**Universidad de las Ciencias Informáticas**
**Facultad 2**


# DESARROLLO DEL MÓDULO RÉGIMEN DE VIDA PARA LOS CENTROS DE RESIDENCIA SUPERVISADA (CRS) DEL SISTEMA DE GESTIÓN PENITENCIARIA (SIGEP) PARA EL SISTEMA PENITENCIARIO VENEZOLANO

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

AUTORES: Maidelis Estevez Monjes.

Adrian Corso Olmo.

TUTORES: MSc. Karina Sánchez Tamayo.

Ing. Eduardo Aranda Pierre.

**Ciudad Habana, mayo 2011**

## DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los ___días del mes de _____ del año ___.

**Maidelis Estevez Monjes**

_____

Firma del Autor

**Adrian Corso Olmo**

_____

Firma del Autor

**MSc. Karina Sánchez Tamayo**

_____

Firma del Tutor

**Ing. Eduardo Aranda Pierre**

_____

Firma del Tutor

## DATOS DE CONTACTO

MSc. Karina Sánchez Tamayo

- Graduado en el 2005 de la carrera Ingeniería Informática en la CUJAE y la UHO.
- Ha trabajado en la Universidad de las Ciencias Informáticas desde septiembre del 2006.
- Obtuvo recientemente la categoría docente de Asistente y la categoría científica de Máster en Gestión de Proyectos Informáticos.
- Ha sido tutor de una tesis en la UCI en temas relacionados con software de gestión desde su vinculación laboral.
- Ha participado en eventos como UCIENCIA, Informática 2009, 2011 y otros eventos.

Ing. Eduardo Aranda Pierre

- Graduado en el 2010 de la carrera Ingeniería en Ciencias Informáticas de la Universidad de las Ciencias Informáticas.
- Se vinculó a la Universidad como profesor en septiembre del 2010.
- Actualmente recibe cursos de posgrado para su superación como profesional.
- Se ha vinculado a los proyectos productivos en el rol de administrador de base de datos.

## AGRADECIMIENTOS

Al culminar este trabajo son muchas las personas que han influido positivamente en la realización del mismo. Les queremos agradecer, especialmente:

A nuestra tutora Karina, que estuvo con nosotros hasta el final, le agradecemos todas las horas de su valioso tiempo dedicadas a la realización de este trabajo de diploma.

A nuestros profesores Roberto Granda Ruiz que sin su ayuda incondicional no hubiera sido posible terminar el trabajo y a Gueorgui Obregón Obregón por todas las ideas que nos brindó guiándonos por el camino correcto.

A la Revolución por brindarnos la oportunidad de ser mejores y por confiar su futuro en nosotros.

A la UCI por permitir conectarnos al futuro y a la Revolución.

A nuestro comandante en Jefe, único e incomparable Fidel Castro Ruz por esta idea extraordinaria al hacer realidad los sueños e ideas de Martí de una sociedad donde todos somos iguales.

Y a todos los que aportaron su granito de arena.

Muchas gracias!!!!

## DEDICATORIAS

### Maidelis:

*Quiero agradecer y dedicar este trabajo ante todo a mis adorados abuelos **Miriam** (mami negra) y **Juan** (papi lindo), que son lo más grande para mí, la luz que siempre me ilumina y guía por el camino correcto, porque sin su amor y apoyo nunca hubiera logrado hacer realidad este gran sueño que parecía inalcanzable, por ser más que mi vida: mi todo.*

*A mi madrecita **Iliana** (mi adorada mami tina), por apoyarme incondicionalmente y por estar siempre ahí cuando la necesité, por ser mi gran inspiración y ejemplo. A mi padre del alma **Elixander** que me ha apoyado siempre con todo su amor y cariño, y ha confiado plenamente en mí, y por fin voy a poder regalarle este momento que con tanto orgullo ha esperado, verme convertida en toda una profesional. A mi padrastro **Arides** que junto a mi mamá me ha apoyado siempre, he podido contar para lo que sea con él, es como un padre para mí y lo quiero y respeto mucho. Este trabajo es dedicado a ustedes, de todo corazón, ya que sin su ayuda y comprensión no hubiese podido lograrlo.*

*A mis hermanitas **Ilianeidis** y **Lisdeima**, que son la luz de mis ojos, y es por ellas que me inspiraba cada día para salir adelante y ser un ejemplo para ellas. A mis hermanos **Leanyer** y **Elixander** a los que quiero y admiro con toda mi alma.*

*A mi tía **Damaris**, por siempre brindarme todo su cariño y su amor, por apoyarme siempre y estar ahí para mí cuando la necesité, ella fue como una madre para mí en estos 5 años de mi carrera. A su esposo **Iván**, mis primos **Klaudia** y **David**, por darme todo su amor. A todos mis tíos y primos, en general a toda mi familia que de una forma u otra me han ayudado siempre.*

*A mi novio amado: **Oscar**, por darme siempre fuerzas para continuar y brindarme todo el amor del mundo, por soportar siempre a esta mujercita tan malcriada y peleona, por ser incondicional, amoroso, por ser el amor de mi vida.*

*A mi querida amiga del alma que es como una hermana para mi **Ivian** (ivita), a la que le agradezco de todo corazón toda la paciencia que ha tenido conmigo, por apoyarme siempre y por brindarme todo su cariño y afecto.*

*A todas mis amistades de la infancia, y a las que he conocido aquí en la UCI, con los que he tenido la bendición de compartir momentos inolvidables, ellos que han estado en todo momento ofreciéndome su apoyo incondicional y todo su amor.*

*A mi compañero de tesis **Adrian**, que hicimos una unión excelente para alcanzar esta meta juntos.*

**Adrian:**

*A mi **madre Edelmis** por brindarme su amor, confianza y apoyo incondicional, por estar siempre a mi lado haciendo posible cada meta que me he puesto en la vida, por comprenderme cuando he tomado una decisión difícil y confiar en mí, por demostrarme que todo es posible cuando se desea y quiere. A mi **padre Humberto** por haber confiado siempre en mí y saber sin dudar, que podía lograrlo. A mi **hermano Yunier** que siempre me apoyado en los momentos precisos desde que éramos niños y a mi **hermanito Richard** para que siga mi ejemplo.*

*A todas mis tías por quererme tanto y darme todo su cariño y apoyo. A mi **tío Vladi** por saber que siempre podré contar con él. A mis primas en especial a **Laura**. A mi **madrina Caridad** la cual ha sido una segunda madre para mí. A mi abuela Berta que aunque no se encuentre físicamente siempre tendrá un espacio dentro de mi corazón, en fin a toda mi familia de Centro Habana y Marianao por TODO el apoyo durante estos 24 años a quienes le debo muchísimo de mi vida. Al fin les regalo lo que tanto esperaron.*

*A mi compañera, amiga y **novia Eyni** por permanecer todo este curso a mi lado apoyándome y queriéndome como nadie, dándome aliento en todo momento. A todos mis viejos amigos que han estado conmigo desde la infancia y a los nuevos amigos que he conocido en estos últimos 5 años que de una forma u otra estuvieron presentes en cada momento de alegría y tristeza. A mi compañera de **tesis Maidelis** porque mejor no la hubiese querido para compartir este triunfo.*

**RESUMEN**

Para contribuir a la solución de los problemas por los que atraviesa el sistema penitenciario venezolano, y dentro del marco del convenio de colaboración Cuba-Venezuela se le da la tarea a la Universidad de las Ciencias Informática (UCI) de realizar el Sistema de Gestión Penitenciaria (SIGEP). Este sistema tiene como misión automatizar los procesos del sistema penitenciario venezolano y garantizar la gestión rápida y segura de toda la información correspondiente a los internos y funcionarios.

El presente trabajo muestra la solución que dieron los autores a partir de las actividades realizadas por ellos como integrantes del SIGEP en los roles de diseñador y programador durante el desarrollo del módulo Régimen de vida del SIGEP. Se utilizó como fuente de trabajo los requisitos definidos y validados con el cliente, así como las definiciones realizadas en el marco del proyecto. Dentro del mismo, se describen las herramientas, tecnologías y metodología utilizadas para el diseño e implementación del módulo, facilitando la comprensión del documento. Se muestran el diseño y la implementación del módulo Régimen de vida a partir del estudio de la arquitectura definida para el proyecto y del análisis de las funcionalidades definidas para el mismo. Para diseñar el módulo se utilizaron diagramas de clase de cada una de las capas que componen la arquitectura. En la implementación de este diseño se usaron herramientas y tecnologías de la plataforma Java dirigidas hacia la programación web.

Al concluir el trabajo, con el cumplimiento de los objetivos trazados, el módulo se integró a la solución SIGEPv2.1 obteniendo resultados satisfactorios en las pruebas realizadas.

**Palabras claves:**

SIGEP, Arquitectura, Régimen, Diseño, Implementación, Módulo.

# ÍNDICE

INTRODUCCIÓN ..... 1
CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA ..... 5
1.1 Introducción ..... 5
1.2 Sistemas Penitenciarios ..... 5
1.2.1 Régimen de vida del subsistema Atención, Supervisión y Orientación para los Centros de Residencia Supervisada (CRS) ..... 6
1.3 Sistemas Informáticos Penitenciarios ..... 6
1.3.1 Sistema de gestión penitenciaria de Ecuador (SIGPEN) ..... 7
1.3.2 Sistema penitenciario del gobierno de Panamá ..... 7
1.3.3 Sistema de Gestión Penitenciaria (SIGEP) ..... 8
1.4 Tecnologías y Herramientas utilizadas en el desarrollo ..... 8
1.4.1 Metodología de desarrollo ..... 9
1.4.1.1 Rational Unified Process (RUP) ..... 9
1.4.1.2 Flujo de trabajo Diseño ..... 10
1.4.1.3 Flujo de trabajo Implementación ..... 11
1.4.1.4 Flujo de trabajo de Pruebas ..... 11
1.4.1.5 Unified Modeling Language (UML) ..... 11
1.4.2 Herramienta de Modelado ..... 12
1.4.3 Plataforma de desarrollo ..... 12
1.4.3.1 Java 2 Platform Enterprise Edition (J2EE) ..... 12
1.4.3.2 Java Servlets ..... 12
1.4.3.3 Java Server Pages (JSP) ..... 13
1.4.3.4 Apache Tomcat 5.5.17 ..... 13
1.4.3.5 Spring Framework v2 .0 ..... 13
1.4.3.6 Hibernate Framework v3.2.0 cr4 ..... 14
1.4.3.7 DOJO Framework v0.4 ..... 15
1.5 Gestor de base de datos ..... 15
1.6 Entorno Integrado de Desarrollo ..... 16
1.6.1 Eclipse 3.4 ..... 16
1.6.1.1 Subclipse v1.0.1 ..... 16
1.6.1.2 SDE (Smart Development Environment) v4.4.1 ..... 16
1.6.1.3 Spring IDE v2.0 ..... 16

1.6.1.4 Hibernate Tools ..... 17
1.7 Sistema de control de versiones ..... 17
1.7.1 Subversion ..... 17
1.8 Formatos de texto ..... 17
1.8.1 JSON ..... 17
1.8.2 XML ..... 18
1.9 Conclusiones parciales ..... 18
CAPÍTULO II. DISEÑO DEL MÓDULO ..... 19
2.1 Introducción ..... 19
2.2 Patrones de diseño ..... 19
2.2.1 Data Access Object (DAO) ..... 19
2.2.2 Fachada (Facade) ..... 20
2.2.3 Controlador (Controller) ..... 20
2.2.4 Modelo-Vista-Controlador (Model-View-Controller, MVC) ..... 20
2.3 Arquitectura del SIGEP ..... 21
2.3.1 Capa de Acceso a Datos ..... 22
2.3.2 Capa de Lógica de Negocio ..... 23
2.3.3 Capa de Presentación ..... 23
2.3.4 Estructura del Sistema ..... 24
2.4 Modelado del Dominio ..... 27
2.5 Modelo de Datos ..... 28
2.6 Modelado de la Capa de Negocio ..... 29
2.7 Modelado de la Capa de Presentación ..... 30
2.8 Descripción de las funcionalidades ..... 31
2.8.1 Módulo Régimen de vida ..... 31
2.8.1.1 Gestionar Régimen de vida laboral ..... 31
2.8.1.2 Gestionar profesiones u oficios del individuo ..... 35
2.8.1.3 Registrar datos laborales ..... 38
2.8.1.4 Modificar datos laborales ..... 38
2.8.1.5 Consultar detalles de datos laborales ..... 38
2.8.1.6 Gestionar Régimen de vida educativo ..... 38
2.8.1.7 Registrar datos educativos ..... 38
2.8.1.9 Modificar datos educativos ..... 39

2.8.1.9 Consultar detalles de datos educativos ..... 39
2.8.1.10 Consultar entradas tardías del residente ..... 39
2.8.1.11 Consultar actividades de atención por individuo ..... 39
2.8.1.12 Consultar detalles de actividad de atención ..... 39
2.9 Conclusiones parciales ..... 39
CAPÍTULO Ill. IMPLEMENTACIÓN Y PRUEBA DEL MÓDULO ..... 40
3.1 Introducción ..... 40
3.2 Modelo de Implementación ..... 40
3.2.1 Diagrama de Componentes ..... 40
3.2.2 Diagrama de Componentes por capas ..... 41
3.4 Implementación de las entidades del dominio ..... 46
3.4.1 Implementación de la Capa de Acceso a Datos ..... 46
3.4.2 Implementación de la capa de negocio ..... 49
3.4.2.1 Transacciones a Nivel de Negocio ..... 50
3.4.3 Implementación de la capa de presentación ..... 51
3.4.3.1 Implementación de los controladores ..... 52
3.4.3.2 Implementación de las páginas Java Server Page (JSP) ..... 52
3.4.3.3 Implementación de la lógica en el cliente ..... 52
3.5 Implementación de la funcionalidad Gestionar Régimen de vida laboral ..... 53
3.6 Pruebas ..... 55
3.6.1 Prueba de Software ..... 55
3.6.2 Herramientas para las pruebas ..... 56
3.6.2.1 JUnit 3.8.1 ..... 56
3.6.2.2 JMeter 2.3.4 ..... 56
3.6.3 Niveles de Pruebas ..... 56
3.6.4 Técnicas de Pruebas ..... 57
3.6.4.1 Pruebas de Funcionalidad ..... 57
3.6.4.2 Pruebas de Confiabilidad ..... 59
3.6.4.3 Pruebas de Rendimiento ..... 59
3.6.4.4 Pruebas Unitarias ..... 62
3.7 Conclusiones parciales ..... 62
CONCLUSIONES ..... 63
RECOMENDACIONES ..... 64

REFERENCIAS BIBLIOGRÁFICAS ..... 65
BIBLIOGRAFÍA ..... 66
ANEXOS ..... **¡Error! Marcador no definido.**
Anexo 1 Diagramas de clase del diseño ..... **¡Error! Marcador no definido.**
Anexo 2 Diagramas de componentes ..... **¡Error! Marcador no definido.**
Anexo 3 Implementación ActividadLaboralDAOImpl ..... **¡Error! Marcador no definido.**
GLOSARIO ..... 75

# INTRODUCCIÓN

En la República Bolivariana de Venezuela existe una grave situación de violencia y corrupción para con la ciudadanía, situación que no solo afecta a las personas sino también al desarrollo del país. Cada día ocurre un sin número de asesinatos, robos y delitos que provocan el miedo en la vida diaria de los venezolanos. Otra situación alarmante son las prisiones venezolanas, en las cuales la pésima alimentación, las malas condiciones higiénicas y sanitarias, la drogadicción, el mal olor, la falta de agua, entre otras condiciones, describen el hábitat de convivencia de los reclusos venezolanos, convivencia que no es nada favorable para dichos reclusos.

Para contrarrestar esta triste situación, con la llegada a la presidencia de esta República del Comandante Hugo Rafael Chávez Frías en el año 1998, se comenzó a desarrollar un profundo programa de transformaciones sociales, para ello era necesario una nueva constitución que sirviera de base legal para realizar todas las actividades que se querían llevar a cabo, por lo que un año después queda aprobada la Constitución de la República Bolivariana de Venezuela.

En este contexto surge el proyecto de Humanización del Sistema Penitenciario el cual integra atención a la salud de los individuos, asesoría especializada y un sistema informático para gestionar y automatizar las actividades y procesos penitenciarios, conocido como el Sistema de Gestión Penitenciaria por sus siglas SIGEP. El objetivo de este proyecto es diseñar, desarrollar e implementar un sistema informático integral que contribuya al control operativo, administrativo y estratégico del sistema penitenciario y además a garantizar el respeto a los derechos de los internos, su actividad de rehabilitación y reinserción en la sociedad.

La Humanización es un proceso que busca refundar el sistema de prisiones, cambiando los “depósitos de hombres” por lugares dignos que permitan a un grupo de hombres y mujeres habilitarse desde la reclusión para la vida en libertad. Para esto es necesario cumplir los derechos fundamentales de los internos:

- Vida, Salud, Educación, Trabajo, Deporte, Cultura y Recreación.
- Dar un trato digno y respetuoso al interno y su familia.
- Propiciar valores y sentido de vida, elevar la autoestima del interno.

En el Sistema Penitenciario Venezolano existen dos tipos de regímenes para clasificar la ubicación de la población penal, el régimen intramuros, que agrupa a los reclusos que permanecen dentro de instalaciones penitenciarias y el régimen extramuros, que es aquella población que se encuentra fuera de los recintos penitenciarios en formas alternativas de cumplimiento de pena como son: las Unidades Técnicas de Supervisión y Orientación (UTSO), aquellas personas que están bajo libertad condicional y los Centros de Residencia Supervisada (CRS), los cuales son una institución, de carácter especial, la cual está conformada por una Edificación tipo Residencia, proyectada para brindar alojamiento, alimentación, tratamiento integral, educación, recreación y asistencia médica básica a la población penal de hombres y mujeres, quienes hayan cumplido por lo menos una tercera parte de la pena impuesta y obtenido una conducta ejemplar, los cuales serán incorporados bajo la medida de Régimen Abierto, dirigido a lograr su reinserción social mediante la atención individualizada y comunitaria, orientado, asesorado y supervisado rigurosamente por un equipo multidisciplinario de profesionales que darán las herramientas necesarias en el ámbito familiar, personal, educativo, laboral, legal y otros (1). La ubicación física de estas instituciones estará, preferiblemente, cerca de zonas laborales y retirada de cualquier otro establecimiento penitenciario.

El Régimen de vida del Residente[1] dentro del CRS será ejecutado en dos fases: Fase de Inducción y Fase de Desarrollo Progresivo.

La Fase de Inducción comprenderá un lapso no mayor a 15 días, dentro de esta fase se encuentra la Fase de Observación con un lapso no mayor de 8 días, en la cual el Residente será informado acerca del reglamento interno y dinámica de la institución, y será abordado por el Equipo de Tratamiento del local para diagnosticar e identificar sus expectativas, habilidades, carencias y necesidades.

Durante la Fase de Observación, el (la) penado(a) permanecerá dentro del CRS y participará en las actividades de inducción: evaluación psicológica, social, familiar, y talleres de información, excepto que tenga empleo, en cuyo caso la Junta de Tratamiento programará un horario acorde a su disponibilidad.

La Fase de Desarrollo Progresivo comenzará inmediatamente después de culminar la Fase de Inducción, e implica la organización y cumplimiento del Plan de Tratamiento Individual del Residente, el

[1] Es el (la) penado(a) que se encuentra bajo la medida de Régimen Abierto, quien la cumplirá en el Centro de Residencia Supervisada.

mismo es el conjunto de actividades terapéuticas, educativas, laborales y recreativas establecidas por el Equipo de Tratamiento.

En el caso de que el Residente cumpla los requisitos que a continuación se mencionan, podrá solicitar un Permiso de Supervisión Especial, el cual le permitirá hospedarse en el domicilio de su apoyo familiar con la obligación de asistir a las actividades contempladas en el Plan de Tratamiento Individual y cumplir las entrevistas con el respectivo Delegado de Prueba, el cual deberá informar periódicamente a la Junta de Tratamiento y al Tribunal que conoce la causa sobre el desenvolvimiento del individuo de acuerdo a su Plan de Tratamiento Individual:

A. Encontrarse bajo un nivel de supervisión mínimo por parte del Delegado de Prueba.
B. Haber estado en el CRS por un tiempo igual o mayor a 12 meses.
C. Tener documentos de identificación en regla.
D. Constancia laboral vigente.
E. Tener Apoyo familiar.
F. Progresividad evidente en las áreas de tratamiento.
G. No haber sido objeto de sanciones disciplinarias.
H. Cualquier otro que considere pertinente el Tribunal de Ejecución que conoce la causa.

La forma de registrar las actividades anteriormente explicadas dentro de los CRS

- No es uniforme y se lleva a cabo de manera manual por lo que sobresale un descontrol de la participación e incorporación de los individuos a las mismas.
- Debido a la difícil consulta de estas actividades y en ocasiones a la pérdida de las mismas no se sabe cuántas se han desarrollado en las instituciones.
- No permite la toma de decisiones en función de la realidad objetiva de cada institución.

A partir de esta situación, se ha definido como ***problema a resolver:*** El sistema que existe actualmente para procesar el Régimen de vida en los CRS es insuficiente para contribuir a la integración de los individuos a la sociedad.

Basado en lo anterior, se ha definido como ***objeto de Estudio*** de este trabajo Atención de los recursos humanos en los CRS.

Para dar solución al problema se propone como ***objetivo General***: Desarrollar el módulo Régimen de vida del subsistema Atención, Supervisión y Orientación en los CRS para contribuir a la integración de los individuos a la sociedad.

Enmarcado en el ***campo de acción***: Régimen de vida de los individuos en los CRS.

Para darle cumplimiento a los elementos antes planteados se especifican los siguientes ***objetivos específicos***:

- Confeccionar el marco teórico de la investigación.
- Analizar los requisitos de software correspondiente al módulo Régimen de vida.
- Desarrollar el módulo Régimen de vida.
- Validar el módulo Régimen de vida.

Los ***resultados esperados*** para este trabajo son los siguientes:

- Modelo de diseño del módulo Régimen de vida del subsistema Atención, Supervisión y Orientación para los Centros de Residencia Supervisada (CRS).
- Modelo de Implementación del módulo Régimen de vida del subsistema Atención, Supervisión y Orientación para los Centros de Residencia Supervisada (CRS).
- Módulo Régimen de vida del subsistema Atención, Supervisión y Orientación para los Centros de Residencia Supervisada (CRS) funcional integrado al SIGEP.

El presente trabajo de diploma está compuesto por tres capítulos de contenido donde se reflejan los siguientes elementos:

**Capítulo 1**: Fundamentación teórica, características de las herramientas y metodologías utilizadas en el diseño e implementación del módulo Régimen de vida del subsistema Atención, Supervisión y Orientación para los CRS.

**Capítulo 2**: Diseño del sistema. Descripción de los elementos principales del diseño a través de artefactos obtenidos como parte del proceso de desarrollo del módulo Régimen de vida.

**Capítulo 3**: Implementación y prueba del sistema. Descripción de los elementos principales de la implementación a través de artefactos obtenidos como parte del proceso de desarrollo del módulo Régimen de vida.

# CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

## 1.1 Introducción

En este capítulo se mostrará una panorámica sobre algunos sistemas informáticos de gestión para sistemas penitenciarios. Se abordarán conceptos que facilitarán la comprensión del presente trabajo y características de las herramientas, tecnologías y metodología previamente definidas en el proyecto SIGEP, utilizadas en el diseño e implementación del módulo Régimen de vida del subsistema Atención, Supervisión y Orientación para los Centros de Residencia Supervisada (CRS).

## 1.2 Sistemas Penitenciarios

Albornoz Berti, define el Sistema Penitenciario, como el conjunto de normas generales, establecidas y específicas referidas a las penas en sí, el modo de su cumplimiento y el tratamiento de los penados y procesados[2] (2).

En el Diccionario de Ciencias Jurídicas, Políticas y Sociales se define sistema penitenciario como régimen penitenciario, definiéndose este último como: "conjunto de normas legislativas o administrativas encaminadas a determinar los diferentes sistemas adoptados para que los penados cumplan sus penas. Se encamina a obtener la mayor eficacia en la custodia o en la readaptación social de los delincuentes. Esos regímenes son múltiples, varían a través de los tiempos; y van desde el aislamiento absoluto y de tratamiento rígido hasta el sistema de puerta abierta con libertad vigilada. Entre ambos extremos existe una amplia gradación" (3). De acuerdo a la situación existente en cada país se define una serie de normas que estandarizan los procesos penitenciarios.

En el caso de la República Bolivariana de Venezuela, tal sistema está constituido por la legislación vigente, los métodos que se emplearán para lograr su funcionamiento, las diferentes dependencias encargadas de su aplicación, los equipos de trabajo y la infraestructura carcelaria.

[2] Procesado: Persona de sexo femenino o masculino que se le señale como autor o participe de un hecho indigno y que ingresa al Sistema Penitenciario en virtud de Auto de Privación Judicial de libertad dictado por el Juez de Control o de Juicio según el estado del proceso.

### 1.2.1 Régimen de vida del subsistema Atención, Supervisión y Orientación para los Centros de Residencia Supervisada (CRS)

El Régimen de vida es el proceso que se encarga de las actividades educativas y laborales del individuo dentro de los CRS así como sus profesiones u oficios. Esta información es utilizada por el Delegado de Prueba[3] para la ejecución de constataciones laborales y para la planificación de las entrevistas. Además permite conocer las actividades de atención en las que ha participado el individuo y las entradas tardías al centro que ha tenido.

## 1.3 Sistemas Informáticos Penitenciarios

En los últimos años con el objetivo de mejorar la situación existente en los sistemas penitenciarios, varios países han venido desarrollando una campaña de modernización, siendo una de las premisas la instalación de sistemas informáticos que ayuden al control y estandarización de los procesos y la información que de estos se genera.

Para desarrollar el módulo de Régimen de vida del subsistema Atención, Supervisión y Orientación para los Centros de Residencia Supervisada (CRS) se realizó un estudio sobre varios sistemas informáticos similares al SIGEP, de los cuales se obtuvo la información de sus respectivos sitios oficiales. Se proporcionan características de cada uno con el objetivo de mostrar sus fortalezas y debilidades.

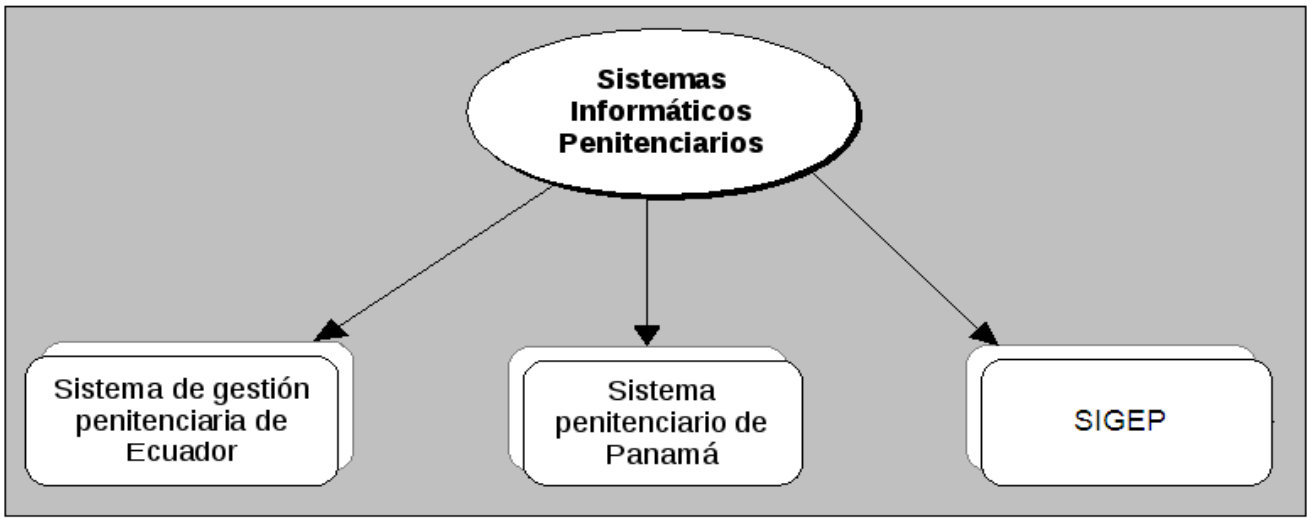



*Figura 1.1: Sistemas Informáticos Penitenciarios*

[3] Delegado de Prueba: es la persona encargada de gestionar el expediente de un penado dentro del CRS.

### 1.3.1 Sistema de gestión penitenciaria de Ecuador (SIGPEN)

***Sitio web oficial:*** *http://www.minjusticia-ddhh.gov.ec*

El SIGPEN es el sistema de gestión penitenciaria de Ecuador. Es un sistema web centralizado, por lo que la información que se maneja sobre los reclusos está constantemente actualizada y se puede acceder desde cualquier lugar siempre que exista conexión a Internet. Brinda información sobre los internos (número de enfermos, internos en pre libertad, habilidades y destrezas, capacidad laboral), áreas administrativas de los centros penitenciarios. Además permite fijar políticas sobre temas específicos mostrando así la flexibilidad del sistema.

Está compuesto por diferentes secciones (Interno, Departamento Jurídico, Departamento Médico, entre otras) las cuales brindan toda la información referente al estado de los reclusos en la sección elegida.

Este sistema tiene  los centros de Rehabilitación Social, los cuales presentan varias actividades que permitan continuar con la preparación de los internos para su reinserción en la sociedad (4). La limitante es que no tienen un Régimen de vida para visualizar todas las acciones que se hayan realizado, ni el registro de las entradas tardías a estas instituciones.

### 1.3.2 Sistema penitenciario del gobierno de Panamá

***Sitio web oficial:*** *http://www.sistemapenitenciario.gob.pa*

El sistema penitenciario panameño ha pasado por un período de modificaciones formado por tres etapas bien definidas: de pre-modernización, de transición y de modernización.

Este período de cambios se inició con la puesta en vigencia de la Ley No.87 de 1941 sobre los establecimientos penales y correccionales. La misma transformó la idea de los establecimientos penitenciarios, de lugares de confinamiento y castigo cruel a centros donde el perfeccionamiento educacional y el amor al trabajo, son ideas inculcadas a las personas bajo detención.

En 1980 las instalaciones del Penitenciario de Gamboa fue destinada a desarrollar un Centro de Rehabilitación Innovador, en el cual la atención de la población penitenciaria estaba encabezada por un equipo interdisciplinario conformado por psicólogo, trabajador social, psiquiatra, abogado, criminólogo y educadores, quienes tenían como función principal, la selección y tratamiento de los privados de libertad, quienes provenían de otros centros penitenciarios, principalmente de la Cárcel

Modelo. Luego se añadieron programas de intervención educativa, social, médica, psiquiátrica y de otra índole (5).

En el año 1997 el gobierno español ofreció una ayuda al gobierno panameño donde entre otros objetivos estaba la transformación de su sistema penitenciario, que incluía la implantación de un software para controlar los diferentes procesos que existen en un centro penitenciario.

Este software consta de una base de datos centralizada y su alcance es registrar y consultar información de cada uno de los reclusos y actividades que se realizan en los centros a nivel nacional.

Existen centros que por falta de infraestructura no están conectados a la base de datos central, lo que trae como resultado que los mismos no se encuentren actualizados sobre el estado de los reclusos en los centros.

El sistema penitenciario panameño tiene permisos laborales y estudio para los reclusos, pero no se visualiza las acciones que se hayan realizado, ni el registro de las entradas tardías de los penados a las instituciones.

### 1.3.3 Sistema de Gestión Penitenciaria (SIGEP)

El Sistema de Gestión Penitenciaria surge en Noviembre del 2006 con el objetivo de diseñar, desarrollar e implementar un sistema informático integral que contribuya al control operativo, administrativo, estratégico del sistema penitenciario, además a garantizar el respeto a los derechos de los internos, su actividad de rehabilitación y reinserción en la sociedad.

Este proyecto en su primera versión se centra en el Régimen Intramuros, el cual agrupa a los reclusos que permanecen dentro de instalaciones penitenciarias, por lo que no se tienen automatizados los procesos Extramuros, que es aquella población que se encuentra fuera de los recintos penitenciarios, en formas alternativas de cumplimiento de pena como son las Unidades Técnicas de Supervisión y Orientación (UTSO), aquellas personas que están bajo libertad condicional, y los Centros de Residencia Supervisada (CRS).

## 1.4 Tecnologías y Herramientas utilizadas en el desarrollo

En el desarrollo de soluciones informáticas la selección de las herramientas correctas tiene una relación directa con el tiempo y la calidad de los artefactos asociados al ciclo de vida del software. En el proyecto SIGEP la selección de herramientas se realizó por el grupo de arquitectura, así como la

definición de tecnologías y los flujos de trabajo por roles realizado por el equipo de analistas. En el presente epígrafe se identificarán las herramientas, tecnologías y metodología utilizada en el desarrollo del trabajo.

### 1.4.1 Metodología de desarrollo

#### 1.4.1.1 Rational Unified Process (RUP)

El problema del software se reduce a la dificultad que enfrentan los desarrolladores para coordinar las múltiples cadenas de trabajo en los grandes proyectos de software. Por lo cual se veían necesitados de un proceso que le integrara las diferentes facetas del desarrollo, un proceso que:

- Proporcione una guía para ordenar las actividades de un equipo.
- Dirija las tareas de un desarrollador por separado y las de un equipo como un todo.
- Especifique que artefactos deben generarse y por quién.
- Ofrezca criterios para el control, la medición de los productos y actividades del proyecto.

Este proceso es RUP, que es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través de UML (Unified Modeling Language), y trabajo de muchas metodologías utilizadas por los clientes.

RUP es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software, diferentes tipos de organizaciones y diferentes tamaños de proyecto.

Esta metodología emplea casos de uso para conducir el proceso de desarrollo, trata de comprender los aspectos estáticos y dinámicos más significativos en términos de arquitectura de software, que en función de las necesidades de los usuarios se refleja en los casos de uso básicos. Además, reconoce que es práctico dividir los proyectos grandes en proyectos más pequeños o mini-proyectos, cada mini-proyecto comprende una iteración que resulta en un incremento (6).

En el proyecto SIGEP se decidió establecer RUP como una referencia para el desarrollo del proyecto, por ser adaptable al contexto y necesidades del mismo. RUP es frecuentemente utilizado en proyectos de gran envergadura como el SIGEP por ser exhaustiva en la generación de documentación basándose en UML. Es práctica para el trabajo con equipos de desarrollo grandes y por tanto difíciles de organizar ya que predefine una serie de roles con tareas y guías de trabajo que reflejan básicamente lo que se espera de ellos. Dentro de los principales roles que RUP define están: ingeniero de requisitos, diseñador, arquitecto, implementador, probador y líder de proyecto.

En la siguiente Figura1.2 se muestran las fases y flujos de trabajo e iteraciones que considera RUP en el desarrollo de un software.

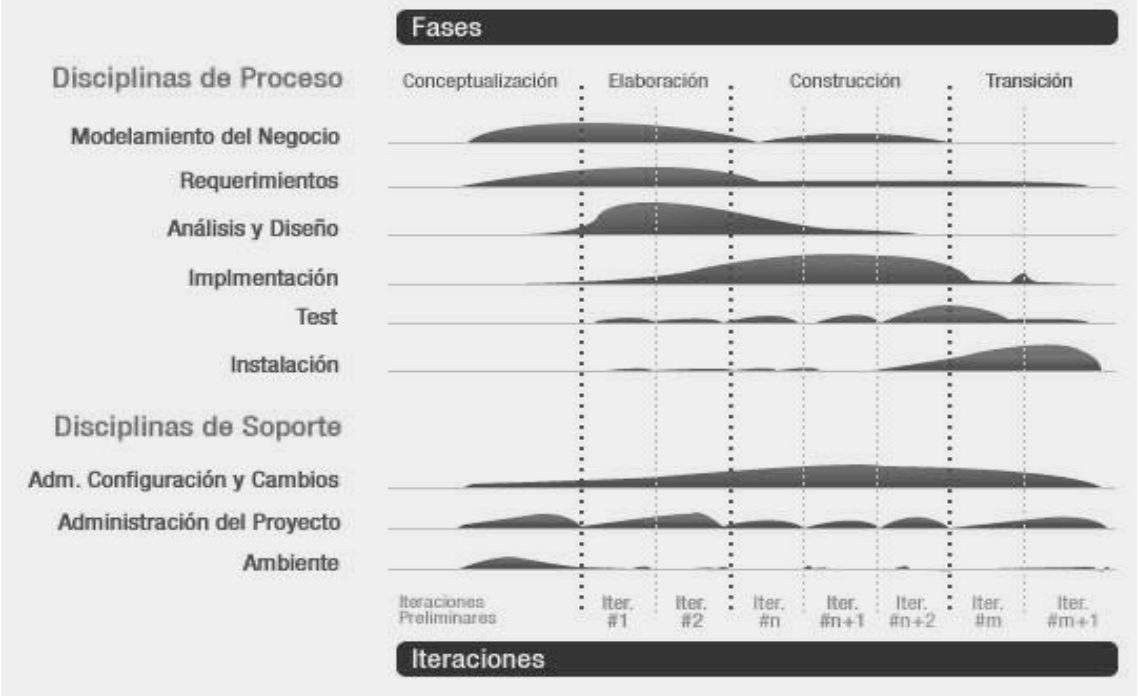



*Figura 1.2: Fases y flujos del RUP*

El desarrollo de la solución que se presenta se centra en los flujos de trabajo propuestos por RUP y adaptados a las características del proyecto SIGEP: diseño, implementación y prueba. El análisis es realizado por el equipo de analistas del proyecto.

### 1.4.1.2 Flujo de trabajo Diseño

Los objetivos del Flujo de Trabajo Diseño son transformar los requerimientos en un diseño de cómo va a ser implementado el sistema, evolucionar hacia una arquitectura del software robusta, adaptar el diseño para que coincida con el ambiente de implementación, diseñando el sistema con un enfoque hacia el rendimiento. Los principales artefactos que propone RUP para esta disciplina son: Modelo de Análisis, Modelo de Diseño, Modelo de Datos y Modelo de Despliegue. Las entradas fundamentales de esta disciplina son las especificaciones de los requisitos de software obtenidas en la disciplina Requisitos. Los trabajadores más importantes de este flujo son el diseñador y el arquitecto.

Según la adecuación hecha por SIGEP de la metodología de desarrollo RUP para nuestro proceso de desarrollo, la cual se especifica en “Visión del sistema de gestión penitenciaria” (7), durante este flujo de trabajo los artefactos que se van a generar son:

Modelo de Diseño: Es un modelo de objetos que describe la realización física de los casos de uso. Contiene subsistemas, paquetes, clases de diseño y diagramas de clases e interacción, sirve de abstracción de la implementación y es usada como entrada fundamental a las actividades de implementación.

Modelo de datos: Describe la representación lógica y física de los datos persistentes de nuestro negocio.

### 1.4.1.3 Flujo de trabajo Implementación

Los objetivos de este flujo son definir la organización del sistema en términos de Subsistemas de Implementación organizados en capas, probar los componentes desarrollados independientemente como unidades, a los cuales se le realiza pruebas de unidad e integrar los resultados en un sistema ejecutable.

Como salida de esta fase obtenemos el Modelo de Implementación el cual incluye: Subsistemas de implementación y sus dependencias, interfaces y contenidos, componentes (ficheros, ejecutables) y dependencias entre ellos.

### 1.4.1.4 Flujo de trabajo de Pruebas

Las pruebas están enfocadas principalmente en la evaluación y determinación de la calidad del producto. Durante esta fase lo fundamental es encontrar y exponer las debilidades en el software.

### 1.4.1.5 Unified Modeling Language (UML)

Es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como RUP), pero no especifica en sí mismo qué metodología o proceso usar. UML proporciona tres beneficios claves: la visualización, la gestión de la complejidad y la comunicación clara. UML ha tenido un enorme impacto en cómo los sistemas software son desarrollados al consolidarse como el lenguaje estándar para el análisis y diseño de estos. Brinda

la posibilidad de establecer una serie de requerimientos y estructuras necesarias antes de escribir el código.

### 1.4.2 Herramienta de Modelado

**Visual Paradigm for UML 3.4 Enterprise Edition**

***Sitio web oficial:*** *http://www.visual-paradigm.com*

Es una herramienta de modelado que soporta: UML, la captura de requisitos, diseño de base de datos, modelado de procesos de negocio para el analista de sistemas y desarrollador de software, además permite realizar el diseño y mantener aplicaciones de software de una forma disciplinada y colaborativa. Visual Paradigm se integra con el IDE Eclipse usado en el proyecto SIGEP permitiendo la generación de código Java y documentación a partir de diagramas; además de mantener dichos diagramas actualizados desde el propio IDE.

### 1.4.3 Plataforma de desarrollo

#### 1.4.3.1 Java 2 Platform Enterprise Edition (J2EE)

***Sitio web oficial:*** *http://java.sun.com/j2ee/overview.html*

La plataforma Java 2 Enterprise Edition (J2EE) es un estándar para el desarrollo de aplicaciones empresariales utilizando el lenguaje de programación Java. Facilita el desarrollo de aplicaciones basándose en pruebas estandarizadas y componentes modulares. Ofrece un conjunto completo de servicios a esos componentes y maneja muchos detalles de comportamiento de la aplicación automáticamente, sin programación compleja. Dentro de las varias especificaciones de APIs[4] que incluye, las que más aportan a la solución SIGEP son: Servlets y Java Server Pages.

#### 1.4.3.2 Java Servlets

***Sitio web oficial:*** *http://java.sun.com/products/servlet/index.jsp*

Un servlet es una clase del lenguaje de programación Java utilizada para ampliar las capacidades de los servidores que alojan aplicaciones accedidas mediante el modelo de programación petición-respuesta. Realizan la función de capa intermedia entre una petición proveniente de un navegador

[4] API: es una interfaz de programación de aplicaciones (del inglés *Application Programming Interface*) es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Web u otro cliente HTTP[5], y las aplicaciones del servidor. Pueden utilizar toda la familia de APIs Java y recibir todos los beneficios del lenguaje, como la portabilidad, el rendimiento y la reutilización. Su función principal es proveer páginas web dinámicas y personalizadas, utilizando para este objetivo el acceso a bases de datos, flujos de trabajo y otros recursos.

### 1.4.3.3 Java Server Pages (JSP)

***Sitio web oficial:*** *http://java.sun.com/products/jsp/overview.html*

Como parte de la familia de la tecnología Java, la tecnología JSP permite el desarrollo rápido de aplicaciones basadas en Web que son independientes de la plataforma. La tecnología JSP separa la interfaz de usuario de la generación de contenidos, permitiendo a los diseñadores cambiar el diseño de la página en general, sin alterar el contenido dinámico subyacente.

### 1.4.3.4 Apache Tomcat 5.5.17

***Sitio Web oficial:*** *http://tomcat.apache.org*

Es un contenedor de servlets creado por la fundación Apache dentro del proyecto Jakarta en un entorno abierto, participativo y publicado bajo la licencia del software de Apache. Implementa las especificaciones de los servlets y de JavaServer Page (JSP). Es usado en numerosas aplicaciones web de gran escala y críticas en una amplia gama de industrias y organizaciones.

Dado que Apache Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

### 1.4.3.5 Spring Framework v2 .0

***Sitio web oficial:*** *http://www.springsource.org*

Es un framework[6] de código abierto que facilita la utilización de la plataforma J2EE haciendo que el desarrollo en la misma sea mucho más fácil, estructurando la aplicación de una forma coherente y productiva. Dentro de las características más importantes que presenta es que está basado en la

[5] HTTP: Hypertext Transfer Protocol. Protocolo a las solicitudes de transferencia de hipertexto y de información entre servidores y navegadores.

[6] Framework: Conjunto de componentes (por ejemplo clases en java) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web.

*inversión de control*[7] y utiliza la programación orientada a aspectos (AOP)[8]. El framework está dividido por módulos que son elegibles dependiendo de las necesidades del desarrollo (8).

En el SIGEP los módulos más utilizados son: *Spring Core* ya que como núcleo del framework constituye la base del sistema, *Spring MVC* que provee su propia implementación del modelo MVC, *Spring ORM (Mapeo de Objeto Relacional por sus siglas en inglés Object Relational Mapping)* que provee a la aplicación de una forma de construir la capa de acceso datos basándose en el patrón DAO e integrándose con el framework Hibernate y *Spring AOP* que provee de un amplio soporte para la programación orientada a aspectos.

### 1.4.3.6 Hibernate Framework v3.2.0 cr4

***Sitio web oficial:*** *https://www.hibernate.org/*

Hibernate es un importante framework de mapeo objeto/relacional y servicio de consultas para Java (disponible también para. Net con el nombre de Nhibernate). Se distribuye bajo los términos de la licencia GNU LGPL[9]. Facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML), detallando cómo es su modelo de datos, qué relaciones existen y qué forma tienen, con esta información Hibernate le permite a la aplicación manipular los datos de la base de datos, operando sobre ellos como objetos, con todas las características de la Programación Orientada a Objetos (POO).

Hibernate genera además las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias. También ofrece un lenguaje de consulta de datos llamado HQL (Hibernate Query Language), y al mismo tiempo APIs para construir las consultas de forma programada.

[7] Inversión de Control: El contenedor inyecta las dependencias al objeto en creación sin esperar que este se las solicite.

[8] AOP (Programación Orientada a Aspecto): Estándar de programación que permite el desarrollo cohesionado ya que separa la lógica de negocio de la aplicación de los servicios del sistema tales como la auditoría y la gestión de transacciones.

[9] LGPL (Lesser General Public License o Licencia Pública General Reducida): es una licencia de software creada por la Fundación para el Software Libre. Permite que software con esta licencia estén integrados en software privativos.

### 1.4.3.7 DOJO Framework v0.4

***Sitio web oficial:*** *http:/www.dojotoolkit.org*

Dojo es un framework que da soporte a la capa de presentación, de código abierto DHTML (Dynamic HTML) escrito en Java Script. Se compone de un conjunto de librerías y widgets[10] pre empaquetados de código Java Script, HTML y CSS, para facilitar el desarrollo de aplicaciones Web que utilicen tecnología AJAX y le brinda a las vistas un conjunto de funcionalidades y un estilo atractivo. Su idea es la de abstraer al desarrollador de las complejidades del DHTML y de las discrepancias existentes entre navegadores, que hacen que el código Java Script a utilizar sea diferente. Algunas de las características que presenta este framework son:

- Oculta el manejo del XMLHttpRequest.
- Manipulación de DOM (Document Object Model).
- Animaciones.
- APIs para el manejo de Ajax y Cometd.
- Manejo de Eventos.
- Drag and Drop.
- Conjunto de Componentes Reutilizables (widgets).
- Gráficos.

## 1.5 Gestor de base de datos

**Oracle 10g**

***Sitio web oficial:*** *http://www.oracle.com*

Es un sistema de gestión de base de datos relacional realizado por Oracle Corporation. Este gestor es robusto, multiplataforma y de gran potencia, generalmente se utiliza en aplicaciones de grandes empresas por su elevado costo de licencias y soporte. Garantiza la seguridad e integridad de los datos, la ejecución de transacciones de forma correcta sin causar inconsistencias, una fácil administración y almacenamiento de grandes volúmenes de datos.

[10] ***Widget:*** es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños.

## 1.6 Entorno Integrado de Desarrollo

### 1.6.1 Eclipse 3.4

***Sitio web oficial:*** *http://www.eclipse.org*

La comunidad Eclipse se compone de personas y organizaciones de una sección transversal de la industria del software. Entre sus herramientas más utilizadas se encuentra el IDE Eclipse. Es una plataforma libre, basada en módulos (plug-ins) agregándole diferentes funcionalidades que posibilitan el desarrollo de las aplicaciones, haciéndolo flexible y extensible. Permite además desarrollar aplicaciones en varios lenguajes de programación como son Java, C++, PHP y otros. En el SIGEP los plug-ins más utilizados son los siguientes:

#### 1.6.1.1 Subclipse v1.0.1

***Sitio web oficial:*** *http://subclipse.tigris.org*

Este plug-in se encarga de manejar la comunicación entre el IDE Eclipse y el servidor Subversion. Dicho servidor es un sistema de control de versiones que permite a varios usuarios trabajar en un mismo proyecto a la vez y es el que se utiliza en el proyecto SIGEP.

#### 1.6.1.2 SDE (Smart Development Environment) v4.4.1

***Sitio web oficial:*** *http://www.visual-paradigm.com/product/sde/ec*

Este plug-in es una herramienta CASE[11] que permite la integración de la herramienta Visual Paradigm con el IDE Eclipse. Soporta todos los diagramas UML generados a lo largo del desarrollo del software. También permite hacer ingeniería inversa para obtener diagramas de clases a partir del código Java y viceversa. Fundamentalmente mantiene la sincronización de las clases con los modelos, además de generar documentación.

#### 1.6.1.3 Spring IDE v2.0

***Sitio web oficial:*** *http://www.springide.org*

Viabiliza el trabajo con Spring ya que contiene un asistente para la creación de nuevos proyectos y archivos de configuración de Spring, posee editores de texto (editor XML con autocompletado) y

[11] Herramienta CASE: Aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software.

gráficos para los archivos de configuración, un visor de los beans configurados y una ampliación del servicio de búsqueda de Eclipse para buscar artefactos de Spring.

#### 1.6.1.4 Hibernate Tools

Facilita el trabajo con el framework Hibernate ya que aporta herramientas de fácil utilización para los programadores: un editor de HQL y un editor de mapeo de XMLs, que posibilita el auto-completamiento y el resaltado de sintaxis. Hibernate Tools es una herramienta de ingeniería inversa de bases de datos que puede generar las clases del modelo de dominio y los archivos de mapeo. También cuenta con varios asistentes para la generación rápida de configuraciones de Hibernate.

### 1.7 Sistema de control de versiones

Es la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. El control de versiones se realiza principalmente en la industria informática para controlar las distintas versiones del código fuente. Sin embargo, los mismos conceptos son aplicables a otros ámbitos como documentos, imágenes, sitios web, etcétera.

#### 1.7.1 Subversion

Soporta el manejo de ficheros y directorios a través del tiempo. Contiene un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios, permitiendo recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos.

### 1.8 Formatos de texto

#### 1.8.1 JSON

***Sitio web oficial:*** *http://www.json.org/json-es.html*

Es un formato de texto utilizado para el intercambio de datos, completamente independiente de los lenguajes de programación como C, C++, C#, Java, JavaScript, Perl, Python y otros. Basado en un subconjunto de JavaScript, utiliza convenciones que son conocidas por los desarrolladores de aplicaciones en distintos lenguajes. Además, mediante el protocolo de llamada a procedimiento remoto (JSON-RPC) permite invocar métodos del servidor desde ficheros JavaScript.

Este formato está compuesto básicamente por dos estructuras: objetos (colección de pares [nombrevalor]) y arreglos (lista ordenada de valores).

Los objetos comienzan con una llave de apertura y terminan con una llave de cierre. Cada nombre es seguido de dos puntos y el valor asignado. Cada uno de estos pares es separado por una coma. Un ejemplo de objeto: {id: 234, nombre:"json"}.

Los arreglos comienzan con corchete izquierdo y terminan con uno derecho. Los valores se separan por comas. Un ejemplo de arreglo: [{id: 123, nombre: "js"}, {id: 456, nombre: "on"}].

### 1.8.2 XML

***Sitio web oficial:*** *http://www.xml.com/*

XML (Extensible Markup Language [lenguaje de marcas extensible]) es un lenguaje de etiquetas desarrollado por World Wide Web Consortium (W3C). Es un lenguaje sencillo y de fácil interpretación por las personas por lo que es muy usado por los desarrolladores de software. Las especificaciones del mismo definen una manera estándar de añadir marcas a los documentos, permitiendo un intercambio de información estructurada entre diferentes plataformas. Puede ser usado en bases de datos, editores de texto, hojas de cálculo, entre otros. Los archivos de configuración de la aplicación están en formato XML, facilitando la gestión de cambios en las configuraciones. Además permite adicionar nuevas etiquetas, lo que lo hace un lenguaje extensible, característica que lo distingue.

## 1.9 Conclusiones parciales

En este capítulo se mostró una panorámica sobre algunos sistemas informáticos de gestión para sistemas penitenciarios de diferentes países, con los cuales no es posible darle solución al problema planteado. Se abordaron conceptos que facilitaron la comprensión del trabajo y características de las herramientas, tecnologías y metodología utilizadas para el desarrollo del módulo Régimen de vida del subsistema Atención, Supervisión y Orientación para los Centros de Residencia Supervisada (CRS). La tecnología de desarrollo que se utilizó fue RUP y las herramientas utilizadas para el desarrollo del módulo son en su mayoría libres, multiplataforma.

# CAPÍTULO II. DISEÑO DEL MÓDULO

## 2.1 Introducción

En el presente capítulo se proporcionan elementos de la solución de diseño por capas, para las funcionalidades definidas en el módulo Régimen de vida. Se realiza una breve descripción de la arquitectura del sistema. Se reflejan los diagramas correspondientes al dominio y a las distintas capas de la aplicación desde un enfoque modular. Se muestran por orden de realización los resultados de cada una de las actividades a realizar definidas en la adecuación hecha por SIGEP de la metodología de desarrollo RUP para la fase de diseño, aunque no se representan todos los diagramas hechos sino solo una muestra representativa de las funcionalidades distintivas del módulo.

## 2.2 Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).Encierran la experiencia que programadores e ingenieros han adquirido en la solución de problemas comunes.

Constituyen soluciones simples y elegantes que ayudan a flexibilizar el código haciéndolo satisfacer ciertos criterios; además de ser una manera más práctica de describir algunos aspectos de la organización del programa. Correctamente elegidos y utilizados contribuyen a una mejor comprensión del diseño o código. A partir de los lineamientos arquitectónicos definidos en el proyecto SIGEP, los patrones de diseño más utilizados son los siguientes:

### 2.2.1 Data Access Object (DAO)

El patrón DAO implementa el mecanismo de acceso requerido para trabajar con la fuente de datos y oculta completamente los detalles de implementación de la fuente a sus clientes. Los componentes del negocio que tratan con los objetos de acceso a datos (DAOs) lo hacen a través de interfaces simples expuestas por estos, dichas interfaces tienen la ventaja de que no sufren modificación alguna cuando cambia la implementación de la fuente de datos subyacente. Lo anterior permite la adopción de diferentes esquemas de almacenamiento sin que esto afecte a clientes o componentes del negocio. En principio este patrón actúa como un adaptador entre el componente y la fuente de datos.

Su principal ventaja es que separa en una capa aparte todo el acceso a datos y abstrae la comunicación con la fuente de datos, esto hace que sea mucho más fácil una migración de fuente de datos en caso de ser necesaria.

En el SIGEP por cada objeto del dominio persistente se define un DAO que expone a través de una interfaz sus funcionalidades donde la implementación está relacionada directamente con la tecnología de persistencia.

### 2.2.2 Fachada (Facade)

El patrón Fachada abstrae las interacciones de los objetos de negocio y proporciona una capa de servicio que expone sólo las interfaces requeridas. Así, oculta a la vista de los clientes las interacciones complejas entre los participantes. La fachada controla las interacciones entre los datos de negocio y los objetos de servicio de negocio que participan en el flujo de trabajo. Encapsula la lógica de negocio asociada con los requerimientos, o sea que su función consiste en proveer una interfaz que servirá de intermediaria entre el cliente y un conjunto de clases o interfaces.

En el SIGEP el patrón es utilizado para separar las diferentes capas. En el caso de la capa de negocio se define una entidad Facade por cada módulo que exporta hacia la capa de presentación las funcionalidades necesarias, de igual forma cada DAO expone las funcionalidades de la capa de acceso a datos. Con la utilización del patrón se minimizan las dependencias que pudieran existir entre capas por lo que un cambio en la capa inferior sería transparente a la capa inmediata superior.

### 2.2.3 Controlador (Controller)

Este patrón propone asignar la responsabilidad de controlar el flujo de eventos de un sistema, a clases específicas llamadas controladores. Los controladores no ejecutan las tareas sino que las delegan en otras clases, generalmente de la capa de negocio, con las que mantiene un modelo de alta cohesión.
En un proyecto como SIGEP esto introduce como ventaja la centralización de toda la gestión de peticiones, permitiendo mayor agilidad en la identificación de errores y corrección de los mismos. Además de aumentar la reusabilidad y el mantenimiento del código.

### 2.2.4 Modelo-Vista-Controlador (Model-View-Controller, MVC)

El patrón MVC es un patrón de diseño que ayuda a darle cierta estructura lógica a la aplicación. Su principal objetivo es separar los datos de la aplicación, la lógica del negocio y la lógica de presentación o interfaz en tres componentes distintos: el modelo que se encarga de la funcionalidad y los datos del

sistema, las vistas que se encargan de mostrar la información al usuario y los controladores que manejan la interacción del usuario con el sistema. De esta manera, se mantiene desacoplada la parte de las listas y la parte del modelo, por lo que no hay que cambiar el modelo por el hecho de cambiar las vistas.

### 2.3 Arquitectura del SIGEP

Para el desarrollo del SIGEP se definió e implementó como arquitectura base ArBaWeb (Arquitectura Base sobre la Web), la cual se basa en los estilos arquitectónicos cliente-servidor y arquitectura en capas. Esta arquitectura tiene como fin: orientar el diseño de las capas lógicas, organizar la forma de codificar según propuestas de convenciones o estándares de códigos y recursos, brindar una estructura física para soportar el código y proponer mecanismos de colaboración entre los componentes integrados en ella (9).

ArBaWeb propone que la aplicación se divida en tres capas: acceso a datos, lógica de negocio y presentación, este modelo de capa permite que cada capa pueda ser modificada tanto como sea posible sin provocar un impacto en las restantes. Una capa no es consciente de lo que le ocurre a la capa superior, su dependencia es puramente con la capa inmediata inferior. Esta dependencia entre capas es normalmente entre interfaces, asegurando que el acople sea el más bajo posible (Ver figura).

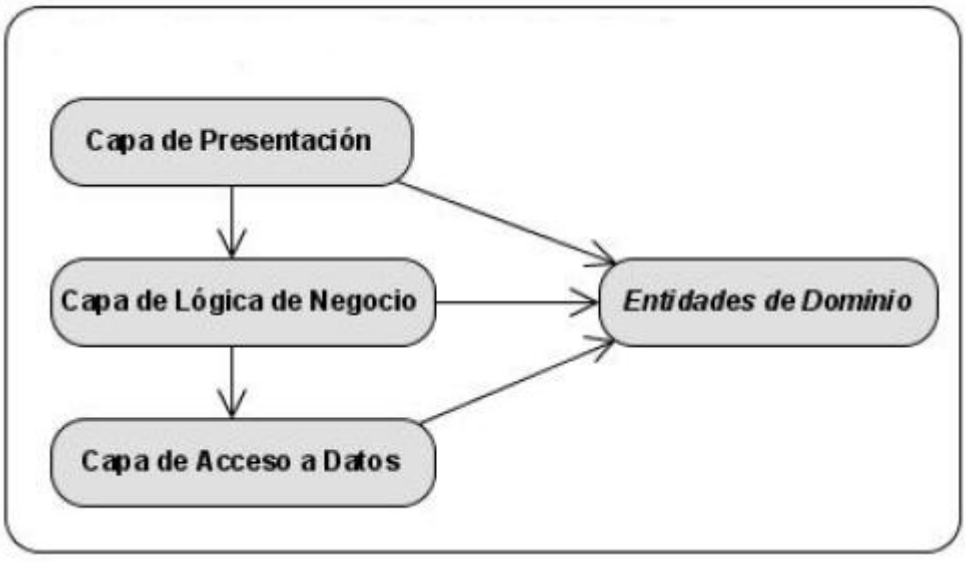



*Figura 2.1: Estructura de capas del sistema*

Un concepto tratado en la figura, es “objetos persistentes del dominio (entidades)”. Este conjunto de objetos puede llegar a considerarse otra capa lógica que puede presentar este tipo de arquitectura.

Estos objetos de dominios no presentan lógica de negocio, esto permite utilizarlos solo como contenedores de información que tradicionalmente son pasados hasta la capa de presentación, en la cual mostrarán los datos que contienen, pero no deben ser modificados, ya que esto solo ocurre dentro de los contextos transaccionales definidos en la capa de servicios de negocio, donde son los objetos de negocio quienes tienen la responsabilidad de la lógica de negocio.

En la arquitectura del SIGEP se define una fachada que representa la interacción entre las Capas de Presentación y Lógica de negocio, la utilización de esta fachada implementa y cumple con el patrón de diseño Facade.

### 2.3.1 Capa de Acceso a Datos

La capa de acceso a datos se encarga de la comunicación con la tecnología de persistencia usada en el proyecto. En esta capa se encuentran los objetos que encapsulan la lógica de acceso a datos o en inglés, Data Access Object (DAO) e interfaces brindadas a través de las cuales la capa de lógica de negocio se comunica con la capa de acceso a datos. Los DAOs encapsulan la persistencia de los objetos de dominios, proveen la persistencia de los objetos transitorios y las actualizaciones de los objetos existentes en la base de datos.

Las implementaciones de los DAOs estarán disponibles para los objetos de la capa de lógica de negocio haciendo uso de la inyección de dependencias entre los objetos de negocio y las instancias de los DAOs, configurada en el contenedor de inversión de control de Spring Framework.

Las implementaciones de los DAOs extienden clases de soporte del framework Spring para el uso de este patrón usando el framework ORM Hibernate, para la realización de sus principales métodos sobre el gestor de base de datos:

- **Métodos para descubrir:** Estos localizan los objetos almacenados para ser usados por la capa de servicios de negocio.
- **Métodos para persistir o salvar:** Estos hacen persistentes a los objetos transitorios.
- **Métodos para eliminar:** Estos eliminan a los objetos guardados en el medio de almacenamiento (generalmente una base de datos).
- **Métodos para conteos y otras funciones agregadas:** Estos devuelven los resultados de operaciones que son más eficientes implementarlas usando funcionalidades de la base de datos (procedimientos almacenas, etcétera.) que iterar sobre los mismos objetos.

### 2.3.2 Capa de Lógica de Negocio

Aunque la lógica de negocio puede estar implementada en los objetos del dominio, ArBaWeb propone la creación de objetos que se encarguen de manejar las transacciones (Managers), garantizando la ejecución de cada una de las acciones que la conforman y en caso contrario, avisar de la imposibilidad de ejecución (10). La comunicación con la capa de acceso a datos es mediante las interfaces de esta última y con la capa de presentación es mediante una fachada (Facade) que contiene todas las funcionalidades de la capa de negocio.

### 2.3.3 Capa de Presentación

En esta capa se utiliza Spring MVC como framework Model View Controller (MVC) para manejar el flujo de datos entre el cliente y las capas inferiores.

Es la responsable de tratar con las interacciones del usuario y obtener los datos que pueden ser mostrados en un formato determinado. Está compuesta por tres tipos de objetos:

- **Controladores:** Estos objetos son responsables de procesar las entradas del usuario en forma de peticiones HTTP, invocando las funcionalidades necesarias expuestas por la capa de servicios de negocio y devolviendo un modelo requerido para ser mostrado.
- **Modelo:** Estos objetos (command) contienen los datos resultantes de la ejecución de la lógica de negocio los cuales son mostrados en la respuesta.
- **Vistas:** Son responsables de mostrar el modelo resultante en la respuesta de la petición. La forma de mostrar el modelo podrá ser de diferentes tipos de vistas, por ejemplo, archivos JSP, HTML, PDF, documentos de Excel, etcétera. Las vistas no son responsables de modificar los datos o incluso de obtener los datos; estas simplemente sirven para mostrar los datos del modelo que han sido suministrados por un controlador.

El cliente interactúa con esta capa directamente utilizando el navegador Web a través de peticiones URL. En el cliente se forma dinámicamente una pequeña capa de presentación que está constituida por código html y JavaScript. Esta capa interactúa directamente mediante peticiones basadas en la tecnología Ajax con los componentes necesarios del servidor creados bajo la tecnología de SpringMVC.

### 2.3.4 Estructura del Sistema

En ArBaWeb se propone como unidad de organización: subsistemas y módulos. Subsistema es: “un conjunto de funcionalidades del sistema que tienen características muy propias y que a su vez están subdivididas en módulos” y módulo es: “un conjunto de funciones que debe realizar el sistema, las cuales son agrupadas por tener características muy similares y se definen en la etapa de diseño”. Un módulo está compuesto por una estructura de paquetes que contribuye a la separación por capas de la aplicación de forma lógica y física (10).

**Estructura de un módulo en el SIGEP**

A continuación se muestra la estructura física del módulo.

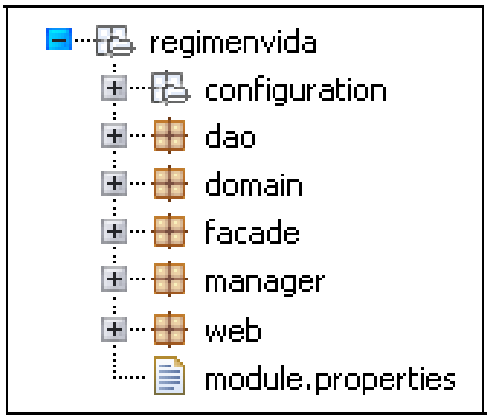



*Figura 2.2: Estructura de paquetes del módulo Régimen de vida*

Cada módulo está compuesto por los siguientes paquetes:

**configuration:** Contienen los archivos que tienen que ver con la configuración del módulo, los “Application-Context” de Spring Framework, los archivos utilizados para la internacionalización, ficheros de propiedades “.properties” y cualquier otro destinado a estos fines.

**dao:** Se encuentran las interfaces DAOs.

**dao.impl:** Se encuentran las implementaciones de las interfaces DAOs.

**dao.map:** Se encuentran los ficheros de mapeo utilizados por Hibernate.

**domain:** Se encuentran todas las entidades persistentes o no del dominio pertenecientes al módulo.

**manager**: Se encuentran las interfaces de los managers de negocio.

**manager.impl**: Se encuentran las implementaciones de los managers de negocio.

**facade:** Se encuentran las interfaces de las fachadas de negocio.

**facade.impl:** Se encuentran las implementaciones de las fachadas de negocio.

**web:** Se encuentran los controladores de la capa de presentación.

**web.command:** Se encuentran las clases utilizadas para guardar datos procedentes de las peticiones web (Command).

**web.editor:** Se encuentran los PropertyEditors.

**web.validator:** Se encuentran las clases utilizadas para validar datos (Validadores).

**Estructura de un Subsistema.**

Un subsistema contiene varios módulos que tienen similitudes. En la Figura 2.3 se expone esta estructura.

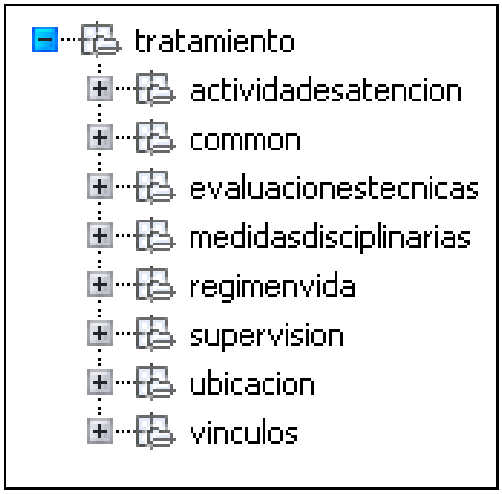



*Figura 2.3: Estructura de paquetes de un Subsistema*

Para cada subsistema existe un módulo común donde se gestionan las funcionalidades comunes nombrado "**common**" (ver Figura 2.4). Los demás paquetes encapsulan los módulos pertenecientes al subsistema.

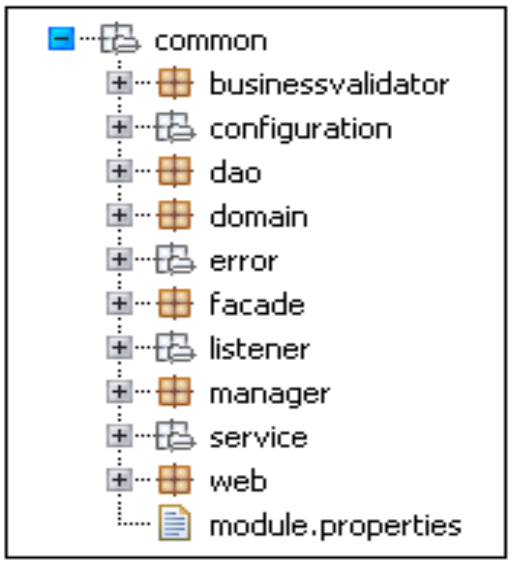



*Figura 2.4: Representación del módulo "common"*

En las siguientes figuras se muestran las estructuras de las “Vistas” del módulo Régimen de vida, ubicadas en la capa de presentación.

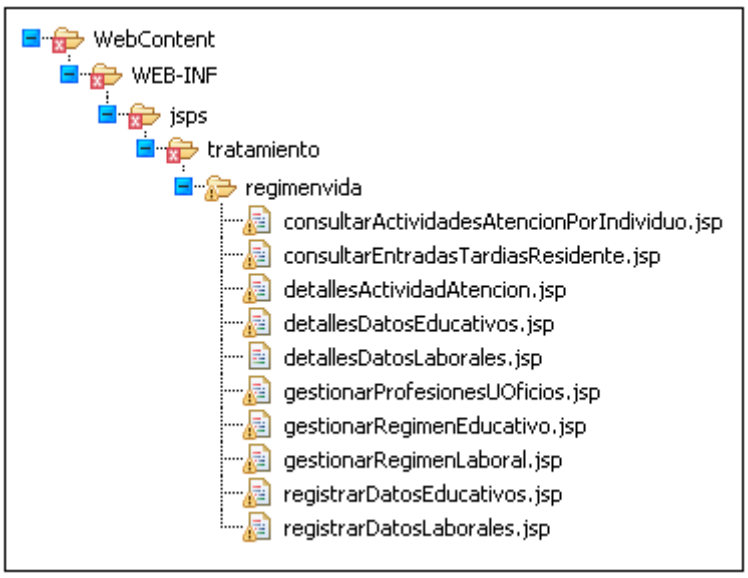



*Figura 2.5: Ubicación de las paginas JSP (Java Server Pages)*

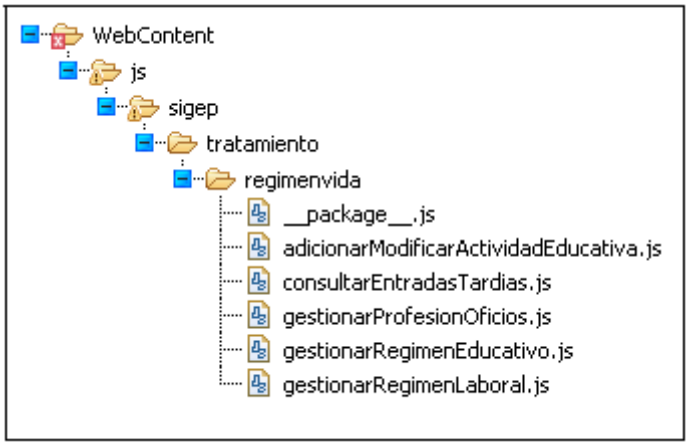



*Figura 2.6: Ubicación de los archivos contenedores del código Java Script*

## 2.4 Modelado del Dominio

El diseño del dominio constituye la entrada principal a las restantes actividades de diseño. En este punto se identifican las entidades que serán gestionadas por la capa de negocio, persistidas o recuperadas por la capa de acceso a datos y mostradas por la capa de presentación. Se identifican los nomencladores. La definición del dominio, en específico de las entidades persistentes sirve como una primera aproximación al diseño definitivo del modelo de datos. En la Figura 2.7 se muestra el diagrama de dominio del módulo Régimen de vida.

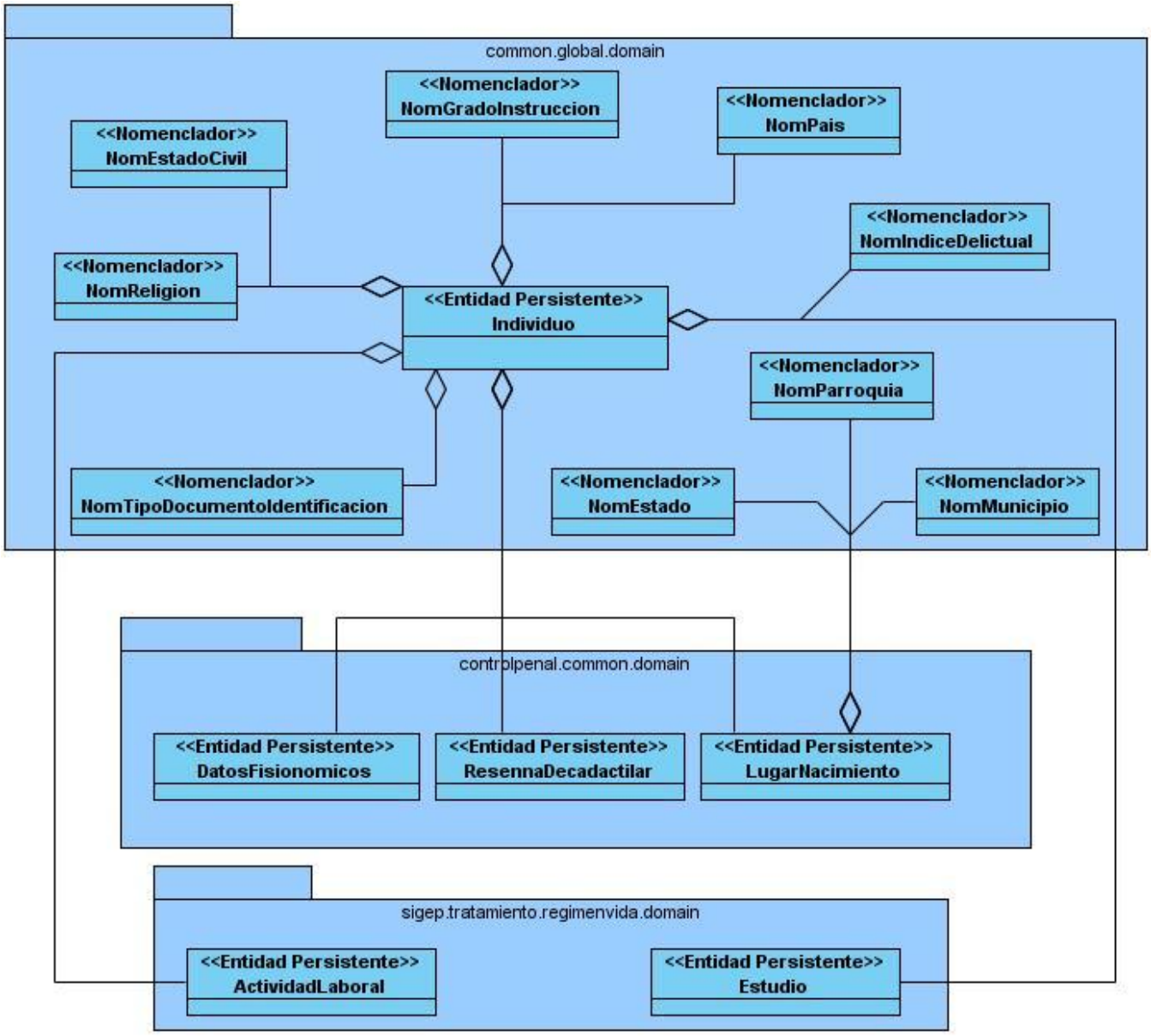



*Figura 2.7: Diagrama de Dominio del módulo Régimen de vida*

## 2.5 Modelo de Datos

A partir del modelado del dominio, donde se definen las clases persistentes y sus relaciones, se obtuvo mediante transformaciones, el modelo entidad-relación del módulo.

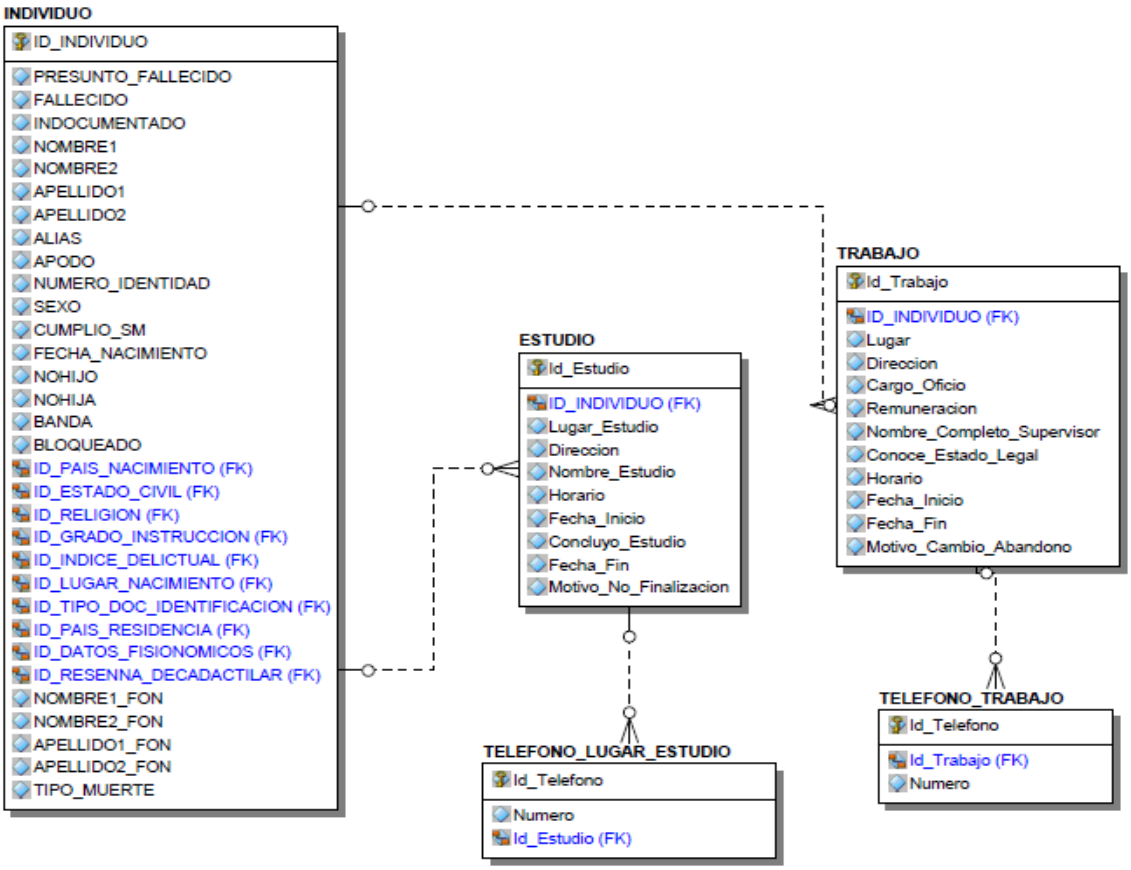



*Figura 2.8: Diagrama entidad relación del módulo Régimen de vida*

## 2.6 Modelado de la Capa de Negocio

El diseño de la capa de negocio contiene las clases necesarias para cubrir las funcionalidades definidas para el módulo y expone, a través de una fachada, sus funcionalidades a la capa de presentación. Si se necesita la comunicación directa de otros módulos o subsistemas con el que se diseña, se define una fachada para ello y se exponen los servicios necesarios.

La fachada tiene la responsabilidad de conocer a los managers, los cuales son los verdaderos objetos de negocio. Los objetos de negocio se definen de acuerdo al agrupamiento lógico de funcionalidades que responden a procesos de negocio. En la Figura 2.9 se muestra el diagrama de clases correspondiente al diseño de la capa de negocio del módulo como ejemplo de lo anteriormente explicado. Se muestran además las interfaces e implementaciones de los managers y la fachada definidas.

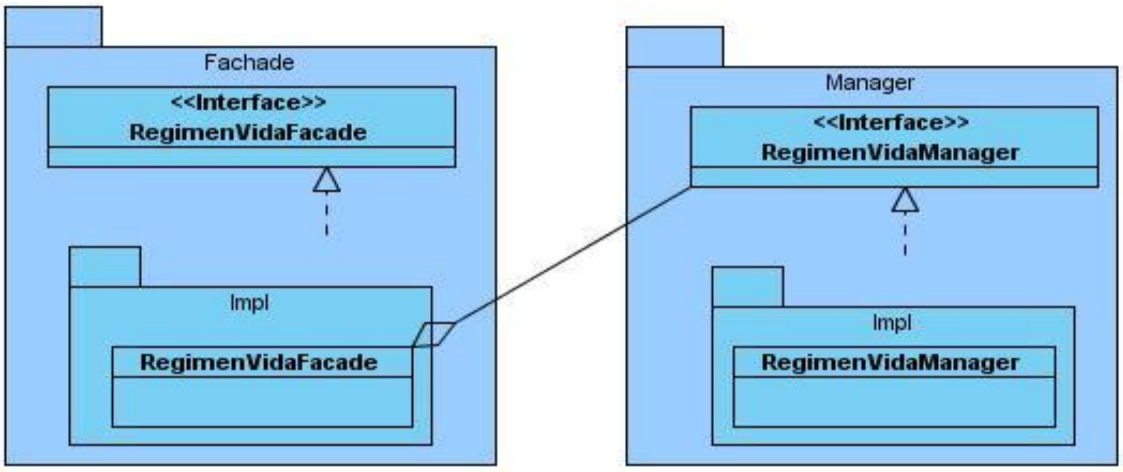



*Figura 2.9: Modelado de la Capa de Negocio del módulo Régimen de vida*

## 2.7 Modelado de la Capa de Presentación

Durante el diseño de la capa de presentación, interviene un componente fundamental y de gran importancia para la realización de cada funcionalidad, el **controlador.** Los controladores se encuentran dentro del contenedor de servlets Tomcat y son los responsables de atender cada petición que llega desde el cliente, construir cada una de las páginas cliente con los datos que estas necesiten o recolectar datos desde la misma para realizar las distintas funcionalidades.

Para el diseño de la capa de presentación se utilizaron las facilidades del módulo MVC del framework Spring, respecto al manejo de peticiones con los controladores (especializaciones de la entidad Controller). Entre los controladores utilizados en el diseño del módulo para la extensión de los controladores del módulo se encuentran el SimpleFormController y MultiActionController.
El controlador SimpleFormController es muy útil para manejar páginas que contengan formularios, ya que permite manejar las peticiones de: solicitud del formulario y envío de los datos al servidor (submit del formulario). Además contiene métodos que posibilitan conocer el tipo de petición que se está haciendo al servidor (solicitud del formulario o envío de datos), así como el envío del formulario con información al cliente y el mapeo de los campos de un formulario con atributos de objetos Java (Command).

El controlador MultiActionController brinda la posibilidad de manejar varias peticiones con una sola clase; donde se crea un método encargado de dar respuesta por cada petición. Extender de este controlador resulta útil si se tienen un conjunto de funcionalidades similares o relacionadas lógicamente. Como ejemplo de ello en el módulo se definió el controlador

RegimenVidaMultiActionController encargado de dar respuesta a las peticiones realizadas por la mayoría de las funcionalidades del módulo.

## 2.8 Descripción de las funcionalidades

Como resultado del flujo de requisitos se definieron las funcionalidades a las que debe responder el diseño y la implementación del módulo Régimen de vida del subsistema Atención, Supervisión y Orientación. Estas funcionalidades se mencionan a continuación:

- Módulo Régimen de vida:
    - Gestionar Régimen de vida laboral.
    - Gestionar profesiones u oficios del individuo.
    - Registrar datos laborales.
    - Modificar datos laborales.
    - Consultar detalles de datos laborales.
    - Gestionar Régimen de vida educativo.
    - Registrar datos educativos.
    - Modificar datos educativos.
    - Consultar detalles de datos educativos
    - Consultar entradas tardías del residente.
    - Consultar actividades de atención por individuo.
    - Consultar detalles de actividad de atención.

A continuación por cada funcionalidad definida se especifica el nombre y una breve descripción, se describen además los diagramas de clases del diseño de algunas funcionalidades. Para alcanzar mayor claridad del diseño, los diagramas están divididos en tres partes (capa de presentación, capa de negocio y capa de acceso a datos y un cuarto diagrama que representa las relaciones entre las diferentes capas).

### 2.8.1 Módulo Régimen de vida

#### 2.8.1.1 Gestionar Régimen de vida laboral

**Breve descripción:** Esta funcionalidad permite consultar el Régimen de vida laboral del residente, donde se muestran todos los empleos en los que se ha desempeñado o se desempeña y brinda las opciones de Adicionar, ver Detalles, Modificar y Eliminar una actividad laboral.

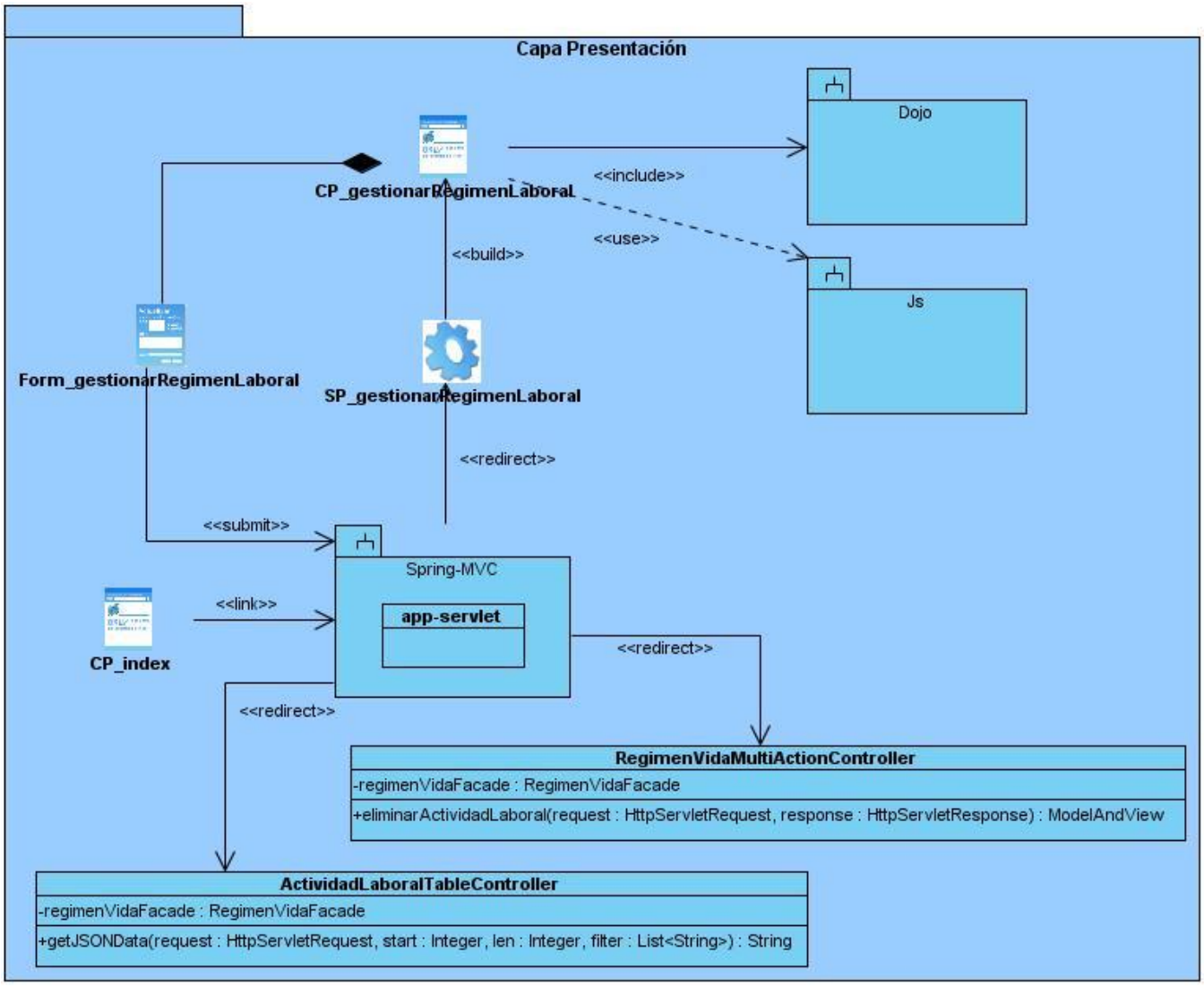



*Figura 2.10: CU: Gestionar Régimen de vida laboral (Capa de Presentación)*

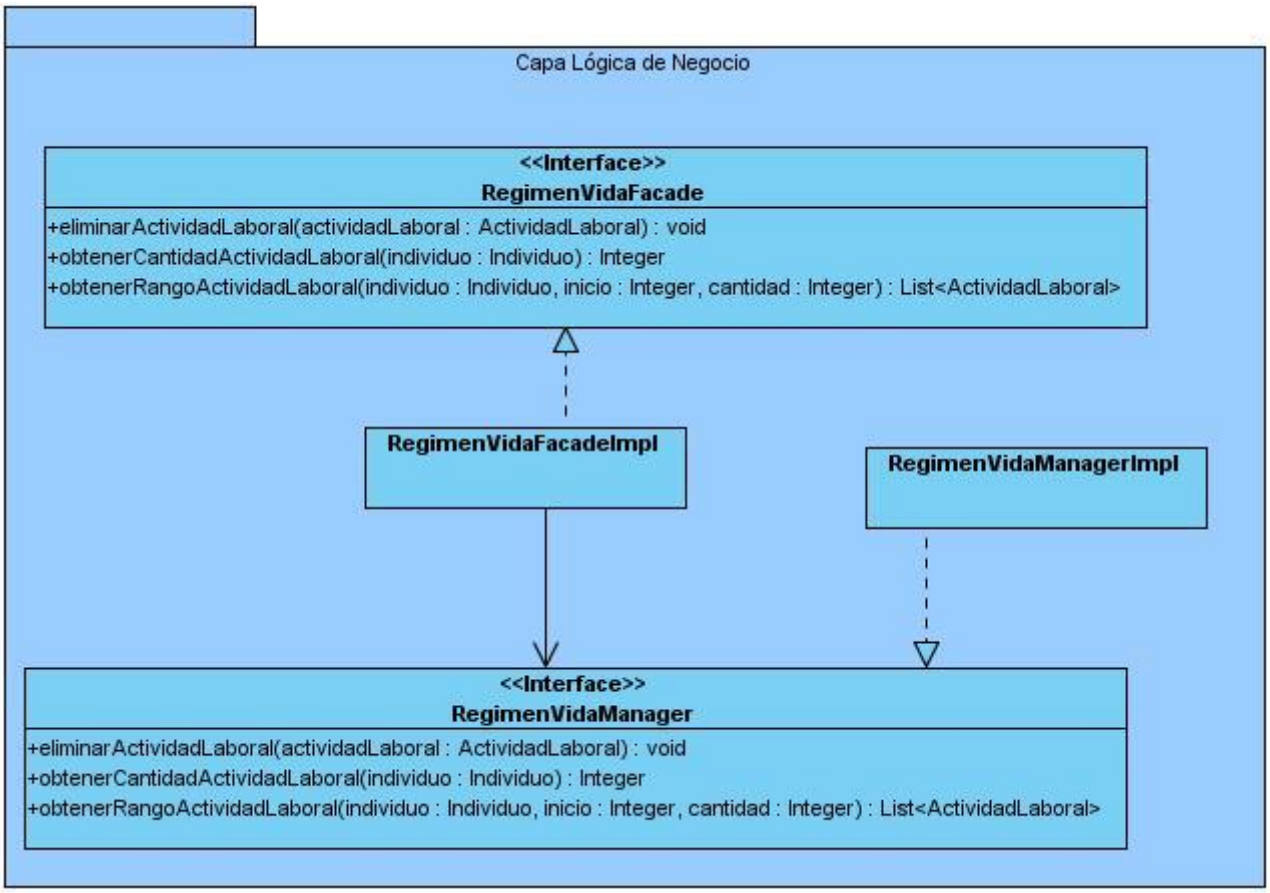



*Figura 2.11: CU: Gestionar Régimen de vida laboral (Capa de Lógica del Negocio)*

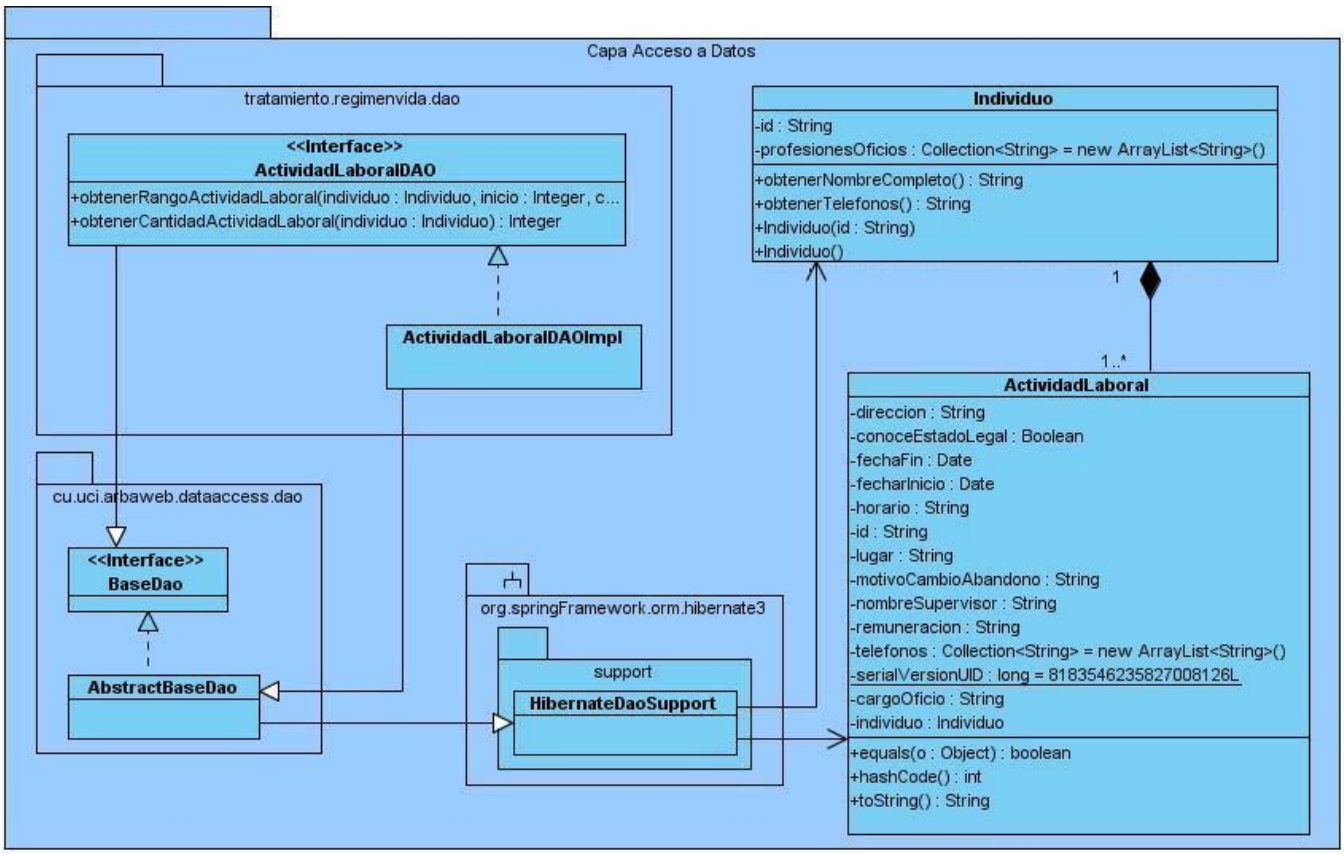



*Figura 2.12: CU: Gestionar Régimen de vida laboral (Capa de Acceso a Datos)*

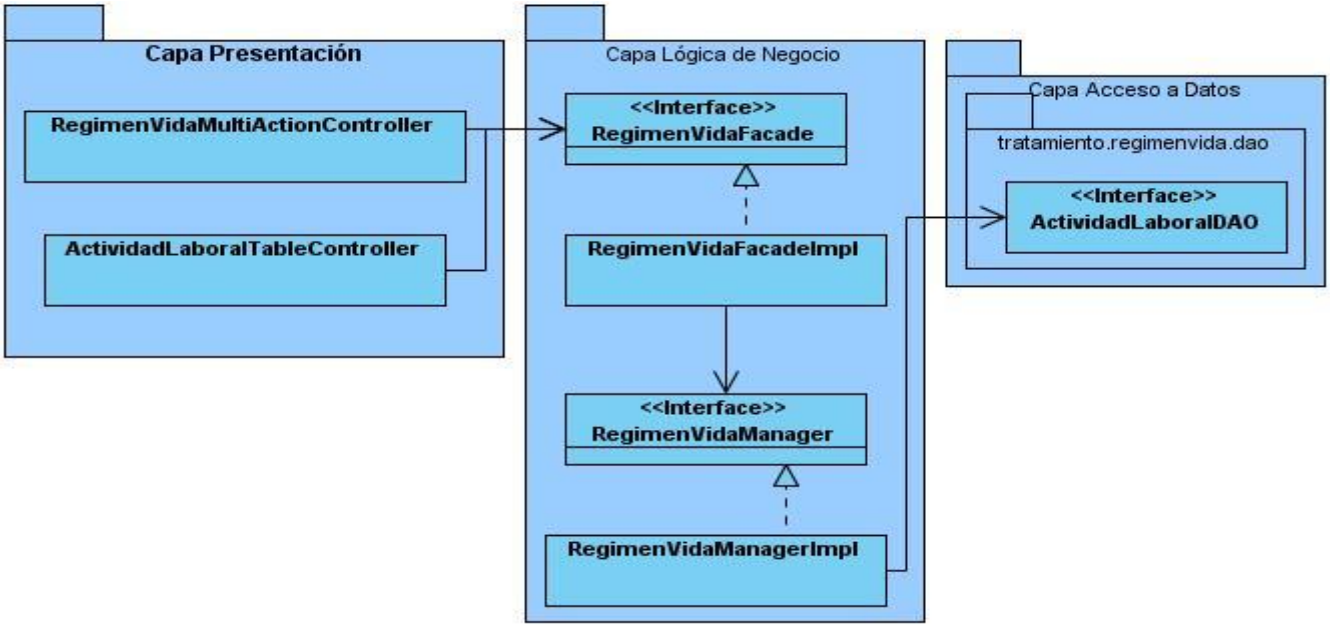



*Figura 2.13: CU: Gestionar Régimen de vida laboral (Relaciones entre Capas)*

### 2.8.1.2 Gestionar profesiones u oficios del individuo

**Breve descripción**: Esta funcionalidad permite mostrar todas las profesiones u oficios del residente, así como registrarlas y/o modificarlas.

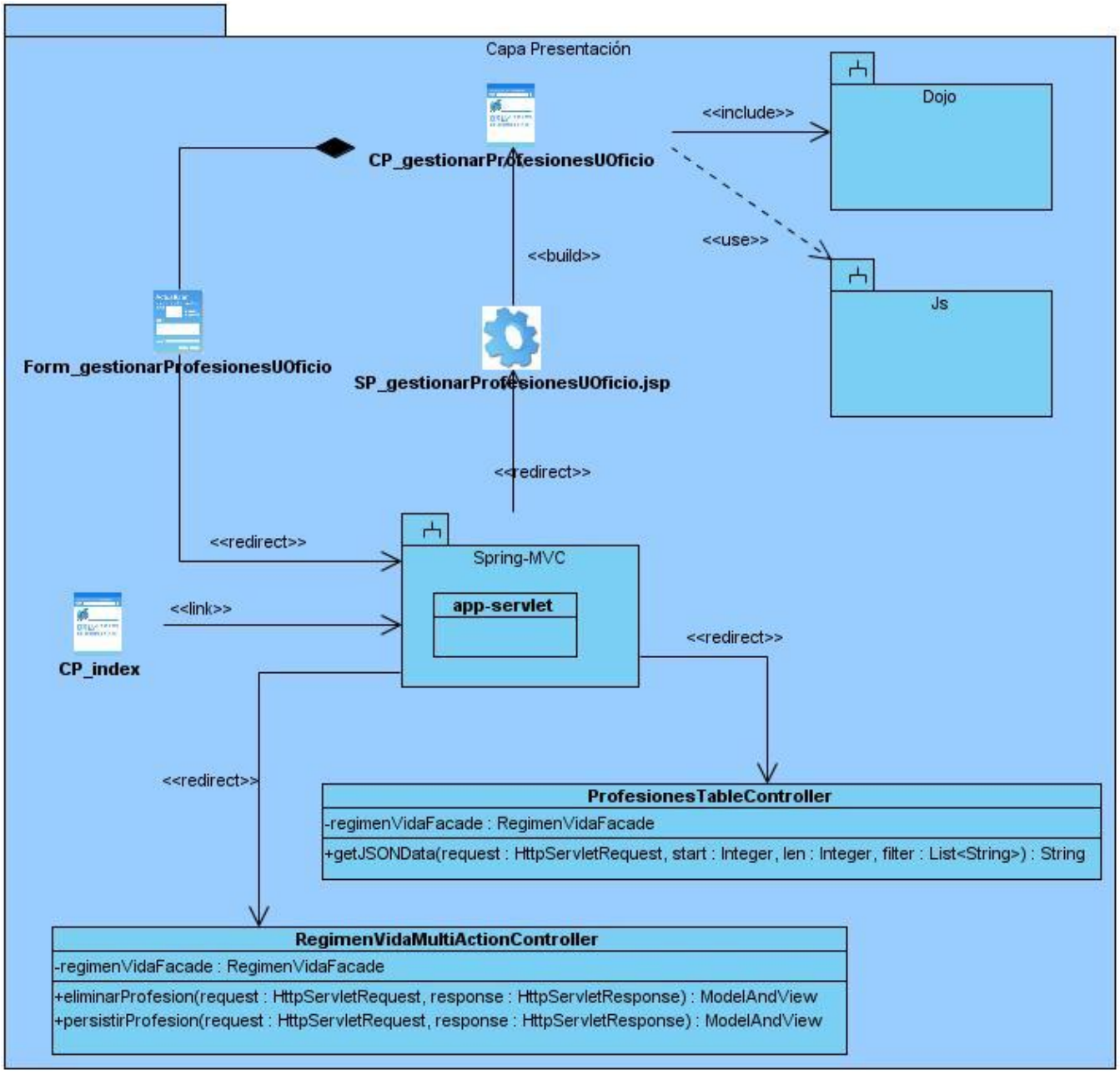



*Figura 2.14: CU: Gestionar profesiones u oficios del individuo (Capa de Presentación)*

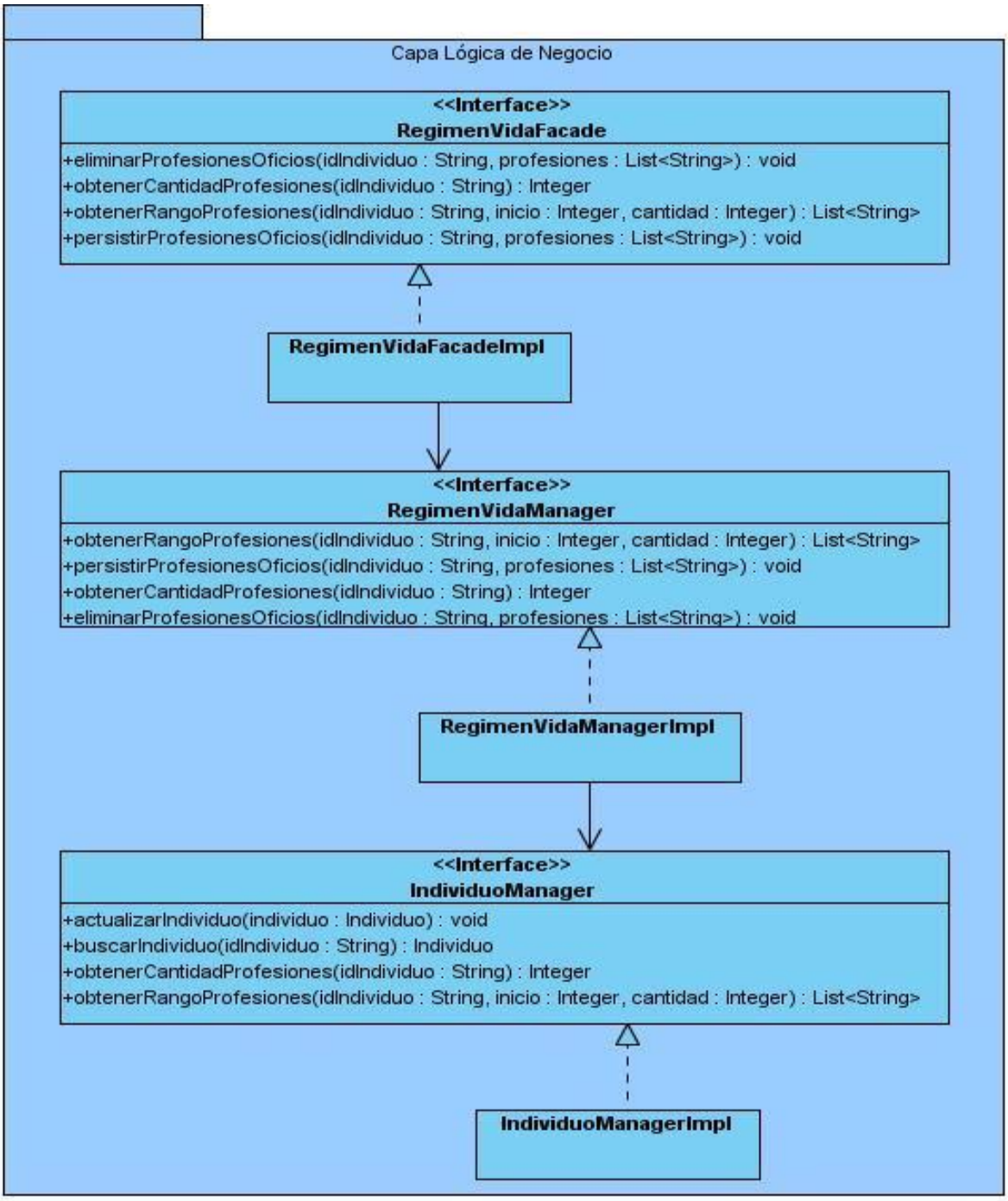



*Figura 2.15: CU: Gestionar profesiones u oficios del individuo (Capa de Lógica de Negocio)*

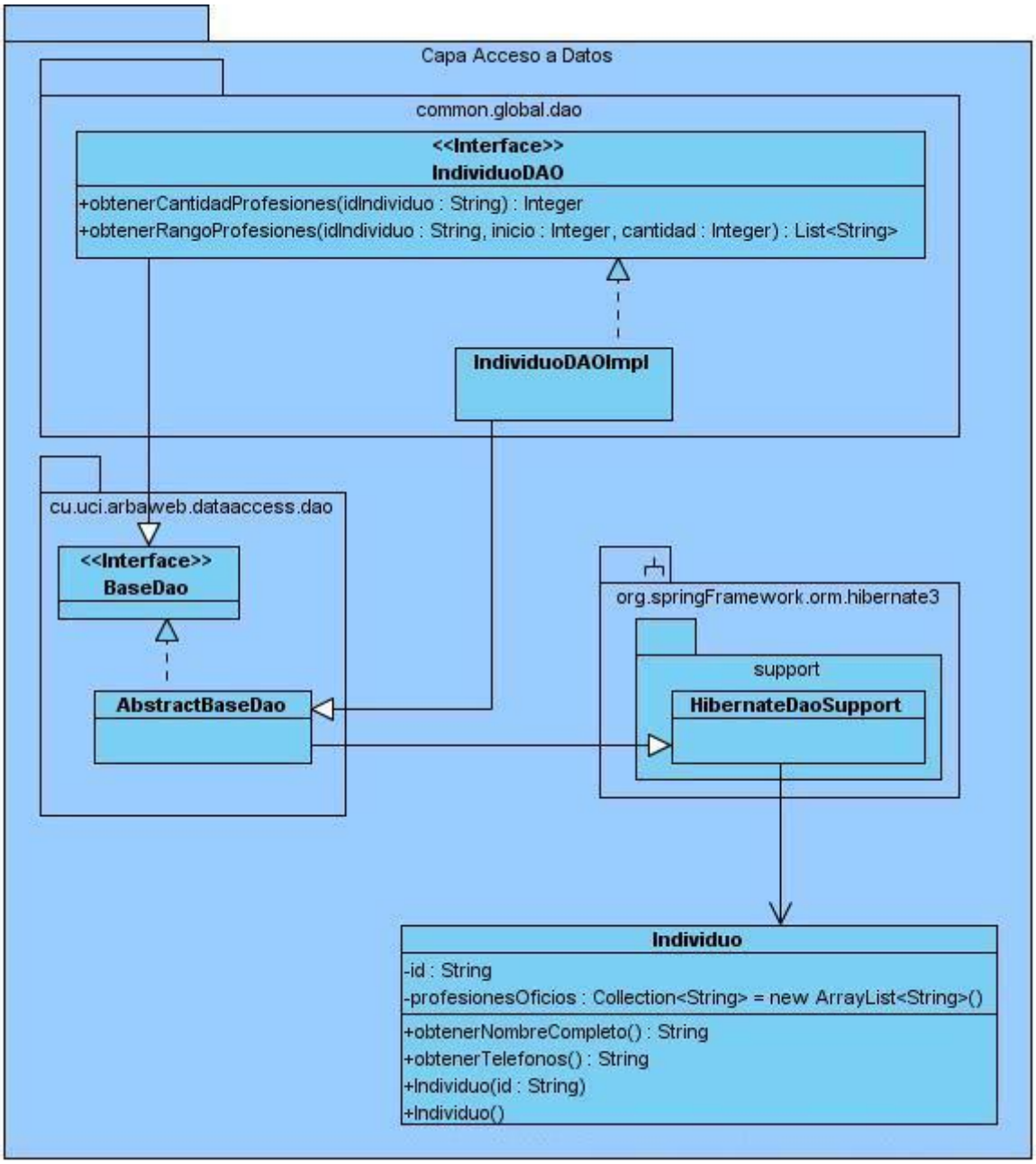



*Figura 2.16: CU: Gestionar profesiones u oficios del individuo (Capa de Acceso a Datos)*

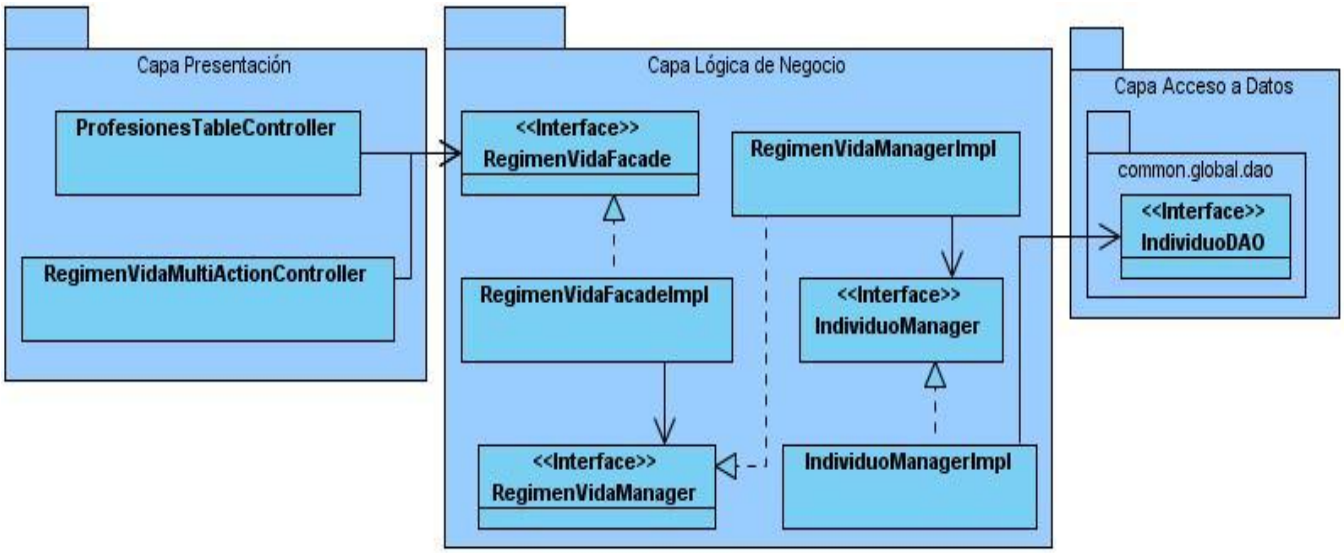



*Figura 2.17: CU: Gestionar profesiones u oficios del individuo (Relaciones entre Capas)*

### 2.8.1.3 Registrar datos laborales

**Breve descripción:** Esta funcionalidad permite registrar un nuevo empleo de un individuo.

### 2.8.1.4 Modificar datos laborales

**Breve descripción:** Esta funcionalidad permite modificar los datos laborales de un individuo registrado previamente.

### 2.8.1.5 Consultar detalles de datos laborales

**Breve descripción:** Esta funcionalidad permite mostrar los detalles de un registro de datos laborales.

### 2.8.1.6 Gestionar Régimen de vida educativo

**Breve descripción:** Esta funcionalidad permite consultar el Régimen de vida educativo del residente, donde se muestran todos los lugares donde ha estudiado o estudia y brinda las opciones de Adicionar, ver Detalles, Modificar y Eliminar una actividad educativa.

### 2.8.1.7 Registrar datos educativos

**Breve descripción:** Esta funcionalidad permite registrar los estudios del residente.

### 2.8.1.9 Modificar datos educativos

**Breve descripción:** Esta funcionalidad permite modificar los datos correspondientes a los estudios del residente registrados previamente.

### 2.8.1.9 Consultar detalles de datos educativos

**Breve descripción:** Esta funcionalidad permite consultar los detalles de un registro de datos educativos del individuo.

### 2.8.1.10 Consultar entradas tardías del residente

**Breve descripción:** Esta funcionalidad permite consultar todas las entradas tardías del residente. Las entradas tardías son aquellas que tienen una fecha y hora de entrada posterior a la fecha y hora de entrada que estaba prevista.

### 2.8.1.11 Consultar actividades de atención por individuo

**Breve descripción:** Esta funcionalidad permite consultar todas las actividades de atención en la que ha participado un individuo.

### 2.8.1.12 Consultar detalles de actividad de atención

**Breve descripción:** Esta funcionalidad permite consultar los detalles de una de las actividades de atención registradas.

Para consultar los diagramas de las demás funcionalidades Ver anexo 1.

## 2.9 Conclusiones parciales

En el capítulo se hizo referencia y se describieron los elementos fundamentales del diseño del módulo Régimen de vida, en seguimiento de los requisitos definidos para el mismo y ajustado a las definiciones arquitectónicas del SIGEP.

# CAPÍTULO III. IMPLEMENTACIÓN Y PRUEBA DEL MÓDULO

## 3.1 Introducción

En el presente capítulo se realiza el flujo de trabajo de implementación, donde se presentan los elementos fundamentales de la fase de implementación del módulo Régimen de vida, se analizan los principales artefactos como el Modelo de Implementación que incluye componentes, subsistemas de implementación y diagramas de componentes, mostrando fragmentos de códigos de la implementación de funcionalidades características del módulo. Además, se realiza un análisis de los resultados de las pruebas realizadas al módulo como parte de la solución SIGEPv2.1 en un ambiente real.

## 3.2 Modelo de Implementación

El Modelo de implementación representa la composición física de la implementación en términos de subsistemas de implementación y elementos de implementación. Describe como los elementos de diseño se implementan en componentes. Dicho modelo se considera el artefacto más significativo del flujo de trabajo de Implementación, debido a la importancia que tiene para los desarrolladores comprender el funcionamiento del sistema desde el punto de vista de componentes y sus relaciones. Este modelo está conformado por el diagrama de componentes.

### 3.2.1 Diagrama de Componentes

Dentro del Modelo de Implementación se encuentran los diagramas de componentes. Los componentes son la parte física y reemplazable de un sistema que está compuesto por un conjunto de interfaces y proporciona la realización de dicho conjunto. Se usan para modelar los elementos físicos que pueden hallarse en un nodo por lo que empaquetan elementos como clases, colaboraciones e interfaces. Son independientes entre ellos y tienen su propia estructura e implementación.

El diagrama de componentes describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y como dependen los componentes unos de otros. Los diagramas de componentes se utilizan para modelar la vista estática de un sistema. Muestran la organización y las dependencias lógicas entre un conjunto de componentes software, sean éstos componentes de código fuente, librerías, binarios o ejecutables. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación.

### 3.2.2 Diagrama de Componentes por capas

A continuación se muestran los componentes referentes al Diagrama de Componentes del CU Gestionar Régimen de vida laboral y del CU Gestionar profesiones u oficios del individuo, pertenecientes al Módulo de Régimen de vida, distribuidos en cada una de las capas y las relaciones entre ellos: Capa de Presentación, Capa de Lógica de Negocio, Capa de Acceso a Datos y un último diagrama donde se muestran las relaciones entre las capas.

- ✓ **Diagrama de Componentes del CU Gestionar Régimen de vida laboral**

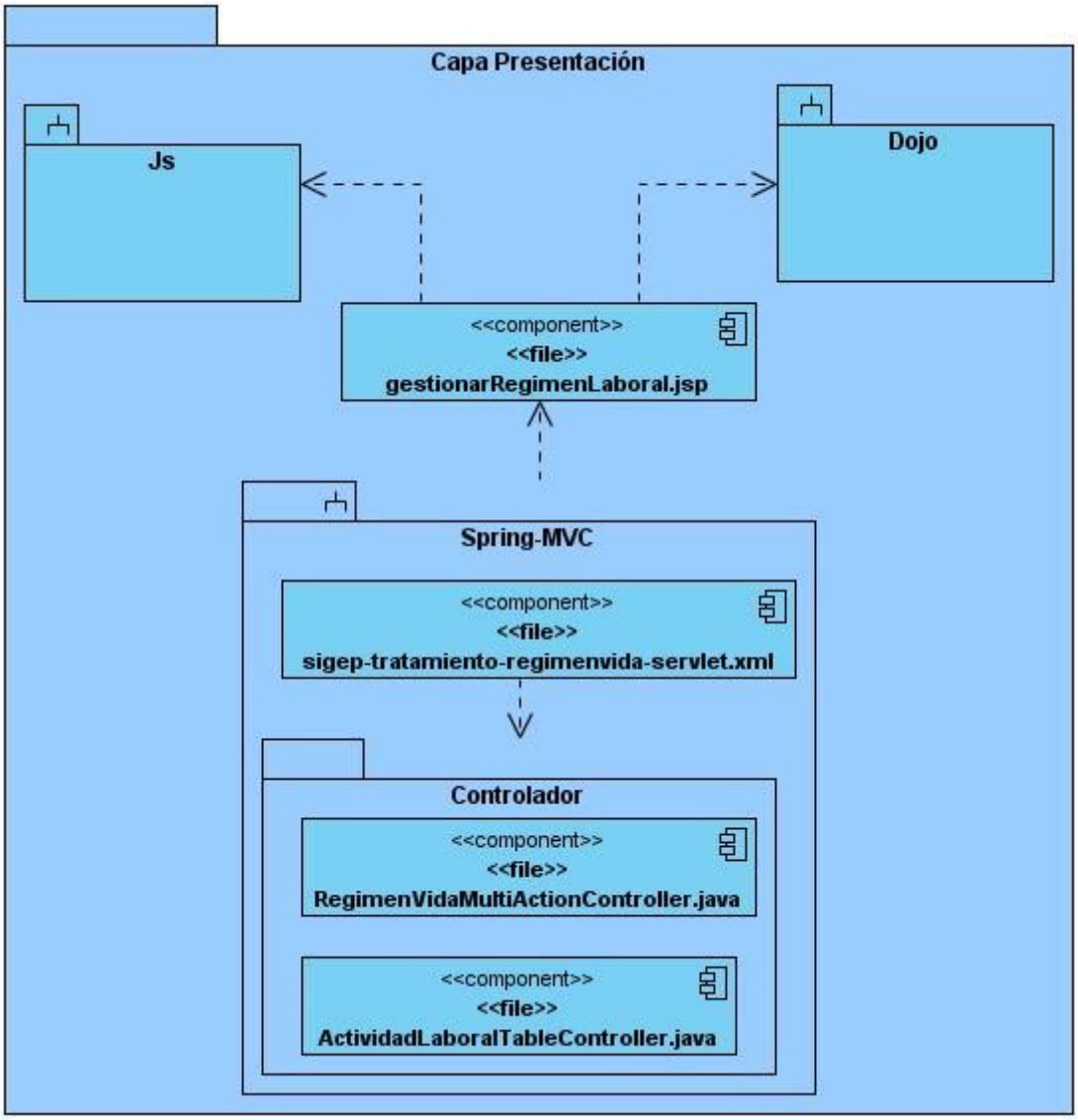



*Figura 3.1: CU Gestionar Régimen de vida laboral (Capa de Presentación)*

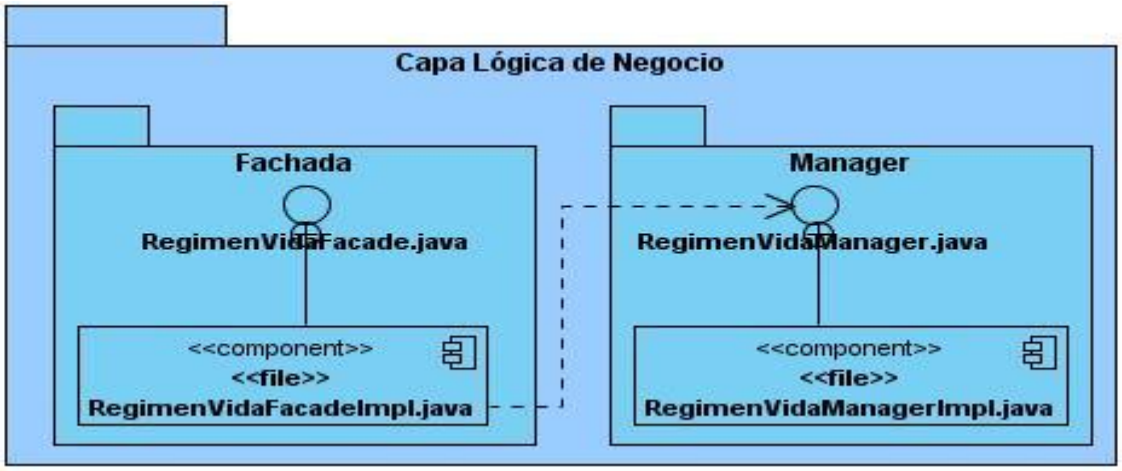



*Figura 3.2: CU Gestionar Régimen de vida laboral (Capa Lógica de Negocio)*

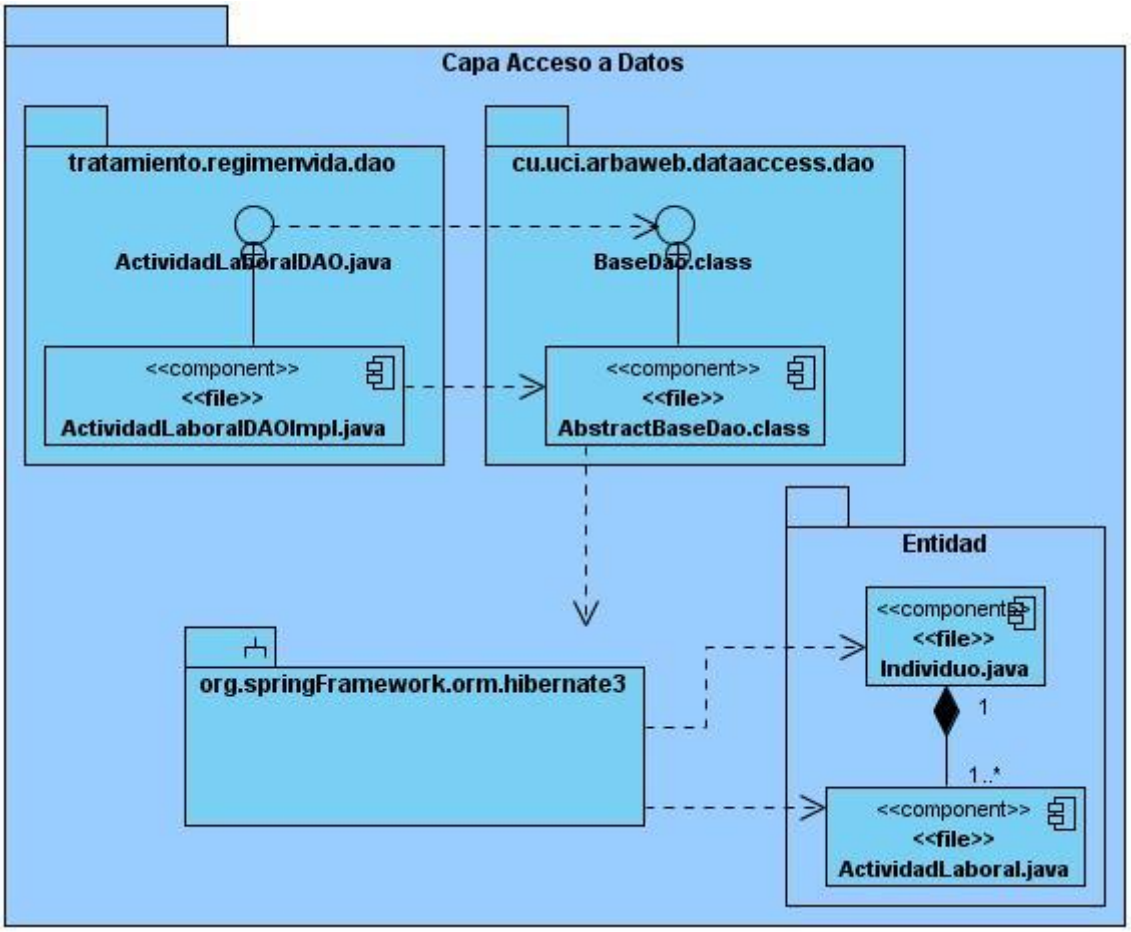



*Figura 3.3: CU Gestionar Régimen de vida laboral (Capa Acceso a Datos)*

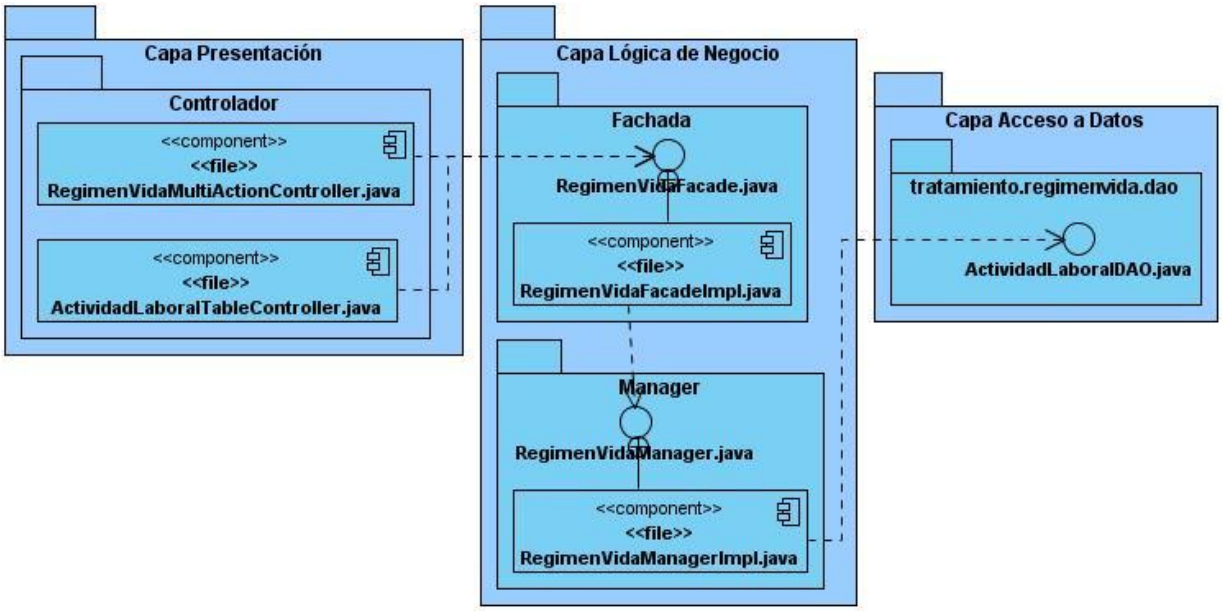



*Figura 3.4: CU Gestionar Régimen de vida laboral (Relaciones entre Capas)*

- ✓ **Diagrama de Componentes del CU Gestionar profesiones u oficios del individuo.**

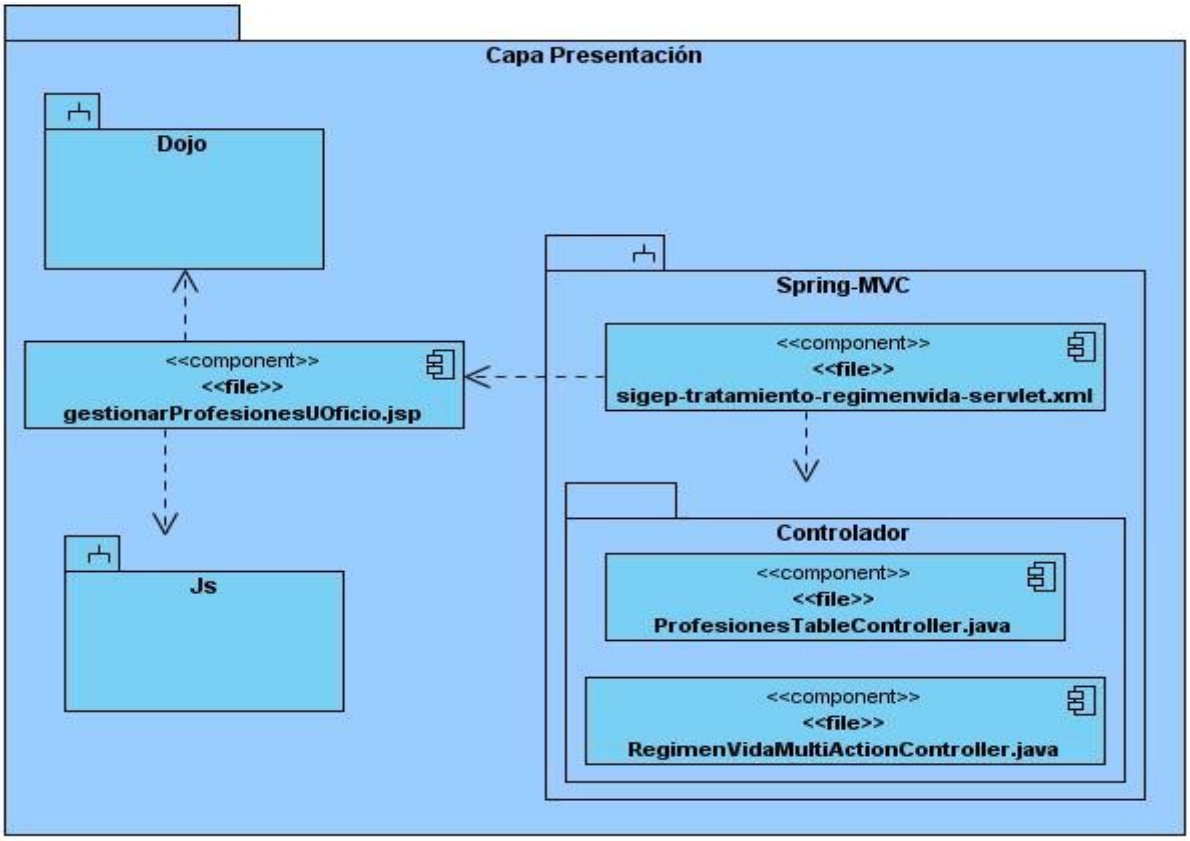



*Figura 3.5: CU Gestionar profesiones u oficios del individuo (Capa de Presentación)*

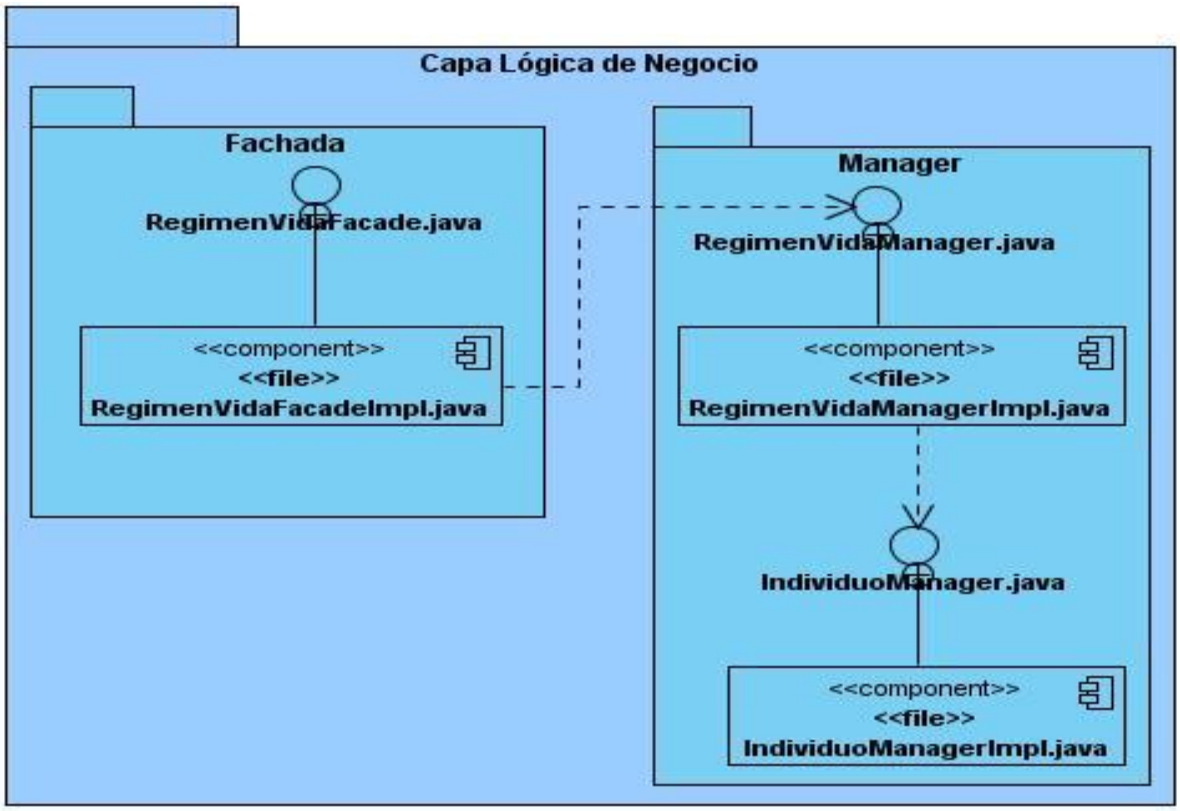



*Figura 3.6: CU Gestionar profesiones u oficios del individuo (Capa de Lógica de Negocio)*

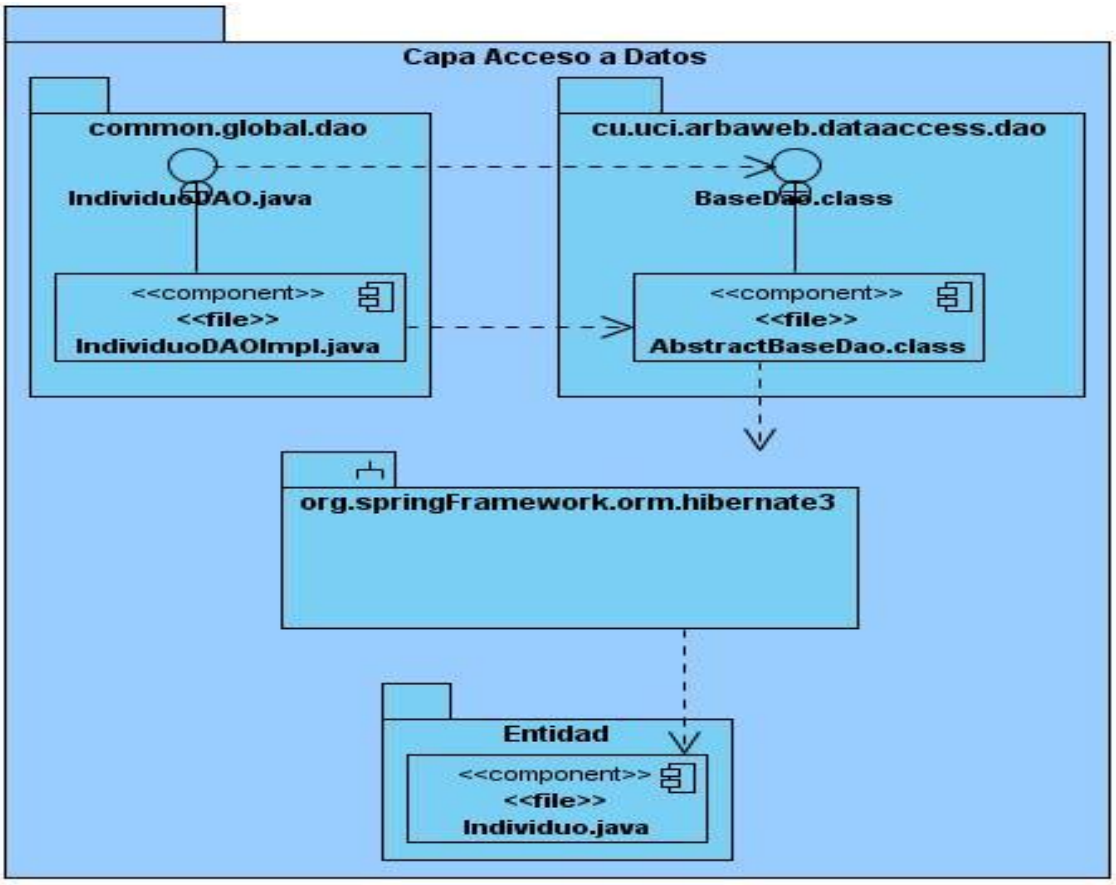



*Figura 3.7: CU Gestionar profesiones u oficios del individuo (Capa de Acceso a Datos)*

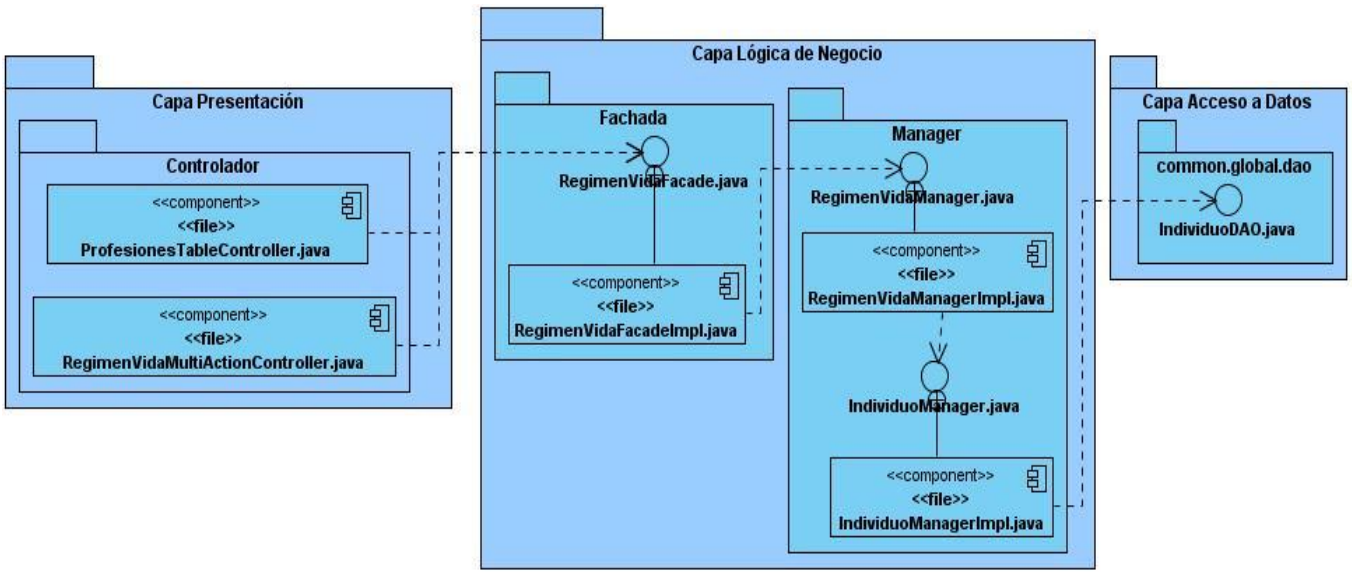



*Figura 3.8: CU Gestionar profesiones u oficios del individuo (Relaciones entre Capas)*

Para consultar los Diagramas de Componentes de las demás funcionalidades ver Anexo 2.

### 3.4 Implementación de las entidades del dominio

Las entidades del dominio suelen tener muy poco comportamiento. Los métodos equals(), que se utiliza para determinar si dos objetos son iguales, el hasCode(), que permite comparar dos objetos para ver si son el mismo, o sea, comparar las referencias de los objetos, es decir sus direcciones de memoria, el toString() que convierte la información de un objeto en una cadena de caracteres string, deben ser implementados en la mayoría de los casos, ya que son frecuentemente utilizados. En este paso se debe refinar la definición de todos los atributos de las entidades de manera que queden listas en un buen por ciento para implementaciones reusables (11).

#### 3.4.1 Implementación de la Capa de Acceso a Datos

La implementación de la capa de acceso a datos se basa en dos tareas fundamentales, la creación de los ficheros de mapeo de Hibernate (hbm.xml) y la implementación de las funcionalidades de los DAOs.

Los ficheros hbm.xml son usados por el framework ORM Hibernate, para mapear los campos entre las tablas del gestor de base de datos y los objetos del dominio de nuestra aplicación. Para la creación de estos importantes ficheros, es de gran ayuda el plug-in de Eclipse, Hibernate Tools que, acelera esta tarea al brindar generación de código en algunos momentos durante la confección de estos ficheros. Según la estructura de paquetes utilizada, los ficheros hbm se colocan en el paquete dao.impl.map.

A continuación se muestra un fragmento del fichero ActividadLaboral.hbm.xml, el cual mapea los campos entre la tabla Trabajo de la base de datos y la clase del dominio ActividadLaboral, para ser usada por la aplicación (Ver Figura 3.10).

```
<hibernate-mapping package="vnz.sigep.tratamiento.regimenvida.domain">
    <class name="ActividadLaboral" table="TRABAJO">
        <id name="id" type="string">
            <column name="ID_TRABAJO" length="20" />
            <generator
                class="vnz.sigep.common.global.util.dataaccess.id.PrefixPersistentGenerator">
                <param name="sequence">SEQ_ESTUDIO</param>
                <param name="maxlo">1000</param>
            </generator>
        </id>
        <many-to-one name="individuo"
            class="vnz.sigep.common.global.domain.Individuo" fetch="select">
            <column name="ID_INDIVIDUO" length="20" />
        </many-to-one>
        <property name="lugar" type="string">
            <column name="LUGAR" not-null="true" />
        </property>
        <property name="direccion" type="string">
            <column name="DIRECCION" not-null="true"/>
        </property>
        <property name="cargoOficio" type="string">
            <column name="CARGO_OFICIO" length="100" not-null="true" />
        </property>
        <property name="remuneracion" type="string">
            <column name="REMUNERACION" length="100" />
        </property>
        <property name="nombreSupervisor" type="string">
            <column name="NOMBRE_COMPLETO_SUPERVISOR" length="100" />
        </property>
        <property name="conoceEstadoLegal" type="boolean">
            <column name="CONOCE_ESTADO_LEGAL" precision="10" scale="0" />
        </property>
```

```
        <property name="horario" type="string">
            <column name="HORARIO" length="100" not-null="true"/>
        </property>
        <property name="fecharInicio" type="date">
            <column name="FECHA_INICIO" length="7" not-null="true" />
        </property>
        <property name="fechaFin" type="date">
            <column name="FECHA_FIN" length="7" />
        </property>
        <property name="motivoCambioAbandono" type="string">
            <column name="MOTIVO_CAMBIO_ABANDONO" length="300" />
        </property>

        <idbag name="telefonos" table="TELEFONO_TRABAJO">
            <collection-id column="ID_TELEFONO" type="string">
                <generator
                    class="vnz.sigep.common.global.util.dataaccess.id.PrefixPersistentGenerator">
                    <param name="sequence">SEQ_TELEFONO_TRABAJO</param>
                    <param name="maxlo">1000</param>
                </generator>
            </collection-id>
            <key>
                <column name="ID_TRABAJO" length="20" />
            </key>
            <element type="string" column="NUMERO" length="20" not-null="true" />
        </idbag>
    </class>
</hibernate-mapping>
```

*Figura 3.10: Configuración del fichero ActividadLaboral.hbm.xml*

En el código anterior se observan etiquetas y algunos de los atributos que estas contienen. El atributo package nos dice cuál es el paquete donde se encuentra la clase que va a ser mapeada. En la etiqueta class se especifican los nombres de la clase y de la tabla de la base de datos. Dentro de la etiqueta id se encuentra el nombre del atributo de la clase que va a almacenar el identificador, el tipo de dato, la columna de la tabla con la cual se hace corresponder y el generador del identificador (generator), en este caso, es la clase *PrefixPersistentGenerator*. En la etiqueta property se configura un atributo de la clase declarando el nombre del mismo, el tipo de dato y la columna de la tabla con la cual va a ser mapeado. Por cada atributo hay que generar una etiqueta property. La etiqueta many-to-one significa que hay una relación de uno a muchos, donde se especifica el nombre del atributo, la clase de este atributo (el tipo de dato) y la columna de la tabla con la cual se mapea.

Las implementaciones de los DAO se guardan en la carpeta dao.impl, ejemplo de estas implementaciones, se muestran en el Anexo 3.

Después de implementar los DAOs el siguiente paso es configurar el fichero de Spring, sigep-tratamiento-regimenvida-dataaccess-context.xml. Esta configuración permite crear el objeto DAO en el

contexto de Spring para luego ser inyectado a los objetos del negocio a través de la inyección de dependencia de Spring, además de inyectar a los objetos de acceso a datos el sessionFactory de Hibernate, que es el objeto que configura al framework Hibernate a partir de los ficheros de mapeo (hbm.xml) y el dialecto a utilizar, según el gestor de base de datos.

En la Figura 3.11 se muestra un ejemplo de la configuración de un DAO en este fichero:

```
<bean id="actividadLaboralDAO" class="vnz.sigep.tratamiento.regimenvida.dao.impl.ActividadLaboralDAOImpl">
    <property name="sessionFactory">
        <ref bean="sessionFactory" />
    </property>
</bean>
```

*Figura 3.11: Configuración de la clase ActividadLaboralDAO en el fichero sigep-tratamiento-regimenvida-dataaccess-context.xml*

### 3.4.2 Implementación de la capa de negocio

Durante esta actividad se implementan en los objetos de negocio los métodos definidos para cada una de las interfaces de los managers ajustándose a las funcionalidades previstas.

Cada método al ser implementado debe de verificar la integridad de los datos de entrada y de salida, e informar a la capa de interfaz de usuario de cualquier eventualidad a través de las excepciones definidas, las cuales serán capturadas por la capa de presentación y mostradas al usuario de una forma adecuada.

La fachada y los managers implementados se configuran en el fichero de configuración del contexto de Spring sigep-tratamiento-regimenvida-business-context.xml. En este fichero se inician los objetos del negocio en el contexto de Spring, se inyectan a cada uno de los managers a través de la inyección de dependencia de Spring los DAOs necesarios para realizar sus funcionalidades, así como la inyección a la fachada de todos los manager que debe conocer para exponer todas las funcionalidades del módulo.

En las siguientes figuras se muestran ejemplos de la configuración de la fachada y del manager regimenVidaManager en este fichero.

```
<bean id="regimenVidaFacade"
    class="vnz.sigep.tratamiento.regimenvida.facade.impl.RegimenVidaFacadeImpl">
    <property name="regimenVidaManager">
        <ref bean="regimenVidaManager" />
    </property>
    <property name="casoManager">
        <ref bean="casoManager"/>
    </property>
</bean>
```

*Figura 3.12: Configuración de la fachada*

```
<bean id="regimenVidaManager"
    class="vnz.sigep.tratamiento.regimenvida.manager.impl.RegimenVidaManagerImpl">
    <property name="actividadLaboralDAO">
        <ref bean="actividadLaboralDAO" />
    </property>
    <property name="estudioDAO">
        <ref bean="estudioDAO" />
    </property>
    <property name="individuoManager">
        <ref bean="individuoManager"/>
    </property>
    <property name="salidasEntradasService">
        <bean class="cu.uci.base.beanresolver.ExternalBean">
            <property name="beanName" value="salidasentradasService" />
            <property name="serviceInterface"
                value="vnz.sigep.common.global.service.SalidasEntradasService" />
        </bean>
    </property>
</bean>
```

*Figura 3.13: Configuración del manager regimenVidaManager*

### 3.4.2.1 Transacciones a Nivel de Negocio

Las transacciones son un elemento de gran importancia durante la implementación de la capa de negocio, de ellas depende que los datos persistentes se mantengan en un estado consistente. Cada método definido en las interfaces de la capa de negocio generalmente se debe ejecutar como una transacción para garantizar la consistencia de la Información en la base de datos.

Para lograr esta tarea se usan las facilidades que brinda Spring para el manejo de funcionalidades como transacciones. Spring, para lograr la ejecución transaccional de las funcionalidades implementadas en los objetos de negocio, se integra con el framework Hibernate y utiliza el soporte para transacciones que brinda esta tecnología de persistencia.

Las declaraciones de las transacciones se realizan en ficheros XML, de forma tal que no es necesario escribir ningún código asociado al manejo de transacciones en los métodos que contienen la lógica de negocio. El framework Spring define un conjunto de etiquetas para manejar las transacciones mediante aspectos entre las que se encuentra:

- <aop:advisor> Contiene dos propiedades fundamentales *pointcut* (una expresión regular que indica el punto de corte donde se aplicará el aspecto, generalmente uno o varios métodos) y *advice-ref* (se declara cómo se va a aplicar la transacción).

A continuación se muestra un ejemplo donde se aplica lo antes explicado:

```
<aop:config>
    <aop:advisor pointcut="execution(* *..*RegimenVidaManagerImpl.*(..))"
        advice-ref="regimenVidaImplMgrAdvice" />

</aop:config>
<tx:advice id="regimenVidaImplMgrAdvice">
    <tx:attributes>
        <tx:method name="*" rollback-for="java.lang.Exception" />
    </tx:attributes>
</tx:advice>
```

*Figura 3.14: Aplicación de transacciones al objeto RegimenVidaManagerImpl*

La expresión *execution* significa: cuando el método sea ejecutado. La expresión entre paréntesis representa los métodos a los que se aplicará la transacción, en este caso: * *..* *RegimenVidaManagerImpl.*(..)*. El primer asterisco significa cualquier tipo de retorno; la expresión *..* *RegimenVidaManagerImpl* significa cualquier clase cuyo nombre termine en RegimenVidaManagerImpl y la expresión *.*(..)* significa cualquier método con parámetros cualesquiera. Con el atributo name se indica el nombre del método al cual se le aplicará rollback al ser lanzada una determinada excepción y con el atributo rollback-for se definen las excepciones que, en caso de ser lanzadas, se desea que fallen las transacciones, en este caso específico a cualquier excepción lanzada fallará la transacción.

### 3.4.3 Implementación de la capa de presentación

La implementación de la capa de presentación consta de dos momentos significativos: la implementación de los controladores y la construcción de la interfaz de usuario asociada a cada petición, junto con los ficheros JavaScripts.

### 3.4.3.1 Implementación de los controladores

La implementación de los controladores es de gran importancia porque manejan todo el flujo web de la aplicación. Los controladores implementan directa o indirectamente la interfaz *org.springframework.web.servlet.mvc.Controller* para insertarse en el flujo de Spring-MVC. Estos componentes son los encargados de la comunicación con la capa de negocio a través de su fachada. Son responsables de validar los datos de entrada de la aplicación, formatear los datos de salida y gestionar el flujo web mostrando las vistas correspondientes. Se configuran en el fichero del contexto de Spring correspondiente a la capa de presentación: *sigep-tratamiento-regimenvida-servlet.xml.*

### 3.4.3.2 Implementación de las páginas Java Server Page (JSP)

Las páginas JSP construyen los documentos HTML que serán las páginas clientes que son mostradas al usuario. Por lo general cada página JSP construye una página cliente o, en ocasiones se juntan un número de ellas para formar una página cliente. Durante la implementación de las páginas JSP, se usan la tecnología JSTL (Java Standard Tag Library), conjunto de librerías de etiquetas simples y estándares muy útil a la hora de realizar ciertas operaciones sobre los datos que son enviados junto con la página JSP desde el controlador y las etiquetas de Spring. Para el diseño gráfico de las JSP usadas se utilizan las hojas de estilo en cascada (Cascading Style Sheets, CSS) definidas para cada uno de los componentes dentro de SIGEP.

### 3.4.3.3 Implementación de la lógica en el cliente

En esta actividad se realiza la implementación de la lógica en el cliente; para esta actividad se usa el lenguaje JavaScript. Es durante esta implementación donde se realizan las validaciones del lado del cliente que sean necesarias y se programa el comportamiento de los componentes gráficos que se utilizan dentro de las páginas JSP, tales como: botones, calendarios, pantallas emergentes de error o información al usuario, tablas entre otros. Los componentes gráficos usados pertenecen a la biblioteca JavaScript DojoToolkit. Además se usa la tecnología AJAX, la cual mantiene la comunicación asíncrona con el servidor, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página, esto permite realizar cambios sobre las páginas sin necesidad de recargarlas, y la tecnología JSON-RPC para el envío de peticiones al servidor de una forma mucho más rápida.

### 3.5 Implementación de la funcionalidad Gestionar Régimen de vida laboral

Para acceder a la gestión del Régimen de vida laboral del residente se selecciona la pestaña de "Régimen de vida" de un expediente, luego se selecciona la pestaña "Laboral" donde se muestra la interfaz de usuario en la cual se observa un resumen de todos los lugares donde ha trabajado o trabaja hasta el momento el individuo. La información que se registra de cada dato laboral (entiéndase por dato laboral a todo lo relacionado con un lugar donde trabajó o trabaja el individuo) es: lugar de trabajo, dirección, teléfonos, fecha en que comenzó a trabajar en el lugar y fecha en que concluyó, si hubiera concluido. La tabla muestra los resultados comenzando por aquellos cuya fecha de finalización no se ha registrado y que representan los centros de trabajo del individuo en el momento. Estos además aparecerán ordenados cronológicamente.

Las peticiones serán atendidas por los controladores ActividadLaboralTableController encargado de proveer los datos de la tabla y RegimenVidaMultiActionController. Este último controlador hereda de la clase de Spring-MVC org.springframework.web.servlet.mvc.MultiActionController que, extiende la capacidad para atender múltiples peticiones; en otras palabras equivale a implementar el flujo de varios controller en un solo controlador a través de un fácil mapeo en el fichero servlet.xml de Spring, en el cual se mapea el nombre de la petición con el controlador que la va a atender y el nombre del método del controlador que atenderá esa petición en específico.

En la Figura 3.15 se muestra la configuración del controlador RegimenVidaMultiActionController dentro del fichero servlet.xml de Spring, la configuración de la facade que compone este controlador, la cual será inyectada al controlador a través de la inyección de dependencia de Spring y las peticiones que este atiende, así como los métodos específicos para cada petición.

```
<bean id="regimenVidaMultiActionController"
    class="vnz.sigep.tratamiento.regimenvida.web.RegimenVidaMultiActionController">
    <property name="regimenVidaFacade">
        <ref bean="regimenVidaFacade" />
    </property>
    <property name="methodNameResolver">
        <bean id="methodNameResolver"
            class="org.springframework.web.servlet.mvc.multiaction.PropertiesMethodNameResolver">
            <property name="mappings">
                <props>

                    <prop key="/oficiosprofesiones/gestionarProfesionesOficios.htm">
                        gestionarOficiosProfesiones
                    </prop>

                    <prop key="/oficiosprofesiones/eliminarProfesionUOficio.htm">
                        eliminarProfesion
                    </prop>
                    <prop key="/oficiosprofesiones/persistirProfesionUOficio.htm">
                        persistirProfesion
                    </prop>

                    <prop key="/actividadlaboral/gestionarActividadLaboral.htm">
                        gestionarRegimenVidaLaboral
                    </prop>

                    <prop key="/actividadeducativa/gestionarActividadEducativa.htm">
                        gestionarRegimenVidaEducativo
                    </prop>
                    <prop key="/actividadeducativa/mostrarDetallesActividadEducativa.htm">
                        mostrarDetallesActividadEducativa
                    </prop>
                    <prop key="/actividadlaboral/detallesActividadLaboral.htm">
                        mostrarDetallesActividadLaboral
                    </prop>
                    <prop key="/actividadlaboral/eliminarActividadLaboral.htm">
                        eliminarActividadLaboral
                    </prop>
                    <prop key="/actividadeducativa/eliminarActividadEducativa.htm">
                        eliminarActividadEducativa
                    </prop>
                    <prop key="/entradastardias/consultarEntradasTardias.htm">
                        consultarEntradasTardias
                    </prop>
                </props>
            </property>
        </bean>
```

*Figura 3.15: Configuración del controlador RegimenVidaMultiActionController*

En la Figura 3.16 se muestra el código donde el controlador devuelve el ModelAndView necesario para construir la interfaz de usuario de gestionar Régimen de vida laboral y en la Figura 3.17 se muestra la interfaz de usuario.

```
public ModelAndView gestionarRegimenVidaLaboral(HttpServletRequest request,
        HttpServletResponse response)
    throws Exception (String id = request.getParameter("id");

    return new ModelAndView("gestionarRegimenLaboral", "id", id);
}
```

*Figura 3.16: Método ModelView del controlador RegimenVidaMultiActionController*


*Figura 3.17: Gestionar Régimen de vida laboral*

## 3.6 Pruebas

**Las Pruebas:** es una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, estos resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente (12).

Esta prueba se desarrolló con los siguientes objetivos: ejecutar pruebas de software bajo condiciones reales, probar el rendimiento de la aplicación.

### 3.6.1 Prueba de Software

A continuación se muestran algunas definiciones de prueba de software:

- Es un elemento crítico para la garantía de la calidad del SW y representa una revisión final de las especificaciones del diseño y de la codificación.
- Es ejecutar el software con determinados datos de entrada y producir resultados que luego serán comparados con los teóricos.

- Es el proceso de ejecución de un programa con la intención de descubrir errores previos a la entrega al usuario final (12).

### 3.6.2 Herramientas para las pruebas

#### 3.6.2.1 JUnit 3.8.1

El JUnit es una herramienta que se utiliza para medir el rendimiento del sistema. La misma, integrada al IDE Eclipse en forma de plug-in, permite que la generación de las plantillas necesarias para la creación de las pruebas de una clase Java se realice de manera automática, lo que facilita al programador, mejor enfoque en la prueba y el resultado esperado, dejando a la herramienta, la creación de las clases que permiten coordinar las pruebas. Por estas ventajas es que se utiliza dicha herramienta para realizar las pruebas unitarias a nivel de desarrollador, teniendo en cuenta, que las mismas deben ser automatizables, completas, reutilizables e independientes.

#### 3.6.2.2 JMeter 2.3.4

**JMeter** es una herramienta de software libre que ofrece la posibilidad de realizar pruebas de carga sobre diferentes aspectos de una aplicación desarrollada en Java. Utilizarla en aplicaciones web para la comprobación de los recursos del sistema, supone una mayor efectividad en el proceso y en la fiabilidad de los resultados. Como herramienta de prueba dispone de varios componentes que facilitan la elaboración de los escenarios de prueba con la ventaja de simular para cada uno de esos escenarios miles de usuarios. De esta manera se verifica el rendimiento del sistema mediante las pruebas de Carga y Estrés y los resultados pueden ser visualizados en diferentes formatos lo cual facilita las labores de análisis para el tester.

Una ventaja de **JMeter** es que permite realizar pruebas de carga sobre cada una de las capas que conforman la aplicación a probar. De esta manera, en un momento dado, es sencillo localizar el foco que está generando contención y está afectando los tiempos de respuesta de un caso de uso específico. (13)

### 3.6.3 Niveles de Pruebas

Cuando se evalúa dinámicamente un sistema software se debe comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el software en su totalidad. Las pruebas se aplican durante todo el ciclo de desarrollo del software para diferentes objetivos y en diferentes escenarios o niveles de trabajo, dentro de estos niveles se encuentran:

**Prueba de Desarrollador**. Es la prueba diseñada e implementada por el equipo de desarrollo. Tradicionalmente estas pruebas han sido consideradas solo para las pruebas de unidad, aunque en la actualidad en algunos casos pueden ejecutar pruebas de integración. Se recomienda que estas pruebas cubran más que las pruebas de unidad.

**Prueba de unidad:** Es la prueba enfocada a los elementos testeables (algo que se pueda probar) más pequeño del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. Mediante esta prueba solo una unidad es probada como tal, como una clase, un paquete de servicio o un subsistema. La prueba de unidad siempre está orientada a caja blanca y son realizadas solamente después de que el programador considera que el componente está libre de errores.

**Prueba de Sistema:** Son las pruebas que se hacen cuando el software está funcionando como un todo. Es la actividad de prueba dirigida a verificar el programa final, después que todos los componentes de software y hardware han sido integrados. Verifica el sistema completo o su aplicación como tal, examinando qué tan bien el sistema cumple con los requerimientos de la organización y su utilidad, seguridad y desempeño. También se realizan estas pruebas a la documentación del sistema.

### 3.6.4 Técnicas de Pruebas

Estas técnicas facilitan una guía sistemática para diseñar pruebas que comprueben la lógica interna de los componentes del software y que verifiquen los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y el rendimiento. Cada nivel de prueba engloba una técnica de prueba específica según los atributos de calidad que se deseen verificar con las pruebas al software.

### 3.6.4.1 Pruebas de Funcionalidad

Dentro de las pruebas de funcionalidad se encuentran:

- **Pruebas funcionales:**

*Objetivo*: Asegurar la funcionalidad apropiada de los módulos, incluyendo el flujo de trabajo, entrada de datos, proceso y recuperación, verificar los requerimientos funcionales, la aceptación de los datos, el proceso, la recuperación y la implementación correcta de las reglas del negocio.

*Caso de prueba*: Un caso de prueba se diseña según las funcionalidades descritas en los casos de usos. Este diseño se elabora previo a la realización de las pruebas funcionales de la aplicación.

Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, para hacer más fructífera la ejecución de las pruebas.

Los casos de prueba pertenecientes al módulo de Régimen de vida del Centro de Residencia Supervisada (CRS) dentro del subsistema Atención, Supervisión y Orientación se pueden encontrar en el expediente del proyecto del SIGEP en su versión 2.1 de la Fase IV. Entre los cuales se encuentra el diseño del caso de prueba del CU Gestionar Régimen de vida Laboral perteneciente al módulo Régimen de vida del CRS (14).

*Método Caja Negra*: En el proceso de pruebas se hace uso de los métodos de prueba de caja negra por ser éste el método utilizado en la liberación del producto software, de ahí el motivo por el que se diseña un caso de prueba por caso de uso del sistema. Los métodos de prueba de caja negra se centran en los requisitos funcionales de software. Además estas pruebas se llevan a cabo sobre la interfaz del software, examina algunos aspectos fundamentales del sistema sin tener mucho en cuenta la estructura lógica interna del software.

En la siguiente tabla se expone un resumen de los resultados de las No Conformidades obtenidos para contabilizar a modo de resumen el total de No Conformidades del módulo Régimen de vida, distinguiendo de ellas cuáles proceden, cuáles no proceden y las resueltas.

| **Módulo Régimen de vida.** | | | |
|---|---|---|---|
| **Total de No Conformidades** | **# de No Conformidades que proceden** | **# de No Conformidades que no proceden** | **# de No Conformidades resueltas** |
| 6 | 6 | 0 | 6 |

*Tabla 3.1 Resumen de las no conformidades del módulo Régimen de vida.*

La prueba realizada de forma general arrojó resultados satisfactorios, a pesar de las no conformidades encontradas. Las inquietudes o dificultades encontradas son cuatro de tipo Solicitud de Cambios y 2 de tipo No Conformidad, las cuales no tenían gran complejidad por lo que fueron resueltas satisfactoriamente.

- **Pruebas de seguridad:**

*Objetivo*: Se realizan para garantizar que el acceso a la información sea accesible únicamente por

las personas autorizadas (Confidencialidad), la fiabilidad de la información (Integridad) y que la información esté disponible cuando se requiera y durante el tiempo que sea necesario (Disponibilidad).
En el caso del Módulo Régimen de vida los permisos de acceso están distribuidos correctamente, y con las pruebas realizadas se obtuvieron resultados satisfactorios.

### 3.6.4.2 Pruebas de Confiabilidad

Se realizan para evaluar el rendimiento y está relacionado también con el control de la detección de errores y de la recuperación para evitar que se produzcan errores. La confiabilidad del software se refiere a la precisión con la que una aplicación proporciona, sin errores, los servicios que se establecieron en las especificaciones originales.

- **Prueba de Stress**

*Objetivo*: Determinar la solidez de la aplicación en los momentos de carga extrema.

Esta prueba se utiliza normalmente para hacer colapsar la aplicación. Se va doblando el número de usuarios que se agregan a la aplicación y se ejecuta una prueba de carga hasta que se colapsa.

*Resumen*: Para las pruebas de Stress se utilizó la herramienta JMeter, pero no se pudo llegar al final de las pruebas, o sea a colapsar el sistema, y hubo que parar los hilos lanzados debido a que la máquina local donde se realizaron dichas pruebas no soporta la creación de tantas instancias, en este caso, solamente soportó hasta 60 usuarios con un tiempo de 180 segundos, conectados simultáneamente en el CRS trabajando en el módulo. En las pruebas de carga se explican detalladamente estas instancias.

### 3.6.4.3 Pruebas de Rendimiento

Las pruebas de rendimiento o desempeño como también se conoce, son las que se realizan, desde una perspectiva, para determinar cuán rápido realiza una tarea un sistema en condiciones particulares de trabajo.

- **Pruebas de carga**

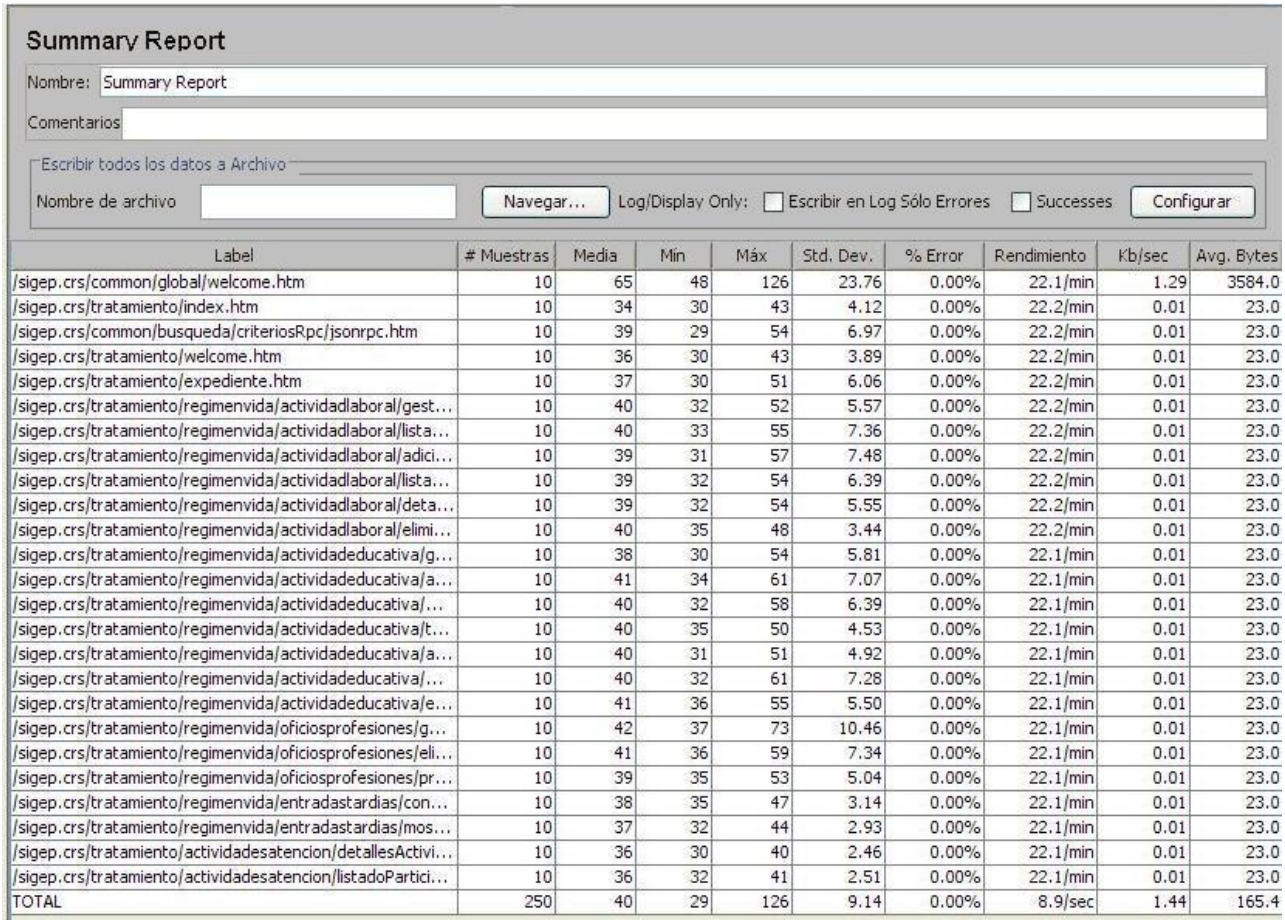
*Objetivo:* Se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperadas.

Esta carga puede ser el número esperado de usuarios concurrentes utilizando la aplicación y que realizan un número específico de transacciones durante el tiempo que dura la carga.

*Resumen:* Las pruebas de carga se realizaron mediante la herramienta JMeter, simulando varias peticiones de un número variable de usuarios.

**Simulación de 10 usuarios con un periodo de subida de 30 segundos**

La siguiente figura muestra que las pruebas se han realizado sin errores. Esto se deduce de la columna representativa del tanto por ciento de errores para cada una de las peticiones asociadas a cada conjunto de muestras. El rendimiento muestra que para una simulación de 10 usuarios junto a un periodo de subida de 30 segundos el servidor es capaz de aceptar una media de 8.9 peticiones por segundo.



Summary Report

Nombre: Summary Report

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Display Only: Escribir en Log Sólo Errores | Successes | Configurar

| Label | # Muestras | Media | Min | Máx | Std. Dev. | % Error | Rendimiento | Kb/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|
| /sigep.crs/common/global/welcome.htm | 10 | 65 | 48 | 126 | 23.76 | 0.00% | 22.1/min | 1.29 | 3584.0 |
| /sigep.crs/tratamiento/index.htm | 10 | 34 | 30 | 43 | 4.12 | 0.00% | 22.2/min | 0.01 | 23.0 |
| /sigep.crs/common/busqueda/criteriosRpc/jsonrpc.htm | 10 | 39 | 29 | 54 | 6.97 | 0.00% | 22.2/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/welcome.htm | 10 | 36 | 30 | 43 | 3.89 | 0.00% | 22.2/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/expediente.htm | 10 | 37 | 30 | 51 | 6.06 | 0.00% | 22.2/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadlaboral/gest... | 10 | 40 | 32 | 52 | 5.57 | 0.00% | 22.2/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadlaboral/lista... | 10 | 40 | 33 | 55 | 7.36 | 0.00% | 22.2/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadlaboral/adici... | 10 | 39 | 31 | 57 | 7.48 | 0.00% | 22.2/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadlaboral/lista... | 10 | 39 | 32 | 54 | 6.39 | 0.00% | 22.2/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadlaboral/deta... | 10 | 39 | 32 | 54 | 5.55 | 0.00% | 22.2/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadlaboral/elimi... | 10 | 40 | 35 | 48 | 3.44 | 0.00% | 22.2/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/g... | 10 | 38 | 30 | 54 | 5.81 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/a... | 10 | 41 | 34 | 61 | 7.07 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/... | 10 | 40 | 32 | 58 | 6.39 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/t... | 10 | 40 | 35 | 50 | 4.53 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/a... | 10 | 40 | 31 | 51 | 4.92 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/... | 10 | 40 | 32 | 61 | 7.28 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/e... | 10 | 41 | 36 | 55 | 5.50 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/oficiosprofesiones/g... | 10 | 42 | 37 | 73 | 10.46 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/oficiosprofesiones/eli... | 10 | 41 | 36 | 59 | 7.34 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/oficiosprofesiones/pr... | 10 | 39 | 35 | 53 | 5.04 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/entradastardias/con... | 10 | 38 | 35 | 47 | 3.14 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/entradastardias/mos... | 10 | 37 | 32 | 44 | 2.93 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/actividadesatencion/detallesActivi... | 10 | 36 | 30 | 40 | 2.46 | 0.00% | 22.1/min | 0.01 | 23.0 |
| /sigep.crs/tratamiento/actividadesatencion/listadoPartici... | 10 | 36 | 32 | 41 | 2.51 | 0.00% | 22.1/min | 0.01 | 23.0 |
| TOTAL | 250 | 40 | 29 | 126 | 9.14 | 0.00% | 8.9/sec | 1.44 | 165.4 |

*Figura 3.18 Simulación de 10 usuarios con un periodo de subida de 30 segundos*

Por cada una de estas filas se muestra la siguiente información:

- **Label**: El nombre de la muestra (conjunto de muestras).
- **# Muestras**: El número de muestras para cada URL.
- **Media**: El tiempo medio transcurrido para un conjunto de resultados.
- **Mín**: El mínimo tiempo transcurrido para las muestras de la URL dada.
- **Máx**: El máximo tiempo transcurrido para las muestras de la URL dada.
- **Error %:** Porcentaje de las peticiones con errores.
- **Rendimiento**: Rendimiento medido en base a peticiones por segundo/minuto/hora.
- **Kb/sec**: Rendimiento medido en Kilobytes por segundo.
- **Avg. Bytes**: Tamaño medio de la respuesta de la muestra medido en bytes.

**Simulación de 60 usuarios con un periodo de subida de 180 segundos**

Realizando la simulación con 60 usuarios considerando un periodo de subida de 180 segundos (de nuevo 3 segundos entre el lanzamiento de cada hilo) los resultados serán los siguientes, teniendo en cuenta que dichos resultando se irán solapando a los ya obtenidos en la simulación anterior.

Summary Report

Nombre: Summary Report

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo | Navegar... | Log/Display Only: Escribir en Log Sólo Errores | Successes | Configurar

| Label | # Muestras | Media | Min | Máx | Std. Dev. | % Error | Rendimiento | Kb/sec | Avg. Bytes |
|---|---|---|---|---|---|---|---|---|---|
| /sigep.crs/common/global/welcome.htm | 70 | 61 | 48 | 169 | 20.21 | 0.00% | 11.4/min | 0.66 | 3584.0 |
| /sigep.crs/tratamiento/index.htm | 70 | 36 | 26 | 59 | 6.72 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/common/busqueda/criteriosRpc/jsonrpc.htm | 70 | 37 | 28 | 68 | 6.65 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/welcome.htm | 70 | 36 | 26 | 67 | 7.31 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/expediente.htm | 70 | 36 | 26 | 65 | 6.55 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadlaboral/gest... | 70 | 39 | 30 | 90 | 8.48 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadlaboral/lista... | 70 | 39 | 30 | 65 | 7.35 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadlaboral/adici... | 70 | 38 | 28 | 67 | 6.88 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadlaboral/lista... | 70 | 38 | 29 | 58 | 5.84 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadlaboral/deta... | 70 | 38 | 29 | 59 | 5.41 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadlaboral/elimi... | 70 | 39 | 29 | 59 | 5.39 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/g... | 70 | 38 | 29 | 55 | 5.77 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/a... | 70 | 40 | 31 | 66 | 7.16 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/... | 70 | 39 | 29 | 63 | 7.51 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/t... | 70 | 39 | 29 | 59 | 5.64 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/a... | 70 | 39 | 31 | 57 | 5.75 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/... | 70 | 39 | 30 | 61 | 6.61 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/actividadeducativa/e... | 70 | 39 | 30 | 57 | 6.03 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/oficiosprofesiones/g... | 70 | 39 | 31 | 73 | 7.02 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/oficiosprofesiones/eli... | 70 | 39 | 30 | 62 | 7.09 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/oficiosprofesiones/pr... | 70 | 38 | 29 | 67 | 6.71 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/entradastardias/con... | 70 | 39 | 30 | 67 | 6.45 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/regimenvida/entradastardias/mos... | 70 | 39 | 30 | 60 | 6.45 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/actividadesatencion/detallesActivi... | 70 | 37 | 28 | 52 | 4.98 | 0.00% | 11.4/min | 0.00 | 23.0 |
| /sigep.crs/tratamiento/actividadesatencion/listadoPartici... | 70 | 37 | 28 | 65 | 5.91 | 0.00% | 11.4/min | 0.00 | 23.0 |
| TOTAL | 1750 | 39 | 26 | 169 | 8.83 | 0.00% | 4.7/sec | 0.76 | 165.4 |

*Figura 3.19 Simulación de 60 usuarios con un periodo de subida de 180 segundos*

En este caso se vuelve a observar que las pruebas se han realizado sin errores. El rendimiento muestra que para una simulación de 60 usuarios junto a un periodo de subida de 180 segundos el servidor es capaz de aceptar una media de 4.7 peticiones por minuto.

#### 3.6.4.4 Pruebas Unitarias

*Objetivo:* Aislar cada parte del programa y verificar que dichas partes individuales son correctas. Estas pruebas facilitan el cambio de código para mejorar la estructura y así asegurarse de que los nuevos cambios no han introducido errores. Permiten además la separación de la interfaz y la implementación, simplificar la integración, y acotar los errores de forma tal que puedan ser localizados fácilmente.

### 3.7 Conclusiones parciales

En este capítulo se presentaron los elementos más significativos en la implementación del módulo Régimen de vida. Se siguieron los lineamientos de la arquitectura del sistema y el diseño definido para el cumplimiento de las funcionalidades pactadas con el cliente.

Se describieron los principales tipos de prueba utilizadas. Para realizar las pruebas unitarias se utilizó la herramienta JUnit y para medir el rendimiento del sistema se ejecutaron pruebas de Carga y Estrés con la herramienta JMeter.

# CONCLUSIONES

Se realizó el diseño y la implementación del módulo Régimen de vida del subsistema Atención, Supervisión y Orientación de la versión 2.1 de la solución SIGEP. Se cumplió con la arquitectura del sistema, se utilizaron las herramientas previstas y se siguieron los flujos de trabajo definidos en el proyecto. Todo lo anterior garantizó la integración del módulo a la solución sin comprometer la estabilidad de los módulos ya implementados y de la solución en su conjunto.

Se logró cubrir todas las funcionalidades definidas para el módulo, cumplir con los objetivos trazados para el presente trabajo y para las tareas del proyecto. El módulo se diseñó e implementó y el proceso quedó documentado satisfactoriamente. Se realizaron además pruebas de funcionalidad, pruebas de confiabilidad y las pruebas de rendimiento de las cuales se obtuvieron resultados satisfactorios.

# RECOMENDACIONES

Se recomienda la ejecución de pruebas al módulo con un gran volumen de información, incorporándole datos de otros sistemas penitenciarios y/o en mayor tiempo de prueba donde se examine el rendimiento del mismo e identifiquen situaciones que no se hayan tenido en cuenta, para garantizar la integridad del módulo, ya que el sistema sólo ha sido probado para una cantidad limitada de datos.

# REFERENCIAS BIBLIOGRÁFICAS


1. [En línea] http://www.dnsp.gob.ve/?q=node/31 .
2. Situación penitenciaria venezolana. [En línea] http://www.monografias.com/trabajos37/situacion-penitenciaria/situacion-penitenciaria2.shtml
3. **Ossorio, Manuel.** *Diccionario de Ciencias Jurídicas, politicas y sociales.*
4. Ministerio de Justicia, Derechos Humanos y Cultos. [En línea] http://www.minjusticia-ddhh.gov.ec.
5. Evolución histórica del Sistema Penitenciario. [En línea] http://www.sistemapenitenciario.gob.pa/uploads/static/23.pdf .
6. **JACOBSON, IVAR, BOOCH, GRADY y RUMBAUGH, JAMES.** *El Proceso Unificado de Desarrollo.* Madrid : PEARSON EDUCACIÓN, 2000.
7. **ARIAS ORIZONDO, ARTURO CÉSAR.** *Visión del sistema de gestión penitenciaria.* CH : ALBET.
8. **WALLS, CRAIG y BREINDENBACH, RAYN.** *Spring in Action.* Greenwich : MANNING, 2008.
9. **PIMENTEL GONZÁLEZ, LUIS ALBERTO.** *Documento de Arquitectura de Software.* Ciudad de La Habana : ALBET, 2007.
10. **PÉREZ RIVERO, IÓSEV y PIMENTEL GONZALEZ, LUIS ALBERTO.** *ArBaWeB: Arquitectura Base sobre la web.* Ciudad de La Habana : UCI, 2007.
11. **Benavides Zaila, Yadira y Gómez Correa, Juan Carlos.** *Diseño e implementación de los módulos Decisiones y Egresos.* 2008.
12. **Pressman, Roger S.** *Ingenieria de SW un enfoque práctico.* s.l. : 5ta edicion.
13. Mejoramiento del Proceso de Pruebas y Corrección de Defectos de Software en un ambiente globalizado. [En línea] [Citado el: 5 de Abril de 2011.] http://chie.uniandes.edu.co/~gsd/index.php?option=com_content&task=view&id=129&Itemid=183
14. *Diseño de Caso de prueba .Caso de Prueba Gestionar Régimen de vida Laboral perteneciente al Módulo Régimen de vida del subsistema Atención, Supervisión y Orientación.* s.l. : expediente de Proyecto del Sistema de Gestión Penitenciaria (SIGEP).

# BIBLIOGRAFÍA


1. Libro: UML y Patrones, Introducción al análisis y diseño orientado a objetos, autor Craig Larman.
2. Jacobson, G. Booch y J. Rumbaugh, El Proceso Unificado de Desarrollo de Software. S.l.: PEARSON EDUCACIÓN S.A, 2000.
3. Estilos Arquitectónicos[En Línea] Disponible en: http://www.willydev.net/descargas/prev/IntroArq.pdf
4. Visual Paradigm 2005 [En línea] Disponible en: http://www.visual-paradigm.com/product/vpuml/
5. BEATON W. Eclipse Platform Technical Overview [En Línea] Disponible en: http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html
6. Framework Dojo [En línea] Disponible en: http://www.dojotoolkit.org/
7. hibernate.org [En línea] Disponible en: Disponible en: http://www.hibernate.org/
8. Conferencia # 5 Pruebas. Ingeniería del Software II. Disponible en: http://teleformacion.uci.cu
9. JMeter Manual de usuario. [En línea] Disponible en: http://www.google.com/url?q=http://www.ejie.net/documentos/Herramientas/JMeter.%2520Manual%2520de%2520usuario%2520v1.1.doc&sa=U&ei=wUTQTeDWIsHYgAeC6dmxDA&ved=0CA0QFjAA&usg=AFQjCNGs6UOCCLMmQSuMp_v60Y7WtN-_CQ

# GLOSARIO

## A

**Arquitectura:** Conjunto de patrones y abstracciones coherentes que guían la construcción del software informático.

## F

**Frameworks:** Conjunto de APIs y herramientas destinadas a la construcción de un determinado tipo de aplicaciones de manera generalista.

## H

**HQL:** Hibernate Query Language (Lenguaje de Consulta de Hibernate).

## I

**IDE:** Integrated Development Enviroment (Ambiente Integrado de Desarrollo).

## M

**Mapear:** Técnica de programación para hacer coincidir o convertir datos entre distintos sistemas (Ejemplo Object-Relational Mapping).

**Método:** Operación que pueden modificar el estado de un objeto u obtener datos sobre el mismo.

## O

**Objeto:** Instancia de una clase encargada de realizar acciones en la ejecución de un programa.

## P

**Persistir:** Almacenar en la base de datos la información referente a algunos objetos.

**Plataforma:** Tecnología básica del software de un ordenador que define como funciona y que otro tipo de software se puede emplear con él.

## S

**SIGEP:** Sistema de Gestión Penitenciaria.