CIS0910IS01

Aplicación web para el cálculo de reservas o pasivos pensionales

Francisco Adolfo Mora Calderón
Edgar Adrian Otálora Neira

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA DE SISTEMAS
BOGOTÁ, D.C.
2010

CIS0910IS01
Aplicación web para el cálculo de reservas o pasivos pensionales

Autor(es):

Francisco Adolfo Mora Calderón
Edgar Adrian Otálora Neira

MEMORIA DEL TRABAJO DE GRADO REALIZADO PARA CUMPLIR UNO DE LOS REQUISITOS PARA OPTAR AL TITULO DE INGENIERO DE SISTEMAS

Director

María Consuelo Franky de Toro

Jurados del Trabajo de Grado

Fabio Antonio Avellaneda Pachón

Alvaro Javier Infante Sepulveda

Página web del Trabajo de Grado

http://pegasus.javeriana.edu.co/~CIS0910IS01/

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA DE SISTEMAS
BOGOTÁ, D.C.
Diciembre, 2010

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERIA
CARRERA DE INGENIERIA DE SISTEMAS

Rector Magnífico

Joaquín Emilio Sánchez García S.J.

Decano Académico Facultad de Ingeniería

Ingeniero Francisco Javier Rebolledo Muñoz

Decano del Medio Universitario Facultad de Ingeniería

Padre Sergio Bernal Restrepo S.J.

Directora de la Carrera de Ingeniería de Sistemas

Ingeniero Luis Carlos Díaz Chaparro

Director Departamento de Ingeniería de Sistemas

Ingeniero César Julio Bustacara Medina

Artículo 23 de la Resolución No. 1 de Junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

AGRADECIMIENTOS

A nuestros familiares y personas cercanas que nos han apoyado a lo largo de la carrera y durante el desarrollo del presente trabajo de grado. A nuestra directora de tesis María Consuelo Franky, por guiarnos y corregir nuestro trabajo.

Contenido

Lista de Ilustraciones

ABSTRACT

The purpose of this document is to present the development process of a Web application that provides a service for calculation of the pension's reserves that private and public entities, insurance companies and pension funds must have in order to pay their employees or affiliates future obligations. This tool was developed by using an application server and the Java EE 5 technology. Open source reporting tools, Rule engines and xml descriptors were also used in order to improve the development process and the quality of the application.

RESUMEN

Este trabajo presenta el proceso de desarrollo de una aplicación web que permite a las compañías aseguradoras, fondos de pensiones y entidades privadas realizar de una manera sencilla y ágil el cálculo de sus obligaciones pensionales. Esta herramienta fue desarrollada mediante el uso de un servidor de aplicaciones junto con la tecnología Java EE 5. Adicionalmente fueron utilizadas herramientas libres de generación de reportes, motores de reglas y descriptores XML para validaciones de archivos de entrada.

RESUMEN EJECUTIVO

En Colombia antes de 1992 se presentaba un grave problema en el tema pensional debido principalmente a la inequidad, a la baja cobertura, y a los múltiples regímenes existentes del sistema pensional en ese momento. Por esta razón se propuso una reforma total al sistema pensional que a su vez terminó extendiéndose para reformar también el sistema de salud. De esta manera surgió la aprobación de la ley 100 de 1993 donde se lograron grandes avances en la financiación y administración, así como en la concepción del sistema pensional [53].

A partir de esta reforma las entidades de seguridad social como las asociaciones de fondos de pensiones o el Instituto del Seguro Social, así como las compañías de seguros y las entidades que manejan regímenes de pensiones especiales como Ecopetrol, están en la obligación de calcular el valor del pasivo pensional de los afiliados que tienen a su cargo para asegurar que cuentan con el capital disponible para el pago de las pensiones. El cálculo del pasivo pensional es para el empleador una evaluación económica, legal y práctica de la situación individual de todos los trabajadores con derechos pensionales a cargo de la empresa [68].

Actualmente en Colombia existen entidades que no tienen recursos para adquirir herramientas tecnológicas adecuadas que realicen estos cálculos actuariales por lo que desconocen el monto de sus obligaciones pensionales futuras; otras entidades deben recurrir a asesorías externas para efectuar las valoraciones de sus pasivos pensionales.

A diferencia de las aplicaciones existentes actualmente en el tema de pensiones, el propósito de este proyecto de grado es el desarrollo de una aplicación en un ambiente Web que permita a una empresa cuantificar el valor actual de las obligaciones pensionales de jubilación, invalidez y sobrevivencia, de acuerdo al Sistema General de Pensiones en Colombia de conformidad con los parámetros que trata la ley 100 sancionada en 1993 y modificada por la ley 797 sancionada en el año 2003.

Previo a la construcción de la aplicación se realizó un levantamiento de requerimientos donde se evaluaron las necesidades del cliente representado por una empresa que requiere calcular sus reservas pensionales, y a partir de estas necesidades, y de reuniones hechas con el cliente

y con la directora del trabajo de grado se llegó a un acuerdo en todos los requerimientos que fueron propuestos.

Posteriormente con base en los requerimientos se definieron cada uno de los módulos que tiene la aplicación, los cuales son; Alimentar Sistema, Cálculos del Sistema, Reportes y Seguridad; cada uno de estos módulos están compuestos por varios casos de uso y con base a estos casos de uso fueron definidas herramientas adicionales que se utilizaron en el proceso de desarrollo.

El **módulo de seguridad** presta todos los servicios a los usuarios que controlan los permisos del ingreso a la aplicación y donde se define a qué opciones del sistema tiene acceso un determinado usuario de acuerdo con su perfil de seguridad. Adicionalmente presta un servicio de auditoría en el cual se registra la actividad de los usuarios sobre la base de datos cuando se altera la información que contiene.

El **módulo Alimentar Sistema** se ocupa de realizar el procedimiento de cargar archivos al sistema con información de los empleados, tablas de esperanza de vida, salarios mínimos y parámetros, las cuales son necesarias para el cálculo del pasivo pensional. En este módulo se utilizaron descriptores XML los cuales sirven para validar el contenido de los archivos que son subidos al servidor de aplicaciones que se utiliza. En algunos casos también se utilizó un motor de reglas para realizar validaciones de negocio.

Para el desarrollo del **módulo cálculos del sistema** fue necesario utilizar intensamente un motor de reglas debido a que permite que el cliente pueda entender y expresar los cálculos y validaciones que se hacen de una manera sencilla. También permite desacoplar la lógica de negocio de los cálculos que obtienen las reservas. Se utilizó el motor de reglas *Drools* que es de libre distribución [72]. Las características de este motor de reglas se describen en la sección 3.3 de este documento. Los cálculos que hace el sistema se ejecutan para cada empleado de las empresas registradas en la aplicación con base en los datos obtenidos de las tablas de parámetros, salarios mínimos, esperanza de vida y empleados ingresadas previamente al sistema.

Por último el **módulo de reportes** le permite al usuario ver el resultado de los cálculos. Los reportes consolidados por empresa muestran los resultados de reservas para una empresa clasificándolos por origen y clase de pensión. Los reportes detallados de los empleados de

una empresa muestran los resultados de las reservas detalladas por empleado clasificadas por origen de pensión. Por último se definieron los reportes consolidados de todas las empresas que muestran los resultados totales de reservas para todas las empresas que se encuentran registradas en el sistema.

Una vez finalizado el proceso de desarrollo se obtuvieron las siguientes conclusiones:

El uso de motores de reglas permite desacoplar la lógica del negocio de la aplicación lo cual beneficia a los clientes ya que les permite realizar modificaciones sin conocer demasiado de programación debido a que las reglas son definidas de forma declarativa. Adicionalmente otra gran ventaja de los motores de reglas es que permiten la modificación de las mismas en tiempo de ejecución.

Por medio del desarrollo del presente proyecto se comprendieron las ventajas ofrecidas por los servidores de aplicaciones en cuanto a escalabilidad, procesamiento distribuido, reducción de costos de mantenimiento entre otras características que permiten que los desarrolladores se vean beneficiados y prefieran elegir este tipo de aplicaciones sobre las que trabajan mediante un modelo cliente servidor.

INTRODUCCIÓN

Según lo dispuesto en el artículo 10 de la ley 100 de 1993, el objeto del subsistema general de pensiones es "… garantizar a la población el amparo contra las contingencias derivadas de la vejez, la invalidez y la muerte mediante el reconocimiento de las pensiones y prestaciones que determina la presente ley, así como propender por la ampliación progresiva de cobertura a los segmentos de población no cubiertos con un sistema de pensiones."[36]

Lo mencionado anteriormente obliga por revisión delegada a la Superintendencia Bancaria, a todas las empresas en funcionamiento y en proceso de liquidación a realizar cálculos que determinen el capital con el que deben contar para cumplir con el pago de las obligaciones pensionales de sus trabajadores. Adicionalmente esta entidad podrá verificar, cuando lo estime conveniente, que el reconocimiento de pensiones, cualquiera que fuere la causa, por parte de las entidades que administren fondos de pensiones, con independencia del régimen, y las entidades aseguradoras de vida, según el caso, se efectúen con sujeción a las disposiciones legales pertinentes, en particular cuando se afecte la garantía estatal de pensión mínima [38].

Por las razones mencionadas anteriormente surgió la idea de realizar este trabajo de grado y la propuesta consiste en el desarrollo de una aplicación web que permita a las compañías con obligaciones pensionales realizar este tipo de cálculos de una manera ágil y eficaz a través de la Web.

Actualmente las aplicaciones que existen de este tipo son limitadas ya que son aplicaciones de escritorio lo cual hace que sea necesario personal de la entidad capacitado para instalar la herramienta en cada uno de los equipos en los cuales se va a utilizar, y se vuelve más complicado si las empresas tienen más de una sucursal en funcionamiento. La solución propuesta en el presente proyecto es construir una aplicación web lo cual tiene como principal ventaja que la información se encuentra centralizada en un servidor y puede ser accedida desde cualquier computador a través de un navegador web sin necesidad de realizar instalaciones adicionales. Cabe anotar que los usuarios que acceden a la aplicación deben contar con los permisos de seguridad necesarios.

Esta aplicación fue desarrollada utilizando la tecnología Java EE 5 que proporciona un conjunto de *frameworks* de infraestructura Java para la construcción de aplicaciones

empresariales entre los cuales se encuentran log4j, Interceptores, Seguridad por roles, anotaciones, Java Server Faces, y un completo modulo dedicado a la persistencia de los datos en este tipo de aplicaciones Web.

Adicionalmente el desarrollo se hará utilizando herramientas libres lo cual representa disminución en los gastos tanto para desarrolladores como para clientes debido a que no es necesario adquirir ningún tipo de licencias.

El objetivo de este trabajo de grado es el de diseñar y construir una aplicación web que brinde soporte a las entidades para calcular las reservas pensionales que deben tener o constituir para el pago de obligaciones pensionales futuras de acuerdo al sistema general de pensiones colombiano. Para el cumplimiento de este objetivo se construyó un producto el cual lleva el nombre "**SisPen**".

Existen actividades adicionales que se realizaron para poder cumplir adecuadamente con el objetivo propuesto. Entre estas actividades se encuentran la investigación y el conocimiento de las normas legales vigentes relacionadas con los pagos de las pensiones y las herramientas tecnológicas necesarias para el desarrollo de este trabajo de grado entre las cuales están *JasperReports* [40], *Drools* [72], Descriptores XML, entre otros. Adicionalmente fue necesario hacer un levantamiento de requerimientos y de casos de uso.

Para la validación del producto final que va a ser entregado al cliente se cuenta con la asesoría de Edgar Otálora Pérez persona que cuenta con más de 10 años de experiencia en el tema de Seguridad Social y cliente principal de la aplicación. Además se harán validaciones parciales con respecto a las herramientas tecnológicas utilizadas con la ayuda de la directora del trabajo de grado, la Ingeniera María Consuelo Franky.

En este documento se describe el proceso detallado de todo lo que fue utilizado para el desarrollo del producto final y se describe el modo en que fueron utilizadas cada una de las herramientas y las razones por las cuales fueron utilizadas. El producto principal lo constituye la aplicación funcional la cual se anexa en un CD con sus respectivos manuales de usuario e instalación entre otros y que además pueden ser descargados de la página web asignada a este trabajo de grado: http://pegasus.javeriana.edu.co/~CIS0910IS01/

Por último se muestran las conclusiones y recomendaciones que se obtuvieron durante el desarrollo de este trabajo de grado y que pueden ser útiles para aquellas personas que tengan algún tipo de interés por este tema. También se presentarán extensiones posibles para este trabajo.

1. Oportunidad, Problemática, Antecedentes

1.1 Descripción del contexto

Los pasivos pensionales son motivo de permanente preocupación en las entidades de seguridad social del orden público o privado que tienen a su cargo el reconocimiento y pago de las pensiones de vejez y jubilación; y de manera especial de los Gobiernos, quienes tienen la obligación de proteger los derechos pensionales de las personas que luego de cumplir con los requisitos exigidos por las normas vigentes o convenciones colectivas, pueden acceder a las pensiones de vejez o de jubilación. Así las cosas, el sistema de seguridad social en Colombia y en varios países del mundo prevé la valoración de los pasivos pensionales [1] o el capital necesario para cubrir las obligaciones futuras derivadas de los pagos vitalicios de las mesadas de la población que ya esta pensionada o de la población con expectativa de pensionarse.

Las reformas pensionales en Colombia, al igual que en el resto de países de América Latina, básicamente han sido motivadas por tres razones: baja cobertura, inequidad e insostenibilidad financiera. Es abundante la literatura que analiza desde diversos enfoques la evolución del sistema pensional colombiano, los cambios introducidos en el modelo y sus implicaciones fiscales y sociales en el tiempo. Las principales conclusiones al respecto indican que en Colombia los estudios técnicos no han sido el motor de tales reformas, en las cuales el tratamiento especial para ciertos grupos ha acentuado los efectos negativos de estas, en detrimento del régimen general [3].

El tema de los pasivos pensionales se maneja a nivel mundial y en el caso colombiano compete a las entidades de seguridad social que administran los regímenes de prima media (Seguro Social) y de ahorro individual con solidaridad (Administradoras de Fondos de Pensiones) señalados en la Ley 100 sancionada en 1993 y en la Ley 797 sancionada en el 2003 [1], así como a las compañías de seguros que comercializan productos de rentas vitalicias, y también a las entidades que administran regímenes especiales de pensiones como Ecopetrol , las Fuerzas Militares, el Magisterio, entre otros. En particular vale la pena señalar que de acuerdo a los parámetros fijados por el Gobierno Nacional mediante la Ley 549 de 1999, las entidades territoriales deberán cubrir el valor de los pasivos pensionales a su cargo,

elaborando previamente el cálculo actuarial de acuerdo a la metodología señalada por el Ministerio de Hacienda y Crédito Público. De la misma manera la valoración de reservas pensionales es fundamental para definir políticas en materia de seguridad social y financiera en aquellas entidades del sector público o privado responsables del pago y la administración de obligaciones pensionales.

Para la valoración de pasivos pensionales se requiere el uso de bases de datos poblacionales y de algoritmos que comprenden formulaciones complejas y cálculos que ya han sido definidos por expertos en cálculo actuarial; de tal forma que se necesitan herramientas de software que sistematicen este proceso, pues de otra forma sería casi imposible de realizar.

Actualmente en Colombia existen entidades que no tienen recursos para adquirir herramientas tecnológicas adecuadas que realicen estos cálculos actuariales por lo que desconocen el monto de sus obligaciones pensionales futuras; otras entidades deben recurrir a asesorías externas para efectuar las valoraciones de sus pasivos pensionales.

Lo que se pretende entonces realizar en este proyecto de tesis es una aplicación web orientada a sistematizar un modelo que permita cuantificar el valor actual de las obligaciones pensiónales de jubilación, invalidez y sobrevivencia, que trata el Sistema General de Pensiones en Colombia para el régimen de prima media con prestación definida, de conformidad con las prestaciones y parámetros que trata la Ley 100 sancionada en 1993 modificada por la Ley 797 sancionada en el año 2003.

1.2 Formulación del problema que se resolvió

Una aplicación web, a diferencia del software de escritorio tradicional cliente/servidor que existe actualmente para hacer estos cálculos, proporciona beneficios como: compatibilidad multiplataforma, está siempre actualizado con la última versión del producto, el acceso es inmediato debido a que no requiere de un proceso de configuración e instalación, existen menos *bugs* debido a que todas las personas que usan el producto trabajan con la misma versión por lo que los defectos se corrigen a medida que se van detectando[4], disminuye el precio debido a que, por ejemplo, se reducen los costos de mantenimiento y actualización de los sitios web, además están disponibles en un ámbito geográfico mundial, a diferencia de las aplicaciones cliente/servidor que están restringidas a un ámbito de red local.

Por otro lado al realizar un producto de software como trabajo de grado se aplican y se ponen en práctica muchos conocimientos adquiridos a lo largo de la carrera para un producto que beneficiaría a alguna(s) entidad(es) que requiera(n) al cálculo de su pasivo pensional conforme a las leyes de seguridad social establecidas en el país. Además nos estamos enfrentando a un reto pues en la actualidad hay software que hace este tipo de cálculos actuariales, por lo tanto esta aplicación debe tener algún campo de innovación que facilite realizar esta tarea y que la diferencie de las demás aplicaciones. De lo mencionado anteriormente surge una pregunta la cual se resolverá durante el desarrollo de este trabajo de grado.

¿Cómo aporta una aplicación web al proceso de cálculo de las reservas pensiónales de las entidades privadas en Colombia que operan de acuerdo con el régimen de prima media?

1.3 Justificación

Debido a la necesidad de las entidades privadas que operan de acuerdo al régimen de prima media, fondos de pensiones y compañías de seguros de calcular el valor de los pasivos pensionales de acuerdo con las normas legales vigentes, y con el objeto de garantizar el cumplimiento del pago de las pensiones de los trabajadores y/o afiliados surge la necesidad de realizar este proceso de una manera ágil y sin verse en la obligación de solicitar asesorías externas a actuarios y personas expertas en el tema.

Por lo tanto surgió la necesidad de crear una aplicación web que agilizara y proporcionara una manera más sencilla de realizar este proceso para los usuarios sin que sea estrictamente necesaria la intervención de expertos para realizar los cálculos.

1.4 Impacto Esperado

Se espera que luego de la aprobación del producto por parte del cliente este pueda ser difundido. El producto puede ser de interés para compañías aseguradoras, compañías de fondos de pensiones, y demás empresas privadas que tengan a su cargo el cálculo del valor de los pasivos pensionales para sus afiliados. Adicionalmente este producto va a ser de gran utilidad para el cliente y para muchos actuarios que son contratados para realizar estos tipos de cálculos para diferentes entidades.

2. Descripción del Proyecto

2.1 Visión global

El motivo principal de la realización de este trabajo de grado es el de agilizar el proceso que realizan las entidades, fondos de pensiones y compañías aseguradoras para el cálculo del pasivo pensional de todos sus afiliados. En el desarrollo de este trabajo se realizó una aplicación de modo que los usuarios de esta pueden: Ingresar archivos con información de la nomina de empleados y retirados de una determinada entidad, o de afiliados a un fondo de pensiones o de una compañía aseguradora. Posteriormente los usuarios pueden obtener reportes con la información de las reservas pensionales que fueron calculadas por el sistema.

2.2 Objetivo general

Diseño y construcción de una aplicación web que brinde soporte a las entidades para calcular las reservas o pasivos pensionales que deben tener o constituir para el pago de obligaciones pensionales futuras de acuerdo al sistema general de pensiones colombiano. Las entidades deben definir el régimen pensional que les aplica.

2.3 Fases Metodológicas o conjunto de objetivos específicos

1. Conocer las normas legales vigentes en Colombia que rigen el sistema prestacional de las pensiones de vejez, jubilación y sobrevivencia.
2. Realizar un levantamiento de información para conocer el proceso que sigue una entidad para calcular su pasivo pensional.
3. Estudio de los *frameworks* de la tecnología Java EE 5 que van a ser usados en el presente proyecto y de las nuevas herramientas de desarrollo con las que no estamos familiarizados.
4. Realizar el análisis, diseño y desarrollo de la aplicación web que permitirá calcular las reservas pensionales de una población definida de la entidad consultada.
5. Realizar la validación de la aplicación web utilizando: la información básica suministrada por una entidad que tenga a su cargo el pago de obligaciones pensionales, los criterios de validación de un actuario y las pruebas funcionales de los resultados con información histórica ya procesada.

2.4 Método que se propuso para satisfacer cada fase metodológica

Las metodologías que fueron aplicadas para cumplir cada una de las metas parciales de este trabajo de grado son:

2.4.1 Documental:

La metodología documental cuenta con varias etapas, algunas de las cuales nos resultan útiles para cumplir con el primer objetivo específico de nuestro proyecto de grado. Las etapas son [5]:

- Obtención de la bibliografía básica sobre el tema: Se investiga en bases de datos, bibliotecas y se consulta con personas conocedoras del tema.
- Lectura rápida del material: Se ubican las principales ideas y se conoce la calidad del material obtenido para hacer una mejor selección de todo lo que se ha encontrado.
- Delimitación del tema: Una vez se ha consultado del tema se puede saber con mayor precisión su dimensión.
- Lectura minuciosa de la bibliografía: etapa que comprende el estudio formal de los documentos hallados en la etapa de obtención de la bibliografía.

2.4.2 Estudio de Casos:

Esta metodología se aplicó para cumplir con el segundo objetivo específico, de esta forma se estudió el proceso que sigue una entidad de la manera más detallada posible teniendo en cuenta su entorno [6], es decir, su carácter (público o privado), el régimen pensional que maneja, el número de empleados, el sector en el que se desenvuelve, entre otras características.

Los pasos generales que fueron seguidos para aplicar esta metodología son [7]:

- Elección del caso (entidad): Se definió el cliente interesado y que se vería beneficiado con el desarrollo de este proyecto.
- Desarrollo del caso: Se realizan las labores necesarias para conocer el proceso seguido para calcular las reservas pensionales programando entrevistas presenciales con el

cliente. En principio se trabajará con información real suministrada por una entidad que realiza este tipo de cálculos.

2.4.3 Analítica:

Se aplicó una metodología analítica para realizar la tercera meta parcial de este trabajo de grado. Al aplicar esta metodología se pudo conocer en detalle las características de los *frameworks* Java EE 5 y las nuevas herramientas de desarrollo con las cuales vamos a interactuar durante el desarrollo del presente proyecto, de tal forma que resulte fácil el uso de las mismas en la etapa de desarrollo de la aplicación. Las principales herramientas que fueron evaluadas fueron Drools [72], JasperReports [40], descriptores XML y creación de directorios Web utilizando el servidor de aplicaciones JBoss.

Se siguieron estas etapas para aplicar la metodología [8]:

- Se adquirió material bibliográfico de los *frameworks* de las nuevas herramientas de desarrollo que fueron usadas a lo largo del proceso de implementación de la aplicación, la mayor parte de esta información fue adquirida a través de la Web debido a que varias de las herramientas utilizadas son proyectos de código abierto por lo cual la documentación está abierta al público general.
- Estudio de los *frameworks* y demás documentación de las nuevas herramientas tecnológicas que van a ser utilizadas para entender su funcionamiento y poder aplicar los conocimientos adquiridos al desarrollo del trabajo de grado.
- Se realizaron ejemplos prácticos de los *frameworks* y de las nuevas herramientas que fueron utilizadas durante el desarrollo de la aplicación. Para probar cada una de estas nuevas herramientas fue necesario instalar cada una de ellas en los equipos de trabajo y utilizar datos de prueba para verificar su correcto funcionamiento.

2.4.4 Programación Extrema:

La programación extrema es una metodología ágil de producción de software. A diferencia de las metodologías tradicionales, la programación extrema tiene las siguientes características [46].

1- Está orientada hacia quien produce y usa el software

2-Reduce el costo del cambio en todas las etapas del ciclo de vida del sistema

3- Combina las que han demostrado ser las mejores prácticas para desarrollar software.

Esta metodología fue utilizada para cumplir el cuarto objetivo del trabajo de grado debido a que fue la que más se ajustó a las necesidades del presente proyecto por las razones que se mencionan a continuación y por el tiempo relativamente corto de duración del proyecto. Esta pone gran énfasis en la adaptabilidad como sucede cuando hay cambios en los requerimientos [9] y el desarrollo del software ya ha comenzado. Las prácticas de la programación extrema que fueron utilizadas durante el desarrollo del proyecto se describen a continuación.

- Rápida Retroalimentación: Pretende generar retroalimentación del cliente que pueda ser interpretada y que permita agregar experiencia en la construcción del sistema tan frecuente como sea posible. Esto permite adicionalmente que los desarrolladores aprendan a probar mejor el sistema así como a diseñar e implementar mejor.
- Simplicidad: Busca simplificar el diseño del software para agilizar el desarrollo del mismo y hacer más fácil su mantenimiento. Siempre se debe procurar escoger el algoritmo, la interfaz, la estructura de datos, el diseño o la estrategia más simple [9].
- Pequeñas Liberaciones: Liberar un sistema simple en producción rápidamente y después generar una nueva versión en un corto periodo de tiempo. El equipo de desarrollo necesita liberar versiones iterativas del sistema a los clientes con frecuencia. Al finalizar cada una de las iteraciones se obtendrá software en producción probado y funcionando correctamente para mostrar al cliente.
- Refactorizar: Esta práctica tiene como principal propósito evitar la utilización de código que ya no es mantenible pero de alguna manera funciona y los desarrolladores temen modificar. Cuando se elimina la redundancia se elimina funcionalidad que no está siendo utilizada y cuando se actualizan diseños obsoletos se está haciendo *"refactoring"* [10].
- Programación por pares: Todo el código de producción está escrito con dos personas trabajando en una sola maquina. Hay dos roles dentro de la pareja.
    - Una persona piensa en la implementación.
    - La otra persona piensa en la viabilidad de la propuesta de la otra persona y en los casos de prueba que pueden fallar realizando la implementación definida, y busca la manera de solucionar los problemas que pueden surgir.

Esta práctica permite adquirir experiencia a los novatos debido a que generalmente trabajan junto a las personas más experimentadas del equipo de trabajo.

- 40 horas semanales: El propósito de esta práctica es producir una implementaciones eficientes ya que si un trabajador trabaja más de 40 horas a la semana es decir no descansa lo suficiente, produce código de mala calidad lo que causa que sea necesario realizar modificaciones futuras lo cual causa retrasos en los proyecto la mayoría de veces.
- Disponibilidad del cliente: Este es el requerimiento más importante de esta metodología debido a que es de vital importancia tener al cliente siempre disponible. Esto se debe a que es necesario que brinde soporte constante al equipo de desarrolladores, además este contacto debe ser presencial preferiblemente. El cliente ayuda a asegurarse que la funcionalidad deseada del sistema está siendo cubierta por los desarrolladores.
- Estándares de codificación: El código debe estar en un formato de manera que se cumplan los estándares impuestos en conjunto por el equipo de trabajo.
- Planeación de liberaciones: Una reunión de planeación de liberaciones se lleva a cabo para generar el plan, el cual abarca el desarrollo de todo el proyecto. Este plan es usado posteriormente para crear planes para cada una de las iteraciones presentes en el desarrollo del proyecto.

3 MARCO TEÓRICO

3.1 Sistema Pensional Colombiano

3.1.1 Descripción

La seguridad social integral representa al conjunto de instituciones, normas y procedimientos, del cual disponen la persona y la comunidad y que proporciona la cobertura inicial de las contingencias, especialmente la capacidad económica de los habitantes del territorio nacional, con el fin de lograr el bienestar individual y la integración de la comunidad [1].

"El sistema General de pensiones en cualquiera de los dos regímenes actuales en Colombia garantiza a sus afiliados y a sus beneficiarios, cuando sea el caso, las siguientes pensiones y/o prestaciones económicas [25]:

a) Pensión de Vejez.

b) Pensión de Invalidez.

c) Pensión de Sobrevivientes.

La pensión de vejez hace referencia al capital al que tiene derecho un afiliado después de cumplir con la edad requerida por la ley. Incluso antes en caso de no cumplir con la edad, si el afiliado cuenta con el monto requerido para hacerlo [26].

La pensión de invalidez es la pensión que pagan las e a los afiliados que queden en condición de invalidez total o parcial sea total o permanente. La condición de invalidez la determina el comité médico de las Aseguradoras de Fondos de Pensiones [26].

La pensión de sobrevivencia es la pensión a la que tienen derecho los beneficiarios de un afiliado fallecido. Los beneficiaros pueden ser: cónyuge, hijos menores de edad o incapacitados, y los padres que sean mayores de 65 anos, o que sean inválidos, o que hayan dependido económicamente del fallecido. [26].

La principal característica del sistema pensional que fue modificado por la ley 100 de 1993 es la existencia paralela de dos regímenes de pensiones: uno basado en beneficios definidos (sistema de reparto, llamado "de prima media") y otro basado en el rendimiento de ahorros en un fondo de capitalización (sistema de ahorro individual) [28].

El régimen de prima media es aquel en el cual los aportes de los afiliados y sus rendimientos constituyen un fondo común que garantiza a los beneficiarios el pago de la respectiva pensión previamente definida en la ley. Este régimen es la base del desarrollo del presente proyecto.

Por otra parte el régimen de ahorro individual hace referencia al "conjunto de entidades, normas y procedimientos, mediante los cuales se administran los recursos privados y públicos destinados a pagar las pensiones y prestaciones que deban reconocerse a sus afiliados"[58]. Este régimen está basado en el ahorro y rendimiento financiero de los pagos que mes a mes realizan los afiliados, y en garantías que ofrecen una pensión mínima y aportes a fondos de solidaridad para los afiliados.

El cálculo actuarial de la reserva matemática para pensiones, estima el valor del capital necesario o reserva requerida para responder por la obligación pensional futura de un grupo de asegurados o afiliados y tiene su fundamento en la formulación de rentas contingentes fraccionadas que considera, entre otros, los parámetros de crecimiento legal de las rentas mensuales, la tasa de interés técnico o tasa real de inversión de las reservas y las tablas de conmutación derivadas de las tablas de mortalidad y de invalidez por género vigentes en Colombia, las cuales tienen implícito el cálculo de las probabilidades de sobrevivencia de una vida (asegurado o afiliado) y de dos vidas (el asegurado o afiliado y el beneficiario de la renta).

Para garantizar el pago de las prestaciones económicas derivadas del otorgamiento de una de las prestaciones antes señaladas, las entidades responsables deben conformar reservas pensionales o determinar los pasivos pensionales, cuyos montos obedecen a un cálculo actuarial en cuanto las mismas dependen de una contingencia relacionada con la ocurrencia de un evento, por ejemplo la esperanza de vida para la pensión de vejez, o la muerte del titular para el caso de la pensión de sobrevivientes.

3.1.2 Proceso de cálculo de las reservas pensionales

Teniendo en cuenta las disposiciones anteriormente descritas es necesario desarrollar la formulación para hacer el cálculo de las reservas correspondientes a los pensionados por IVM (Invalidez, Vejez, Muerte) [29]. Para poder llegar a estas formulaciones que son la base para el desarrollo de la aplicación del presente trabajo de grado se requiere de elementos adicionales entre los cuales se debe tener en cuenta las tablas de mortalidad que consisten en

un cuadro estadístico en el que partiendo de un grupo de personas de edad uniforme, se van contabilizando las muertes ocurridas año tras año y la esperanza de vida de cada edad.

Adicionalmente se requieren tablas de parámetros con el valor del interés y el IPC (Índice de precios al consumidor), así como una tabla con la información de salarios mínimos definidos para un año determinado.

Una vez definidas las diferentes fórmulas y parámetros técnicos y legales a utilizarse para realizar el cálculo de las reservas pensionales se espera que al final del proceso se generen reportes que le permitan al usuario pertinente conocer el valor actual (pasivo pensional o reserva) tanto global como individual que se requiere para cumplir con sus obligaciones pensionales futuras.

La aplicación desarrollada realiza la evaluación de la reserva requerida a un corte determinado para el caso de un grupo de personas que a la fecha de corte estén pensionadas por vejez, invalidez o sobrevivencia y es extensiva al cálculo actuarial de reservas de personas que aún no se han pensionado pero que tienen la expectativa de hacerlo en el futuro dado que se tiene estimado el período de diferimiento o tiempo que falta para pensionarse.

3.1.3 Convenciones, Información Básica

La información básica para cada asegurado corresponde a la estructura señalada en el formato 394 proforma F.3000-74 [27], teniendo en cuenta las siguientes convenciones según el origen de la pensión: vejez, sustitución originada en el fallecimiento de un asegurado o afiliado pensionado, invalidez o sobrevivencia originada en el fallecimiento de un asegurado o afiliado que aún no se ha pensionado.

1. Origen de la pensión vejez: En los campos referidos al Parentesco A se incluye la información del pensionado. En los campos referidos al Parentesco B se incluye la información del primer beneficiario y se calcula la reserva de jubilación para el pensionado referido en el Parentesco A y reserva de sobrevivencia para el pago del beneficio al beneficiario referido en el Parentesco B en caso de fallecimiento del titular referido en el parentesco A.
2. Si el origen de la pensión es sustitución, en los campos referidos al Parentesco A se incluye la información del pensionado o causante de la sustitución. En los campos

referidos al Parentesco B se incluye la información del primer beneficiario. Se calcula solo la reserva de sobrevivencia para el beneficiario referido en el parentesco B.

3. Origen de la pensión invalidez: En los campos referidos al Parentesco A se incluye la información del inválido. En los campos referidos al Parentesco B se incluye la información del primer beneficiario y se calcula la reserva de jubilación para el asegurado o afiliado referido en el Parentesco A y reserva de sobrevivencia para el beneficiario referido en el Parentesco B.
4. Si el origen de la pensión es sobrevivencia de un asegurado o afiliado fallecido sin haberse pensionado: En los campos referidos al Parentesco A se incluye la información del fallecido. En los campos referidos al Parentesco B se incluye la información del beneficiario al que se le otorgó la pensión de sobrevivientes; solo se calcula la reserva de sobrevivencia para el beneficiario referido en el parentesco B.

3.1.4 Bases Técnicas

a) Tablas de Mortalidad

Existen varios factores que se deben tener en cuenta a la hora de intentar establecer un valor aproximado para la mortalidad de los seres humanos como por ejemplo el nivel económico, el factor social, cultural, demográfico, entre otros. La información base sobre la cual se obtienen los datos relacionados con la mortalidad es la relacionada con los registros de defunciones ya que con base a ellos se derivan las tablas de mortalidad. En estas tablas derivadas se observan aspectos de gran relevancia para diferentes grupos de edad con respecto a la probabilidad de mortalidad en un periodo determinado de tiempo. Adicionalmente permite obtener una medida resumen de la mortalidad general: la esperanza de vida al nacer, no afectada por la composición por edad de la población y se derivan, así mismo, las funciones indispensables para la elaboración de las proyecciones de población.

Estas tablas contienen diversos índices que representan cambios en las condiciones y en la dinámica de la mortalidad a medida que son definidas nuevas tablas de mortalidad que pueden determinar ciertos comportamientos los cuales son [54]:

- Los cambios en el nivel de mortalidad, reflejados en la esperanza de vida al nacer.
- Las transformaciones de la estructura de la mortalidad por genero y edad

- La evolución de la diferencias por genero en la esperanza de vida.

Las últimas tablas de mortalidad que son utilizadas en los cálculos de pensiones fueron emitidas por la Superintendencia financiera el 30 de Julio de 2010 [14]. En estas nuevas tablas se estableció un nuevo patrón de mortalidad que refleja incrementos en la esperanza de vida de los colombianos. En general la industria considera que estas nuevas tablas son un gran avance debido a que permiten su actualización de acuerdo con la situación actual de la sociedad.

Para el caso de nuestro proyecto fueron definidas cuatro tablas de mortalidad; hombres válidos, mujeres válidas, hombres inválidos, y mujeres invalidas [15].

b) Tasas de crecimiento de rentas intereses

Crecimiento de pensiones (k): Corresponde a la tasa estimada de crecimiento de las futuras mesadas pensionales.

Varía anualmente con base en la variación del índice de precios al consumidor (IPC) ponderada de los tres últimos años, según lo dispuesto en el Decreto 2783 de 2001 del Ministerio de Hacienda y Crédito Público [16].

Interés técnico real (i): Corresponde a la tasa de interés real que se espera rinden en el futuro las reservas constituidas.

Varía anualmente según lo dispuesto en el Decreto 2783 de 2001 del Ministerio de Hacienda y Crédito Público [16].

c) Tablas de salarios mínimos

Estas tablas contienen los salarios mínimos por año en Colombia y son utilizadas para calcular la reserva pensional de las organizaciones.

3.1.5 Formulación: Notación Internacional

Definiciones

R_T: Reserva Total.

R_j: Reserva de jubilación.

R_s: Reserva de Sobrevivencia.

$R_{A/F}$: Reserva de Auxilio Funerario.

P = Valor de la pensión o renta mensual.

B = Valor de la prima.

x = Edad del trabajador a la fecha de corte.

y = Edad del beneficiario para el caso de invalidez.

t = Período de diferimiento en años para acceder al beneficio.

n. = Duración de la renta temporal

k = Tasa de crecimiento de las futuras mesadas pensionales.

- V: Factor de descuento financiero.

$$V = \frac{1}{1+i}$$

- i: Como se mencionó anteriormente es la tasa de interés técnico real.
- D_x: Conmutación actuarial definida así

$$D_x = V^x l_x$$

en donde:

l_x: Número de personas vivas de edad *x*, según las tablas de mortalidad

V^x: Factor de descuento financiero para las personas vivas de edad *x*

$C_x, Mx, D_{xiy}, Nxy = \text{conmutaciones}$ Actuariales

- d_x: número de personas que fallecen a la edad *x*, según tablas de mortalidad.

$C_x = V^{x+1} d_x$ Conmutación actuarial

$$Mx = \sum_{t=0}^{110} c_{x+t} \text{ Conmutacion actuarial}$$

- A_x:

Valor actual de $1 pagaderos a los beneficiarios al final del año de la muerte del trabajador de edad *x*.

$$A_x = \frac{M_x}{D_x}$$

$$Nx = \sum_{t=0}^{110} D_{x+t}$$

$$D_{xiy} = V^{(x+y)/2} l_x l_y$$

$$Nxy = \sum_{t=0}^{110} D_{x+t;y+t}$$

$(Va)_x^{12}$:

Valor actual de una renta vitalicia anual de $(1+k) para una persona de edad *x*, con tasa anual de crecimiento de k pagadera en doce fracciones vencidas de 1/12 cada una.

$$(Va)_x^{12} = (Va)_x\left(1 + \frac{11}{24}k\right) + \frac{11}{24}(1+k)$$

$$(Va)_x = \frac{N_{x+1}}{D_x}$$

- $(Va)_x^2$:

  Valor actual de una renta anual vitalicia $(1+k), con tasa anual de crecimiento de K, pagadera en dos fracciones vencidas de ½ cada una.

$$(Va)_x^2 = (Va)_x\left(1 + \frac{1}{4}k\right) + \frac{1}{4}(1+k)$$

- $(Va)^{12}x/y$:

  Valor actual de una renta anual vitalicia $(1+*k*), con tasa de crecimiento anual de *k*, pagadera mensual vencida de 1/12 al cónyuge de edad *y* a la muerte del jubilado de edad *x*.

$$(Va)_{x/y}^{12} = (Va)_{x/y}\left(1 + \frac{11}{24}k\right)$$

$$(Va)_{x/y} = (Va)_y - (Va)_{xy}$$

$$(Va)_{xy} = \frac{N_x + 1 : y+1}{D_{xy}}$$

- $(Va)_{x/y}^2$:

  Valor actual de una renta anual vitalicia $(1+k), con tasa de crecimiento anual de *k*, pagadera en dos fracciones vencidas de 1/2 al cónyuge de edad y a la muerte del jubilado de edad *x*.

$$(Va)_{x/y}^2 = (Va)_{x/y}\left(1 + \frac{1}{4}k\right)$$

- tE_x:

  Valor actual de $1 pagadero a una persona de edad x, dentro de *t* años, sí a los *t* años se encuentra viva.

$$tE_x = \frac{D_x + t}{D_x}$$

- tE_{xy}:

  Valor actual de $1 pagadero a una pareja de edad xy, dentro de t años, sí a los t años ambos viven.

$$tE_{xy} = \frac{D_x + t, y + t}{D_{xy}}$$

- A_F :

  Valor del auxilio funerario señalado en el artículo 51 de la Ley 100/93

Origen de la pensión: Vejez

Reservas

- Para los asegurados o pensionados con 14 mesadas anuales**:**
  - R_j: Reserva requerida para pagar las pensiones futuras de un pensionado de edad *x*.

$$R_j = \left((12P(Va)^{12}_{x+t}) + (2B(Va)^{2}_{x+t})\right)tE_x$$

  - Cuando el beneficiario es el cónyuge con hijos o el cónyuge mayor de 30 años sin hijos o el padre o la madre (o ambos, en cuyo caso *P*=*P*/2 y *B*=*B*/2):
  - R_s: Reserva requerida para pagar las pensiones futuras a un beneficiario de edad *y* cuando fallezca la persona de edad *x*.

$$R_s = \left((12P(Va)^{12}_{x+t/(y+t)}) + (2B(Va)^{2}_{x+t/(y+t)})\right)tE_x$$

  - Cuando el beneficiario es el cónyuge menor de 30 años sin hijos.

R_s = (12P$(Va)^{12}_{x+t/(y+t)\urcorner 20}$ + 2B.$(Va)^{2}_{xtt/y+t\urcorner 20}$) $_tE_{xy}$.

- Cuando el beneficiario es un hijo menor de 25 años.

R_s = (12P$(Va)^{12}_{x+t/(y+t)\urcorner 25-(y+t)}$ + 2B$(Va)^{2}_{x+t/(y+t)\urcorner 25-(y+t)}$) tE_{xy}

- En las conmutaciones anteriores se considera si el estado del pensionado y del beneficiario es válido o invalido.

- Para los asegurados o pensionados con 13 mesadas anuales:

R_j = ([12P $(Va)^{12}_{x+t}$] + [$B(Va)^{2}_{x+t}$]). tE_x

- Cuando el beneficiario es el cónyuge con hijos o el cónyuge mayor de 30 años sin hijos o el padre o la madre (o ambos, en cuyo caso P=P/2 y B=B/2)

R_s = (12P$(Va)^{12}_{x+t/(y+t)}$+ $B(Vai)^{2}_{x+t/(y+t)}tE_{xy}$

- Cuando el beneficiario es el cónyuge menor de 30 años sin hijos.

R_s = (12P$(Va)^{12}_{x+t/(y+t)\urcorner 20}$ + $B(Va)^{2}_{x+t/(y+t)\urcorner 20}$) tE_{xy}.

- Cuando el beneficiario es un hijo menor de 25 años.

R_s= (12P$(Vav)^{12}_{x+t/(y+t)\urcorner 25-(y+t)}$ + $B(Vav)^{2}_{x+t/(y+t)\urcorner 25-(y+t)}$) tE_{xy}.

En las conmutaciones anteriores se considera si el estado del pensionado y del beneficiario es válido o invalido

$R_{A/F}$=$(A_{x+t}A_f)tE_x$

$R_T = R_j + R_s + R_{A/F}$

Si se ha reconocido la pensión de sobrevivientes por fallecimiento del pensionado

$R_s = R_j$ calculada con la edad del beneficiario

$R_T = R_s$

Origen de la pensión: invalidez

Reservas

- Para los asegurados o afiliados inválidos con 14 mesadas anuales**:**
  - R_j: Reserva requerida para pagar las pensiones futuras de un pensionado de edad *x*.

    $R_j = ([12P(Va)^{12}_{x+t}] + [2B\ (Va)^{2}_{x+t}]).\ tE_x.$
  - Cuando el beneficiario es el cónyuge con hijos o el cónyuge mayor de 30 años sin hijos o el padre o la madre (o ambos, en cuyo caso P=P/2 y B=B/2)
  - R_s: Reserva requerida para pagar las pensiones futuras a un beneficiario de edad *y* cuando fallezca la persona de edad *x*.

    $R_s = (12P(Va)^{12}_{x+t/(y+t)} + 2B.(Va)^{2}_{x+t/y+t})\ tE_{xy}.$
  - Cuando el beneficiario es el cónyuge menor de 30 años sin hijos

    $R_s = (12P(Va)^{12}_{x+t/(y+t)\urcorner 20} + 2B.(Va)^{2}_{xtt/y+t\urcorner 20})\ tE_{xy}.$
  - Cuando el beneficiario es un hijo menor de 25 años.

    $R_s = (12P(Va)^{12}_{x+t/(y+t)\urcorner 25-(y+t)} + 2B.(Va)^{2}_{xtt/y+t\urcorner 25-(y+t)})\ tE_{xy}$
- Para los asegurados o afiliados inválidos con 13 mesadas anuales**:**

  $R_j = ([12P(Va)^{12}_{x+t}] + [B\ (Va)^{2}_{x+t}]).\ tE_x.$
  - Cuando el beneficiario es cónyuge con hijos o el cónyuge mayor de 30 años sin hijos o el padre o la madre (o ambos, en cuyo caso P=P/2 y B=B/2)

    $R_s = (12P(Va)^{12}_{x+t/(y+t)} + B.(Vai^{2}_{x+t/y+t})\ tE_{xy}.$
  - Cuando el beneficiario es cónyuge menor de 30 años sin hijos

    $R_s = (12P(Va)^{12}_{x+t/(y+t)\urcorner 20} + B(Va)^{2}_{x+t/(y+t)\urcorner 20})\ tE_{xy}.$
  - Cuando el beneficiario es un hijo menor de 25 años.

    $R_s = (12P(Vav)^{12}_{x+t/(y+t)\urcorner 25-(y+t)} + B(Vav)^{2}_{x+t/(y+t)\urcorner 25-(y+t)})\ tE_{xy}.$

- En las conmutaciones anteriores se considera que el pensionado es inválido y el estado del beneficiario (valido o inválido).

$$R_{A/F} = (A_x + {}_tA_f)\,{}_tE_x$$

$$R_T = R_j + R_s + RA_f$$

Origen de la pensión: Sobrevivencia por fallecimiento del asegurado o afiliado no pensionado

Reservas

- Para los beneficiarios con 14 mesadas

$$R_s = ([12P\ (Va)^{12}_{x+t}\ ] + [2B\ (Va)^{2}_{x+t}\ ]).\ {}_tE_x$$

- Para los beneficiarios con 13 mesadas

$$R_s = ([12P\ (Va)^{12}_{x+t}\ ] + [B\ (Va)^{2}_{x+t}\ ]).\ {}_tE_x$$

$$R_T = R_s$$

Personal con renta temporal

Reservas

- Para los sobrevivientes que sólo tienen derecho a una renta temporal 14 mesadas en el año

$$R_s = (12P.(Va)^{12}_{x+t\,\urcorner\,n} + 2\,B.(Va)^{2}_{x+t\,\urcorner\,n})\ {}_tE_x$$

- Para los sobrevivientes que sólo tienen derecho a una renta temporal 13 mesadas en el año

$$R_s = (12P.(Va)^{12}_{x+t\,\urcorner\,n} + 2\,B.(Va)^{2}_{x+t\,\urcorner\,n})\ {}_tE_x$$

$$R_T = R_s$$

3.2 Aplicaciones Empresariales Web

3.2.1 Descripción

Una aplicación web es un sistema de información que opera en el entorno de internet. El navegador web (Firefox, Internet Explorer, etc.), es el que permite la interacción entre el usuario y la aplicación como tal. El protocolo HTTP (Hyper Text Transfer Protocol) es un protocolo utilizado a nivel de la aplicación para transferencia de información entre sistemas de manera rápida [32]. Este protocolo es el que permite la conexión entre el navegador y el servidor web y es tolerante a fallos.

Uno de los principales componentes sobre el cual se basan las aplicaciones web, es el uso de HTML (Hyper Text Markup Language) el cual es un lenguaje descriptivo que usan los diseñadores para crear páginas web independientes de la plataforma usada. En otras palabras, HTML especifica un conjunto de reglas que los diseñadores siguen para crear documentos que son entendidos por diferentes aplicaciones que funcionan bajo diferente hardware y sistemas operativos [33].

Existen varias tecnologías que pueden servir para realizar este proyecto, están las tecnologías con servidor de aplicaciones como .Net y Java EE5, y sin servidor de aplicaciones como PHP, Perl y Python, a continuación se explican las principales características de cada una de estas tecnologías, y los factores de comparación que se tuvieron en cuenta para escoger la tecnología apropiada para el presente proyecto.

3.2.2 Tecnologías que no utilizan servidor de aplicaciones

Las siguientes son las tecnologías que no usan servidor de aplicaciones.

a) PHP

Es un lenguaje de “código abierto” interpretado, de alto nivel, contenido en páginas HTML y ejecutado en el servidor. El código PHP siempre es ejecutado en el servidor [34], por lo tanto cuando el cliente realiza una solicitud al servidor web, se ejecuta un paso intermedio denominado proceso previo. En este punto el compilador PHP procesa el código que recibe de la solicitud del cliente y pasa el resultado al servidor web, una vez terminado este proceso finalmente el servidor web envía el resultado al cliente.

Generalmente con PHP se desarrollan aplicaciones web relativamente pequeñas pero en algunos casos se encuentran sistemas grandes como Amazon, desarrollado en PHP, al igual que manejadores de contenido (como Moodle) y ambientes colaborativos (como gForge). Estos sistemas grandes desarrollados en PHP rara vez ofrecen operaciones transaccionales y características de alta seguridad. PHP no fue concebido para soportar estas características por lo cual se deben programar apoyándose en el motor de base de datos.

b) Perl

Perl es un lenguaje de alto nivel que permite desarrollar prototipos de forma rápida y ágil, es de código abierto y tiene una historia que supera los 20 años, además es multiplataforma y no está vinculado a ninguna empresa [48]. Perl se puede usar para programación en la Web ya que cuenta con capacidades de manipulación de texto y con un ciclo de desarrollo rápido, tiene desarrollados varios *web frameworks* y se puede integrar con diferentes motores de base de datos [48]. La principal desventaja de Perl es el tiempo de ejecución de los programas debido a que son compilados cada vez que se ejecutan por lo que consume muchos recursos de máquina.

Igualmente para asegurar requerimientos no funcionales como seguridad y manejo de transacciones, las aplicaciones Perl deben apoyarse en otras herramientas como los motores de bases de datos.

c) Python

Python es un lenguaje de programación usando en un amplio rango de aplicaciones y entre sus principales características está el ser claro y con una sintaxis legible, ser orientado a objetos, y tener un manejo de errores basado en excepciones, entre otras [49]. Python se integra fácilmente con otros lenguajes de programación como .Net, Java o C++, corre en casi cualquier plataforma, es amigable y fácil de aprender, además es de código abierto lo cual implica que se puede usar y distribuir libremente [49]. Así como en los lenguajes mencionados anteriormente para asegurar requerimientos no funcionales como seguridad y manejo de transacciones las aplicaciones Python deben apoyarse en otras herramientas como motores de bases de datos.

3.2.3 Tecnologías que utilizan servidor de aplicaciones

Un Servidor de Aplicaciones es un producto de software que permiten administrar las sesiones de los clientes, alojan la lógica de negocio y se conectan con los recursos de cómputo entre los que se incluyen datos, transacciones y contenido.

Los servidores de aplicaciones típicamente funcionan como sistema de comunicación para una o más bases de datos para transmitir datos procesados hacia y desde una interfaz de usuario. La interfaz de usuario para este tipo de aplicaciones usualmente se despliega a través de un navegador web, pero también se puede presentar de otras maneras a los clientes que pertenezcan a una misma red distribuida.

La principal función de los servidores de aplicaciones es la de procesar la lógica de negocio en una arquitectura multicapas. La arquitectura de tres capas se refiere a los tres diferentes niveles que generalmente se manejan en este tipo de aplicaciones, el cliente, el servidor de aplicaciones y la base de datos.

a) .NET Framework

.NET *framework* es una plataforma de Microsoft que permite la compilación y ejecución de aplicaciones y servicios web XML. .NET tiene dos componentes principales [17]: Common Language Runtime y la biblioteca de clases base de .NET. Common Language Runtime o motor en tiempo de ejecución administra el código en tiempo de ejecución y proporciona servicios como la administración de memoria, la administración de subprocesos y comunicación remota, promoviendo la seguridad y solidez del código. El código ejecutado bajo el motor en tiempo de ejecución se denomina código administrado mientras que al resto del código se le llama código no administrado.

El otro componente principal de .NET, la biblioteca de clases base es una colección orientada a objetos de tipos reutilizables que se usan para desarrollar aplicaciones que van desde las herramientas de interfaz gráfica de usuario, pasando por las de línea de comandos, hasta las aplicaciones basadas en ASP.NET [50].

En la siguiente ilustración se muestran los dos componentes principales de .NET *framework* y

su relación con otros componentes del sistema; como se puede observar en la figura las aplicaciones web trabajan bajo el servidor IIS (*Internet Information Server*).

Ilustración 1 .Net framework [50]

.NET es una plataforma de desarrollo que soporta varios lenguajes de programación como son Visual Basic, Visual C#, IronRuby e IronPython, entre otros; además cuenta con opciones de acceso flexible a datos provistas por ADO.NET [51].

A continuación se mencionan algunas de las ventajas ofrecidas por el framework.

- Unifica los modelos de Programación
- Simplifica el proceso de desarrollo
- Proporciona un entorno seguro y robusto de ejecución
- Es independiente del lenguaje de programación
- Interoperabilidad ofrecida con aplicaciones externas
- Simplifica la administración e instalación de aplicaciones

Las aplicaciones .NET se encuentran alojadas dentro de un contenedor, que proporciona una variedad de servicios que son necesarios para el desarrollo de aplicaciones empresariales entre los cuales se encuentran transacciones, la seguridad, y los servicios de mensajería [59].

b) Java EE 5

La plataforma Java Empresarial actual ofrece soporte para un gran número de las tecnologías de computación distribuida y servicios de redes [35]. Estos servicios vienen desde la versión J2EE y son los siguientes:

- JDBC: es un API que permite el envío de sentencias SQL a un servidor de base de datos así como de capturar los resultados de las consultas que envía el servidor.
- RMI: Es un modelo de programación que proporciona un acercamiento genérico de alto nivel a la computación distribuida. Solo funciona si el cliente y el servidor están implementados en Java.
- Java IDL (CORBA): Es un modelo de programación distribuida similar a RMI. Pero a diferencia de RMI este modelo funciona sin importar el leguaje de programación en el que se encuentre implementado el cliente o el servidor.
- JNDI: Permite a los programas Java tener transparencia con respecto a la ubicación de objetos o datos de acuerdo con un grupo de valores de atributos específicos.
- Enterprise JavaBeans: Es un modelo de componente para unidades de lógica de negocio y datos del negocio.
- JMS (*Enterprise Messaging*): es el API de java que permite trabajar con el envío asincrónico y recepción de mensajes en red a través de colas de mensajes controlados por el servidor JMS.

Una nueva versión de la plataforma tecnológica se introdujo en el año 2006 esta es conocida como la versión empresarial de Java EE 5. El propósito de la plataforma Java EE 5 es de proporcionar a los desarrolladores un poderoso grupo de librerías que permite reducir en gran margen el tiempo de desarrollo, reduciendo la complejidad de la aplicación, y mejorando el desempeño de la misma.

El propósito de la tecnología Java EE 5 es minimizar el numero de artefactos que deben ser creados y mantenidos, simplificando el proceso de desarrollo. Java EE 5 soporta la inyección

de anotaciones dentro del código fuente sin necesidad de que mantener artefactos en otro lugar.

Una anotación es un modificador o metadato que proporciona información adicional para el servidor respecto a las clases, interfaces, constructores, métodos, campos, parámetros, y variables locales que son definidos dentro del proceso de desarrollo de una aplicación [60]. Por ejemplo una anotación puede reemplazar la interfaz y la implementación requerida para un servicio web. Las anotaciones también pueden reemplazar campos adicionales que el programa requiere, y que se mantienen separadamente [60]. Por medio de estas anotaciones estos archivos separados que eran usados anteriormente y también conocidos como descriptores, ya no resultan útiles. A continuación se mencionan las principales funciones de las anotaciones:

- Remplazar algunos de los descriptores existentes
- Eliminar la necesidad de definir interfaces adicionales (ejemplo java.rmi.Remote)
- Permiten que se observe la configuración de un componente en el mismo lugar donde se encuentra definido.

Java EE5 proporciona anotaciones para las siguientes tareas entre otras:

- Desarrollo de aplicaciones empresariales
- Definición y uso de Servicios Web
- Asociar clases con tecnología Java con archivos XML
- Asociar clases con tecnología Java con Bases de Datos
- Asociar Métodos con operaciones
- Especificación de dependencias externas
- Especificación de información de despliegue, incluyendo atributos de seguridad.

Un desarrollador puede simplemente ingresar la información como anotaciones directamente en el código fuente java, y el servidor Java EE se encarga de configurar el componente en el momento del despliegue y en tiempo de ejecución.

Persistencia en JEE 5

El *framework* de persistencia (Java Persistence API) se introduce en la plataforma Java EE 5. Este componente es un estándar de persistencia y un marco de asociación de objetos

relacionales para el lenguaje java. JPA forma parte de la especificación EJB 3.0 sobre la tecnología Java EE 5. También proporciona el paquete se servicios javax.persistence y su principal función es la de permitir la persistencia de objetos Java (POJOs) en una base de datos relacional.

La persistencia en java está compuesta por 3 áreas fundamentales [41]:

- El API de persistencia de java
- El lenguaje que permite realizar las consultas.
- El modelo de objetos y el modelo relacional de los metadatos.

Las entidades se encuentran definidas dentro del API de persistencia de java, el cual proporciona un modelo orientado a objetos para lograr un manejo adecuado de los datos en las aplicaciones Java.

Enterprise Java Beans

Un Enterprise Java Bean es un componente de software del lado del servidor que puede ser desplegado en un ambiente de *n* capas distribuido. Un EJB es desplegado en un contenedor de EJB. En otras palabras, un contenedor es aquel que se encarga de proporcionar los servicios para todos los componentes EJB. Existen varios tipos de EJB los cuales se describen a continuación:

1. **EJB de Sesión**: Este tipo de EJB está hecho con el propósito de implementar una porción de la lógica de una aplicación solicitada por un cliente que puede ser un servlet, un JSP o un applet. Existen dos tipos de EJB de sesión:
    a. EJB sin estado (*stateless*): este tipo de EJB no es modificado al recibir peticiones de los clientes, es decir el estado siempre permanece igual, solo se encarga de recibir datos y retornar resultados. Internamente el contenedor del EJB crea un conjunto de instancias con el propósito de que el *bean* pueda ser utilizado por múltiples clientes de manera concurrente [64].
    b. EJB con estado (*stateful*): un EJB con estado es aquel en el cual las variables definidas almacenan información que recibe mientras dura la conexión de un cliente con el componente. Es posible que un cliente interactué varias veces con el

componente y se modifique el valor de las variables a lo largo de la conversación [64].

2. **EJB de Entidad**: un EJB de entidad representa un objeto de negocio en un mecanismo de almacenamiento persistente. Algunos ejemplos de objetos de negocio son clientes, órdenes y productos. En Java EE 5 el mecanismo de almacenamiento es una base de datos relacional. Usualmente cada EJB de entidad representa una tabla en la base de datos relacional [61].
3. **Mensajería**: Quizás el framework más importante es JMS que viene acoplado desde el principio a un servidor de aplicaciones, y que permite realizar tareas masivas con alto paralelismo utilizando múltiples colas de mensajes.

Servicios Proporcionados por los EJBs

Estos *EJBs* mencionados anteriormente proporcionan un conjunto de servicios en el desarrollo de aplicaciones web que se describen a continuación.

1. *Servicios de distribución*:
    a. Método de Invocación Remota (RMI): este método es usado por el cliente para acceder a los EJB y viceversa. Debe ser definida una interfaz remota para cada uno de los EJB que se tengan definidos. El contenedor genera el objeto EJB a partir de las definiciones de la interfaz remota [62].
    b. Protocolos: EJB ha adoptado IIOP como el protocolo estándar para proporcionar interoperabilidad.
2. *Manejo de Estado*: el manejo de estados se realiza a través de los EJB de sesión los cuales fueron definidos previamente en este documento.
3. *Manejo de Persistencia*: el manejo de persistencia se realiza a través de los EJB de entidad los cuales fueron definidos previamente en este documento.
4. *Manejo de Transacciones*: La arquitectura de los Enterprise JavaBeans soporta transacciones distribuidas en aplicaciones basadas en componentes [62]. Aunque esta tecnología también puede ser usada en la implementación de sistemas no transaccionales, el modelo fue diseñado para soportar transacciones distribuidas incluso con bases de datos distribuidas.

5. *Seguridad*: los EJB utilizan los servicios de seguridad de java entre los cuales se incluyen servicios de autenticación y autorización para restringir el acceso a objetos y métodos seguros. La tecnología EJB automatiza el uso de la plataforma Java de modo que no sea necesario codificar explícitamente las rutinas de seguridad de Java.

3.2.4 Ventajas de trabajar con Servidores de Aplicaciones

Un servidor de aplicaciones permite reducir el tamaño y la complejidad de las aplicaciones contenidas en él permitiendo a estos programas compartir recursos y funcionalidad de una manera organizada y eficiente. Lo servidores de aplicaciones proporcionan beneficios en las área de usabilidad, escalabilidad, mantenibilidad entre otras propiedades no funcionales [69].

Las principales ventajas que proporcionan los servidores de aplicaciones son:

- Permite el desacoplamiento entre el cliente y la lógica de negocio.
- Facilidad en administración y mantenimiento del sistema.
- Existe un punto central para el almacenamiento de la información.
- Independencia de la plataforma o sistema operativo (Servidores de aplicaciones Java).

Además de los aspectos mencionados anteriormente, los servidores de aplicaciones dan una solución a muchas de las debilidades de las aplicaciones desarrolladas bajo el modelo cliente servidor.

El principal beneficio que brindan los servidores de aplicaciones es la capacidad de separar la carga de trabajo entre varios computadores. Un servidor tiene una cantidad finita de procesamiento sin importar el costo de la inversión. Por lo tanto esta característica brinda la posibilidad de invertir en servidores de rango medio lo cual proporciona una mayor capacidad de cómputo y permite reducción de costos [74].

Otra ventaja de los servidores de aplicaciones es que permiten que las bases de datos utilizadas por las aplicaciones estén ubicadas en diferentes lugares. Esto permite que las fuentes principales de la información se ubiquen en donde más procesamiento va a ser realizado. Esto permite que el tráfico de red disminuya porque gran parte del procesamiento se realiza de manera local.

Los servidores de aplicaciones mejoran el proceso de desarrollo de software separándolo en tareas más pequeñas y proporcionando *frameworks* que permiten la reutilización de código. Adicionalmente introducen un nuevo concepto que busca mejorar el modelo cliente servidor

por medio de la aparición de una capa adicional en las aplicaciones que desacopla la lógica de negocio del resto de la aplicación.

3.2.5 Comparación entre tecnologías para el desarrollo web con servidores de aplicaciones

Pese a que PHP es muy utilizado dada su facilidad de programación y por ende alta velocidad de desarrollo, además de ser una tecnología de código abierto, fue descartado para el desarrollo del presente proyecto debido a la falta de experiencia por parte de los miembros del equipo de trabajo y debido a que los servicios prestados por los servidores de aplicaciones proporcionan varias ventajas a los desarrolladores sobre esta tecnología como pool de conexiones a la base de datos y servicios transaccionales y seguridad. Estos son servicios que ya son proporcionados por los servidores de aplicaciones y que no deben ser implementados por los desarrolladores.

JEE y Microsoft.NET tienen como propósito simplificar el desarrollo de aplicaciones web proporcionando un conjunto de componentes y servicios. Por medio de los componentes estandarizados y servicios a su disposición, los desarrolladores pueden concentrarse en la lógica del negocio sin necesidad de tener que codificar servicios adicionales.

Estas dos plataformas fueron creadas para permitir a los desarrolladores la construcción de aplicaciones “por capas”. Una aplicación de tres capas consiste en una capa cliente, una capa servidor y otra de datos [63]. En una aplicación web típica el usuario selecciona un link, este hace que se transmita una petición al servidor de aplicaciones. Como respuesta el servidor de aplicación accede a la información que se encuentra almacenada en un motor de base de datos, luego crea una página web y presenta los resultados de la petición al usuario[63]. JEE y .NET proporcionan servicios para facilitar las transacciones en las cuales se pueden ejecutar procesos de negocio que pueden almacenar datos en la base de datos.

Una diferencia clave entre estas dos plataformas se presenta porque JEE es un estándar abierto que se ejecuta en múltiples plataformas, mientras que .NET es propiedad de Microsoft y funciona solamente sobre Windows. Los productos JEE son ofrecidos por múltiples distribuidores mientras que para la tecnología .NET los productos solo son ofrecidos por Microsoft. Es importante notar que JEE no es como tal un producto sino un conjunto de estándares sobre los cuales se desarrollan productos.

Se consideraron ciertas características al momento de realizar la comparación:

- **Escalabilidad**: Es la capacidad de mejorar recursos para ofrecer una mejora (idealmente) lineal en la capacidad de servicio [65]. La característica principal es que cuando se pretenda agregar carga adicional a la aplicación esto se logre agregando recursos y no modificando la estructura de la aplicación. Para cumplir con este requerimiento ambas tecnologías ofrecen diversas opciones entre las cuales se encuentran la utilización de tecnologías de organización por clústeres que incluyen equilibrio de carga en la red y de los componentes. Adicionalmente esto permite mejorar la disponibilidad de los sitios web.
- **Portabilidad**: Habilidad del software para ser transformado de un entorno a otro [ISO 9126].

Esta característica como se mencionó anteriormente está a favor de las aplicaciones JEE debido a que funcionan bajo cualquier sistema operativo mientras que las aplicaciones .NET funcionan únicamente bajo ambiente Windows.

- **Rendimiento**: Desde el punto de vista de los usuarios de las aplicaciones construidas usando alguna de estas dos tecnologías el rendimiento se mide en términos de tiempos de respuesta.

En cuanto al despliegue de aplicaciones y en cuanto a los retardos o tiempos de respuesta de una tecnología con respecto de la otra a medida que se aumenta el número de usuarios no se presentan grandes diferencias entre las dos tecnologías [57]. Estos servidores web soportan métodos de estabilidad que permiten distribuir la carga de los sitios en función del hardware utilizado, esto permite que se obtengan mejores tiempos de respuesta para los usuarios.

- **Seguridad**: Habilidad para prevenir acceso no autorizado accidental o deliberado a programas o datos [ISO 9126]. JEE y .NET proporcionan servicios de seguridad pero con enfoques diferentes. Los servicios de autorización y autenticación en .NET son proporcionados a través de repositorios de identificación. JEE no especifica qué métodos deben ser usados para realizar estas funciones, dejando estas funciones a los desarrolladores o distribuidores. A pesar de no ser obligatorio, la funcionalidad de autorización y autenticación es proporcionada a través de JAAS (Java authorization and authentication service) [56]. Ambas plataformas usan conceptos similares para el manejo

de acceso a los recursos por parte de los usuarios y el código, con los permisos y el concepto de la asociación de roles a permisos siendo clave para ambas. Pero JEE es más fino en el modelo de roles pues se puede proteger a nivel de método de componente de negocio mientras que .NET solo a nivel de componente.

- **Costo Monetario del Sistema**: JEE puede ofrecer soluciones con variaciones en los precios de acuerdo con las necesidades del usuario que puede requerir inversiones costosas debido a la capacidad de ejecución en múltiples plataformas, pero también pueden ser muy baratas, incluso gratuitas. Por otra parte las plataformas Windows que requiere .NET pueden llegar a ser más económicas que las plataformas UNIX que usualmente son utilizadas por JEE. El precio de las licencias para una determinada plataforma varía de acuerdo a las utilidades que son ofrecidas a los desarrolladores (Oracle permite utilizar la implementación de su estándar de JEE de forma gratuita).
- **Lenguajes de Programación soportados**: la plataforma JEE soporta única y exclusivamente el lenguaje Java, mientras que .NET esta creado para soportar varios leguajes ofrecidos por Microsoft entre los cuales se encuentran Visual Basic, C#, o C++, además de ofrecer interoperabilidad entre todos los lenguajes mencionados. A continuación se muestra un cuadro comparativo con algunas características de las dos tecnologías:
- **Persistencia**: Tanto Java EE5 como NET ofrecen una manejo de persistencia mediante la definición de entidades que representan objetos del dominio que están relacionados con la información almacenada en la base de datos. Java EE5 maneja la persistencia mediante el framework JPA mientras que .NET lo hace mediante el Entity Framework.

	JEE	.NET
Modelo de Componentes	**Sí(EJB)**	**Si (DCOM)**
Independencia de plataforma	**Si**	**No, Windows Únicamente**
Lenguaje de programación	**Java**	**C#, VB, C++, J#...**
Costo del producto	**Algunos gratuitos**	**Comercial**

Acceso a base de datos	**JDBC**	**ADO.NET**
Contenido web dinámico	**JSP,JSF**	**ASP.NET**
Tipo de Tecnología	**Estándar**	**Tecnología**
Persistencia	**JPA**	**Entity Framework**

Ventajas de trabajar con Java EE 5

Debido a que este es un proyecto con enfoque empresarial, entre las características principales que se requieren se encuentra la parte de seguridad, así como el manejo de transacciones debido a que en algunas ocasiones la aplicación será accedida por varios usuarios al mismo tiempo de modo que se hace indispensable el manejo de la concurrencia, así como la atomicidad de las transacciones que se realizan en la base de datos. Esta es la labor que se encarga de realizar el servidor de las aplicaciones Java EE 5.

Otro aspecto de gran importancia a favor de Java EE 5 es la capacidad de mantenibilidad que tienen este tipo de aplicaciones. Estas aplicaciones se basan en el modelo orientado a objetos, además se maneja una arquitectura definida previamente que se divide en capas, que separa la presentación, la lógica de negocio, y la persistencia de la información. Esto permite que la complejidad a la hora de encontrar problemas o realizar modificaciones no sea tan grande como lo sería si se estuviera trabajando con otro tipo de arquitectura.

Los servidores de aplicaciones Java EE 5 ofrecen facilidades de operación en *cluster* para lograr alta escalabilidad, permitiendo atender un número creciente de usuarios simultáneos mediante la configuración de servidores adicionales en el *cluster*.

Las aplicaciones Java EE 5 pueden lograr buenos tiempos de respuesta aplicando patrones en el desarrollo y aprovechando las facilidades automáticas de *pool* de conexiones que ofrece el servidor de aplicaciones. Siguiendo los patrones, las conexiones a las bases de datos se realizan desde los componentes de negocio y no desde las páginas o elementos de la capa web.

Por otro lado, al usar Java EE 5 tenemos acceso a *frameworks* libres como JSF (JavaServer Faces), JPA (Java *Persistence* API) y Seam.

Se escogió realizar el presente proyecto utilizando la plataforma JavaEE debido a que ofrece los servicios que son necesarios en cuanto a seguridad, rendimiento y escalabilidad, además las herramientas que utilizamos para cumplir con el estándar no tienen ningún costo para el equipo de trabajo. Por otro lado JEE nos da la posibilidad de ejecutar la aplicación en sistemas operativos Linux haciendo una previa configuración. En cuanto a la persistencia se identifico que Java EE 5 por medio del framework JPA brinda ciertas ventajas a los desarrolladores debido a que evita la escritura de código JDBC y permite que los desarrolladores se enfoquen en la lógica del negocio. En cuanto a seguridad JEE es más fino en el modelo de roles pues se puede proteger a nivel de método de componente de negocio mientras que .NET solo a nivel de componente.

3.3 Motores de Reglas

Previo al surgimiento de los motores de reglas los sistemas tecnológicos tenían el reto de almacenar reglas de negocio dentro del código de la aplicación o de construir su propio motor de reglas. La definición de reglas dentro de la lógica de la aplicación implicaba cambios en el código y publicar nuevamente la aplicación cada vez que cambiaba alguna regla [66]. Esto causaba que un sistema fuera muy costoso para mantener y riesgoso de modificar a medida que las reglas de negocio evolucionaban.

Con el tiempo aparecieron los motores de reglas, que de alguna forma resolvieron los problemas que se presentaban antes de su aparición.

El propósito principal de los motores de reglas es el de permitir cambios rápidos a las reglas de negocio sin tener que realizar modificaciones en el código que forma parte de la lógica de negocio. También se busca que los sistemas con motores de reglas se puedan extender para permitir que los usuarios con conocimientos del domino de negocio puedan hacer modificaciones en las reglas [66]. Las reglas son declarativas, entendibles por el cliente y no forman parte del código de la aplicación.

3.3.1 ¿Por qué usar un motor de reglas?

a) Programación Declarativa

La principal ventaja de un motor de reglas es la programación declarativa ya que por medio de las reglas se pueden expresar de manera sencilla soluciones a problemas complejos y por lo tanto es más fácil verificar las soluciones generadas. Las reglas son mucho más fáciles de leer que el código como tal.

Los sistemas de reglas están en capacidad de resolver problemas de una alta complejidad proporcionando una explicación de cómo se llego a la solución y por qué se tomó cada decisión en el camino para llegar a esa solución [71].

b) Separación de lógica y datos

Los datos se encuentran en los objetos del dominio, y la lógica es definida mediante de las reglas. El motor de reglas separa los datos de la lógica de negocio lo cual puede ser considerado como una ventaja en la mayoría de los casos. Esta separación permite que la lógica de negocio sea mucho más fácil de mantener si se producen cambios en el futuro debido a que solo es necesario modificar las reglas que se ven afectadas sin tener que recompilar la aplicación. En lugar de tener la lógica definida en muchos objetos del dominio todo puede ser organizado en uno o muchos archivos de reglas diferentes [71].

c) Velocidad y Escalabilidad

Los algoritmos utilizados para el desarrollo de los motores de reglas proporcionan formas muy eficientes de relacionar los patrones de las reglas con los objetos del dominio definidos. Es especialmente eficiente cuando se tienen conjuntos de datos que cambian en pequeñas porciones de modo que el motor de reglas pueda recordar asociaciones similares hechas previamente [71].

d) Centralización del conocimiento

Por medio del uso de las reglas, se crea un repositorio del conocimiento (*knowledge base*) el cual es ejecutable, esto permite que toda la información necesaria sea almacenada en un solo punto representado por el repositorio, por ejemplo una política de negocio. Las reglas son tan

fáciles de leer que en muchas ocasiones pueden ser utilizadas como parte de la documentación de una determinada aplicación [71].

e) Reglas Fáciles de Entender

Los sistemas de reglas proporcionan facilidad en la explicación de las reglas adoptadas.

3.3.2 ¿Cuándo usar un motor de reglas?

Se recomienda usar un motor de reglas en cada aplicación que contenga cierta lógica de programación la cual necesita cambiar continuamente. Incluso si una aplicación contiene solamente un pequeño grupo de reglas que cambian constantemente, se puede ver muy beneficiada separando las reglas del resto de la lógica de negocio [42].

Si una determinada aplicación usa demasiados *if/then* esta puede ser simplificada con un motor de reglas [42]. Las organizaciones en general cuentan con expertos del dominio disponibles que tienen un amplio conocimiento sobre las reglas de negocio y procesos de una organización. Este tipo de personas generalmente no son muy técnicas pero pueden ser muy lógicas. Las reglas les pueden permitir expresar la lógica que definen en sus propios términos. Desde luego estas personas deben estar en capacidad de pensar crítica y lógicamente.

3.3.3 Reglas y Formas de Representación del conocimiento

El conocimiento es visto como un concepto en un nivel alto de abstracción. Por ejemplo, ‘40’ es un dato, ‘400 grados al mediodía’ puede ser información relacionada con el clima en algunas situaciones, pero la expresión ‘la mayoría de personas se siente mejor cuando la temperatura crece sobre los 40 grados’ es conocimiento acerca de información del clima y sobre las personas. La comprensión de que el conocimiento se expresa en la forma de reglas es lo que permite la concepción del conocimiento como algo más abstracto que la información [52].

Existen diferentes tipos de conocimiento que el ser humano está en capacidad de identificar con facilidad; el conocimiento relacionado con objetos, eventos, desempeño de tareas, entre otros.

El conocimiento de causalidad se expresa generalmente como una cadena de sentencias las cuales relacionan causa y efecto. Una sentencia típica puede ser ‘si se hierven tomates con los

aderezos correctos se obtiene como resultado salsa de tomate.' Este tipo de conocimiento puede ser bien representado por un conjunto de reglas que pueden ser encadenadas usando proposiciones lógicas [52].

Para el propósito de este proyecto estamos interesados en la representación del conocimiento por medio de reglas de producción las cuales son generalmente conocidas como instrucciones *IF/THEN*.

El lado izquierdo, A, de una regla de producción de la forma ´If A then X´ es llamado cláusula antecedente mientras que el lado derecho, X, es la consecuencia. La regla puede ser interpretada de muchas maneras: si alguna condición se satisface entonces alguna acción es apropiada; si alguna sentencia es verdadera entonces otra sentencia puede ser inferida; si una cierta estructura sintáctica está presente entonces otra puede ser generada gramaticalmente. En general la A y la X pueden ser sentencias complejas construidas a partir de sentencias más simples usando conectores AND, OR, o el operador NOT, estos operadores solamente pueden ser usados en el antecedente y posteriormente se define la consecuencia.

Las reglas de producción son fáciles de entender para los humanos en general, y cada regla representa una pequeña e independiente porción de conocimiento, y por lo tanto puede ser fácilmente agregada o sustraída del la base del conocimiento.

Debido a que las reglas son en principio independientes una de la otra, soportan un estilo declarativo de programación lo cual permite que se reduzcan los problemas de mantenimiento. Sin embargo es necesario tener precaución debido a que no se están tomando en cuenta las reglas contradictorias las cuales pueden producir en algunos casos ineficiencia y en otros casos conclusiones equivocadas.

Otra de las ventajas que ha sido explotada en los sistemas basados en reglas es la facilidad con la que un motor de reglas está en capacidad de guardar registro del uso de las reglas definidas por parte de las aplicaciones y por lo tanto proporcionar explicaciones acerca del razonamiento del sistema. Por último estos sistemas de producción no requieren muchos recursos del procesador pero en cambio requieren una cantidad relativamente alta de memoria y de almacenamiento secundario [52].

3.3.4 Inferencia y conocimiento

Hay dos preguntas importantes acerca de la representación del conocimiento. Primero está la pregunta de cómo se representa el conocimiento en el cerebro animal o humano y a continuación se pregunta cuales son las estructuras que pueden ser usadas para la representación computacional. "La inteligencia artificial ha sido definida como el uso de las facultades mentales mediante la utilización de modelos computacionales" [52]; este significado no concuerda con el que buscan los desarrolladores de sistemas basados en conocimiento. Cabe notar que no existe certeza alguna sobre el funcionamiento del cerebro humano, y por lo tanto nadie está en capacidad de dar una aproximación acertada para la mejor representación del conocimiento computacional formalmente. Hasta que se realicen avances fundamentales lo mejor que pueden hacer los desarrolladores de este tipo de sistemas es el uso de cualquier formalismo que mejor se adapte a la tarea que se pretende realizar [52].

Aparte de ser abstracto en varios niveles, el conocimiento tiene relación con las acciones. El conocimiento tiene una estrecha relación con la información debido a que es el encargado de procesarla. Esto significa que nosotros entendemos el conocimiento pero lo que se procesa es información. No es útil por ejemplo que las personas respondan de manera favorable a la temperatura alta si no hay forma de medirla. El uso del conocimiento conduce a la formación de planes de acción.

Desde este punto de vista, la inferencia es similar al conocimiento como el procesamiento lo es a la información. Inferencia es el método usado para transformar percepciones en una forma adecuada para que puedan ser convertidas en acciones [52]. A partir de las experiencias vividas por un ser humano se puede observar individual y colectivamente que ciertas acciones dan como resultado ciertas consecuencias. Este fenómeno descrito anteriormente se conoce con el nombre de causalidad, el cual indica que la acción A 'causa' la percepción B.

En lo que se refiere a la inferencia no se puede esperar que los computadores tengan la misma capacidad del cerebro humano para hacer razonamientos de cualquier tipo. La inferencia en el área de ingeniería del conocimiento se define como el proceso formal por medio del cual se obtienen conclusiones a partir de premisas [52]. Existen varios métodos de inferencia que caracterizan a los Motores de Reglas, pero los principales son el encadenamiento hacia

adelante y el encadenamiento hacia atrás a continuación se explican los métodos mencionados.

a) Encadenamiento hacia adelante:

Para entender de una manera adecuada el método de encadenamiento hacia adelante, se considera un sistema cuyo conocimiento está representado de la forma de reglas de producción y cuyo dominio es la validez de proposiciones abstractas: A, B, C… [52]

La base del conocimiento está compuesta de las siguientes reglas:

Regla1: A and B and C implica D

Regla2: D and F implica G

Regla3: E implica F

Regla4: F implica B

Regla5: B implica C

Regla6: G implica H

Regla7: I implica J

Regla8: A and F implica H

Se asume que le fue solicitado al sistema verificar si la proposición H es verdadera dado que las proposiciones A y F también son verdaderas. Suponiendo que el computador almacena las reglas en un dispositivo secuencial de modo que el acceso a las reglas se debe hacer en orden.

La estrategia que se va a utilizar se conoce como encadenamiento hacia adelante y consiste en pasar por las reglas hasta que una de ellas se dispare, posteriormente se puede continuar hasta que todas las reglas hayan sido procesadas una vez, o es posible continuar disparando reglas hasta que se ha llegado a la conclusión deseada o hasta que la base de datos de proposiciones validadas no sea alterada mas por el proceso y por lo tanto no es posible llegar a la conclusión.

Como se asume que A y F son verdaderas, si se aplican todas las reglas a esta base de datos las únicas reglas que son disparadas son la 4 y la 8 y en la regla 8 se le asigna el valor verdadero a H, que es lo que se busca.

Del método anterior existen dos métodos de parada, cuando H se convierte en verdadero o cuando la base de datos no sufre más cambios después de aplicadas las reglas. La selección de una de estas dos técnicas depende del propósito para el cual fue construido el sistema. Un efecto interesante del procedimiento anterior consiste en que cuando se prueba la proposición requerida no se necesita ninguna computación adicional.

Estas estrategias son conocidas como encadenamiento hacia adelante [52] o razonamiento directo de datos, porque se comienza con datos conocidos y se aplican las reglas sucesivamente para poder encontrar la implicación de los resultados. Esta estrategia es apropiada en situaciones en donde los datos son costosos de coleccionar pero la cantidad de los mismos es baja [52].

b) Encadenamiento hacia atrás

Hay una forma totalmente diferente de abordar el problema mencionado, es decir probar la proposición que H es verdadera y consiste comenzar con la meta deseada 'H es verdadera' e intentar encontrar evidencia para que esto sea cierto. Este método se conoce como encadenamiento hacia atrás [52]. Se usa generalmente cuando lo único que se necesita es probar H y no se está interesado en los valores de otras proposiciones. Esta técnica no será utilizada en el presente proyecto debido a que los principales motores de reglas utilizados actualmente utilizan el método de encadenamiento hacia adelante mencionado anteriormente.

c) Rete

Rete es un algoritmo que evalúa un predicado declarativo contra un conjunto cambiante de datos en tiempo real. Este algoritmo construye una red ordenada de tipo árbol con todos los patrones que son definidos en las condiciones y se usa para revisar contra la base de conocimiento evitando la generación de ciclos internos. [20]

Este algoritmo es mucho más eficiente determinando la relevancia de las reglas, dados unos datos particulares, comparado con la eficiencia de las instrucciones *if/then/else* o *select/case*. A medida que crece el número de reglas aumenta la ventaja del algoritmo sobre el código procedimental.

Cuando aumenta la cantidad de reglas almacenadas en la base de conocimiento, el algoritmo de encadenamiento hacia adelante mencionado anteriormente puede responder de manera

muy lenta debido a que se realizan muy pocos cambios en la memoria de trabajo en cada ciclo cuando se evalúan las reglas.

3.3.5 Componentes y características técnicas de un Motor de Reglas

Los sistemas basados en reglas son sistemas de computación que pueden dar sugerencias o tomar decisiones en un área estrechamente definida o cercana al nivel de un humano experto [52]. Existen dos clases de este tipo de sistemas: sistemas que toman decisiones los cuales son principalmente usados para controlar procesos y para aplicaciones de tipo financieras, y sistemas que actúan como apoyo a las decisiones que se requieren tomar, dando sugerencias de posibles soluciones pero no tomando decisiones autónomas.

La característica más importante a nivel de arquitectura de este tipo de sistemas es que el conocimiento acerca de un problema es almacenado de manera separada del código que aplica el conocimiento para resolver un problema determinado usando una representación por reglas del conocimiento adquirido.

El repositorio con pedazos de conocimiento en un Motor de Reglas es conocido como la base de las reglas o la **base de conocimiento** y los mecanismos que aplican estos conocimientos a los datos se conoce como **motor de reglas** o motor de inferencia.

La base de conocimiento se encarga de manejar las reglas, así como también las consultas y referencias a recursos externos de reglas y hechos llamados conjunto de premisas [67].

Estas características se observa en la figura que se muestra a continuación.

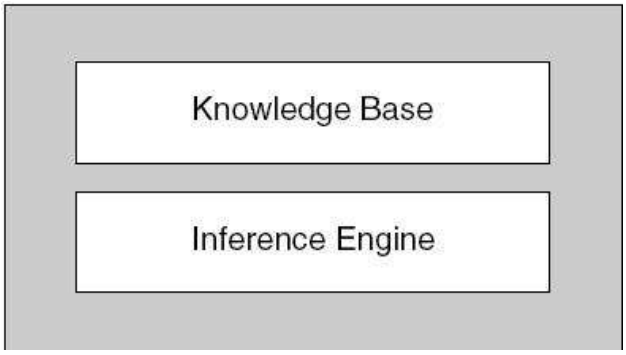

Ilustración 2 Componentes Motor de Reglas [52]

Existen cuatro componentes fundamentales en un BRMS (Business Rule Management System). [52]

1. El conjunto de símbolos y la manipulación de valores los cuales comparten todos los sistemas de computación haciéndolos similares a los lenguajes de programación.
2. La estructura de la base del conocimiento, incluyendo algunas técnicas que permitan a las aplicaciones hacer uso del conocimiento de una manera racional para estar en capacidad de resolver el problema que se intenta resolver.
3. El motor de inferencia que encadena las reglas establecidas para llegar a conclusiones validas.
4. Por último se tiene el repositorio, en el cual se almacenan las reglas y pueden ser manipuladas de diferentes maneras.

La base del conocimiento y el motor de inferencia están separados uno del otro para facilitar el mantenimiento. Adicionalmente, la base del conocimiento usualmente contiene diferentes tipos de conocimiento que incluye típicamente objetos, procedimientos y relaciones de causalidad.

3.3.6 Motor de Reglas Drools

Existen varios sistemas de reglas en el mercado, pero durante el desarrollo de este proyecto se utilizó la herramienta **Drools** [75] por la facilidad de integración que tiene con el servidor de aplicaciones Jboss el cual fue utilizado en el presente proyecto, y por el hecho de estar desarrollado para ser utilizado por aplicaciones escritas en Java.

Drools es una implementación extendida del algoritmo **rete** [75] descrito anteriormente ajustado al lenguaje Java. La adaptación de rete a una interfaz orientada a objetos permite expresiones más naturales de reglas de negocios con respecto a los objetos de negocio. Este motor de reglas utiliza un método de encadenamiento hacia adelante para evaluar las reglas (ver Encadenamiento hacia adelante, 4.4.4 a)), Drools asocia la semántica del rete normal relacional con un modelo orientado a objetos compatible con Java.

El motor de inferencia en Drools relaciona los hechos y los datos con las reglas de producción para inferir conclusiones que resultan en acciones. Una regla de producción es una estructura compuesta por 2 partes como se muestra a continuación [71].

```
When
     <Condiciones>
Then
     <Acciones>
```

A continuación se hace una descripción de las principales características del motor de reglas Drools.

a) Base del Conocimiento

La base de conocimiento de Drools es un repositorio que almacena todas las definiciones de conocimiento de la aplicación. Puede contener reglas, procesos, funciones y tipos de modelos.

La base del conocimiento como tal no contiene instancias de los datos, estos son conocidos como hechos; para almacenar estas instancias se crean sesiones a partir de la base de conocimiento que además permiten iniciar las instancias de los procesos definidos. La creación de la base del conocimiento consume muchos recursos computacionales mientras que la creación de la sesión es muy ligera, de modo que es recomendable que solo sea creada una base de conocimiento con múltiples sesiones.

El siguiente fragmento de código muestra un ejemplo acerca de cómo se crea una base de conocimiento a partir de código Java.

```
KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase();
```

b) Sesión de Conocimiento sin estado

Una sesión sin estado puede ser llamada como una función, que recibe valores de entrada y retorna resultados, algunos de los usos comunes de este tipo de sesiones son [71]: validaciones, cálculos, filtros.

A continuación se muestra un ejemplo detallado de una sesión sin estado:

Primero se define una clase que representa la información de un empleado.

```
public class Empleado {

   private String nombre;
   private int edad;
   private String cargo;
   private boolean valido;

   // metodos get y set
```

Luego de definir el modelo se procede a definir una regla que permita validar un empleado de acuerdo a restricciones definidas.

```
rule "Is of valid age"
when
   $a : Empleado( edad < 18 )
then
   $a.setValido( false );
end
```

Para que se pueda activar la regla que se acaba definir es necesario agregar los datos a la sesión insertando una o varias instancias de la clase que fue definida previamente. Después de la inserción el motor de reglas se encarga de evaluar las reglas contra las restricciones. En la regla anterior se definió una variable $a que permite referenciar el objeto. El carácter '$' es opcional pero ayuda a diferenciar definición de variables de los nombres de los campos. En el proceso de evaluación de cada regla definida se realiza un consulta sobre la base de conocimiento donde se obtienen las instancias de los objetos que cumplen con las restricciones definidas.

Luego de haber definido el archivo de reglas es necesario crear la base del conocimiento que contiene una colección de todas las reglas compiladas y listas para ejecutarse. Las reglas se compilan a través del un objeto de tipo **KnowledgeBuilder**, a continuación se muestra como se realiza este proceso.

```
KnowledgeBuilder kbuilder = KnowledgeBuilderFactory.newKnowledgeBuilder();
  kbuilder.add( ResourceFactory.newClassPathResource("Descuentos.drl"),  ResourceType.DRL);
  KnowledgeBuilderErrors errors = kbuilder.getErrors();

  if (errors.size() > 0) {
   for (KnowledgeBuilderError error: errors) {
    System.err.println(error);
   }

   throw new IllegalArgumentException("Could not parse knowledge.");
  }

  KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase();
  kbase.addKnowledgePackages(kbuilder.getKnowledgePackages());
```

En la instancia del KnowledgeBuilder es necesario definir el nombre del archivo de reglas que se va a compilar, además del tipo de archivo que por lo general es un archivo .DRL; luego de crear el KnowledgeBuilder se verifica si se producen errores en la compilación, si el proceso se completa con éxito se procede a crear la base de conocimiento a la cual se agrega el conjunto de reglas que ya fue previamente compilado de modo que se asegura que las reglas están escritas correctamente.

```
StatelessKnowledgeSession ksession = kbase.newStatelessKnowledgeSession();
Empleado empleado1 = new Empleado("Edgar", 16, "Desarrollador");
Empleado empleado2 = new Empleado("Adolfo", 17, "Analista");
Empleado empleado3 = new Empleado("Juan", 19, "Futbolista");
ksession.execute(empleado1);
ksession.execute(empleado2);
ksession.execute(empleado3);
```

El fragmento de código mostrado ejecuta las reglas definidas. Como los empleados 1 y 2 son menores de 18 años los registros se marcan como inválidos, para el empleado 3 el registro no se marca como inválido.

c) Sesión de Conocimiento con estado

Las sesiones con estado tienen una vida más larga y permiten cambios iterativos con el tiempo. Este tipo de sesiones se usan comúnmente para resolver problemas de este tipo: monitoreo, diagnóstico, logística.

A continuación se ilustra un ejemplo [71] de monitoreo que permite entender mejor las sesiones con estado. Se definieron cuatro clases y un archivo de reglas:

```
public class Cuarto {
   private String nombre
   // métodos get y set acá
}
public class Extintor {
   private Cuarto cuarto;
   private boolean encendido;
   // métodos get y set acá
}
public class Fuego {
   private Cuarto cuarto;
   // métodos get y set acá
}
public class Alarma{
}
```

En este ejemplo se asume que una casa puede tener muchos cuartos por lo tanto las reglas deben expresar relaciones entre los objetos

```
rule "Cuando hay fuego encienda en extintor"
when
   Fuego($cuarto : cuarto)
   $extintor : Extintor(cuarto==$cuarto, encendido==false)
then
   modify($extintor){setEncendido(true)};
   System.out.println("Encendido el extintor para el cuarto" + $cuarto.getNombre());
end
```

En la regla definida anteriormente se utiliza la palabra reservada *"modify"*. Esta palabra permite modificar los datos y le permite al motor de reglas estar pendiente a las modificaciones de estos datos para razonar a partir de estas modificaciones de nuevo. Este proceso se llama inferencia y forma parte esencial de las sesiones con estado.

En la regla anterior se verifica si hay un incendio en algún cuarto y el detector de ese incendio está apagado, entonces en la consecuencia se cambia el valor del detector a encendido, y se imprime por consola el cambio que se registro en el detector de ese cuarto.

Para ejecutar las reglas se sigue el mismo proceso mencionado para la sesión sin estado, creando la base de conocimiento y posteriormente la sesión y agregando los datos.

d) El lenguaje de reglas

Archivos de Reglas

Un archivo de reglas es un archivo con extensión .drl. En este tipo de archivos se pueden definir múltiples reglas, consultas y funciones. También se pueden definir reglas en múltiples

archivos lo cual puede ayudar a gestionar la definición de las mismas cuando el número de reglas es elevado.

Reglas

En la siguiente imagen [71] se muestra un esquema con cada una de las partes que componen una regla. Cada regla está compuesta por un grupo de atributos que pueden ser usados en la definición de la regla.

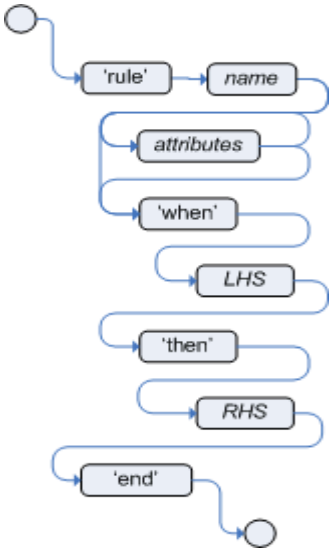

Ilustración 3 Componentes de las reglas [71]

Una regla define que cuando un grupo particular de condiciones se cumple, entonces se hace lo que se especifica como una lista de acciones o consecuencias las cuales se definen en el '*then*' sobre cada una de las instancias del objeto seleccionado. A diferencia del *if/else*, un *when* no está atado a una evaluación específica o a un determinado momento en el tiempo, simplemente indica que se evaluará continuamente un regla durante el tiempo que el motor de reglas dure en ejecución.

Una regla debe tener un nombre único dentro de un mismo archivo, de lo contrario se producirá un error en la compilación. Si el nombre de la regla contiene espacios, entonces debe ser definido dentro de comillas dobles.

Atributos de las reglas

Los atributos de las reglas proporcionan una manera declarativa de influenciar el comportamiento de una regla. Es necesario entender estos atributos para hacer un uso adecuado del motor de reglas Drools; a continuación se muestran los principales.

- **No-loop**: cuando la consecuencia de la regla modifica un hecho puede ocurrir que se active de nuevo causando recursión. Asignándole true a no-loop la regla solo será evaluada una vez y los cambios en los datos serán ignorados.
- **Ruleflow-group**: brinda control para activar las reglas por grupos.
- **Salience**: este valor es un entero. Por defecto es cero y permite organizar las reglas definiendo prioridades por medio de los números asignados a cada regla, de tal forma que se ejecutan primero las reglas con menor valor en el atributo *Salience*.
- **Activation-group**: la primera regla que se dispare que forme parte de un *activation group* cancela la ejecución de las demás reglas que pertenecen al mismo *activation-group.*
- **Dialect**: especifica el lenguaje que va a ser usado para escribir las reglas. Existen dos dialectos disponibles Java u MVEL.
- **Date-expires**: un regla no puede ser activada si la fecha actual es posterior al atributo date-expires que fue definido.
- **Duration**: un entero que define el tiempo que va a esperar una regla para ser disparada si todavía es verdadera.

Componente condicional de las reglas

El componente condicional de las reglas está donde se define la palabra ‘when’ y se compone de uno o más elementos condicionales, si se deja vacio este segmento la regla se redefine y la condición siempre será verdadera de modo que se activará una vez siempre que se inserta una nueva sesión a la base del conocimiento.

Los elementos condicionales funcionan a través de uno o más patrones. En su forma más simple, es decir sin restricciones, un patrón se asocia a un hecho de un tipo determinado. Si por ejemplo se tiene definida una clase ‘Empleado’ eso significa que el patrón asociará todas las instancias del objeto existentes en la base de conocimiento. Para obtener una referencia a

un objeto se debe definir una variable como por ejemplo '$e', un ejemplo de la asociación se muestra a continuación:

```
$e : Empleado()
```

Dentro del paréntesis es donde se pueden establecer restricciones para que se tengan en cuenta solo ciertas instancias de los objetos que se ingresan a la base de conocimiento.

En las restricciones pueden ser usados los conectores and (&&), or (||) además de diversos operadores, estos son utilizados de manera muy similar a como se hace en Java lo cual permite que no se requiera de mucho tiempo para entender la sintaxis de este lenguaje. En la imagen que se muestra se puede observar un patrón definido con restricciones.

```
// El tipo de empleado es "temporal" and edad < 24, and ciudad is bogota
Empleado ( tipo == "temporal" && edad < 24, ciudad == "bogota" )
```

En el ejemplo mostrado se tienen tres restricciones las cuales se encuentran separadas por ',' y '&&'. En el orden de prioridades de evaluación de componentes tiene prelación el conector &&, luego el conector || y por último se ejecuta el conector',’. Aunque se pueden establecer nuevas prioridades de evaluación por medio del uso de paréntesis para separa los conectores.

Un campo se deriva de un método al que se puede acceder por medio del objeto. Estos son los métodos *get ()* y *set()*. Por ejemplo volviendo a la clase Empleado, el patrón *Empleado (tipo==”Temporal*”) aplica el método *getTipo ()* que se define en la clase empleado.

Acciones

Esta parte de las reglas es la que se define después de la palabra '*then*' y debe contener una lista de acciones a ser ejecutadas. No se debe usar código condicional en esta parte de la regla debido a que todas las condiciones necesarias para llegar a la acción se deben definir en el '*when*'. No debe contener muchas instrucciones debido a que debe ser declarativo y se debe poder entender con facilidad. El principal propósito de las acciones es el de insertar, modificar o eliminar parte de los datos con los cuales está trabajando el motor de reglas.

4 DESARROLLO DEL TRABAJO

En este capítulo se describen las estrategias utilizadas para construir los módulos del sistema Sispen, haciendo uso de motores de reglas, descriptores XML, opciones de seguridad y herramientas que permiten la elaboración de reportes. Estos módulos son los siguientes:

- Módulo de Seguridad: Presta todos los servicios a los usuarios que controlan los permisos del ingreso a la aplicación y donde se define a qué opciones del sistema tiene acceso un determinado usuario de acuerdo con su perfil de seguridad.
- Módulo de Auditoria: Este modulo permite llevar un registro detallado de las acciones que los usuarios realizan sobre la base de datos.
- Módulo Alimentar Sistema: se ocupa de realizar el procedimiento de cargar archivos al sistema con información de los empleados, tablas de esperanza de vida, salarios mínimos y parámetros, las cuales son necesarias para el cálculo del pasivo pensional.
- Módulo Cálculos del Sistema: Este módulo realiza el proceso de cálculos de las reservas por medio del uso de un motor de reglas para facilitar la modificación de estas por parte de los clientes.
- Módulo de Reportes: le permite al usuario ver el resultado de los cálculos los cuales son generados en archivos que son almacenados en el directorio web de la aplicación con formato PDF.

4.1 Generación inicial de la aplicación mediante los generadores Seam y Taylor

Con la herramienta Taylor [12] se hizo el modelo UML de las entidades persistentes de la aplicación y se generaron las clases Java correspondientes, incluyendo las anotaciones necesarias. El modelo de entidades se puede ver en el manual de mantenimiento que forma parte de los anexos.

El generador de Seam [22] le dió las bases al presente proyecto, ya que con este se creó la estructura inicial de la aplicación, se generaron los casos de uso CRUD de las entidades que teníamos inicialmente, y sirvió para ir agregando casos de uso a medida que la aplicación se iba desarrollando. A continuación se describen las tareas del generador que fueron utilizadas.

Tarea *setup*: Esta tarea permite definir las propiedades que va a tener el proyecto que se desea crear. Como resultado de esta tarea, el driver JDBC definido es instalado en el servidor de aplicaciones Jboss y se crea el archivo de propiedades del proyecto *build.properties* bajo el subdirectorio seam-gen.

Tarea *create-project*: esta tarea permite la creación del proyecto Seam inicial sobre el cual va a ser desarrollada la aplicación. La creación del proyecto se basa en las propiedades definidas en el paso anterior (tarea *setup*). Los componentes creados son los siguientes:

- Librerías y descriptores necesarios
- Herramienta ANT que permite compilar y publicar la aplicación.
- Archivos de propiedades de idiomas.
- Datasource que permite la conexión con la base de datos.
- Páginas iniciales de ingreso al sistema.
- Módulo de autenticación por defecto.

Tarea *generate-UI*: Esta tarea genera casos de uso CRUD para todas las entidades existentes en el proyecto (que previamente fueron generadas con la herramienta Taylor). Después de la generación se pueden ver las páginas de cada uno de los casos de uso que permiten la modificación, inserción, consulta y eliminación de una instancia de las entidades definidas. Además se pueden ver las relaciones entre entidades por medio de la navegación entre páginas.

Como resultado de la generación se agregara a la página inicial de la aplicación el menú de cada uno de los casos de uso CRUD que fueron generados para acceder a las respectivas páginas.

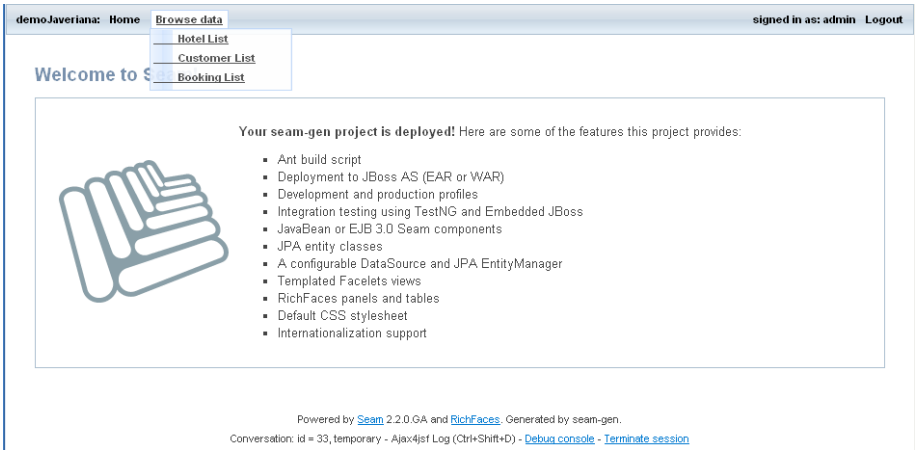

Ilustración 4 Generación páginas CRUD

En caso de que no se quieran generar casos de uso CRUD para todas las entidades definidas basta con añadir un comentario en la anotación @Entity de cada una de las entidades que no se quieren generar.

Tarea ***new- action***: esta tarea permite la generación de nuevos casos de uso y de sus respectivas páginas de tipo acción sin forma. Además de la página con un botón para invocar la acción, se genera un *EJB* de sesión sin estado y un test unitario JUNIT.

Para observar el funcionamiento de la página generada se debe republicar el proyecto y así se puede apreciar una página como la mostrada en la siguiente figura.

Ilustración 5 tarea *new-action*

4.2 Módulo de Seguridad basado en roles finos y perfiles

4.2.1 Uso del módulo CincoSecurity

En el desarrollo de este proyecto se tomó como base el módulo de seguridad de código abierto CincoSecurity [72] para asegurar el acceso a la aplicación. Este módulo ofrece una gran flexibilidad para proteger cada uno de los métodos de los EJB por roles finos en una aplicación desarrollada en Java EE 5 por medio de anotaciones. También es posible proteger por roles el acceso a una página y a los elementos que componen las paginas JSF. Este módulo introduce el concepto de perfiles de seguridad los cuales tienen asociados un conjunto de roles.

Este módulo de seguridad le proporciona al usuario varios casos de uso. El caso de uso para gestión de perfiles de seguridad muestra cada uno de los módulos que son utilizados por la aplicación, los casos de usos asociados a ese módulo, y por cada caso de uso muestra los servicios ofrecidos por este. Cada servicio corresponde a un método de un EJB que soporta el caso de uso y para cada servicio existe un rol de seguridad asociado. El caso de uso de gestión de perfiles permite definir un nuevo perfil de seguridad seleccionando los roles asociados al nuevo perfil.

El caso de uso gestión de usuarios permite registrar un nuevo usuario y asociarlo a uno o muchos perfiles de seguridad.

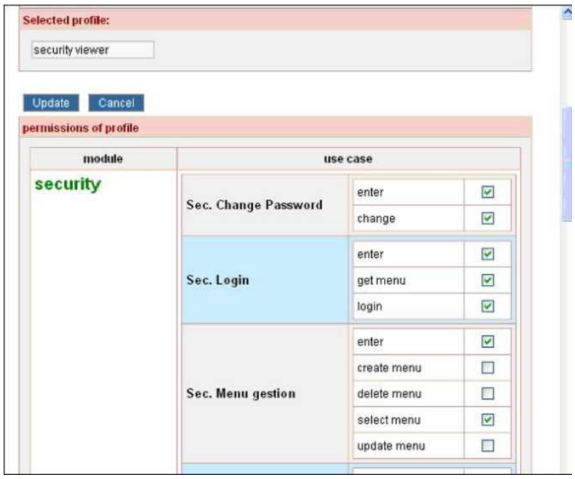

Ilustración 6 Perfiles de Usuarios y Servicios

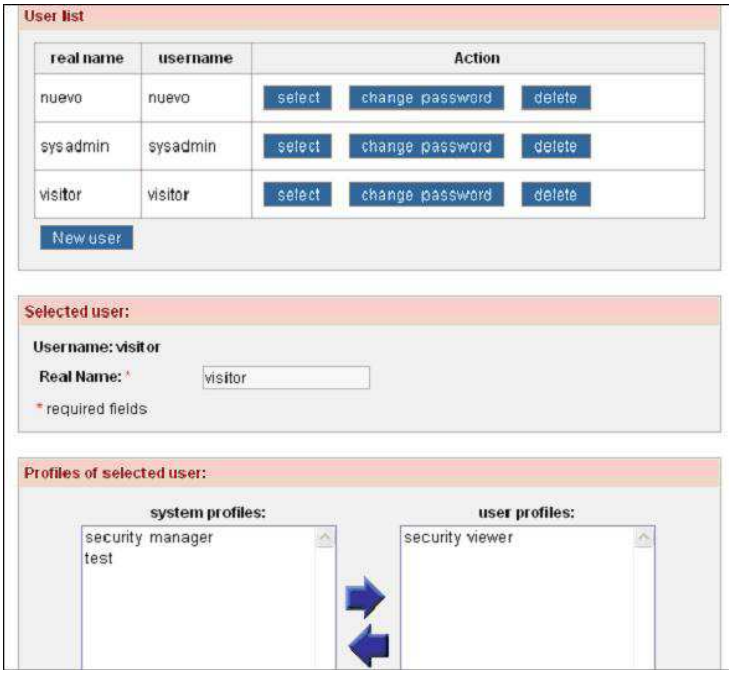

Ilustración 7 Asociación de Usuarios y Perfiles de seguridad

La aplicación "**SisPen**" obtuvo beneficios incorporando este módulo ya que por ejemplo es capaz de capturar los roles asociados al usuario actual autenticado y carga solamente las opciones permitidas para ese usuario en el menú de opciones. Además la aplicación "SisPen" va a tener los casos de uso para la gestión de perfiles de seguridad y de usuarios, y CRUD's para registrar módulos, casos de uso y servicios. De esta manera "SisPen" puede continuar protegiendo los elementos de nuevos casos de uso y de nuevos módulos.

A continuación se hace una breve descripción de cada uno de los demás casos de uso que ofrece el sistema CincoSecurity, un caso de uso representa un conjunto de servicios implementados por un EJB de sesión junto con un conjunto de páginas JSF que permiten invocar estos servicios:

- CRUD de Módulos: Un modulo es un conjunto de casos de uso relacionados. Cincosecurity brinda la posibilidad de agregar nuevos casos de uso y agregarlos a módulos definidos.
- CRUD de Servicios: Un caso de uso está compuesto por servicios que son representados a través de métodos o acciones de los EJB de sesión . Este modulo permite definir cada

uno de los servicios y asociarlos a un caso de uso determinado. Estos servicios definidos serán invocados por las paginas jsf durante la ejecución de la aplicación.

- CRUD de Menús: la gestión de menús permiten agregar cada uno de los casos de uso y módulos del sistema al menú principal de la aplicación. Generalmente los casos de uso son definidos como submenús de los módulos a los cuales pertenecen.
- Parámetros de la aplicación: este caso de uso permite la definición de parámetros adicionales que son necesarios para la aplicación como por ejemplo el valor del tiempo máximo de duración de una sesión o de una conversación de los usuarios que ingresan al sistema.

4.2.2 Proceso de Agregación de un nuevo caso de uso

Una vez acoplado el módulo de CincoSecurity a una aplicación generada Seam, siguiendo las indicaciones del manual de CincoSecrity [73], se pueden agregar nuevos casos de uso manualmente o por medio de los generadores de Seam.

El manual de CincoSecurity indica las precauciones y pasos a seguir para registrar en la aplicación nuevos módulos, nuevos casos de uso y nuevos servicios, de modo que los roles asociados con los nuevos servicios sean incorporados a los perfiles de seguridad. De esta manera se puede seguir manteniendo la seguridad a medida que se extiende la aplicación.

4.3 Módulo de Auditoria mediante *listeners* y métodos *call-back*

4.3.1 Descripción

El objetivo de este módulo es llevar un registro detallado de las operaciones que se realizan sobre la base de datos por parte de los usuarios. Se hace un registro de antes y después de una operación hecha sobre la base de datos mediante el uso de los métodos *Callback*.

- Métodos *Callback*: estos son métodos que se ejecutan cuando se presentan ciertos eventos determinados. Se pueden definir este tipo de métodos antes y después de las distintas operaciones hechas sobre la base de datos de la aplicación. Un método de una entidad puede ser designado como un método Callback para recibir notificaciones a partir de un evento particular [2].

- *EntityListener*: Un entity listener es una clase sin estado y se define usando la anotación @EntityListeners. Esta clase se utiliza para definir las acciones que se van a ejecutar cada vez que sea invocado un método Callback. [2]

A continuación se describen los métodos Callback existentes y cómo pueden ser utilizados. [2]

Método Callback	Descripción.
@PostLoad	Se invoca el método definido con esta anotación después de cargar una entidad. A partir de la base de datos
@PostPersist	Se invoca el método definido con esta anotación después de persistir una entidad. En la base de datos
@PostRemove	Se invoca el método definido con esta anotación después de remover una entidad. En la base de datos
@PostUpdate	Se invoca el método definido con esta anotación después de actualizar una entidad. En la base de datos
@PrePersist	Se invoca el método definido con esta anotación antes de persistir una entidad.
@PreRemove	Se invoca el método definido con esta anotación antes de remover una entidad
@PreUpdate	Se invoca el método definido con esta anotación antes de actualizar una entidad

4.3.2 Descripción del proceso de auditoría.

El proceso para la construcción del módulo de auditoría que se siguió en el proyecto fue el siguiente:

- Definir una clase Entity Listener encargada de implementar los métodos *Callback* descritos en la sección anterior (más adelante se mostrará un ejemplo de esta clase).
- Agregar anotación de la clase Entity listener a cada clase entidad que vaya a ser auditada: es necesario agregar la anotación @EntityListeners con el nombre de la clase Entity listener Por ejemplo, si se tiene una entidad llamada Parameter que va a ser auditada, su encabezado sería el siguiente:

```
@Entity
@EntityListeners ({Listener.class})
public class Parameter implements java.io.Serializable {
```

- Contenido de la clase *Listener*:

  Esta clase es la encargada de registrar todas las acciones de los usuarios sobre la base de datos. Cada vez que un usuario hace una operación se crea y se persiste una nueva instancia de la entidad **Auditoria** donde se guarda el nombre del usuario, la fecha, la hora y la acción que fue realizada por el usuario. Estas acciones pueden ser de diferentes tipos:

  - Persistencia:
  - Eliminación:
  - Actualización

  Para el caso del presente proyecto el nombre de la clase es “Listener.java”.

  Para poder cumplir con el objetivo de agregar un nuevo registro de auditoría con la nueva acción hecha por el usuario es necesario seguir los siguientes pasos:

- Obtener una instancia del EntityManager que es el framework manejador de persistencia el cual no se puede obtener por medio de anotaciones (inyección) debido a que es una clase Java normal y no un EJB de sesión
- Obtener el usuario de la sesión actual, a partir de la variable "customer" del contexto de sesión utilizando el entity manager obtenido de manera programática como se mencionó en el paso anterior.
- Construir una nueva instancia de Auditoria, pasando al constructor la información del usuario y la hora actual
- Persistir la nueva instancia de Auditoria

Como ejemplo se muestra el método callback que se ejecuta después de persistir una entidad en la base de datos:

```
@PostPersist
public void postPersist(Object obj) throws ParseException, NamingException {
   if (obj instanceof Employee) {
      grabarAuditoria("Crear Employee", "@PostPersist",
String.valueOf(((Employee)obj).getId()));
   }
   else if (obj instanceof Customer) {
      grabarAuditoria("Crear Customer", "@PostPersist", ((Customer)obj).getUsername());
   }
}
```

La anotación @PostPersist permite que este método sea invocado cada vez que se haya persistido un entidad en la base de datos.

4.4 Módulo Alimentar Sistema: validar archivos de datos mediante descriptores XML y reglas

4.4.1 Resumen

El módulo Alimentar Sistema se compone de cuatro casos de uso: Cargar Información al Sistema, Cargar Tablas de Cálculos, CRUD de Usuarios y CRUD de Empresas.

Los casos de uso CRUD de Usuarios y CRUD de Empresas fueron relativamente sencillos de desarrollar pues uno de ellos estaba parcialmente desarrollado por el módulo de seguridad CincoSecurity (CRUD de Usuarios) y el otro se obtuvo mediante los generadores del *framework* Seam.

El proceso seguido para desarrollar los casos de uso Cargar Información al Sistema y Cargar Tablas de Cálculos fue similar porque en ambos se requiere leer un archivo de entrada con extensión .csv, verificar los datos que vienen en el archivo, y finalmente guardar los datos en la base de datos. Estos dos casos de uso utilizan el directorio web de archivos de la aplicación SisPen para sacar de allí los descriptores XML necesarios y alojar los archivos subidos por el usuario, así como los generados por la aplicación.

4.4.2 Directorio Web de Archivos

El directorio web de archivos es de gran utilidad para la aplicación SisPen, pues le permite tener acceso y alojar los diferentes archivos con los que ésta funciona. Los casos de uso Cargar Información al Sistema y Cargar Tablas de Cálculos leen los archivos de descriptores XML y escriben archivos subidos por el usuario en el directorio web. El directorio web de archivos de la aplicación SisPen es la carpeta PENSIONES.war que se pone en la carpeta *...\jboss-5.1.0.GA\server\default\deploy* del servidor JBoss; esta carpeta contiene a su vez varias carpetas para guardar los archivos de una forma organizada. Una de las carpetas más importantes dentro de PENSIONES.war es la que contiene a los descriptores XML, llamada "Descriptores", allí están alojados todos los descriptores con los que se verificará que la información subida por los usuarios a SisPen es correcta, de igual manera el usuario administrador de SisPen podrá navegar por algunas de las carpetas del directorio web para ver los archivos subidos por los usuarios a la aplicación.

El directorio web de archivos debe ser configurado como se indica en el manual de instalación de la aplicación SisPen.

4.4.3 Cargar Información al Sistema

Este caso de uso se encarga de guardar en la base de datos información correspondiente a los empleados de una empresa. Para realizar esta función se leen los datos del archivo de entrada, se validan estos datos, y luego se guardan en la base de datos. La parte más importante de este caso de uso es el proceso de validación de datos ya que allí se usan descriptores XML y el motor de reglas Drools [71].

Para validar los datos ingresados al sistema usamos la librería de manejo de archivos desarrollada por la empresa CincoSoft [74] la cual se basa en describir el formato de cada archivo con un descriptor XML para poder validar tipos. Estos descriptores permiten definir un archivo con extensión .xml donde se especifica el formato de los datos que van a ser ingresados, permitiendo para cada dato detallar el tipo de dato que se ingresa, como por ejemplo *long*, *double*, *text*, entre otros. Si el tipo de dato que se ingresa no llega a corresponder con el definido en el descriptor se genera un mensaje de error que se guarda en una lista de *Strings*, este error indica el valor que no pudo ser leído y el nombre del campo donde surgió el error.

A continuación se muestra un ejemplo de un descriptor XML con tres campos, en la parte superior se especifica que los datos van a estar separados por ";" y en los datos se especifica el tipo de dato que debe tener lo que se ingresa y el valor, que en este caso puede ser cualquier valor ya que se pone como vacío; la parte que indica el formato se usa para tipos de datos Date, si queremos indicar que la fecha debe seguir cierto orden:

```
<?xml version="1.0"?>

<global-flat-file superoot='sispen' name='SisPen,yyyy,_,MM,_,dd,_,hh,_,mm,_,ss,

  <!-- ========================== -->
  <!-- FORMATO DATO               -->
  <!-- ========================== -->

  <flat-file mode='separator' separator=';' root='sispen' register='DATO'>

    <field name='ano'        position='-1' type='long' format='' value='' />
      <!-- Aca poner comentarios relacionados con el atributo ano-->

    <field name='ipc' position='-1' type='double' format='' value='' />
      <!-- Aca poner comentarios relacionados con el atributo ipc-->

    <field name='interes'      position='-1' type='double' format='' value='' />
      <!-- Aca poner comentarios relacionados con el atributo interes-->

  </flat-file>

</global-flat-file>
```

Para mostrar los errores surgidos del ingreso de datos se genera un archivo de texto con el mismo nombre del archivo original más un guíon y un "no", quedando así: "*nombreArchivo-no.txt*"; en este archivo por cada registro incorrecto se incluye el número de la línea que tuvo el error, la línea que tuvo el error, y la línea de error que genera el código de descriptores XML, a continuación se muestra el ejemplo de lo que resulta después de un error:

```
Línea 1
1;2;1.5;5.5;2;2.5;5.5;12345;05/06/2005;76543;25/07/2002;0;3;5;7;1;2;1234567890;1;06,
origenDeLaPension: El valor del campo es '0' y no es correcto porque es menor a 1
```

Para mostrar los registros correctos que pudieron ser ingresados a la base de datos se genera un archivo de texto con el mismo nombre del archivo original más un guión y un "si", quedando así: "*nombreArchivo-si.txt*", dentro de este archivo, por cada registro correcto se incluye el número de la línea correcta, la línea completa correcta, y una notificación de que el

registro fue ingresado a la base de datos con determinado identificador; a continuación se muestra un ejemplo de un registro de una línea correcta:

```
Línea 14
1;2;1.5;5.5;2;2.5;5.5;12345;05/06/2005;76543;25/07/2002;3;1;5;7;
*** Registro ingresado a la base de datos con id = 15470
```

Los archivos de registros correctos e incorrectos se muestran por medio de enlaces en la página de la aplicación encargada de este caso de uso, una vez el usuario ha terminado de cargar exitosamente un archivo, como se muestra en la siguiente figura.

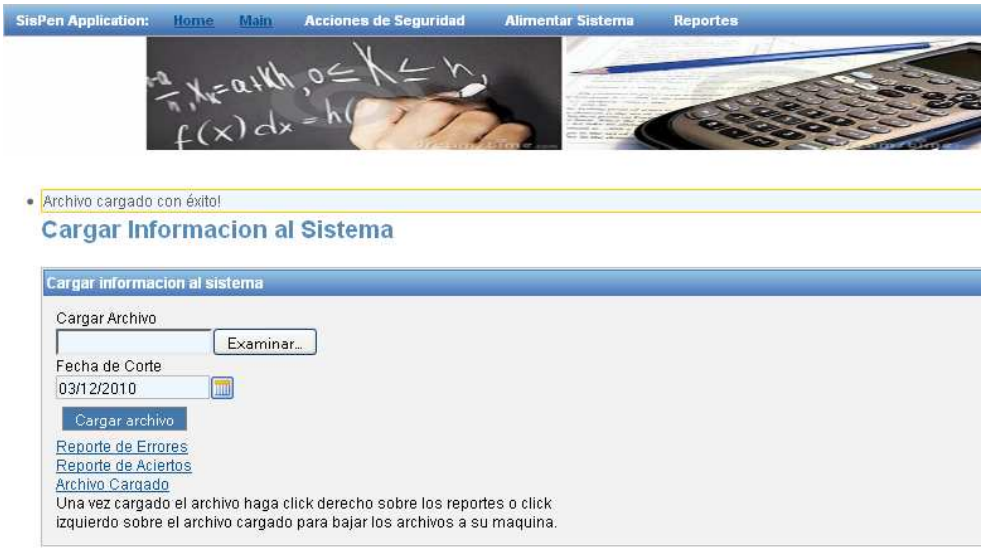

Ilustración 8 Cargar Información al sistema

Adicionalmente a las validaciones de tipo, se realizaron validaciones de negocio sobre los datos de cada archivo que se quiere subir a la aplicación. Para ello se utilizó el motor de reglas Drools [71]; por medio de las reglas se validó que los valores ingresados estuvieran dentro de un rango.

Para validar los datos que se ingresan al programa mediante las reglas se siguió este proceso: se pasan los datos ingresados a un objeto en Java; ese objeto es insertado dentro de otro objeto que además tiene una lista de errores y una lista de aciertos; este último objeto se inserta a la sesión de conocimiento del motor de reglas, y se disparan las reglas. En las reglas se determina si los campos del objeto son incorrectos, de serlo se ingresa una línea de error en

la lista de errores que luego va a ser mostrada en un archivo de texto como se explicó anteriormente en esta sección.

A continuación se muestra el ejemplo de una regla que valida el salario mínimo de tal forma que éste sea mayor o igual a cero, de no serlo se ingresa un error a la lista de errores, y se marca una variable booleana para indicar que el registro no es bueno:

```
rule "Verificación SalarioMinimo - salario >= 0"
  when
    salarioMinimoExt : SalarioMinimoExtendido( parametro.salarioMinimo < 0 )
  then
    salarioMinimoExt.setRegistroBueno( false );
    salarioMinimoExt.addErrores("salario: El valor del campo es '" + salarioMin
        "' y no es correcto porque es menor a 0");
end
```

Desafortunadamente la información de entrada de empleados en varios campos puede ser vacía, imposibilitando la validación mediante descriptores XML y reglas; en estos casos se verificó manualmente si el campo venía vacío o no, de tal forma que si venía vacío no sacara error. Sin embargo no se pudo hacer la validación mediante las dos herramientas mencionadas anteriormente.

4.4.4 Cargar Tablas de Cálculos

Este caso de uso se encarga de guardar en la base de datos registros con información que debe ser utilizada para realizar los cálculos de reservas pensionales. Para hacer esto se sigue el mismo proceso que se siguió en el caso de uso Cargar Información al Sistema, es decir, se lee información de un archivo .csv, se verifica la información, y luego se guarda en la base de datos; para verificar la información se usan descriptores XML y reglas, los cuales se explicaron en la sección anterior.

Las tablas que pueden ser subidas al sistema son seis: mujeres válidas, mujeres inválidas, hombres válidos, hombres inválidos, parámetros cálculo y salarios mínimos.

4.5 Módulo Cálculos del Sistema: mediante reglas declarativas y el motor de reglas Drools

4.5.1 Resumen

El módulo cálculos del sistema es uno de los más importantes de la aplicación SisPen porque en este se desarrolla la más importante funcionalidad del sistema que consiste en calcular las reservas en dinero que deben tener las empresas para cumplir con sus obligaciones pensionales futuras.

Este módulo se compone de cuatro casos de uso: Calcular Invalidez Vejez Jubilación, Calcular Sobrevivencia, Calcular Auxilio Funerario; y Calcular Total Reservas. Todos los casos de uso que componen este módulo se llevan a cabo en el mismo archivo del programa por lo que su desarrollo en general fue semejante, por esta razón sólo vamos a explicar un cálculo en detalle ya que con uno que se explique se expone el funcionamiento de los demás cálculos del sistema.

Como queda claro después de leer el marco teórico del presente documento, para desarrollar los cálculos se usó un motor de reglas llamado *Drools* [71], este motor de reglas nos permitió simplificar la lógica del negocio para que sea más fácil entender y modificar.

Para usar el motor de reglas en la aplicación SisPen se declaran dos archivos asociados a este, uno con extensión .drl donde se declaran todas las reglas y otro que es una clase Java donde se crea una base de conocimiento y se asocia al archivo .drl. Luego se retorna la base de conocimiento a un Bean Java y a partir de esta base de conocimiento se crea la sesión de conocimiento, que es donde se insertan los objetos Java y se disparan las reglas. A continuación se muestra la clase Java donde se crea la base de conocimiento:

```
public static KnowledgeBase readKnowledgeBase()
throws Exception {
    KnowledgeBuilder kbuilder =
        KnowledgeBuilderFactory.newKnowledgeBuilder();
    kbuilder.add(
            ResourceFactory.newClassPathResource("ReglasCalculosEmpleado.drl"),
            ResourceType.DRL);
    KnowledgeBuilderErrors errors = kbuilder.getErrors();
    if (errors.size() > 0) {
        for (KnowledgeBuilderError error: errors) {
            System.err.println(error);
        }
        throw new IllegalArgumentException("Could not parse knowledge.");
    }
    KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase();
    kbase.addKnowledgePackages(kbuilder.getKnowledgePackages());
    return kbase;
}
```

Para usar las clases mostradas en el ejemplo anterior no fue necesario incluir ninguna librería a la aplicación, pues estas ya venían incluidas desde la generación del proyecto con el *framework* Seam.

A continuación se muestra un ejemplo donde se declara una base de conocimiento, se declara una sesión de conocimiento, se insertan objetos a la sesión de conocimiento y se disparan las reglas:

```
//Acá se crea la base de conocimiento y se ejecutan las reglas
KnowledgeBase kbase = null;
try {
    ////////////////////////// Cargar las reglas//////////////////////////////
    kbase = DroolsParametroCalculo.readKnowledgeBase();
    StatefulKnowledgeSession ksession = kbase.newStatefulKnowledgeSession();
    KnowledgeRuntimeLogger logger =
    KnowledgeRuntimeLoggerFactory.newFileLogger(ksession, "test");
    for(int p=0; p<listaParametrosCalculoExt.size(); p++){
        ksession.insert(listaParametrosCalculoExt.get(p));
    }
    ksession.fireAllRules();
} catch (Exception e) {
    e.printStackTrace();
}
```

Los archivos con extensión .drl deben estar en el subdirectorio *"resources"* de la aplicación SisPen, además es necesario aclarar que se usaron diferentes archivos de reglas .drl; uno para especificar las reglas de cálculos y otros para especificar las reglas de validación de los datos de entrada.

4.5.2 Calcular Invalidez Vejez Jubilación

El cálculo de invalidez vejez jubilación se hace para cualquier afiliado o asegurado pensionado o con expectativa de pensionarse y representa el dinero que debe tener una entidad para pagarle a una persona una pensión.

El proceso que se sigue para hacer el cálculo de invalidez vejez jubilación es el siguiente:

- Se termina de cargar el archivo de empleados a la aplicación SisPen
- Se calculan datos que deben ser usados en las reglas
- Se insertan a la sesión de conocimiento del motor de reglas todos los objetos empleados (cargados en memoria a partir de la base de datos) y tablas de cálculos necesarias para hacer los cálculos
- Se disparan las reglas
- Se guardan en la base de datos las modificaciones hechas a los objetos empleados con los resultados de los cálculos.

A continuación se muestra un ejemplo de una regla donde se calcula Invalidez Vejez Jubilación. En esta regla se utilizan tres tipos de objetos inicialmente: un empleado donde finalmente van a quedar los resultados de los cálculos, un EJB soporteReglas que como su nombre lo indica ayuda a las reglas a hacer persistir la información del empleado en la base de datos, y una instancia del objeto esperanzaDeVida, que tiene datos de parámetros que son usados para realizar los cálculos.

Para los objetos empleado y esperanzaDeVida se validan ciertos datos dentro del paréntesis que sigue a su declaración. Al final de la regla, después de hacer los cálculos, se hace guardar la información del empleado llamando un método del EJB soporteReglas:

```
rule "Empleado - Cálculo 1.1.a OrigenDeLaPension=1, ClaseDePension=1 - InvalidezVejezJubilación"
 when
    empleado : Empleado( origenDeLaPension == OrigenDeLaPension.INVALIDEZ1 && claseDePension == ClaseDePension.VITALICIAINMEDIATA1
    && estado == Estado.INVALIDO2 )
    soporteReglas : SoporteReglasBean( )
    //Se saca de las tablas de mortalidad la fila que necesitamos
    esperanzaDeVida : EsperanzaDeVida( y == empleado.edad && sexo == empleado.sexo && estado == empleado.estado )
 then
    double factorRenta = 0;
    if(empleado.getNumeroDeMesadas() == 14 ){
        factorRenta = ( 12*esperanzaDeVida.getA12y() ) + ( 2*esperanzaDeVida.getA2y() );
    }
    if(empleado.getNumeroDeMesadas() == 13 ){
        factorRenta = ( 12*esperanzaDeVida.getA12y() ) + ( 1*esperanzaDeVida.getA2y() );
    }
    BigDecimal invalidezVejezJubiliacion = BigDecimal.valueOf( factorRenta*empleado.getMesada() );
    empleado.setInvalidezVejezJubiliacion(invalidezVejezJubiliacion);
    //System.out.println( "Desde la Regla, Cálculo InvalidezVejezJubilación: " + empleado.getInvalidezVejezJubiliacion() );
    soporteReglas.guardarEnLaBaseDeDatos(empleado);
end
```

La anterior regla representa las fórmulas R_j = ([12P$(Va)^{12}_{x+t}$] + [2B $(Va)^{2}_{x+t}$]). tE_x y R_j= ([12P $(Va)^{12}_{x+t}$] + [$B(Va)^{2}_{x+t}$]). tE_x que están en el marco teórico de este documento, donde se explican estas fórmulas. Además esta regla, al igual que todas las demás deja todo el proceso de cálculo por adentro para que el cliente de la aplicación pueda entenderla e incluso modificarla, el único proceso que se hace en el Bean *soporteReglas*, y que se oculta al cliente, es el de guardar la información en la base de datos.

4.6 Módulo Reportes: mediante el reporteador Jasper

4.6.1 Descripción

Este módulo permite mostrar a los usuarios los resultados del cálculo de las reservas de pasivos pensionales para las diferentes empresas que se encuentran registradas en el sistema.

Para el desarrollo de este módulo fue utilizada una librería de código abierto que facilita la elaboración de reportes y que está escrita en Java, esta librería se llama JasperReports [40] Para mayor agilidad se utilizó la herramienta IReport [21] que ofrece un entorno visual para el diseño de reportes e integra la librería JasperReports, además le brinda al usuario diversas herramientas útiles para diseñar ejecutar y exportar los reportes.

La principal ventaja ofrecida por JasperReports es la capacidad de exportar reportes en diferentes formatos (*pdf*, *html* o *XML*) de acuerdo con las necesidades de los clientes. La integración con la aplicación web desarrollada se facilita debido que los reportes interactúan

únicamente con la base de datos y solamente requieren de parámetros definidos en la aplicación para generar los resultados deseados.

4.6.2 Proceso de elaboración de los Reportes

a) Establecer Fuente de Datos

La herramienta IReport [21] soporta conexiones JDBC a la base de datos, para esto solamente es necesario tener el servicio de la base de datos en funcionamiento, ya que a través de la interfaz grafica proporcionada por IReport se definen los parámetros de conexión, como se muestra a continuación.

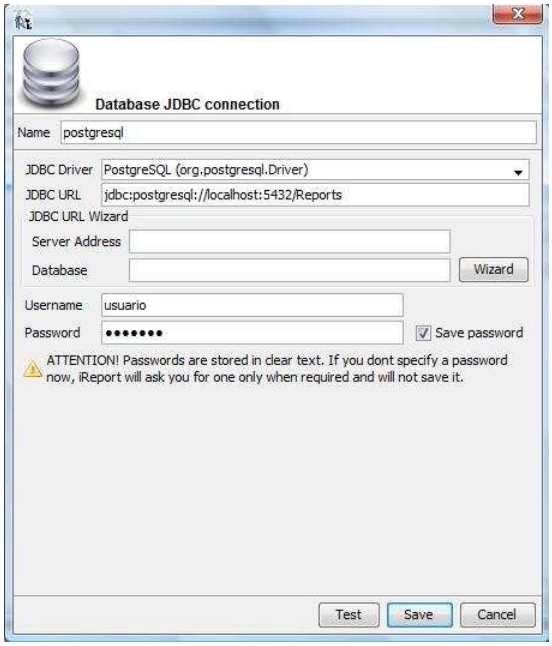

Ilustración 9 Definir Fuente de Datos

La herramienta contiene los controladores que permiten la conexión mediante JDBC con algunos de los motores de bases de datos más utilizados del mercado. Para el presente proyecto se utilizó el controlador que permite establecer la conexión con una base de datos PostgreSQL.

Luego de definir los datos de conexión es posible probarla para asegurarse de que funciona correctamente mediante el botón Test, como se muestra en la figura anterior.

b) Definir Consulta SQL

Se deben establecer los campos necesarios para el diseño del reporte y esto se logra definiendo una consulta SQL general que contiene todos los campos a usar en el reporte y que se va a utilizar como base en el proceso de elaboración del reporte. Mediante el asistente de creación de reportes se puede definir la consulta, o se puede crear una a través de una interfaz que ofrece la herramienta que permite construir la consulta agregando tablas, relacionándolas y seleccionando los campos necesarios para obtener los resultados esperados, como se muestra en la siguiente figura.

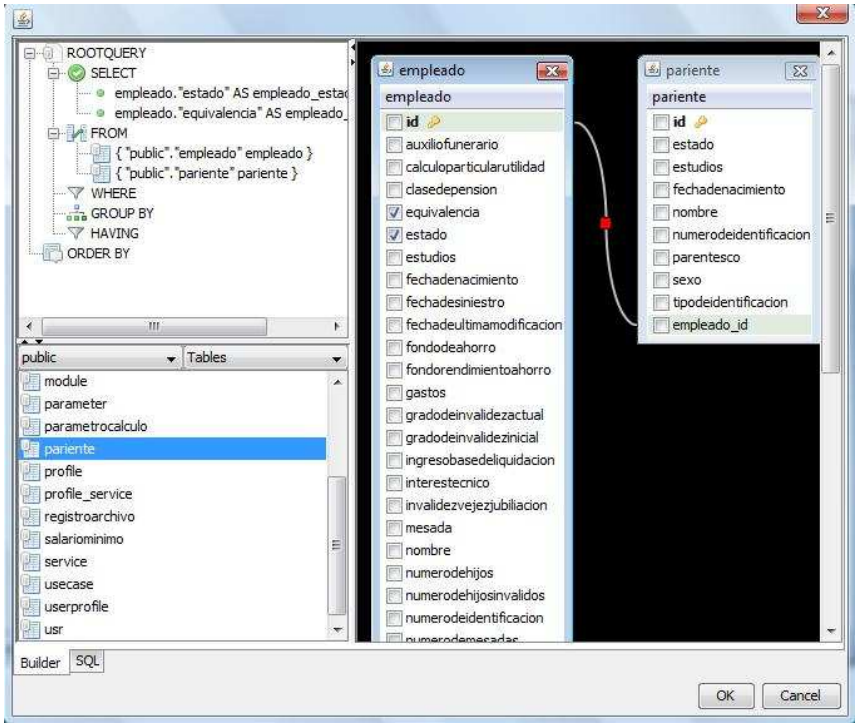

Ilustración 10 Definir Consulta SQL

c) Definir Campos de Agrupación

Al establecer la consulta es posible agruparla por cualquiera de los campos que fueron definidos (como se muestra en la siguiente figura), esto permite especificar secciones adicionales que pueden ser insertadas en el diseño del reporte, por ejemplo los totales y pueden ser o no utilizadas a la hora de elaborarlo.

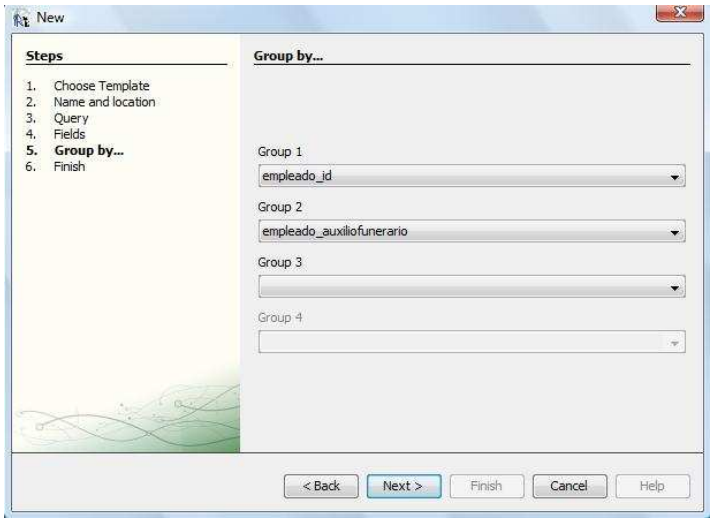

Ilustración 11 Definir Campos de Agrupación

La herramienta automáticamente genera un titulo para la sección agrupada que generalmente se identifica con el nombre del campo por el cual se quiere agrupar.

d) Definir secciones del reporte

Es necesario agregar todas las secciones que van a ser parte del reporte y contienen algún tipo de información. Cada una de estas secciones puede ser configurada de manera independiente a las demás y cada sección tiene un propósito diferente en la elaboración de los reportes. A continuación se describe la función de cada una de las secciones de un reporte.

Ilustración 12 Secciones de un reporte

- Titulo: Es la primera sección de un reporte y se imprime solo una vez. Puede ser mostrado en una sola página.
- Encabezado de Pagina: Esta sección se imprime una por cada página que forme parte del reporte y define el encabezado de cada una de ellas.
- Encabezado de Columna: permite definir el encabezado de cada columna que compone el reporte.
- Detalle: Se muestra repetidamente por cada una de las filas generadas a partir del resultado de la consulta definida.
- Pie de Página de Columna: se muestra al final de cada una de las páginas generadas si existen datos generados en la sección de detalles.
- Pie de Página: Se muestra en cada página después del pie de página de una columna con excepción de la última página en donde no se muestra.
- Resumen: solo se muestra una vez al final del reporte y generalmente es utilizado cuando los reportes requieren generar totales de alguno de los campos obtenidos a partir de la consulta.

e) Definir parámetros:

Los parámetros son objetos de un reporte los cuales son usados para pasar datos al reporte generado. Los datos pueden ser pasados desde otro reporte o desde una aplicación. Estos datos son usados para poder llenar el reporte generado de manera dinámica.

Ilustración 13 Definición de Parámetros

Es necesario definir el tipo de dato y el nombre de los parámetros que van a ser utilizados, adicionalmente la herramienta permite probar el parámetro definido que fue agregado mediante un cuadro de texto que solicita el valor del parámetro antes de realizar la consulta para generar los resultados.

f) Definir Variables:

Estas variables permiten definir valores de datos que pueden ser modificados durante la ejecución de la aplicación. En un reporte cualquier resultado de procesamiento de información puede ser almacenado en una variable que puede ser mostrada en el reporte o usada para procesar datos adicionales.

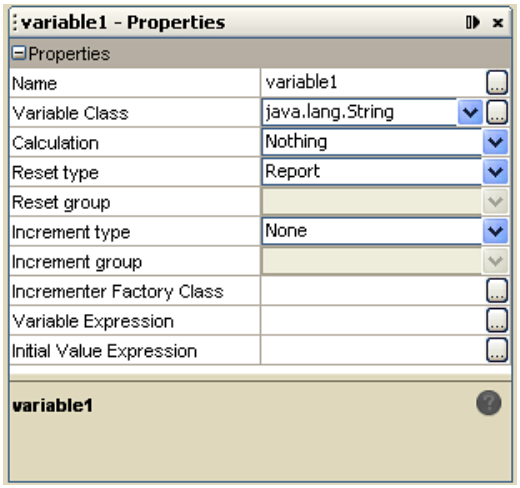

Ilustración 14 Definición de Variables

Se debe definir el nombre de la variable, el tipo de dato, entre otros atributos, y se puede escoger si la variable va a ser usada para hacer una operación de cálculo de las que son usadas en el lenguaje SQL. También se puede definir una expresión para la variable utilizando los métodos Java que están definidos para el tipo de dato de la variable. Se pueden realizar operaciones con cualquiera de los campos, variables y parámetros previamente definidos.

g) Integración de Reportes con la aplicación

Para integrar JasperReports [40] a la aplicación Seam simplemente hubo que agregar la librería *jasperreports-3.7.4.jar* al proyecto y a la lista de librerías que se van a desplegar con

la aplicación, para hacer esto se incluyó el nombre de la librería en el archivo "*deployed-jars-ear.list*" que está en la raíz de la aplicación. A medida que se exportan reportes a diferentes formatos de archivo hay que ir agregando librerías adicionales.

Para generar un reporte desde la aplicación Seam se requiere el archivo con extensión .jasper del reporte, un atributo tipo *Map* donde van los parámetros que se pasan al reporte y un atributo tipo *connection* donde van las características de la base de datos, adicionalmente se requiere un EJB desde el cual se puedan ejecutar los métodos proporcionados por JasperReports. A continuación se muestra un ejemplo de generación de un reporte desde un método de un EJB que es ejecutado cuando se presiona un botón de la aplicación.

```
String archivoJasper = rootDir + "\\Reportes\\ConsolidadoEmpleado1.jasper";
String archivoJrprint = rootDir + "\\Reportes\\ConsolidadoEmpleado1.jrprint";
String archivoPdf = rootDir + "\\Reportes\\ConsolidadoEmpleado1.pdf";

try {
    Connection connection;
    String url = "jdbc:postgresql://localhost:5432/reservas";
    connection = DriverManager.getConnection(url, "usuario", "vep794");

    Map parameters = new HashMap();
    parameters.put("EmpresaId",user.getEmpresa().getId().intValue());
    parameters.put("fechaCorte", fechaCorte);
    //log.info("FechaCorte: " + fechaCorte.toString());

    JasperFillManager
    .fillReportToFile(
            archivoJasper,
            archivoJrprint,
            parameters, connection);
    JasperExportManager
    .exportReportToPdfFile(
            archivoJrprint,
            archivoPdf);

    exitoGenerandoReporte = true;
} catch (SQLException sqlException) {
    sqlException.printStackTrace();
} catch (JRException jrexception){
    log.error(jrexception.getMessage());
    jrexception.printStackTrace();
}
```

Las rutas de los archivos leídos y generados en la imagen anterior están en el directorio web de la aplicación SisPen, pues al final de este proceso el cliente de la aplicación podrá descargar los reportes a su máquina.

5 RESULTADOS Y REFLEXIÓN SOBRE LOS MISMOS

Como resultado del presente proyecto se generó una aplicación web que facilita el cálculo de las reservas pensionales para entidades públicas, privadas, fondos de pensiones y compañías de seguros. Esta aplicación presenta las siguientes características.

Definiciones de perfiles: Esta característica se desarrolló con base al módulo de código abierto CincoSecurity el cual permite una gran flexibilidad para restringir el acceso de los usuarios del sistema a los diferentes servicios ofrecidos de acuerdo a los privilegios de cada uno de ellos. Esto se debe a la protección de acceso a los métodos de los EJB por medio de roles finos utilizando anotaciones en el *framework* Java EE 5.

Seguimiento de operaciones en Base de Datos (auditoría): mediante el módulo de auditoría se registran todas las modificaciones hechas sobre la base de datos, ya sean actualizaciones, inserciones o borrado de datos. Esto se hace posible mediante el uso de métodos *Callback* que permiten la ejecución de métodos antes y después de operaciones hechas sobre la base de datos.

Procesamiento de archivos planos: Esta aplicación permite procesar y almacenar en la base de datos archivos planos con información de empleados y parámetros necesarios para realizar los cálculos de reservas pensionales, además se hace un proceso previo de validación realizado por medio de descriptores XML y archivos de reglas para comprobar la validez de la información ingresada.

Uso de motor de reglas: por medio del uso de un motor de reglas fue posible definir los cálculos del sistema de una manera declarativa de modo que estos sean fáciles de entender para los expertos de negocio y además les sea posible modificarlos en caso de que alguno de ellos cambie con el tiempo. Esto se hace posible debido a que por medio del motor de reglas se separa la lógica de negocio de la aplicación.

Generación de reportes: los informes realizados utilizando herramientas especializadas permiten mostrar los resultados a los clientes y permiten que ellos descarguen los archivos generados a sus máquinas.

Validación: El proceso de validación de la aplicación a lo largo del desarrollo se realizó mediante información de prueba que nos facilitó el cliente. Para comprobar los resultados arrojados por la aplicación se cargaron los archivos de prueba que teníamos y se compararon los resultados obtenidos con los resultados de prueba con información correcta de las reservas que ya había sido comprobada previamente. Adicionalmente el cliente pudo ver el avance de la aplicación a lo largo de todo el proceso por medio de prototipos que se le presentaban luego de finalizar cada uno de los módulos definidos. Así obtuvimos la retroalimentación necesaria y nos aseguramos de que lo desarrollado era realmente lo que el cliente esperaba. Luego de dar por terminado el proceso de desarrollo se validaron los resultados del cálculo de las reservas utilizando información real la cual solicitamos a una compañía de seguros (Seguros La Equidad). Después de realizar las validaciones y de mostrar la aplicación al cliente se obtuvieron los resultados esperados, sin embargo esta última validación también nos sirvió para darnos cuenta que la aplicación necesita un cambio, pues cuando se ingresan grandes cantidades de información la aplicación falla por *time out* ya que tiene transacciones muy largas, estas transacciones deberán hacerse por lotes para que no se presenten fallas.

Esta aplicación proporciona ciertas ventajas sobre las aplicaciones de escritorio que en general son utilizadas para realizar este tipo de tareas debido a que funciona sobre un servidor de aplicaciones el cual presta servicios adicionales de seguridad, escalabilidad y capacidad de procesamiento distribuido. Además tiene la ventaja de estar en el entorno internet.

La mayoría de productos analizados y encontrados durante la parte inicial del proyecto están desarrolladas bajo el modelo cliente servidor por lo cual surgió la idea del desarrollo del presente proyecto, para proporcionar ventajas adicionales.

6 CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

Conclusiones

Por medio del presente proyecto los miembros del equipo de trabajo adquirieron experiencia en el desarrollo de aplicaciones de tipo empresarial utilizando *frameworks* de Java EE 5. Igualmente se adquirió destreza en el uso de distintas herramientas que pueden ser integradas con este tipo de aplicaciones, como las que permiten realizar reportes, validar archivos planos, y realizar definiciones de reglas de negocio declarativas y desacopladas del código de la aplicación que brindan facilidad de entendimiento por parte del cliente.

En cuanto a la metodología propuesta para el desarrollo de la aplicación se definió usar programación extrema. Esta metodología nos brindó muchas ventajas debido a que permite establecer un contacto constante con el cliente por lo tanto al finalizar el desarrollo de cada uno de los módulos el cliente tiene la posibilidad de ver el avance de la aplicación, dar su opinión al respecto y proporcionar retroalimentación al equipo de trabajo. Esto nos permitió hacer las correcciones respectivas de una manera ágil y tener claridad en los requerimientos solicitados por el cliente.

Mediante una consulta por parte de los integrantes del equipo de trabajo se adquirieron los conocimientos necesarios para comprender y realizar una implementación adecuada del proceso de cálculo de las reservas. Adicionalmente se aprendieron a utilizar las diferentes herramientas necesarias para el desarrollo del presente proyecto, para lo cual se destinó tiempo antes del inicio del desarrollo del proyecto debido a que algunos conceptos no fueron abordados a lo largo de la carrera.

Al hacer uso de un servidor de aplicaciones se comprendieron las ventajas que ofrecen en cuanto a escalabilidad, procesamiento distribuido, reducción de costos de mantenimiento entre otras características que permiten que los desarrolladores se vean beneficiados y prefieran elegir este tipo de aplicaciones sobre las que trabajan mediante un modelo cliente servidor.

Recomendaciones

Para este tipo de trabajos de aplicación práctica es necesario tener un cliente definido y tratar de mantener un contacto permanente con él o buscar reuniones que sean los más frecuentemente posibles con el objetivo de que evalué en todo momento el avance del proceso de desarrollo de la aplicación de acuerdo con pequeñas liberaciones generadas continuamente por parte de los desarrolladores. Esto permite evitar malentendidos o desarrollar módulos que no fueron solicitados y evita la pérdida de tiempo y aumento de costos durante el desarrollo, además de que se garantiza la satisfacción del cliente en la entrega del producto final.

Trabajos Futuros

La aplicación está desarrollada para la valoración de reservas de pensiones (reservas matemáticas) para las compañías de seguros que cubren los riesgos de vejez invalidez y muerte, de acuerdo a las especificaciones técnicas y formatos establecidos por la Superintendencia Financiera de Colombia para esas compañías , pero puede adaptarse o extenderse para la valoración de esas reservas de acuerdo a los requerimientos y condiciones pensionales establecidos por otras entidades (Fondos de pensiones , Ministerio de Hacienda, Superintendencia de Sociedades, Superintendencia de Puertos y Transportes, etc.) o gobiernos.

Se puede extender para la valoración de reservas derivadas de convenciones o pactos colectivos; puede adaptarse también, para realizar proyecciones y simulaciones de obligaciones pensionales para efectos de planeación presupuestal o para definir nuevas políticas en materia de pensiones (por ejemplo modificaciones en las edades para acceder a la pensión, o modificaciones en el monto de la pensión, o modificaciones en los períodos mínimos de cotización).

En cuanto a la implementación realizada debe pasarse a una nueva versión que trabaje las transacciones por lotes sobre la base de datos cuando va a realizar cálculos mediante reglas de negocio, como ya se mencionó en los resultados. Esa nueva versión es necesaria pues la actual tiene problemas de transacciones muy largas que fallan por *time out* cuando hay demasiados empleados por procesar.

7 POST-MORTEM

Metodología propuesta vs. Metodología realmente utilizada.

Para realizar el análisis, diseño y desarrollo de la aplicación web "**SisPen**" fue propuesto el uso de la metodología programación extrema. Se trató de cumplir con todos los principios de esta metodología aunque en ocasiones no fue así debido a que el tiempo nos lo impidió. Por esta razón algunas veces no se realizó la programación por pares debido a que era necesario dividirnos el trabajo para cumplir con el cronograma.

Se realizó la validación de la aplicación web utilizando la información suministrada por una entidad que tiene a su cargo el pago de obligaciones pensionales, los criterios de validación de un actuario y las pruebas funcionales de los resultados con información histórica ya procesada: gracias a la metodología escogida fue posible realizar validaciones continuas a la aplicación. Además ayudó bastante a cumplir esta metodología el constante contacto que se mantuvo con el cliente.

Efectividad en la estimación de tiempos del proyecto

Se presentó una variación en los tiempos estimados y los tiempos realmente empleados en cada uno de los módulos desarrollados debido a que en muchas de las actividades nos tomó más tiempo del presupuestado aprender a utilizar las herramientas que iban a ser usadas. Por otra parte, en módulos como el de reportes se estimó más tiempo del que fue empleado debido a que se adquirió destreza en el manejo de la herramienta *IReport* rápidamente. En general los tiempos se fueron ajustando a lo largo del proyecto y se pudo cumplir y completar todas las actividades que fueron propuestas al inicio del proyecto.

8 - REFERENCIAS Y BIBLIOGRAFÍA

1. Referencias

[1] Miguel Ángel Monroy Saavedra. (1996) *Sistema De Seguridad Social Integral Colombiano*, 1ª Edición, Bogotá.

[2]Reza Rahman, Derek Lane (2007) *EJB 3 in Action*, Editorial Manning

[3] Loredana Helmsdorff. (2007) *Ampliación De Cobertura Del Sistema Pensional Colombiano y Atención Al Adulto Mayor*, Departamento de Planeación Nacional.

[4] Paul Graham. (2001) *The other road ahead*.

http://www.paulgraham.com/road.html

[5] Ávila Baray, H.L. (2006) *Introducción a la metodología de la investigación*, Edición electrónica.
www.eumed.net/libros/2006c/203/

[6] M. Castillo. (2008, Octubre 21 2005). *Método de estudio de caso*.

http://www.usn.edu.mx/artman/publish/article_16.shtml

[7] E. Gómez Sánchez, Y.A. Dimitriadis, J.I. Asensio Pérez, M. Rodríguez Cayetano, M.L. Bote Lorenzo y G. Vega Gorgojo. (2003), *Aplicación y evaluación del estudio de casos como técnica docente en el área de ingeniería telemática*. Universidad de Valladolid

www.gsic.uva.es/uploaded_files/6B_2.pdf

[8] Ruiz Limón, R. (2007) *Historia y evolución del pensamiento científico*, Edición electrónica gratuita.

www.eumed.net/libros/2007a/257/

[28] Consuelo Uribe Mallarino. (1999) *Alcances de la seguridad social en Colombia*, Pontificia Universidad Javeriana, Papel político 9-10.

[29] Julián Andrés Duque Barreto. (2005) *Aproximación metodológica del cálculo de las reservas pensionales para los afiliados activos de la caja nacional de previsión*, Universidad Nacional.

http://matematicas.unal.edu.co/academia/programas/documentos_tesis/8.pdf

[36] Fasecolda Federación de Aseguradores Colombianos, (1999) *Seguridad Social En Colombia*.

[44] Jboss Drools Team (2010) Drools Expert User Guide, v5.1.1,

http://downloads.jboss.com/drools/docs/5.1.1.34858.FINAL/drools-expert/html_single/index.html

[52] Ian Graham, (2007) *Business rules management and Service oriented Architecture, A pattern language*, Editorial Wiley

[74] Rick Leander. (2000) *Building Application Servers*, Editorial Cambridge University Press

2. Bibliografía

[9] V. Toro (2009), Extreme programming *El enfoque Extreme Programming*, 23 Salón de Informática de la ACIS, Septiembre de 2003.

[10] D. Wells. (2006, February 17, 2006). *Extreme programming: A gentle introduction*: http://www.extremeprogramming.org/

[11] R. Jeffries. (Noviembre 8 de 2001). *What is extreme programming?*

http://www.xprogramming.com/xpmag/whatisxp.htm

[12] SourceForge.NET (2008) *Taylor, model driven architecture on rails.*

http://taylor.sourceforge.net/index.php/Overview

[13] C. Richardson (February 27, 2006) *What POJO is programming?* SYS-CON Media, Inc.

http://java.sys-con.com/node/180374

[14] Superintendencia Financiera (2010), *Resolución número 1555*

www.superfinanciera.gov.co/NormativaFinanciera/Archivos/r1555_10.doc

[15] Ministerio de hacienda y crédito público (1994), *Decreto 2852.*

www.presidencia.gov.co/prensa_new/decretoslinea/1994/diciembre/26/dec2852261994.pdf

[16] Ministerio de hacienda y crédito público (2001), *Decreto 2783.*

www.presidencia.gov.co/prensa_new/decretoslinea/2001/diciembre/20/dec2783202001.doc

[17] Thuan L. Thai, Hoang Lam (2002) *.NET Framework Essentials*, Editorial O´Reilly

media.wiley.com/product_data/excerpt/98/07645482/0764548298.pdf

[18] Juan Carlos Rubio (2008) *Enterprise Java Beans*, Centro Informático Científico de Andalucía

http://www.slideshare.net/jcrubio/curso-ejb3

[20] Ricardo Alanís Barrera. (2005) *Sistemas Expertos e Inteligencia Artificial*, Universidad de Burgos.

pisuerga.inf.ubu.es/cgosorio/SExInArt/trabajos/CLIPS.pdf

[21] JasperForge (2010) *the report designer for JasperReports and JasperServer*

http://jasperforge.org/projects/ireport

[22] María Consuelo Franky (2009), *Guía de los generadores del framework Seam*, Pontificia Universidad Javeriana.

[25] Superintendencia Financiera. (2006), *Pensión de vejez – Pensión de invalidez – Requisitos.*

[26] Jorge Rojas. (1999), *La Rentabilidad del sistema privado de pensiones en el Perú.*

http://www.pucp.edu.pe/departamento/economia/images/documentos/DDD160.pdf

[27] Superintendencia financiera. (2009), *Circular Externa 011*

www.superfinanciera.gov.co/NormativaFinanciera/Archivos/ce011_10.doc.

[32] José Luis Álvarez (2007), *Introducción al protocolo HTTP*, Universidad de Huelva.

www.uhu.es/josel_alvarez/NvasTecnProg/recursos/ProtocoloHTTP.pdf

[33] Kris Jamsa, Suleiman Lalani, Steve Weakly (1998), *Web Programming*, Editorial McGraw-Hill.

[34] The php group. (2010), *Página oficial de PHP*. 2010

http://www.php.net/

[35] Davis Flanagan, Jim Farley, William Crawford y Kris Magnuson. (1999) *Java Enterprise in a Nutshell*, Editorial O´Reilly)

[38] Secretaria del Senado Colombiano. (1994) *Decreto 1284*

http://www.secretariasenado.gov.co/senado/basedoc/decreto/1994/decreto_1284_1994.html.

[40] JasperSoft Corporation. (2009) *JasperReports open source java reporting library*. http://jasperforge.org/plugins/project/project_home.php?projectname=jasperreports

[41] Eric Jendrock, Jennifer Ball, Debbie Carson, Ian Evan, Scott Fordin, Kim Haase, (2010) *The Java EE5 Tutorial, Introduction to the Java Persistence API*, Oracle.

http://download.oracle.com/javaee/5/tutorial/doc/bnbpz.html

[42] Chris Moran. (2004), *Does your project need a rule engine*, Utilizer, Inc.

http://java.sys-con.com/node/45082

[45] FICO, Decision Management Blog (2007) *Submission to Production Rule Representation*.

http://www.edmblog.com/weblog/PRRCore.pdf

[46] Alejandro Aguilar Sierra. (2002) *Introducción a la programación extrema*.

http://www.willydev.net/descargas/articulos/general/IntroXP.PDF

[48] Perl.org. (2010) *The Perl programming language*.

http://www.perl.org/

[49] Python software foundation (2010), *Python Programming Language*.

http://www.python.org/

[50]Microsoft Corporation (2010), *Información general y conceptual sobre .NET Framework*

http://msdn.microsoft.com/es-es/library/zw4w595w.aspx

[51] Microsoft Corporation (2009) *.NET Framework overview*.

http://www.microsoft.com/net/overview.aspx

[53] Fernando Tobón. (2003) *La encrucijada del sistema pensional colombiano: Causas de la crisis y perspectivas de la solución*, Perfil de Coyuntura Económica.

http://aprendeenlinea.udea.edu.co/revistas/index.php/coyuntura/article/viewFile/2283/1841

[54] Ciro Leonardo Martínez Gómez. (1997) Colombia. *Tablas Abreviadas De Mortalidad Por Sexo Para Fechas Censales y Estimaciones Quinquenales*, DANE.

ftp://190.25.231.247/books/LB_10354_1995_2025.pdf

[56] Ger Mulcachy. (2009), *J2EE vs .NET*, CGISecurity

www.cgisecurity.com/.../J2EEandDotNetsecurityByGerMulcahy.pdf

[57] Miguel Azocar, (2004) *Elección de la mejor herramienta de aplicaciones empresariales, J2EE vs .NET*, Universidad Técnica Federico Santa María.

Aluos.elo.utfsm.cl/~azcar/ramos/2_semestre_2004/Seminario_redes_Computadores/Informe1.doc

[58] Congreso de la república de Colombia (1993) *Ley 100 de 1993, sistema de seguridad social integral.*

http://www.laseguridad.ws/consejo/consejo/html/biblioteca-legis/ley_100.pdf

[59] Chad Vawter and Ed Roman. (2001) *J2EE vs. microsoft.NET A comparison of building XML-based web services*, The middleware Company.

http://www.mimerhellas.com/html/library/J2EE-vs-DotNET.pdf

[60] The eclipse Foundation. (2010), *Java EE 5 support for annotations*

http://help.eclipse.org/galileo/index.jsp?topic=/org.eclipse.jst.j2ee.doc.user/topics/cannotations.html

[61]Oracle Corporation. (2010), *The J2EE tutorial*, Oracle Corporation

http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/EJBConcepts4.html

[62] Technische Informatik Letzte Änderung. (2006) *EJB services.*

http://www-ti.informatik.uni-tuebingen.de/os390/java/ejbcb02.pdf

[63] IBM. (2003), *J2EE vs. .NET*, IBM Corporation.

http://www-01.ibm.com/software/smb/na/J2EE_vs_NET_History_and_Comparison.pdf

[64] Matt Raible. (2006), *EJB3 conceptos básicos.*

http://www.jroller.com/jl_monteagudo/entry/ejb3_conceptos_b%C3%A1sicos

[65] Microsoft Corporation. (2010) *Objetivos de diseño visual studio.net escalabilidad.*

http://msdn.microsoft.com/es-es/library/aa292172%28VS.71%29.aspx

[66] David Sawyer. (2010) *Bussines rules Engines.*

http://it.toolbox.com/blogs/enterprise-apps/it-architecture-101-business-rule-engines-36865

[67]Jens Dietrich, (2005) *On the Test-Driven Development and Validation of Business Rules*, Massey University.

http://ibis.in.tum.de/docs/docs_staff/docs_paschke/docs/ista2005-final.pdf

[68] T. G. TG consultores. (2010, Calculo actuarial.)

http://tgconsultores.net/htm/calculo.html

[69] Oracle Corporation. (2003) *Oracle application server 10g migrating from WebLogic 10g.*

http://download.oracle.com/docs/cd/B12428_15/migrate.904/b10425/asmwl01.htm#1009807

[71] Drools Expert User Guide:
http://downloads.jboss.com/drools/docs/5.1.1.34858.FINAL/drools-expert/html/index.html

[75] JBoss Drools Team. (2009) *The business logic integration platform.*

[72] María Consuelo Franky, Víctor Toro, Rodrigo López,(2010) "*CincoModule: Módulo de seguridad basado en roles finos"*, 5CCC: Quinto Congreso Colombiano de Computación, Cartagena (Colombia), ISBN: 978-958-8387-40-6.

[73] María Consuelo Franky, Víctor Manuel Toro, Rodrigo López, (2009), *CincoSecurity Module based on fine roles*

http://sourceforge.net/projects/cincosecurity/

9 - ANEXOS

Los siguientes documentos forman parte de los anexos y se encuentran disponibles en la página del presente trabajo de grado: http://pegasus.javeriana.edu.co/~CIS0910IS01/

- Inventario de módulos y casos de uso.
- Documento Hacer Usos (Requerimientos).
- Documento de Especificación de Requerimientos (SRS).
- Encuestas hechas al cliente.