, Responsabilidades y Colaboración (C.R.C) permite al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de programación. 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, las clases⁸ que colaboran con cada responsabilidad. A continuación se exponen una representación de las tarjetas obtenidas durante el proceso de desarrollo de los módulos de Administración y configuración:

Tabla 5: Tarjetas C.R.C UsuarioIndicadorEditManager

Clase: UsuarioIndicadorEditManager	
Responsabilidades	Clases relacionadas
La responsabilidad de asignarle permiso a un usuario a que tenga acceso a determinado indicador recae sobre esta clase.	UsuarioList IndicadorList

Tabla 6: Tarjetas C.R.C UsarioDetailsManager

Clase: UsarioDetailsManager	
Responsabilidades	Clases relacionadas
Esta clase tiene la responsabilidad de mostrar los datos de un usuario, las trazas y las operaciones	TrazaList RolOperacion

⁸ El nombre de estas clases no se acentúan y pueden ser compuestos

asociadas a este.	Operación UsuarioOperacionList Traza OperacionList Usuario
-------------------	--

Tabla 7: Tarjetas C.R.C UsuarioDeleteManager

Clase: UsuarioDeleteManager	
Responsabilidades	Clases relacionadas
Esta clase tiene la responsabilidad de eliminar un usuario y la configuración de la página principal que está asociada a este.	Usuario ConfPagPrincipal

2.3 Arquitectura de 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, de la siguiente manera:

- Capa de presentación (Vista):** En la capa Vista 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. La vista además incluye los 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 postgresSQL.

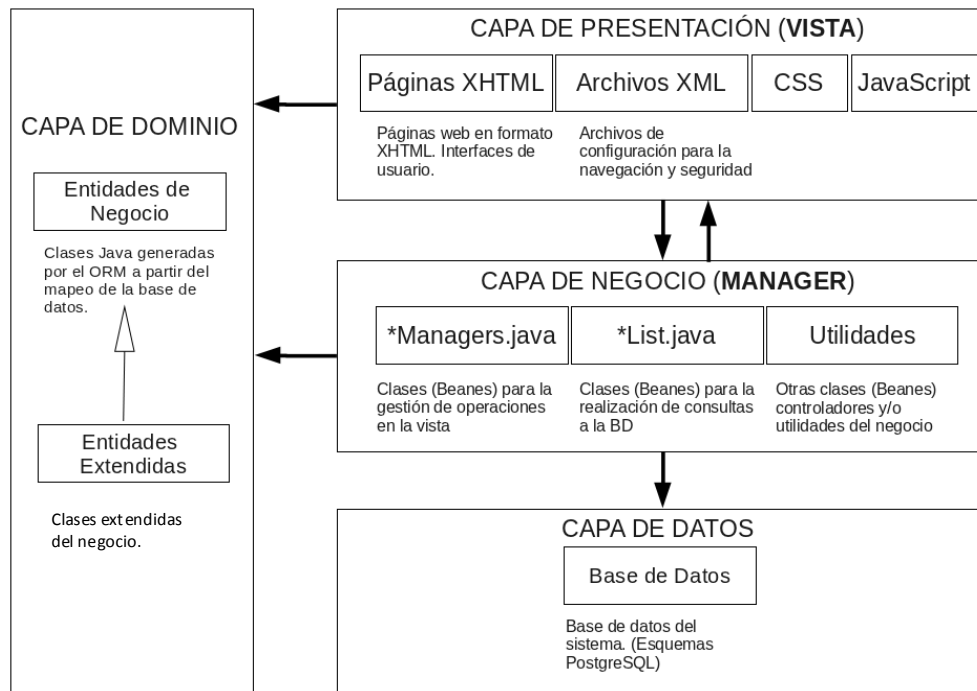


Figura 4: Implementación de la arquitectura en capas.

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: a) 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; b) un proyecto EAR, en el cual se realiza el empaquetamiento del sistema luego de su compilación, previo a su despliegue en el servidor de aplicaciones JBoss; c) y el proyecto EJB, donde se define toda la capa de negocio (Figura 5).

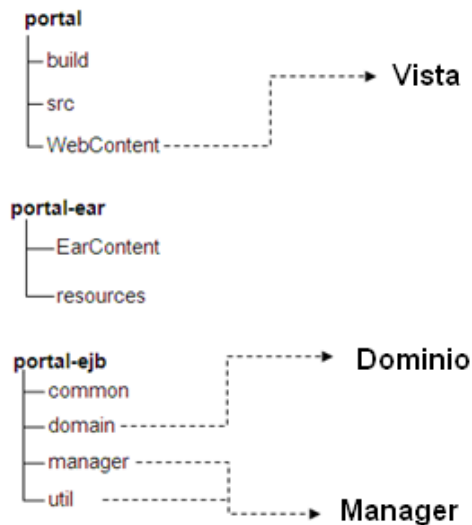


Figura 5: Distribución física de componentes de la arquitectura

2.3.1 Arquitectura de seguridad

La seguridad, en sentido general, cuenta con dos mecanismos fundamentales: a) el mecanismo de autenticación, a partir del cual se identifica al usuario; b) el mecanismo de autorización, mediante el cual se determina si un usuario tiene los permisos suficientes para acceder o no ha determinado recurso⁹. Para implementar estos mecanismos se hizo uso de las clases *Identity* y *Credentials* que provee el framework Seam, 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.

⁹ Entiéndase como recurso una funcionalidad (método) del sistema, una vista (página XHTML), un objeto digital (fichero, indicador, informe, entre otros).

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.

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, 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 a 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>
    </redirect>
  </rule>
</navigation>
</page>

```

2.2.2 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.

En este epígrafe se presenta el modelo lógico correspondiente a cada una de las Historias de Usuario agrupadas en la **Tabla 2**. 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.

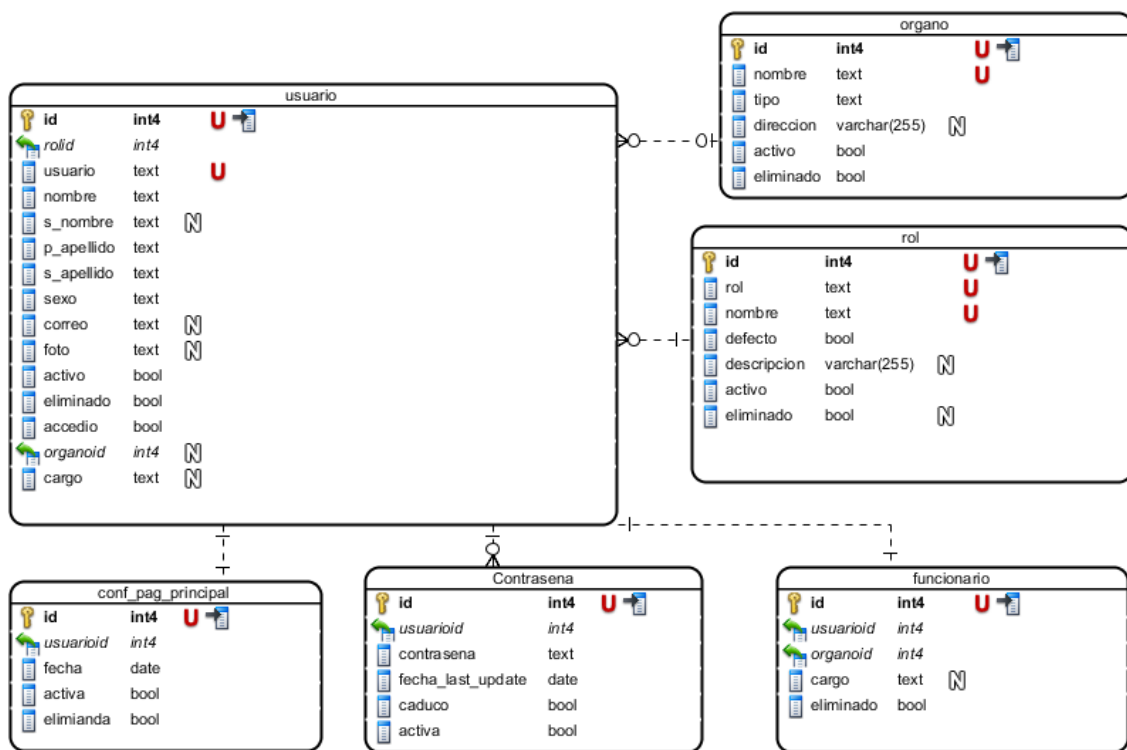


Figura 6: Entidades relacionadas con la HU Administrar Usuario

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.

2. 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 7) *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.

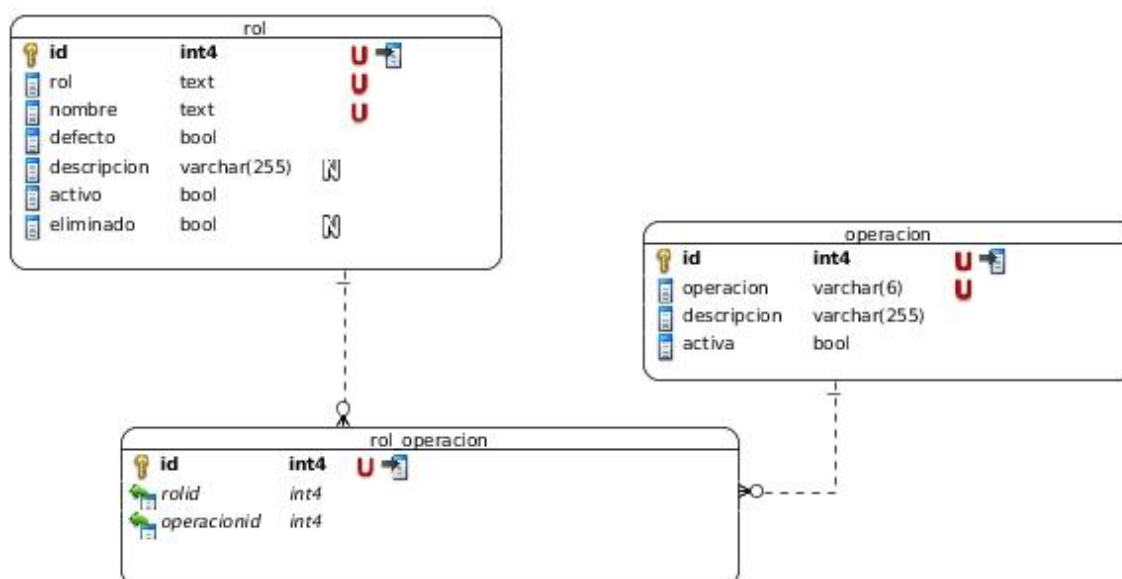


Figura 7: Entidades relacionadas a la HU Administrar Rol

3. 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 8).

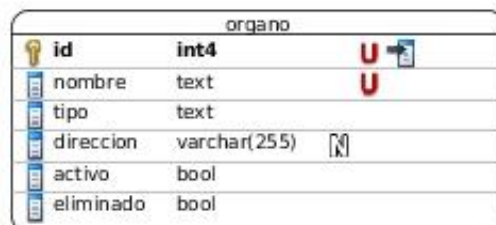


Figura 8: Entidades relacionadas a la HU Administrar Órgano

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 9: Entidades relacionadas a la HU Registrar y listar trazas

5. Modelo de datos de Historia de Usuario *Administrar Indicador*:

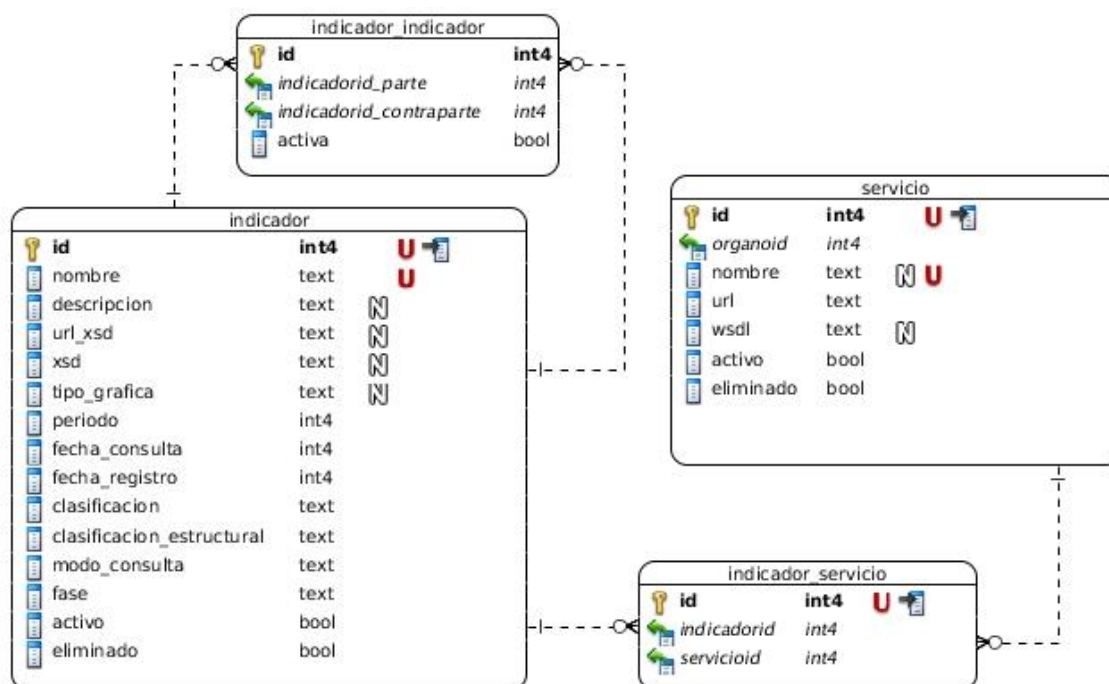


Figura 10: Entidades relacionadas con la HU Administrar Indicador

Administrar Indicador define los escenarios: a) registrar indicador; b) editar un indicador; c) eliminar indicador; d) buscar indicador; e) mostrar detalles de un indicador y f) listar indicadores.

Para los escenarios a), b) y c) solo se emplea la entidad **indicador**; el resto de los escenarios emplean todas las entidades del modelo (Figura 10).

6. Modelo de datos de Historia de Usuario *Administrar Informe*:

La Historia de Usuario *Administrar Informe* responde al diseño por los usuarios del sistema de los informes que agruparán reportes de indicadores registrados en el sistema. Los escenarios que esta HU define son los mismos que la anterior HU (*Administrar Indicador*). En este caso, los escenarios de registro, edición y eliminación involucran las entidades: **informe**, en la cual se almacenan los datos generales de un informe; **funcionario**, de la que se extrae la llave del funcionario que emite este informe; **informe_indicador**, resultado de una relación de *muchos a muchos* y en donde se especifican todos los indicadores que agrupa dicho informe. El resto de los escenarios de

la HU (listar, mostrar detalles y buscar), emplean todas las entidades del modelo (Figura 11).

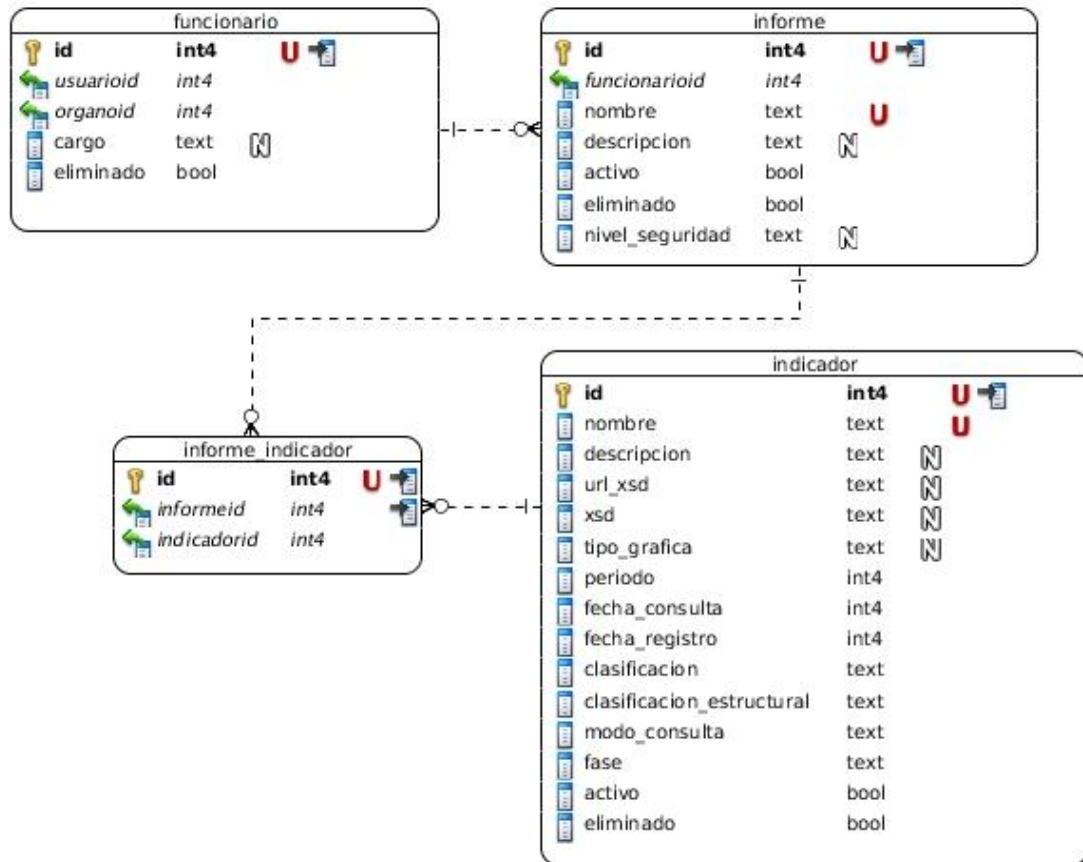


Figura 11: Entidades relacionadas a la HU Administrar Informe

7. 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 12) **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.

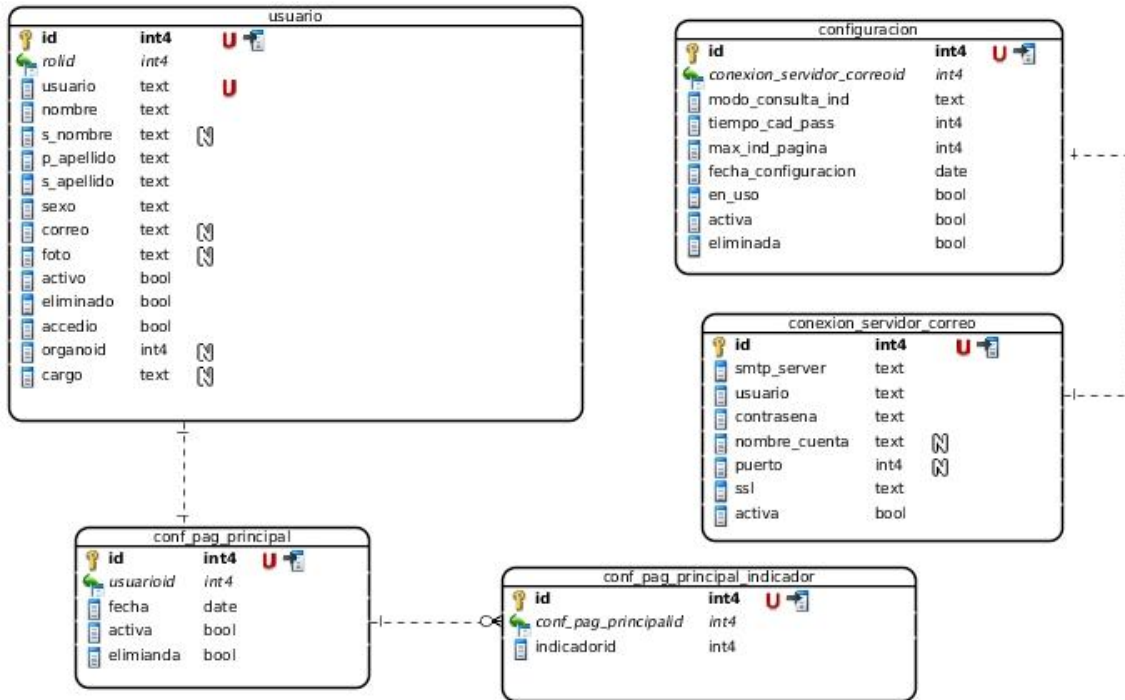


Figura 12: Entidades relacionadas con la HU Administrar Configuración del sistema

Para el caso de la configuración de atributos globales del sistema, se definen los escenarios: a) editar configuración general; b) registrar configuración de conexión a servidor de correo electrónico; c) editar configuración de conexión a servidor de correo electrónico; y d) listar configuraciones de conexión a servidor de correo electrónico. Para estos escenarios se emplean las entidades **configuración** y **conexión_servidor_correo**.

8. Modelo de datos de Historia de Usuario *Administrar permisos adicionales de usuario:*

Administrar permisos adicionales de usuario define el escenario editar permisos adicionales. Este escenario emplea la entidad **usuario_operacion** para asociar a un usuario las operaciones que, adicionalmente a las que tiene asignadas por su rol, podrá realizar en el sistema.

En este modelo (Figura 13) se utiliza el nomenclador de operaciones (entidad **operacion**) para extraer aquellas operaciones que el usuario no tiene asignadas en su rol, y a partir de las cuales determinará cuáles realizará adicionalmente.

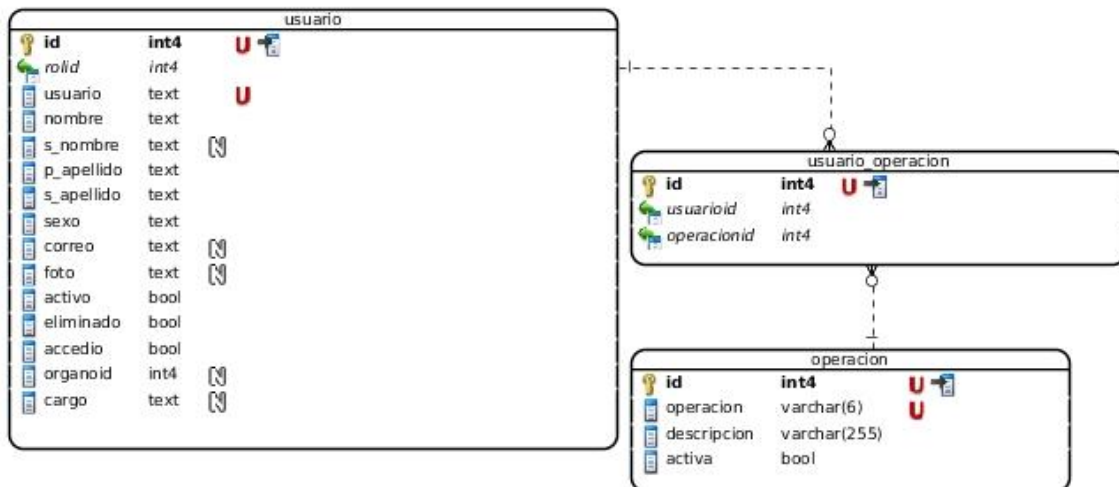


Figura 13: Entidades relacionadas con la HU Administrar permisos adicionales de usuario

9. Modelo de datos de Historia de Usuario *Administrar permisos a Indicadores*:

Esta Historia de Usuario define el escenario editar permisos de un usuario a realizar reportes de un determinado indicador. Un usuario, por defecto, no tiene permiso de realizar reportes de ningún indicador registrado en el sistema. Esta configuración inicial puede editarse a partir del registro o eliminación en la entidad *nusuario_indicador_permiso* del modelo (Figura 14) aquellos indicadores a los que el usuario tendrá o no permisos de realizar reportes.

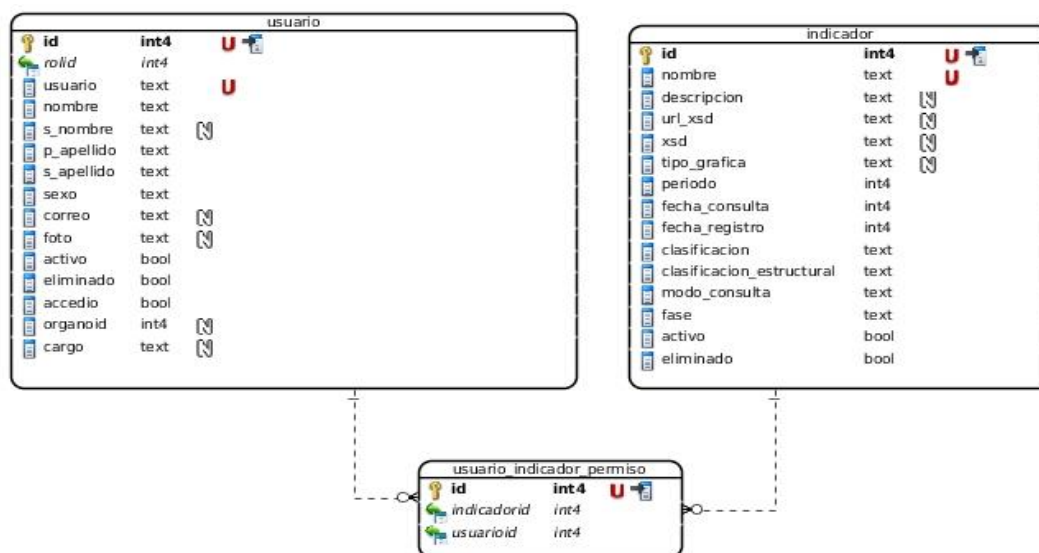


Figura 14: Entidades relacionadas a la HU Administrar permisos a indicador

10. Modelo de datos de Historia de Usuario *Administrar permisos a informes*:

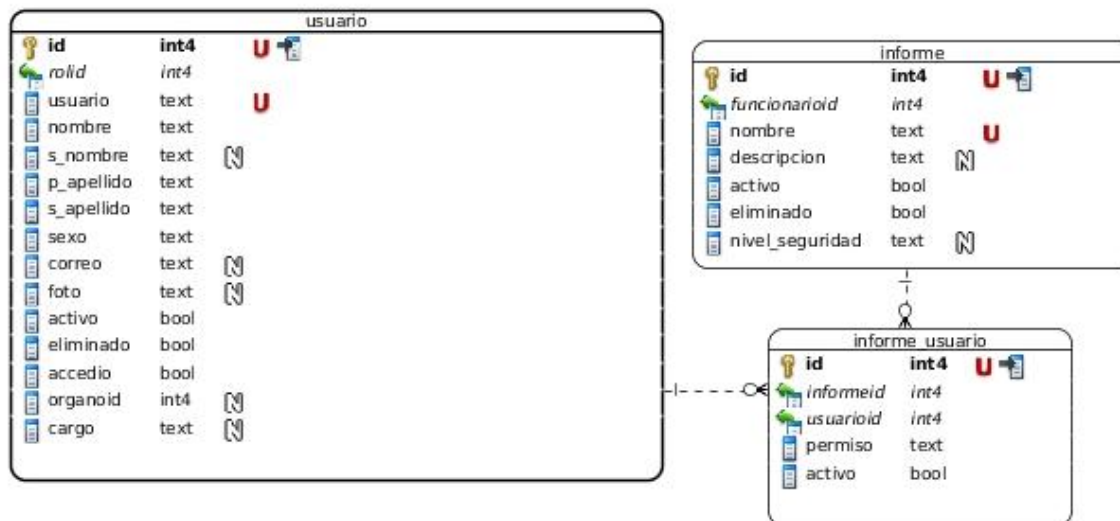


Figura 15: Entidades relacionadas a la HU Administrar permisos a informes

Esta Historia de Usuario es muy similar a la anterior. La diferencia radica en que se define una tipología de permisos para un informe: permiso de modificación; permiso de lectura; permiso de modificación y lectura; o ninguno. En la entidad **informe_usuario** se especifica el permiso que tendrá un usuario de cada uno de los informes a los que se le podrá asignar algún permiso de acceso. Aquellos informes que contengan indicadores a los que el usuario no podrá realizar reportes, serán informes cuyo permiso permanente es ninguno.

11. 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 todas las entidades del modelo (Figura 16), 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.

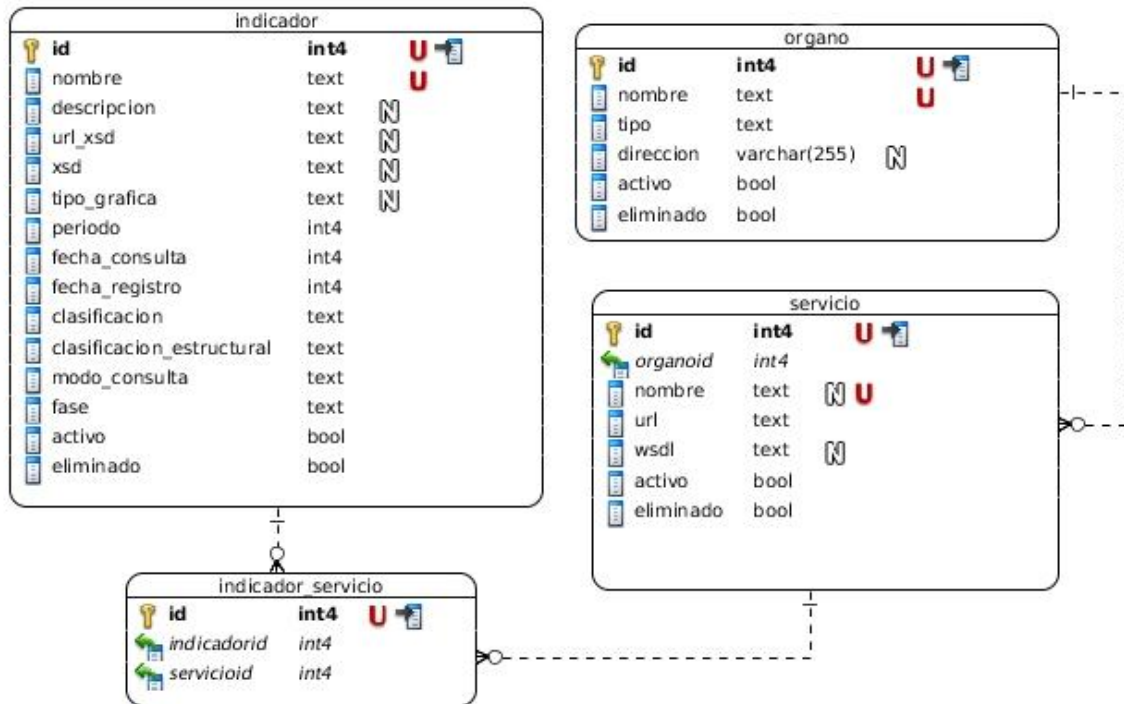


Figura 16: Entidades relacionadas a la HU Administrar Servicios Web

12. Modelo de datos de Historia de Usuario *Listar y editar operaciones del sistema*

Esta Historia de Usuario solo define los escenarios: a) editar activación de operación del sistema; b) mostrar detalles de una operación; y c) listar operaciones del sistema.



Figura 17: Entidades relacionadas con la HU Administrar Operaciones

El primer escenario involucra la entidad **operacion** del modelo (Figura 17), en la cual modifica el estado del atributo **activa** entre los valores **true** y **false**. El resto de los escenarios utilizan todas las entidades del modelo.

13. Modelo de datos de Historia de Usuario *Administrar asociación parte contraparte*:

Esta Historia de Usuario define el escenario editar asociación parte-contraparte. Esta asociación es resultado de una relación recursiva de la entidad **indicador** con ella misma. El resultado de esta relación recursiva, la entidad **indicador_indicador**, registra las asociaciones parte-contraparte entre dos o más indicadores; donde un indicador parte puede tener varios indicadores contraparte, a partir de los cuales se valide la consistencia del indicador parte.

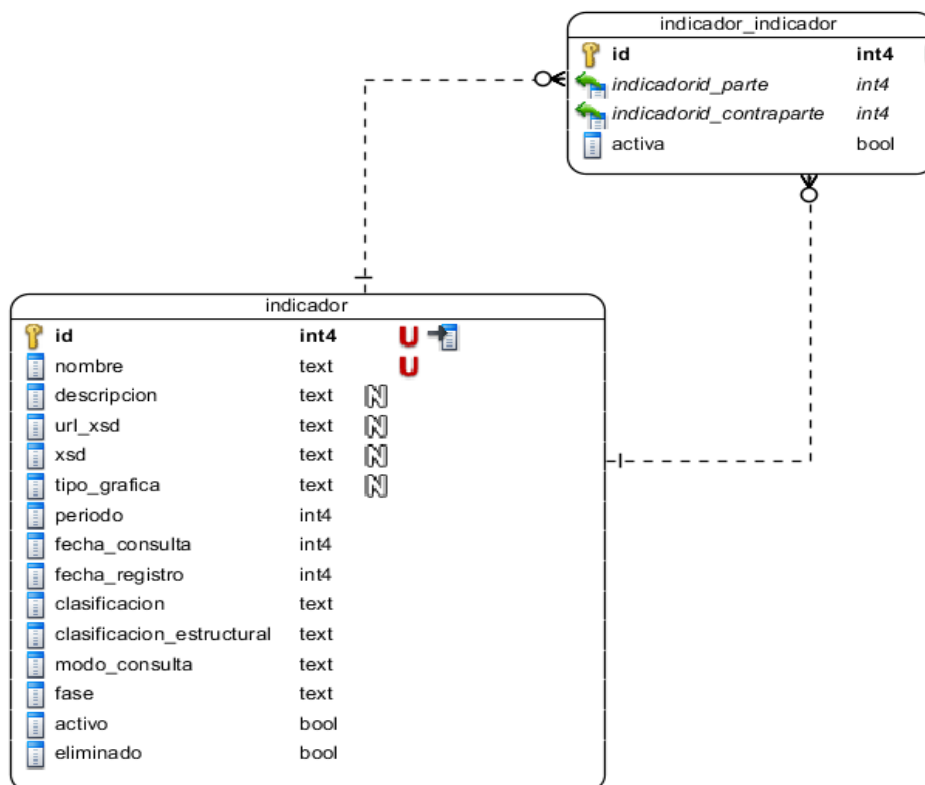


Figura 18: Entidades relacionadas con la HU Administrar asociación parte-contraparte

14. Modelo de datos de Historia de Usuario *Administrar contraseña*:

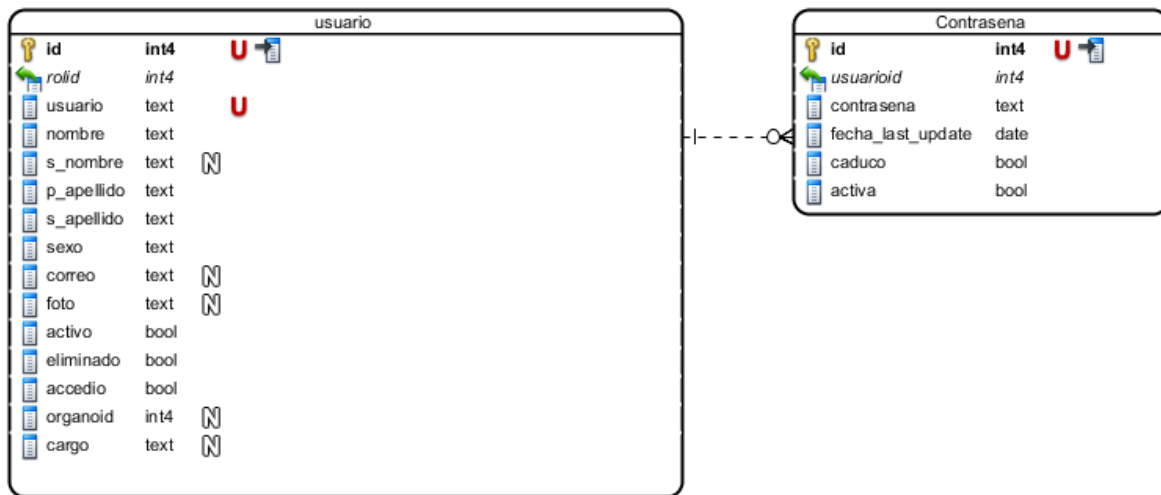


Figura 19: Entidades relacionadas con la HU Administrar Contraseña

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. Ambos escenarios emplean las entidades del modelo (Figura 19).

2.2.3 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. Algunos de los patrones empleados en el desarrollo de los módulos de Administración y Configuración se encuentran los de comportamiento, perteneciente a la familia GRASP (General Responsibility Assignment Software Patterns) (Larman, 2004) GRASP son patrones generales de software, utilizados en la asignación de responsabilidades. Aunque se considera que más que patrones propiamente dichos, son una serie de buenas prácticas recomendables en el diseño de software. A continuación se ejemplifica algunos de estos patrones utilizados. Los patrones GRASP empleados en este trabajo son:

- **Patrón *Controller* (Controlador):**

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

```
<div class="actionButtons">
  <h:commandButton id="save"
    value="Guardar"
    action="#{usuarioEditManager.save}"/>
</div>
```

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. En 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*. El siguiente segmento de código representa cómo se asocia el método ***save*** del controlador ***UsuarioEditManager.java*** al atributo ***action*** del componente botón correspondiente a la interfaz ***UsuarioEdit.xhtml***.

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.

- **Patrón Experto (*Expert*):**

Este patrón es el principio básico de asignación de responsabilidades. Fue empleado en las clases controladoras para la creación de objetos y la implementación de métodos. Por ejemplo, en cada controlador en el que se implementa el método ***save*** se requiere de la información necesaria para construir el objeto correspondiente antes de persistirlo en la base de datos. Empleando el mismo caso anterior, en el

controlador ***UsuarioEditManager.java*** se crea un objeto del tipo ***Usuario*** a partir de los valores de los atributos del objeto recibidos de los campos de la interfaz ***UsuarioEdit.xhtml***.

- **Patrón Creador (*Creator*):**

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***.

- **Patrón *Alta Cohesión*:** Este patrón se emplea en la definición de las entidades de negocio del sistema, las cuales almacenan coherentemente toda la información que requieren para la modelación de los objetos.
- **Patrón *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 de que provee el framework Seam, no obstante por su utilizad se señala su empleo en esta sección.

Otros patrones empleados además de los patrones de comportamiento GRASP, fueron los patrones de la familia *Gang of Four* (GOF) (Metsker, 2002). Entre los patrones GOF empleados se encuentran los siguientes patrones:

- **Patrón Solitario (*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; tal y como se muestra en el siguiente segmento de código de la clase **FechaHora.java**:

```
private FechaHora() {
    String[] ids = TimeZone.getAvailableIDs(-5 * 60 * 60 * 1000);

    pdt = new SimpleTimeZone(-5 * 60 * 60 * 1000, ids[0]);

    pdt.setStartRule(Calendar.APRIL, 1, Calendar.SUNDAY, 2 * 60 * 60 * 1000);
    pdt.setEndRule(Calendar.OCTOBER, -1, Calendar.SUNDAY, 2 * 60 * 60 * 1000);

    calendar = new GregorianCalendar(pdt);
}

public static FechaHora getInstance() {
    if (obj == null) {
        obj = new FechaHora();
    }

    return obj;
}
```

2.2.4 Diagrama de despliegue

Este diagrama se utiliza para modelar el hardware utilizado en la implementación del sistema y las relaciones entre sus componentes. El servidor de aplicaciones web JBoss es el que despliega todos los sistemas de la plataforma, los que interactúan con las 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.

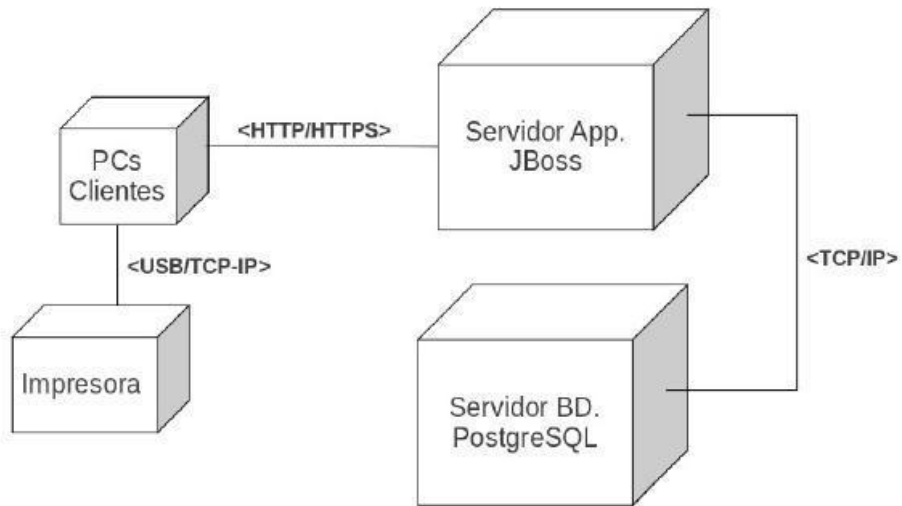


Figura 20: Diagrama de Despliegue

2.2.5 Conclusiones del Capítulo

El Plan de Iteraciones, dado la prioridad de las Historias de Usuario, determinó el periodo de implementación de esta, las cuales fueron guiadas y orientadas por las tarjetas C.R.C. A fin de obtener un sistema¹⁰ 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 en el sistema y la autorización de acceso a los recursos del sistema. La relación entre las entidades, la descripción de los escenarios e ilustración de los patrones utilizados facilitan la comprensión y análisis del diseño.

¹⁰ Portal del Grupo de Coordinación de la Política Penal y Penitenciaria

Capítulo III Implementación y Prueba

Introducción al capítulo

En el presente capítulo se detallan las iteraciones llevadas a cabo durante la construcción de los módulos 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 de los módulos.

3.1 Fase de Implementación

En esta fase se realiza la implementación de las Historias de Usuario que fueron agrupadas en cada iteración. 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.

3.1.1 Tareas de programación 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: En la Tabla 15 se establece los tiempos estimados y reales para cada Historia de Usuario que esté relacionada a la iteración.

Tabla 8: Tiempo estimado y real de implementación por cada Historia de Usuario

Módulos	Historias de Usuario	Tiempo de Implementación (Semanas)	
		Estimación	Real
Administración	Administrar Usuario	3	3.5
	Administrar Órgano	3	2.9
	Administrar Rol	3	3.2
	Administrar contraseña	1	1

	Administrar permisos adicionales de usuario	1	1
--	---	---	---

Tabla 9: Tareas de programación por Historia de Usuario

Historias de Usuario	Tareas de programación por Historias de Usuario
Administrar Usuario	<ul style="list-style-type: none"> ➤ Autenticar usuario ➤ Crear usuario ➤ Eliminar usuario ➤ Mostrar detalles de usuario ➤ Modificar datos de usuario ➤ Listar usuario registrado en el sistema
Administrar Órgano	<ul style="list-style-type: none"> ➤ Registrar nuevo órgano ➤ Eliminar órgano ➤ Modificar datos de órgano ➤ Listar órgano ➤ Buscar órgano ➤ Mostrar detalles de órgano
Administrar Rol	<ul style="list-style-type: none"> ➤ Crear rol ➤ Eliminar rol ➤ Listar rol ➤ Mostrar detalles de rol ➤ Editar rol
Administrar contraseña	<ul style="list-style-type: none"> ➤ Cambiar contraseña ➤ Cambiar contraseña sin verificación
Administrar permisos adicionales de usuario	<ul style="list-style-type: none"> ➤ Editar permisos adicionales a un usuario

Iteración 2:

Tabla 10: Tiempo estimado y real de implementación por Historia de Usuario

Módulos	Historias de Usuario	Tiempo de Implementación (Semanas)	
		Estimación	Real
Administración	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

Tabla 11: Tareas de programación por Historia de Usuario

Historias de Usuario	Tareas de programación por Historias de Usuario
Administrar Informe	<ul style="list-style-type: none"> ➤ Crear informe ➤ Eliminar informe ➤ Listar informe ➤ Mostrar detalles de un informe ➤ Editar permisos extras de un informe
Administrar Indicador	<ul style="list-style-type: none"> ➤ Crear indicador ➤ Eliminar indicador ➤ Listar indicador ➤ Mostrar detalles de un indicador ➤ Editar indicador
Administrar permisos a Informes	<ul style="list-style-type: none"> ➤ Editar permisos de acceso a un informe
Administrar permisos a indicadores	<ul style="list-style-type: none"> ➤ Editar permisos de generar reporte de un indicador
Administrar asociaciones	<ul style="list-style-type: none"> ➤ Definir asociación parte-contraparte entre indicadores

parte contraparte	<ul style="list-style-type: none"> ➤ Eliminar asociación parte-contraparte entre indicadores ➤ Listar asociación parte-contraparte entre indicadores ➤ Mostrar detalles asociación parte-contraparte entre indicadores
-------------------	---

Iteración 3:

Tabla 12: Tiempo estimado y real de implementación por Historia de Usuario

Módulos	Historias de Usuario	Tiempo de Implementación (Semanas)	
		Estimación	Real
Configuración	Administrar Configuración del sistema	3	3.2
	Administrar		
Administración	Registrar y listar Trazas	3	3.1
	Administrar Servicios Web	3	3.2
	Listar y editar operaciones del sistema	3	2.5

Tabla 13: Tareas de programación por Historia da Usuario

Historias de Usuario	Tareas de programación por Historias de Usuario
Administrar Configuración del Sistema	<ul style="list-style-type: none"> ➤ 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 ➤ Configurar modo de consulta de los indicadores
Registrar y listar Trazas	<ul style="list-style-type: none"> ➤ Visualizar las trazas ➤ Registrar trazas del sistema
Administrar Servicios Web	<ul style="list-style-type: none"> ➤ Crear servicios web ➤ Eliminar servicios web ➤ Listar servicios web

	<ul style="list-style-type: none"> ➤ Mostrar detalles de un servicios web ➤ Editar servicios web
Listar y editar operaciones del sistema	<ul style="list-style-type: none"> ➤ Visualizar las operaciones ➤ Registrar las operaciones del sistema

3.2 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.

3.2.1 Pruebas de caja blanca

Durante la implementación de los módulos 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 esta técnica son:

1. **Paso 1:** Construcción del grafo de flujo asociado al segmento de código que se desea probar.
2. **Paso 2:** Cálculo de la complejidad ciclomática del grafo (A Complexity Measure, 1976) (Lopez, et al., 2010).
3. **Paso 3:** Determinación de un conjunto básico de caminos independientes.
4. **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 se 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
    }
}
```

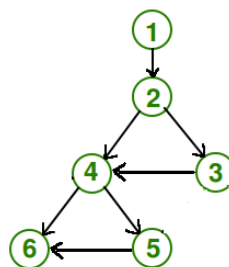


Figura 21: Grafo de flujo asociado a la funcionalidad Mostrar detalles de usuario

Dado el segmento de código, se construye el grafo de flujo (Figura 6) y posteriormente se realiza el cálculo de la complejidad ciclomática a partir de tres variantes (A Complexity

Measure, 1976) 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 total de aristas
- **N:** Cantidad total de nodos
- **p:** Cantidad de componentes conectados en una arista

De manera que:

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

$$V(G) = 3$$

- **Variante 2:** Calculo 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:** Calculo 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.

3.2.2 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 tres iteraciones de pruebas, en las cuales se evaluó el funcionamiento de la aplicación (Figura 34).

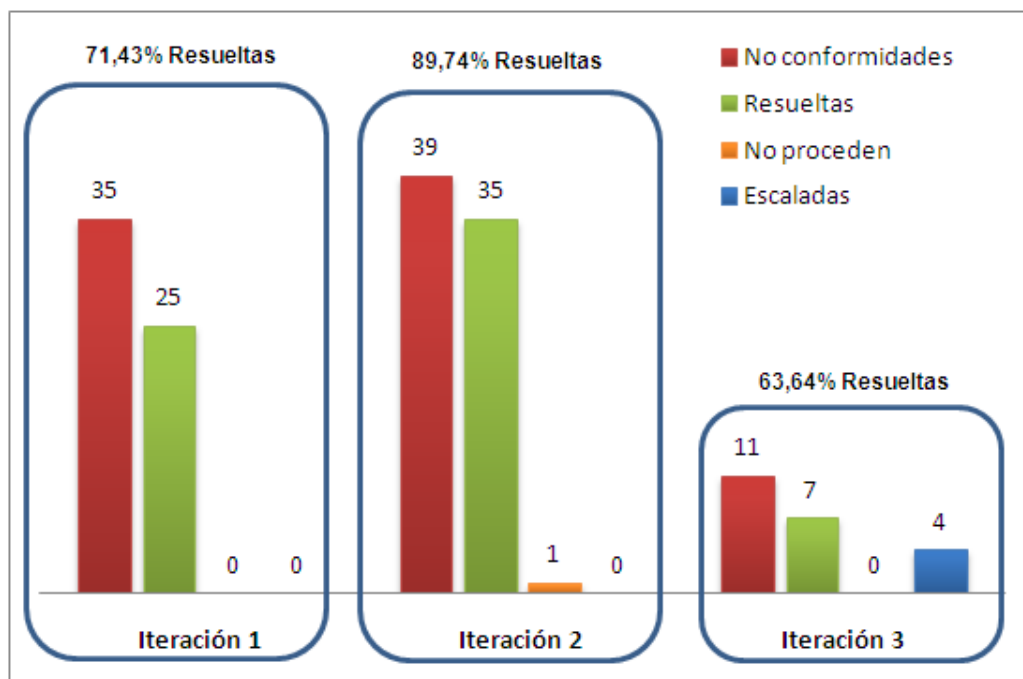


Figura 22: Resultados de las pruebas funcionales en tres iteraciones

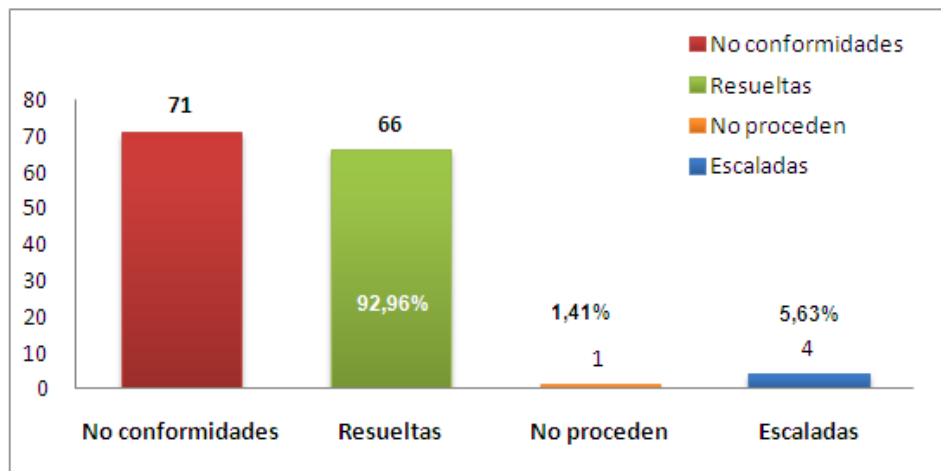


Figura 23: Resultados generales de las pruebas funcionales

De manera general, las pruebas permitieron identificar 71 no conformidades, de ellas se resolvieron 66, lo cual representa el 92,96% del total. Las 4 no conformidades escaladas evidencia la necesidad de realizar una cuarta iteración luego de que se haya dado respuesta sobre estas no conformidades. Las no conformidades escaladas son:

1. En las búsquedas de las operaciones registradas en el sistema aparece la operación "**ADM00**", la cual se refiere a la posibilidad de realizar cualquier operación en el sistema. Esta no conformidad deberá ser evaluada con los funcionarios del Grupo de Coordinación, especialmente con el MININT, para identificar cuál es la solución a ejecutar.
2. El sistema presenta en la interfaz **OrganoEdit.xhtml** el campo "Nombre del órgano". Evaluar la utilización del término "órgano" para referirse a instituciones que no jueguen un rol de órgano sino de organismo, empresa, dirección u otra forma de organización. Evaluar con los funcionales del Grupo de Coordinación, cuál es la terminología apropiada para esto.
3. El nombre de usuario, a especificar en la interfaz **UsuarioEdit.xhtml** acepta números. Verificar con los funcionarios del Grupo de Coordinación si esto deberá ser posible.
4. El acceso a la página de **login.xhtml** provoca el cierre de sesión del usuario actualmente autenticado.

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 tercera iteración, la cual permitió cerrar el ciclo de pruebas:

Tabla 14: Caso de prueba Autenticar Usuario

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 15: Caso de prueba Crear 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 introducidos 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 16: Caso de prueba Eliminar Usuario

Caso de Prueba Funcional	
Código: HU1_P3_I3	Historia de Usuario: No.1 Administrar Usuario
Nombre: Eliminar usuario	
Descripción: Prueba para verificar la funcionalidad eliminar usuario.	
Condiciones de Ejecución: El usuario debe estar registrado en el sistema y tener permiso para eliminar un usuario.	
Entrada / Pasos de ejecución: El usuario selecciona la opción eliminar.	
Resultado Esperado: El sistema debe eliminar el usuario seleccionado.	
Evaluación de la Prueba: Prueba satisfactoria.	

3.3 Conclusiones del capítulo

Las tareas de implementación fueron realizadas en el plazo e iteración acordada, brindando el funcionamiento del sistema al cual se le realizó un conjunto de pruebas. Las pruebas realizadas a la solución se terminaron en el tiempo acordado dando un resultado satisfactorio, lo que demuestra que la efectividad del desarrollo dirigido por pruebas, juega un papel fundamental en el proceso de desarrollo de software. A partir de las pruebas realizadas, se pudo evidenciar que todas las funcionalidades quedaron implementadas, independientemente de las inconformidades escaladas a los funcionales del Grupo de Coordinación.

Conclusiones Generales

A partir de la realización de este trabajo, se llegaron a las siguientes conclusiones:

1. El Portal del Órgano se considera un DSS ya que cumple con las características generales de este tipo de sistema, identificadas en el marco teórico elaborado.
2. Se identificaron e implementaron 52 requerimientos de administración y 15 de configuración (actualmente en proceso de firma por parte del cliente).
3. Además de los 67 requerimientos asociados a la administración y configuración del sistema, se identificaron requerimientos para los módulos de Reportes, Información y Colaboración.
4. Las pruebas funcionales realizadas validaron el cumplimiento de las funcionalidades del sistema.

Recomendaciones

Se recomienda:

1. Validar los requerimientos del portal con el cliente a partir de la firma de los mismos.
2. Implementar los módulos de Reportes, Información y Colaboración del sistema.
3. Realizar las pruebas de aceptación de los módulos de Administración y Configuración.
4. Realizar una cuarta iteración de pruebas a partir de la consulta realizada a los funcionales del Grupo de Coordinación, sobre las no conformidades identificadas en el proceso de prueba y escaladas a este nivel.

Bibliografía

1. *A Complexity Measure*. **McCabe, T. J.** 1976. 4, Octubre de 1976, IEEE Transactions on Software Engineering, Vols. Vol. SE-2, págs. 308 - 320 .
2. **BEATON, W.** 2006. Eclipse Platform Technical Overview. [En línea] 2006. <http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html>.
3. *Beginning JBoss Seam® From Novice to Professional*.
4. **Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato.** 2004. *Version Control with Subversion*. 2004.
5. **dssmexico.** <http://www.dssmexico.com>. [En línea] [Citado el: 22 de Febrero de 2012.] <http://www.dssmexico.com/productos/bi-architecture.html>.
6. **Farley, Jim.** 2007. *Practical JBoss® Seam Projects*. New York : Steve Anglin, 2007.
7. **Haettenschwiler.** 1999. *Neues anwenderfreundliches Konzept der Entscheidungsunterstützung*. Zurich : s.n., 1999.
8. **Hogguit, Onier and Hernández, Madelaine.** 2011. *Arquitectura de Interoperabilidad*. La Habana : s.n., 2011. Tesis de Ingeniería.
9. **Larman, Craig.** 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Tercera Edición. 2004. ISBN:0131489062.
10. **Lopez, Angel.** 1997. *Java la programación del futuro*. Buenos Aires : Moreno S.A, 1997.
11. **Lopez, Miguel and Habra, Naji.** 2010. *Relevance of the Cyclomatic Complexity Threshold for the Java Programming language*. Pennsylvania State University. 2010. Artículo de Base de Datos Institucional.
12. **Marchioni, Francesco.** 2009. *JBoss AS 5 Development*. Birmingham : Packt Publishing Ltd, 2009.
13. **Metsker, Steven John.** 2002. *Design Patterns Java™ Workbook*. s.l. : Addison Wesley, 2002.
14. **Michael J. Scott, Morton.** 1971. *DSS are interactive computer-based systems, which help decision makers utilize data and models to solve unstructured problems*. Boston : s.n., 1971.

15. **Morton, Michael S. Scott. 1971.** *Management Decision Systems: Computer-Based Support of Decision Making.* Boston : Harvard University, 1971.
16. **Nusairat, Joseph Faisal. 2007.** *Beginning JBoss Seam.* New York : Editorial Board, 2007.
17. **PostgreSQL-es. 2010.** PostgreSQL. [En línea] 2010. http://www.postgresql.org/es/sobre_postgresql..
18. **Power, D. J. 2002.** *Decision support systems: concepts and resources for managers.* s.l. : Quorum Books, 2002.
19. —. DSSResources. [En línea] [Citado el: 1 de Mayo de 2012.] <http://dssresources.com/history/dsshistory.html>.
20. **Power, Daniel J. 2012.** Decision Support Systems Resources. [En línea] 2012. [Citado el: 1 de Mayo de 2012.] <http://www.dssresources.com/>.
21. —. **2012.** Decision Support Systems Resources. [En línea] 2012. <http://www.dssresources.com/>.
22. **Pressman, Roger S. 2002.** *Ingeniería de Software, un enfoque práctico.* s.l. : McGraw-Hill Companies,, 2002.
23. **Quin, Liam. W3C.** [En línea] [Citado el: 1 de Junio de 2012.] <http://www.w3.org/standards/xml/>.
24. **Ralph H. Sprague, Eric D. Carlson. 1982.** *Building effective decision support systems.* New York : Englewood Cliffs, 1982.
25. Seam - Contextual Components. [En línea] [Citado el: 2012 de Mayo de 1.] <http://www.seamframework.org/Documentation>.
26. **Valdés González, Jorge Luis. 2012.** *Modelo de intercambio de datos basado en la homogenización del formato de intercambio.* Departamento de Informática Jurídica, Universidad de las Ciencias Informáticas. La Habana : s.n., 2012. Ponencia de evento. UCIENCIA 2012.
27. **Wells, Don. 2009.** Extreme Programming. [En línea] 28 de September de 2009. [Citado el: 1 de Junio de 2012.] <http://www.extremeprogramming.org/>.