



ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE CIENCIAS
DE LA COMPUTACIÓN**

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

**PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL
TÍTULO DE INGENIERO EN SISTEMAS E INFORMÁTICA**

**TEMA: Control del proceso de retorno de dinero de planillas de
seguros utilizando una herramienta BPM libre integrada al
sistema hospitalario Innovativa Salud.**

AUTOR: SHWEIZER BONILLA ARIE

DIRECTOR: ING. VICTOR PALIZ

SANGOLQUÍ

OCTUBRE 2015

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

CERTIFICADO

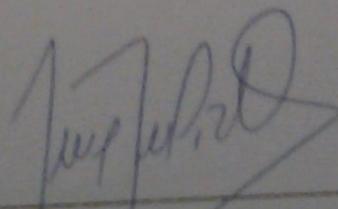
Ing. Victor Paliz

CERTIFICA

Que el trabajo titulado "Control del proceso de retorno de dinero de planillas de seguros utilizando una herramienta BPM libre integrada al sistema hospitalario innovativa salud." realizado por el señor Schweizer Bonilla Arie, ha sido guiado y revisado periódicamente y cumple normas estatutarias establecidas por la Universidad de las Fuerzas Armadas - ESPE.

El mencionado trabajo consta de un documento empastado y un disco compacto el cual contiene los archivos en formato portátil de Acrobat (PDF). Se autoriza al señor Schweizer Bonilla Arie, que el material se entregue al Ing. Mauricio Campaña, en su calidad de Director de la Carrera.

Sangolquí, Octubre de 2015



ING. VICTOR PALIZ
DIRECTOR DE TESIS

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

DECLARACIÓN DE RESPONSABILIDAD
SHWEIZER BONILLA ARIE

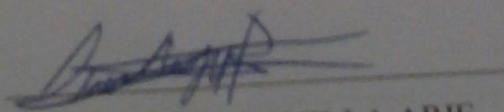
DECLARA QUE:

El proyecto de grado denominado "Control del proceso de retorno de dinero de planillas de seguros utilizando una herramienta BPM libre integrada al sistema hospitalario innovativa salud." ha sido en base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan en la bibliografía.

Consecuentemente este trabajo es de mi autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, Octubre de 2015



SHWEIZER BONILLA ARIE

C.I: 1716041031

UNIVERSIDAD DE LAS FUERZAS ARMADAS – ESPE
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

AUTORIZACIÓN DE PUBLICACIÓN
SHWEIZER BONILLA ARIE

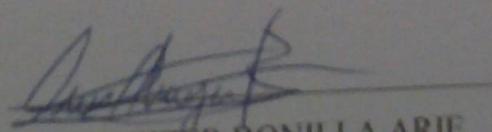
AUTORIZAN:

A la Universidad de las Fuerzas Armadas – ESPE la publicación en la biblioteca virtual de la institución del trabajo "Control del proceso de retorno de dinero de planillas de seguros utilizando una herramienta BPM libre integrada al sistema hospitalario innovativa salud.", cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y autoría.

Consecuentemente este trabajo es de mi autoría.

En virtud de esta declaración, nos responsabilizamos del contenido, veracidad y alcance científico del proyecto de grado en mención.

Sangolquí, Octubre de 2015



SHWEIZER BONILLA ARIE

C.I: 1716041031

DEDICATORIA

A mis padres por su amor, consejos, apoyo y esfuerzo.

por brindarme todas las herramientas que me

permitieron llegar hasta aquí

y por forjarme como la

persona que soy en

la actualidad.

AGRADECIMIENTO

Quisiera agradecer y reconocer la colaboración de todas las personas que hicieron posible que esta tesis se realizara, es por ello que agradezco a los ingenieros que durante años en la institución han impartido sus conocimientos. Sus enseñanzas se ven plasmadas en el contenido teórico y práctico en esta tesis y en mí en lo profesional y personal, en especial a mi Tutor el Ingeniero Victor Paliz, por todo su tiempo, consejos y apoyo brindado al desarrollo de este proyecto, que ayudó a que esta idea se convierta en realidad. Quisiera agradecer al equipo de trabajo de Innovativa Salud por permitirme realizar mi tesis junto a ellos, por su asesoramiento, consejos y recursos.

Tabla de contenido

CERTIFICADO	¡Error! Marcador no definido.
DECLARACIÓN DE RESPONSABILIDAD	¡Error! Marcador no definido.
AUTORIZACIÓN DE PUBLICACIÓN	¡Error! Marcador no definido.
DEDICATORIA	iv
AGRADECIMIENTO	v
CAPÍTULO 1 INTRODUCCIÓN	12
1.1 PROBLEMÁTICA	12
1.2 OBJETIVOS	13
a) Objetivo General	13
b) Objetivos Específicos.....	13
1.3 ALCANCE.....	13
CAPÍTULO 2 MARCO TEÓRICO	15
2.1 Las tecnologías Web	15
2.1.1 Tecnologías en el cliente	15
2.1.2 Tecnologías en el servidor.....	15
2.2 Procesos	16
2.3 Comparación entre herramientas BPMS.....	17
2.3.1 BonitaSoft.....	18
2.3.2 jBPM.....	18
2.3.3 Bizagi.....	19
2.3.4 ProcessMaker.....	19
2.3.5 Activiti BPM.....	20
2.4 Servidores de Aplicaciones Web	23
2.4.1 WildFly.....	23

2.4.2	GlassFish	23
2.5	Entornos de Desarrollo IDE.....	24
2.5.1	Eclipse	24
2.5.2	Netbeans	25
2.6	Innovativa Salud	26
2.7	Tecnologías de Integración.....	27
2.7.1	SOAP.....	27
2.7.2	REST	28
CAPÍTULO 3.-	ANÁLISIS Y DISEÑO.....	30
3.1	Análisis de requerimientos.....	30
3.1.1	Introducción.....	30
3.1.2	Ámbito del sistema	30
3.2	Descripción General.....	32
3.2.1	Funciones del producto.....	32
3.2.2	Características de los usuarios	33
3.2.3	Suposiciones y dependencias.....	33
3.2.4	Requisitos futuros	33
3.3	Requisitos Específicos	34
3.3.1	Requisitos Funcionales	34
3.3.2	Interfaces externas	35
3.3.3	Atributos del sistema	35
3.4	Análisis del sistema.....	36
3.4.1	Descripción del funcionamiento	36
3.4.2	Descripción de la información.....	37
3.5	Diseño del sistema	44
3.5.1	Diagrama de implementación:.....	44

3.5.2	Diseño del proceso.....	45
3.6	Proceso en BPMN2.....	48
3.7	Diseño de reportes gerenciales.....	50
CAPÍTULO 4 INTEGRACIÓN Y PRUEBAS		57
4.1	Proceso de integración del proceso con Innovativa Salud.....	57
4.2	Problemas presentados en la integración.	65
4.3	Integración de reportes gerenciales.....	67
4.4	Pruebas realizada	69
4.4.1	Requisito RF1:.....	69
4.4.2	Requisito RF2:.....	75
4.4.3	Requisito RF3:.....	81
4.4.4	Requisito RF5:.....	85
4.4.5	Requisito RF7:.....	87
4.4.6	Requisito RF6:.....	91
4.4.7	Requisito RF8:.....	93
4.4.8	Requisito RF10:.....	95
CAPÍTULO 5.- Conclusiones y recomendaciones		98
5.1	Conclusiones	98
5.2	Recomendaciones	99
CAPÍTULO 6.- ANEXOS		100
6.1	Código Fuente.....	100
6.1.1	Anexo 1: Actividad Actual.....	100
6.1.2	Anexo 2: Aprobacion:	102
6.1.3	Anexo 3: Generar:	104
6.1.4	Anexo 4: Iniciar Proceso:	106
6.1.5	Anexo 5: Ping:.....	107

6.1.6	Anexo 6: Realizar Tarea:	109
6.1.7	Anexo 7: Revision:	111
6.1.8	Anexo 8: Proceso Bean:	113
6.1.9	Anexo 9: Pom.xml de InnovativaRest:	116
6.2	Glosario de Términos.....	118
BIBLIOGRAFÍA		120
Tabla de contenido.....		vii

Índice de Diagramas:

Diagrama 1 Implementación	44
Diagrama 2 Caracterización del Proceso	45
Diagrama 3 Proceso Original de Aprobación de Planilla.....	46
Diagrama 4 Proceso Recomendado	47
Diagrama 5 Proceso BPMN2	48

Índice de Tablas:

Tabla 1 Cuadro Compartivo	21
---------------------------------	----

Índice de Figuras:

Figura 1 Distribución de gráficos	56
Figura 2 Pantalla de Facturación	60
Figura 3 Inicio de Proceso	61
Figura 4 Generar Planilla.....	61
Figura 5 Atención Médica	62

Figura 6 Enviar Planilla.....	63
Figura 7 Planilla Requiere Cambios.....	63
Figura 8 Facturación de planilla.....	64
Figura 9 Pago de Factura.....	65
Figura 10 standalone-full.xml.....	66
Figura 11 Security Domain.....	67
Figura 12 Resultado CP001.....	70
Figura 13 Resultado CP002.....	71
Figura 14 Resultado CP003.....	73
Figura 15 Resultado CP004.....	74
Figura 16 Resultado CP005.....	76
Figura 17 Resultado CP006.....	77
Figura 18 Resultado CP007.....	79
Figura 19 Resultado CP008.....	80
Figura 20 Resultado CP009.....	81
Figura 21 Resultado Alternativo CP009.....	82
Figura 22 Resultado CP010.....	83
Figura 23 Resultado Alternativo CP010.....	84
Figura 24 Resultado CP011.....	85
Figura 25 Resultado CP012.....	86
Figura 26 Resultado 1 CP014.....	88
Figura 27 Resultado 2 CP014.....	88
Figura 28 Resultado 3 CP014.....	88
Figura 29 Resultado 4 CP014.....	89
Figura 30 Resultado 1 CP015.....	90
Figura 31 Resultado 2 CP015.....	90
Figura 32 Resultado 1 CP016.....	91
Figura 33 Resultado 2 CP016.....	92
Figura 34 Resultado CP017.....	94
Figura 35 Resultado CP018.....	96
Figura 36 Resultado CP019.....	97

CAPÍTULO 1 INTRODUCCIÓN

1.1 PROBLEMÁTICA

El presente proyecto tiene como finalidad implementar un Business Process Management (BPM), el cual servirá para controlar el proceso de aprobación de la planilla de seguros, esto debido a que existe falta de control en el actual manejo de pagos de seguros. Los administradores de los centros médicos necesitan ser capaces de verificar qué instancia retrasa o detiene el proceso para cada una de las planillas generadas. La definición de un proceso y la administración del mismo con una herramienta BPM da a los administradores la capacidad de identificar y corregir los problemas existentes que generan la importante suma en cuentas por cobrar, esta herramienta permitirá a los administradores verificar si la falla es interna (dentro del centro médico) o externa (compañías de seguros).

En caso de que la falla sea interna tendrá la capacidad de tomar decisiones informadas. En caso de que la falla es externa se tendrá que informar a la compañía de seguro.

El proceso se integrará al sistema de gestión hospitalaria Innovativa Salud, desarrollado por el centro de transferencia tecnológica de la Universidad de las Fuerzas Armadas. El sistema posee un módulo de facturación el cual administra la generación de planillas. En este punto es necesario iniciar el proceso de aprobación; la solución planteada con BPM permitirá a la gerencia controlar el proceso de aprobación de las planillas desde este punto hasta el cobro final de la factura, al estar integrado el BPMS al sistema Innovativa Salud el control se realizará internamente por lo que los usuarios no tendrán la necesidad de cambiar la manera en que actualmente realizan sus actividades, el proceso en la actualidad se realiza de forma manual lo que complica el seguimiento de un trámite impidiendo conocer el estado actual de cualquier planilla.

Este proceso es necesario en la mayoría de centros médicos del país por lo que integrar un BPM al sistema médico significará para los administradores una solución integral a varios de sus problemas con un solo gasto, dando así a Innovativa Salud una

ventaja competitiva que se suma a las que actualmente posee. Esta integración facilitará además el manejo del área contable y permitirá a los centros de salud brindar un mejor servicio a sus clientes.

El proyecto además fomenta el trabajo conjunto del Centro de Transferencia y Desarrollo Tecnológico de la Universidad de las Fuerzas Armadas ESPE (CTT-ESPE-CECAI) con el Departamento de Ciencias de la Computación (DECC) y la Carrera de Ingeniería en Sistemas e Informática.

1.2 OBJETIVOS

a) Objetivo General

Controlar el proceso de retorno de dinero de planillas de seguros utilizando una herramienta BPM libre integrada al sistema hospitalario Innovativa Salud.

b) Objetivos Específicos

- Determinar herramienta BPMS a ser utilizada.
- Obtener el proceso de aprobación de planilla de seguros utilizando un manejador de procesos de negocio.
- Integrar el proceso de aprobación de planillas al Sistema Innovativa Salud .
- Presentar reportes gerenciales que permitan monitorear el proceso de aprobación de planillas y brinden información para la toma de decisiones.

1.3 ALCANCE

Desarrollar un proyecto BPM con el cual se pueda realizar el control y seguimiento del proceso de aprobación de planillas de seguros, este proceso se definirá con BPMN2¹ y se implementará en el manejador de procesos de negocios definido a través de un estudio comparativo entre las principales herramientas libres. El BPM

¹ Segunda versión de la notación para BPM

(Business Process Manager) seleccionado deberá permitir integrar el proceso a un entorno Enterprise de Java.

Detalladamente el proceso tendrá las siguientes etapas:

Generación de planilla

- Generar planilla inicial.
- Ingreso de información hospitalaria
- Verificación de datos ingresados.
- Envío de planilla.

Aprobación

- Evaluación de aseguradora.
- Corrección de planilla.
- Reenvío de planilla.

Envío de Factura

- Selección de planillas aprobadas.
- Envío de factura final.
- Ingreso del valor a cuentas por cobrar del centro médico.
- Confirmación de pago de factura de la aseguradora.

La integración con el Sistema Innovativa Salud se realizará en el módulo de facturación, se presentarán datos de procesos y la situación actual mediante informes gerenciales permitiendo a los administradores tomar decisiones informadas.

CAPÍTULO 2 MARCO TEÓRICO

2.1 Las tecnologías Web

2.1.1 Tecnologías en el cliente

Las tecnologías del lado del cliente son aquellas que se realizan en el equipo del usuario final, para este caso se realizó una aplicación web por lo que la tecnología utilizada se ejecuta en el navegador.

El lenguaje de marcas de hipertexto, es como su nombre lo sugiere un lenguaje que permite agregar marcas (etiquetas) al texto original, estas marcas nos permitirán codificar la estructura o la presentación del texto. El hipertexto definido como “Texto que contiene elementos a partir de los cuales se puede acceder a otra información” (Real Academia Española, 2015) nos brinda la funcionalidad que define al lenguaje. El hipertexto como tal se desarrolló años antes de que Tim Berners-Lee junto a Robert Cailliau presentaran Word Wide Web.

El primer documento en el que se describe al lenguaje fue HTML Tags, publicado 1991 en el que lista las etiquetas a ser usadas en HTML, la manera en que estas deben ser abiertas y cerradas y la posibilidad de incluir atributos dentro de las mismas. (w3, 2015)

Desde 1990 HTML ha tenido varios estándares iniciando con HTML 2.0 definido en RFC 1866, en la actualidad se encuentra definida la versión 5 de HTML desarrollada gracias al impulso de WHATWG y fue publicada en Octubre de 2014. En esta última versión se destacan las etiquetas canvas 2D, 3D, audio y video.

2.1.2 Tecnologías en el servidor

En el lado del servidor de aplicaciones también existen varias tecnologías que permiten desarrollar funciones complejas que facilitaran la interacción con el usuario, en esta sección se analizan las más importantes en función de la tecnología utilizada para el desarrollo de este trabajo, la primera tecnología es JSF diseñado para ser flexible, Java Server Faces incluye un grupo de APIs que ayudan a manejar estado, eventos, validaciones de los componentes dentro de la interfaz de usuario y una librería

JSP que facilita su uso dentro de las páginas que usen esta tecnología. Su objetivo es definir la separación entre la lógica de la aplicación y la presentación de la misma. (Oracle, 2015).

JSP acrónimo de Java Server Pages, es una tecnología para crear páginas web que sean rápidas de realizar y fáciles de mantener, JSP es considerada como una extensión de Java Servlet technology juntas estas tecnologías entregan al programador un entorno atractivo para el desarrollo de páginas dinámicas siendo más eficiente que otras tecnologías web, otra de las ventajas que posee JSP es que gracias a su base en java es posible ejecutar las aplicaciones en varias plataformas sin necesidad de realizar cambios debido principalmente al uso de máquinas virtuales java. (Oracle, 2015)

Java Enterprise Edition es una plataforma de desarrollo y ejecución de software que puede ser desarrollada en varias plataformas muchas de código abierto como Netbeans o Eclipse lo que reduce el costo inicial del desarrollo.

A partir de la plataforma JEE 5 se incluye un API de persistencia que ofrece un mapeo objeto/relacional que facilita la gestión de datos dentro de los beans de java si como le los componentes web y los clientes en otras aplicaciones. (Java, 2015)

2.2 Procesos

La gestión de procesos se ha revelado como una de las herramientas que permiten mejores resultados para el cumplimiento de la misión y visión dentro de cualquier institución, además de facilitar conceptos de calidad como el compromiso del personal, enfoque al cliente, toma de decisiones basadas en hechos y la mejora continua, esta mejora se logra gracias al ciclo de vida de los procesos el cual posee las siguientes etapas: (AuraPartal, 2015)

1. **Modelización:** Utilizando la notación BPMN 2.0 se diseñan y construyen los procesos mediante el uso de tareas, eventos, compuertas de decisión, etc. Una vez definido el proceso es necesario definir actores, roles, formularios y bases de conocimiento.
2. **Simulación:** Es posible dos tipos de simulación, simulación estática y real esto quiere decir es que es posible simular únicamente con el modelo del proceso o ejecutarlo y probar el proceso tal como será puesto en ejecución.

3. **Ejecución:** Semejante a la fase de producción en desarrollo de software, el proceso que se ha generado en la etapa de modelización es utilizado diariamente por los usuarios que trabajan dentro del proceso.
4. **Monitorización:** En esta etapa se analizan los resultados obtenidos en la etapa anterior, se verifica si se están obteniendo los resultados deseados y en los tiempos requeridos.
5. **Optimización:** Tras el análisis de los resultados de la puesta en producción del proceso es necesario pulir el proceso para obtener los resultados deseados o para mejorar el proceso en general.

La gestión por procesos genera trabajo estandarizado, cumplimiento de las tareas en tiempos adecuados además de facilitar la coordinación y el control

2.3 Comparación entre herramientas BPMS

“Business Process Management (BPM) es un conjunto de métodos, herramientas y tecnologías utilizados para diseñar, representar, analizar y controlar procesos de negocio operacionales. BPM es un enfoque centrado en los procesos para mejorar el rendimiento que combina las tecnologías de la información con metodologías de proceso y gobierno. BPM es una colaboración entre personas de negocio y tecnólogos para fomentar procesos de negocio efectivos, ágiles y transparentes.” (Kian Garimella, 2015)

Un BPMS es el sistema o software necesario para automatizar los procesos que han sido previamente modelados facilitando su mantenimiento y control, para el modelado existen varios estándares, pero el más difundido dentro de los BPMS es BPMN (Business Process Modeling Notation) en la actualidad la mayoría de BPMS utilizan la notación BPMN además de notaciones propias.

En cuanto a las metodologías aún se pueden utilizar las metodologías de Ingeniería de Software pero únicamente para el desarrollo de componentes debido principalmente a que dichas metodologías posee modelos que son comprendidos principalmente por personal técnico y debido a la filosofía implícita dentro de BPM es

necesario que todo el personal pueda entender los diagramas para así poder comprender su función dentro de los procesos y dentro de la empresa. (Pérez, 2015)

2.3.1 BonitaSoft

BonitaSoft posee 3 módulos principales, Bonita Studio que posee los módulos de process modeler, que es un editor gráfico de proceso con base en BPMN 2.0 el cual además posee una versión colaborativa, 100 conexiones, templates, simulación de procesos y desarrollo de aplicaciones con drag and drop. Bonita Engine que posee aplicación java y Rest, administración de tareas humanas, motor transaccional y Multi-tenancy. User XP email, etiquetas y categorías, soporte de múltiples lenguajes, tareas de delegación y administración avanzada.

BonitaSoft posee varias ediciones, este trabajo se centrará en la versión community, en esta versión no es posible administrar perfiles, generar templates de procesos, reusar formularios, sub tareas y no posee reportes. (BonitaSoft S.A, 2012)

2.3.2 jBPM

jBPM es un motor de flujo de trabajo desarrollado completamente en JEE actualmente se encuentra en su versión 6.1 es desarrollado por Red Hat y posee licencia Apache 2.0. Al igual que BonitaSoft soporta procesos en BPMN 2.0 pero posee además su propio lenguaje de definición de procesos conocido como jPDL.

Como parte de la comunidad JBoss jBPM se basa en máquinas virtuales de procesos, para el desarrollo el proyecto posee varias opciones, la principal herramienta para desarrolladores es un complemento de Eclipse, pero existen en la actualidad plugins para Netbeans, existe la posibilidad de definir proyectos y procesos desde una interfaz web pero esta interfaz en las versiones 5.0 tiene un enfoque hacia el usuario final.

jBPM no posee reportes integrados pero puede utilizarse en conjunto con otra herramienta de JBoss de código abierto, es posible además realizar simulación de procesos y control de versiones. (The JBoss jBPM team, 2014)

2.3.3 Bizagi

Semejante a BonitaSoft, Bizagi también posee 3 módulos, Bizagi Modeler es el encargado de realizar los modelos de proceso basado en BPMN al igual que los BPMS anteriores, posee también la característica de colaboración entre equipos que permite la discusión del equipo de trabajo para definir procesos a diferencia de Bonita y al igual que jBPM es posible simular los procesos antes de implementarlos.

Bizagi Studio es el módulo que permite generar una aplicación a partir de los procesos modelados, es además posible generar formularios sin código, simplemente utilizando herramientas drag and drop, al igual que con las otras herramientas es posible definir reglas de negocio pero se diferencia de jBPM por su capacidad de realizar balanceo de carga.

Bizagi Engine proporciona la plataforma de trabajo que permitirá a múltiples usuarios ingresar al sistema y realizar los procesos necesarios en un ambiente considerado amigable y de fácil utilización. (Bizagui, 2014)

2.3.4 ProcessMaker

Una parte importante de todo BPMS es la forma en la que se definen los procesos, dentro de ProcessMaker se conoce como diseñador a la herramienta que permite a los analistas de negocio diseñar un proceso utilizando funcionalidad drag and drop en una interfaz web. La funcionalidad de arrastrar y soltar es también utilizada en el diseño de formularios donde ya se incluyen certificados PKI² con campos para firmas digitales. Una de las diferencias entre ProcessMaker y los BPMS mencionados anteriormente es que sus formularios permiten ocultar campos dentro del formulario y crear documentos de salida, documentos que pueden poseer información de auto llenado.

Al igual que los BPMS analizados, ProcessMaker posee varios servicios web que facilitan su integración a programas de terceros pero además esta herramienta facilita la creación de triggers web.

² Public Key Infraestructure

La interfaz de usuario conocida dentro del sistema como bandeja de entrada es como su nombre lo sugiere semejante a la de un correo electrónico lo que facilita al usuario su uso con la característica adicional de permitir organizar las peticiones.

La administración documental de esta herramienta se realiza internamente por lo que resulta más sencillo buscar y encontrar documentos creados o procesados dentro del sistema incluyendo también aquí el control de versiones. (ProcessMaker, 2015)

2.3.5 Activiti BPM

Activiti se basa en jBPM y conserva su motor de actividades, pero pretende utilizar herramientas más sofisticadas, sus módulos aunque se dividen en las mismas 3 categorías generales del resto de BPMS poseen un mayor grado de independencia, por ejemplo en el modelado posee los módulos Modeler y Designer siendo el primero una herramienta para compilar BPMN 2.0 y el segundo un plugin de Eclipse que procesa el diseño en BPMN 2.0.

Activiti Engine es el corazón del Activiti desarrollado en Java con el cual se permite al usuario actualizar el proceso y facilita el levantamiento y ejecución de los procesos, además permite continuar con los procesos de forma asíncrona.

La interfaz de usuario es el módulo Activiti Explorer el cual provee acceso a Activiti Engine para todos los usuarios, permitiendo a estos ver sus tareas pendientes, las que él es candidato o crear y completar tareas nuevas.

Esta herramienta posee los reportes integrados permitiendo generar estadísticas de las actividades como por ejemplo el tiempo promedio para la realización de una actividad o proceso específicos. (Activiti, 2015)

En la siguiente tabla se resumen las características de los BPMS analizados para facilitar su comparación.

BPMS/ CARACTERISCA	Bonita Soft	jBPM	Bizagi	ProcessMaker	Activiti BPM
Administración de perfiles	No en community	Si	Si	Si	Si
Generación de reportes	No en community	Si, como modulo	Si	Si	Si
Simulación	No en community	Si	Si	No	No
Modelamiento de procesos	Si	Si	Si	Si	Si
Soporte para BPMN2	Si	Si	Si	Si	Si
Soporte técnico	Muy bueno	Excelente	Muy bueno	Bueno	Bueno
Nivel de Documentación	Muy bueno	Muy bueno	Muy bueno	Bueno	Bueno
Licencia	LGPL	Apache License 2.0	Modeler Freeware Studio Freeware Studio pagado	Affero GPL	Apache License 2.0
Base de tecnología	Java	Java	Java y .NET	php	Java
Balanceo de carga	No	No	Si	No	No

Tabla 1 Cuadro Comparativo

Las diferentes opciones de BPMS poseen muchas características en común, entre ellas las opciones de integración hacia otros sistemas; sin embargo existen características necesarias para este proyecto que no posee todos los BPMS libres mencionados anteriormente, por ejemplo una característica necesaria son los reportes gerenciales que BonitaSoft no posee en su versión libre y debido a las características del proyecto no se podría utilizar. La mejor opción es jBPM, herramienta que además de brindar el soporte de JBoss ofrece un chat en línea muy activo que permite realizar preguntas directamente al equipo de desarrollo del proyecto. Otra de las características importantes que impulsaron el uso de este BPMS es la tecnología en la que fue desarrollado. Este proyecto al igual que Innovativa Salud es desarrollado en Java EE lo que facilitará la integración y la edición en caso de ser necesaria. jBPM utiliza una licencia Apache que permite a los usuarios usar el software, distribuirlo, modificarlo y distribuir copias modificadas que son las características necesarias para poder implantarlo junto con Innovativa Salud. Otros BPMS proporcionan características útiles pero no necesarias para este caso como el balanceo de carga realizado por Bizagi, esta característica puede mejorar el tiempo y la cantidad de trabajo realizado por cada uno de los empleados del centro médico, pero se consideró que esta característica no constituye un factor crítico de éxito.

La bandeja de entrada de ProcessMaker, nombre con el que se conoce a su interfaz de usuario es otra característica que permite a un proyecto sobresalir en comparación con otros BPMS libres, sin embargo es una característica que no será utilizada por el requerimiento funcional 2 detallado en la sección 3.3 requisitos específicos en el que se especifica que la interfaz de usuario será Innovativa Salud.

Activi posee características semejantes a jBPM pues nació a partir de este proyecto pero se ha concentrado en su funcionamiento conjunto con Alfresco facilitando la gestión documental, característica que no sería utilizada. Las características básicas son similares entre todos los BPMS aquí analizados, el diseño del proceso se puede realizar mediante un ambiente de desarrollo web gráfico utilizando características drag and drop y los formularios se pueden editar de manera semejante, por lo que no se tomará en cuenta estas particularidades ni el soporte que todas poseen para la notación BPMN2.

2.4 Servidores de Aplicaciones Web

La selección de jBPM como la herramienta BPMS implica la utilización de dos servidores de aplicaciones web. JBoss ahora conocido como WildFly que es el servidor de aplicaciones propio de la herramienta BPMS y Glassfish servidor en el que se encuentra funcionando el sistema médico actualmente.

2.4.1 WildFly

WildFly es la evolución de JBoss AS puede considerarse como la versión 8 del servidor de aplicaciones JBoss desarrollado por Red Hat, al igual que sus predecesores WildFly es desarrollado completamente en java y con proyectos de código abierto e implementa JEE. (WildFly, 2015)

Las principales características de este servidor de aplicaciones son:

- Velocidad al inicializarse debido a que los procesos no críticos son mantenidos *congelados* hasta que sea necesario su primer uso.
- Posee un enfoque hacia la reducción del consumo de memoria.
- Provee subsistemas a modo de plugin que pueden ser añadido y removidos dependiendo de la necesidad del desarrollador lo que permite optimizar su tamaño.
- Dentro de las características más importantes dentro de este proyecto esta su soporte para JEE 7 e implementa los últimos estándares de JEE.

2.4.2 GlassFish

El servidor de aplicaciones GlassFish desarrollado completamente en Java originalmente por Sun Microsystems fue adquirido por Oracle Corporation, compañía que aportó el módulo de persistencia TopLink para el mapeo objeto relacional, permite implementar las tecnologías definidas en plataforma JEE.

La comunidad de usuarios GlassFish es una de las más activas, permitiendo descargas gratuitas a usuarios, partners que contribuyen con características y testers que convierten a este proyecto en uno de los más completos.

Su licenciamiento es dual Common Development and Distribution License y General Public License o GNU. Esta licencia en parte proporciona el nombre al servidor de aplicaciones, pues su traducción al español sería “Pez de cristal” especie de pez llamada así debido a su cuerpo es transparente y permite observar sus huesos, seméjate a las cualidad de la licencia GNU. (Serra Manchado, 2010)

2.5 Entornos de Desarrollo IDE

Un IDE³ es un software que facilita a los programadores el desarrollo de software, gracias a que integra un editor de código fuente y un debugger como características básicas, en la actualidad la mayoría de IDE poseen además características de autocompletado, análisis de código fuente y un compilador como es el caso de Eclipse y Netbeans, entornos que serán usados dentro del proyecto.

2.5.1 Eclipse

Es un conjunto de herramientas de software libre y código abierto multiplataforma, fue desarrollado originalmente por IBM, actualmente es administrado por Eclipse Foundation que es una organización que procura incentivar el desarrollo de una comunidad de código abierto además de desarrollar un conjunto de productos complementarios.

El artículo de Sean Michael Kerner “Five Years On: The Future of Eclipse” se refiere al software como una herramienta que permite a los desarrolladores a concentrarse en distinguir el producto desarrollado, en lugar de reunir esfuerzos en crear diferentes ambientes de desarrollo para diferentes plataformas.

Fue liberado bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License. Free Software Foundation ha expresado que las licencias son de software libre, pero son inconciliables con Licencia pública general de GNU (GNU GPL).

³ Ambiente de desarrollo integrado.

“La base para Eclipse es la plataforma de cliente enriquecido. Los siguientes componentes constituyen la RCP⁴: (Aguilar Mosquera & Chico Macias, 2014)

- Plataforma principal - inicio de Eclipse, ejecución de plugins
- OSGi - una plataforma para bundling estándar.
- El Standard Widget Toolkit (SWT) - Un widget toolkit portable.
- JFace - manejo de archivos, manejo de texto, editores de texto
- El Workbench de Eclipse - vistas, editores, perspectivas, asistentes

2.5.2 Netbeans

Producto libre sin restricciones de uso, actualmente es gestionado por Oracle Corporation, fue fundado en código abierto por Sun Microsystems. Es una plataforma que permite crear proyectos de escritorio o webs, se utiliza el lenguaje de programación Java el cual tiene varias herramientas como:

Aplicaciones de escritorio:

Frameworks (Java Swing)

Servicios reutilizables

Para el desarrollo web:

- **JavaServer Faces (JSF)**
 - Construcción de interfaces basadas en plantillas,
 - Rápida creación de componentes por composición,
 - Fácil creación de funciones y librerías de componentes
- **JavaServer Pages (JSP)**

Permite la utilización de código Java mediante scripts.

NetBeans es modular, cada archivo Java contiene clases java escritas para interactuar con las APIs de la plataforma, contando además con el archivo especial Manifest file identificándolo como modulo.

⁴ del Inglés Rich Client Platform.

Los módulos pueden ser creados de manera independiente volviéndose extensibles para cualquier desarrollador. Cuenta con varias versiones la más actual es NetBeans 8.0.1 (NetBeans, 2015)

2.6 Innovativa Salud

El Centro de Transferencia y Desarrollo Tecnológico de la ESPE “Innovativa” nace de los centros de capacitación informática (CECAI). Centros que tenían por misión la capacitación en informática para aquellos soldados que quedaron discapacitados tras la guerra del Cenepa en 1995 permitiéndoles ser productivos para su familia y para el país. En 2010 se transformó en la unidad de capacitación y productividad. Actualmente realiza transferencia tecnológica mediante el desarrollo de proyectos, capacitación, asesoría y consultoría. Dentro de los proyectos desarrollados se encuentra Innovativa Salud. (Innovativa, 2015)

Sin duda las Ciencias de Salud y Medicina son las áreas que mayores beneficios obtienen gracias a tecnología de información (TI) tanto en aspectos operativos como de administración y atención al cliente, suponiendo un ahorro en la gestión hospitalaria y de centros médicos. Esto motivó al desarrollo de Innovativa Salud un proyecto que consiste en desarrollar e implementar un sistema médico en los centros de salud dentro de la Universidad de las Fuerzas Armadas ESPE y comercializarlo bajo licenciamiento Open Source LGPL a otros centros de salud Militares o clínicas privadas. Al ser el proyecto escalable se dispone de un software genérico que cubre como mínimo los siguientes procesos.

Procesos Administrativos:

- Facturación
- Compras
- Bodegas
- Contabilidad

Módulos hospitalarios

- Admisión
- Historia Clínica

- Consulta Externa
- Emergencia
- Hospitalización
- Radiología
- Laboratorio Clínico y Microbiología
- Farmacia
- Quirófano

El licenciamiento libre es parte esencial del sistema Innovativa Salud por lo que el desarrollo del proceso debe necesariamente realizarse en herramientas libres para preservar el licenciamiento del proyecto. (Innovativa)

Esta condición será necesaria en cualquier proyecto futuro que sea parte de la línea de investigación informática médica para proyectos de tesis.

2.7 Tecnologías de Integración

Debido a que el BPMS debe comunicarse con Innovativa Salud, es necesario analizar las herramientas de integración que posee jBPM para así poder seleccionar la más adecuada para las necesidades propias de este proyecto.

2.7.1 SOAP

SOAP es un protocolo de invocación de objetos basado en XML y fue desarrollado originalmente para que las aplicaciones distribuidas se comuniquen a través del protocolo HTTP para esto extiende con una cabecera que identifica a un mensaje como SOAP y con un payload XML que contiene los datos reales.

La idea básica detrás de SOAP consiste en facilitar la comunicación entre varios web services coordinando los pasos entre cada uno. SOAP no depende del sistema operativo, del lenguaje de programación o del modelo de objeto.

Los objetivos de SOAP son proveer de un protocolo estándar para obtener datos en XML, que este protocolo pueda ser extendido, no provee una comunicación bidireccional y no genera objetos por referencia. SOAP debe ser capaz de funcionar sobre la infraestructura existente y que los programadores solo necesiten conocer la manera en que se invoca a un servicio. (Koftikian, 2015)

SOAP está basado en el intercambio de mensajes, los mensajes son encapsulados en sobres o <Envelope> que contiene un <Header> que es opcional y un <body> obligatorio. El body es el área del mensaje SOAP en la que se produce el intercambio de datos todo el mensaje SOAP <Envelope> se encapsula dentro del protocolo HTTP puede ser tanto una petición GET como POST en caso de falla el error será el mismo usado dentro de http. (Papazoglou, 2015)

Las principales ventajas de SOAP son:

- Simplicidad
- Portabilidad
- No tiene problemas con firewalls
- Usa estándares abiertos.
- Interoperatividad.
- Aceptación universal

2.7.2 REST

Representational State Transfer es un estilo de arquitectura que se encuentra basado en estándares HTTP, URL, HTML entre otros. REST pretende utilizar una de las características que permitieron a la web convertirse en la más importante aplicación distribuida, esta característica es el manejo de URLs la que permiten tejer una red para la comunicación de mayores aplicaciones, al tratarse de una arquitectura podría considerarse a http como una aplicación en REST, pero para este proyecto y analizando a REST como método de integración entre varios sistemas hare referencia a aquellos servicios web basados en REST.

Grandes empresas como Google consideran a SOAP muy complicado por lo que han migrado a la utilización de servicios web basados en REST lo que da origen a un debate de SOAP vs REST y existen argumentos a favor y en contra de ambas tecnologías, por ejemplo hay quienes apoyan el uso de REST argumentan que este tipo de servicio web permite un mayor número de usuarios y que SOAP genera brechas de seguridad en un firewall.

Entre las principales características de REST se tiene que cada objeto soporta las operaciones estándares definidas, menor consumo de recursos, posee una dirección única para cada instancia, las operaciones son definidas en el mensaje, stateless⁵ del lado del servidor y puede ser usado con https. (Barbero, 2015)

En una visión a futuro SOAP posee una gran desventaja y es la incapacidad de exponer interfaces en común a varios socios por lo que si la tendencia actual continúa es posible que REST ocupe en su totalidad los servicios web prueba de esto es que “Amazon posee ambos estilos de uso de sus servicios Web. Pero el 85% de sus clientes prefieren la interfaz REST” (Navarro Marsler, 2015)

⁵ No posee estado.

CAPÍTULO 3.- ANÁLISIS Y DISEÑO

3.1 Análisis de requerimientos

3.1.1 Introducción

En este documento, se presentaran las necesidades y características del proceso de aprobación de planillas de seguro para su correcto funcionamiento e integración al sistema Innovativa Salud.

3.1.2 Ámbito del sistema

El presente proyecto tiene como finalidad implementar un Business Process Management (BPM), el cual servirá para controlar el proceso de aprobación de la planilla de seguros, esto debido a que existe falta de control en el actual manejo de pagos de seguros. Los administradores de los centros médicos necesitan ser capaces de verificar qué instancia retrasa o detiene el proceso para cada una de las planillas generadas. La definición de un proceso y la administración del mismo con una herramienta BPM da a los administradores la capacidad de identificar y corregir los problemas existentes que generan la importante suma de cuentas por cobrar, esta herramienta permitirá a los administradores verificar si la falla es interna (dentro del centro médico) o externa (compañías de seguros).

En caso de que la falla sea interna tendrá la capacidad de tomar decisiones informadas. En caso de que la falla es externa se tendrá que informar a la compañía de seguro.

El proceso se integrará al sistema de gestión hospitalaria Innovativa Salud, desarrollado por el Centro de Trasferencia Tecnológica de la Universidad de las Fuerzas Armadas ESPE.

El sistema posee un módulo de facturación el cual administra la generación de planillas. En este punto es necesario iniciar el proceso de aprobación. La solución planteada con BPM permitirá a la gerencia controlar el proceso de aprobación de las planillas desde este punto hasta el cobro final de la factura, al estar integrado el BPMS al sistema Innovativa Salud el control se realizará internamente por lo que los usuarios

no tendrán la necesidad de cambiar la manera en que actualmente realizan sus actividades, el proceso en la actualidad se realiza de forma manual lo que complica el seguimiento de un trámite que impide conocer el estado actual de cualquier planilla.

Este proceso es manejado de manera semejante en la mayoría de centros médicos del país y cada uno de ellos posee los mismos inconvenientes con las cuentas por cobrar. La integración de un BPM al sistema médico significará para los administradores una solución integral a varios de sus problemas con un solo gasto, dando así a Innovativa Salud una ventaja competitiva que se suma a las que actualmente posee. Esta integración facilitará además el manejo del área contable y permitirá a los centros de salud brindar un mejor servicio a sus clientes.

El proyecto además fomenta el trabajo conjunto del Centro de Transferencia y Desarrollo Tecnológico de la Universidad de las Fuerzas Armadas ESPE (CTT-ESPE-CECAI) con el Departamento de Ciencias de la Computación (DECC) y la Carrera de Ingeniería en Sistemas e Informática.

Para conseguir controlar el proceso de retorno de dinero de planillas de seguros será necesario comparar las herramientas BPMS libres disponibles en el mercado según las necesidades propias del proyecto y seleccionar la más apropiada, sobre esta herramienta se diseñara el proceso que posteriormente será integrado al Sistema Innovativa Salud, esta integración permite obtener datos del proceso que serán utilizados en la generación de reportes consiguiendo de esta manera un mayor control del proceso.

Detalladamente el proceso tendrá las siguientes etapas:

Generación de planilla

- Generar planilla inicial.
- Ingreso de información hospitalaria
- Verificación de datos ingresados.
- Envío de planilla.

Aprobación

- Evaluación de aseguradora.
- Corrección de planilla.
- Reenvío de planilla.

Envío de Factura

- Selección de planillas aprobadas.
- Envío de factura final.
- Ingreso del valor a cuentas por cobrar del centro médico.
- Confirmación de pago de factura de la aseguradora.

El sistema no permitirá acceso al personal de aseguradoras por lo que el registro de la aprobación de planillas de seguros lo realizará el usuario encargado de la emisión de las mismas dentro del centro médico.

3.2 Descripción General

El proyecto requiere el desarrollo de un proceso en BPMN que permita controlar la aprobación de planillas de seguros generadas en el sistema Innovativa Salud, por lo que el BPMS deberá estar integrado al sistema médico. Innovativa Salud es un sistema multiempresa lo que implica diversidad en usuarios y permisos, permisos que deberán ser considerados dentro del BPMS.

3.2.1 Funciones del producto

El producto realizará seguimiento a las planillas de seguros, facilitando el control sobre éste proceso, el seguimiento se realizará utilizando a Innovativa Salud como interfaz de usuario, debido a que en este sistema se realizan actualmente las operaciones y actividades definidas dentro del proceso de aprobación de planillas, al realizarse cada una de las actividades Innovativa Salud utilizará servicios dentro de jBPM que le permitan al BPMS realizar el seguimiento del proceso. Los datos obtenidos mediante la herramienta deberán ser presentados en informes que describan de manera clara el funcionamiento de este proceso.

3.2.2 Características de los usuarios

En la actualidad el sistema Innovativa Salud se encuentra funcionando en los centros médicos de la Universidad de las Fuerzas Armadas ESPE sin embargo al tratarse de un sistema multiempresa estas implementaciones no serán las únicas y el número y variedad de usuarios se incrementaran a medida que se aumente la cantidad de implementaciones.

Para los centros médicos al interior de la Universidad se han considerado los siguientes roles de usuario:

CAJA: El personal de caja será el principal usuario del proceso encargado de generar las planillas y enviarlas a las aseguradoras.

MÉDICO: La función del médico dentro del sistema será ingresar el diagnostico dentro de la planilla para su posterior auditoria por parte de las aseguradoras.

ADMINISTRACIÓN: El administrador controlará el correcto funcionamiento del proceso y tendrá acceso a los informes generados por el BPMS.

3.2.3 Suposiciones y dependencias

- Conectividad entre equipos de los clientes y el servidor de aplicaciones.
- Innovativa Salud proporcionara la interfaz de usuario para el proceso de aprobación de planillas.

3.2.4 Requisitos futuros

- Será necesario crear nuevos procesos, usuarios y reportes en casos en que Innovativa Salud se implemente en nuevos centros médicos.
- El BPMS podrá ser usado para controlar otros procesos dentro de los centros médicos o la totalidad de estos.

3.3 Requisitos Específicos

3.3.1 Requisitos Funcionales

Número de requisito	RF1
Nombre de requisito	Proceso generado en BPMS
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Fuente del requisito	stakeholder
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Eencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Número de requisito	RF2
Nombre de requisito	Integrar proceso a la interfaz de usuario de Innovativa Salud
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Fuente del requisito	stakeholder
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Eencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Número de requisito	RF3
Nombre de requisito	Utilizar usuarios de Innovativa Salud en BPMS
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Fuente del requisito	stakeholder
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Eencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Número de requisito	RF4
Nombre de requisito	Utilizar postgres como base de datos de BPMS
Tipo	<input type="checkbox"/> Requisito <input checked="" type="checkbox"/> Restricción
Fuente del requisito	stakeholder
Prioridad del requisito	<input type="checkbox"/> Alta/Eencial <input checked="" type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Número de requisito	RF5
Nombre de requisito	Generación de reportes del proceso
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Fuente del requisito	stakeholder
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Eencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Número de requisito	RF6
Nombre de requisito	Seguimiento de planillas de manera individual
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Fuente del requisito	stakeholder
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Eencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Número de requisito	RF7
Nombre de requisito	La facturación final se realiza en por grupos de planillas pero el proceso es individual para cada una
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Fuente del requisito	stakeholder
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Eencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Número de requisito	RF8
Nombre de requisito	En caso en que el BPMS no pueda ejecutarse, innovativa Salud debe poder seguir operando
Tipo	<input type="checkbox"/> Requisito <input checked="" type="checkbox"/> Restricción
Fuente del requisito	stakeholder
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Eencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Número de requisito	RF9
Nombre de requisito	Si se utilizan servicios es necesaria la creación de una capa intermedia a modo de bus de servicios
Tipo	<input type="checkbox"/> Requisito <input checked="" type="checkbox"/> Restricción
Fuente del requisito	stakeholder
Prioridad del requisito	<input type="checkbox"/> Alta/Eencial <input checked="" type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Número de requisito	RF10
Nombre de requisito	Innovativa salud es concebido como sistema multiempresa por lo que el proceso debe funcionar de igual manera
Tipo	<input checked="" type="checkbox"/> Requisito <input type="checkbox"/> Restricción
Fuente del requisito	stakeholder
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Eencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

3.3.2 Interfaces externas

El usuario utilizará Innovativa Salud para la ejecución del proceso por lo que la interfaz de usuario de este sistema será en la que se realicen las tareas del proceso, el seguimiento y reportes podrán ser visualizados en la interfaz de jBPM.

3.3.3 Atributos del sistema

Debido a que no podemos detener la atención a los clientes del centro médico es importante que si debido a un error el BPMS no puede realizar el seguimiento de las planillas Innovativa Salud permanezca en funcionamiento.

Los usuario en jBPM serán los mismos de Innovativa Salud y deberán estar clasificados por el centro médico al que pertenecen.

3.4 Análisis del sistema

3.4.1 Descripción del funcionamiento

Se plantea la realización de un módulo dentro del sistema innovativa salud que gestione el proceso de aprobación de planillas de seguros en la que existen una serie de usuarios quienes realizaran las actividades del proceso que se detallan a continuación. Los usuarios que forman parte del proceso serán creados y tendrán perfiles de acuerdo con sus perfiles en Innovativa Salud. Permitiendo así integrar de mejor manera al proceso desarrollado en jBPM con el sistema médico.

El proceso inicia con la cita médica realizada por el personal de caja y el paciente para esto se debe tener historial médico registrado dentro del sistema, si esta condición no se cumple el sistema solicitará la creación del historial médico sin embargo esta tarea no es considerada como parte del proceso de aprobación de una planilla de seguro, la cita emitida incluye al nuevo cliente en el listado de pacientes por atender del doctor sea general o de especialidad, durante la atención el doctor llenará datos importantes para la posterior auditoría de la empresa aseguradora como el motivo de la consulta, medicamentos recetados, si es una primera consulta o seguimiento entre otros datos que se agregan tanto a la planilla de seguros como al historial del paciente.

Existe la posibilidad de que estos datos no sean ingresados en casos en los que el paciente solicite exámenes médicos solicitados por médicos externos por lo que necesariamente la atención será obviada, en ambos casos antes de enviar las planillas estas deben ser aprobadas tras una revisión de los datos que estas contienen en este punto el administrador puede decidir cancelar la planilla y terminan con el proceso.

El siguiente paso dentro del proceso es la auditoria de las planillas realizado por las empresas aseguradoras, esta auditoría al tratarse de un proceso interno dentro de las instituciones no será parte de este módulo, sin embargo es necesario conocer la respuesta de estos organismos por lo que la auditoria será definida como la recepción de la respuesta a las planillas. Las respuestas pueden ser de aprobación y de rechazo, en caso de rechazo será necesaria una edición de la planilla que permita cumplir con

los requerimientos de la aseguradora para la aprobación final del documento por lo que la planilla será enviada nuevamente hasta quedar aprobada.

El listado de planillas aprobadas están listas para su facturación, la cual se realiza de manera similar a las facturas emitidas a los clientes del centro médico con la salvedad de que en este caso la factura es emitida hacia la aseguradora con todos los datos de las planillas aprobadas, como resulta lógico estas empresas no pagan las planillas en efectivo por lo que el valor total de la factura entra en las cuentas por cobrar de Innovativa Salud. Para terminar el proceso es necesario ingresar el pago de la factura.

3.4.2 Descripción de la información

El módulo que se propone desarrollar requiere la siguiente información, la misma que es listada a continuación, se encuentra dentro de la base de datos necesaria para jBPM y en la base de datos de Innovativa Salud, por lo que no es necesaria la creación de nuevas tablas, y no se listarán la totalidad de datos necesarios para estos sistemas pues la mayoría no influyen en este proyecto. Fue necesaria la edición de la tabla que almacena los datos de los usuarios dentro de Innovativa Salud, y añadir un campo que identifique al proceso en la cabecera de la factura o planilla, siendo estos los únicos cambios necesarios por lo que se lo describirá con mayor detalle:

- **Instancia del Proceso**
 - *Intanceid*:: Código que identifica la instancia del proceso
 - *Lastmodificationdate*: Fecha en que fue modificado el proceso por última vez
 - *Lastreaddate*: Fecha de última lectura
 - *Processid*: Id del proceso
 - *Processinstancebytearrayoid*: Código de instancia del proceso
 - *Startdate*: Fecha de inicio
 - *State*: Estado del proceso

- **Task**

- *Id*: Identifica a la tarea
- *Archived*: entero que identifica el archivo
- *Allowedtodelegate*: Identifica a quien se puede delegar una tarea
- *Formname*: Nombre del formulario
- *Priority*: Prioridad de la tarea
- *Subtaskstrategy*: Estrategia para subtareas
- *Activationtime*: momento de activación de la tarea
- *Creadeon*: Fecha de creación
- *Deploymendid*: Fecha de despliegue
- *Documentaccesstype*: Tipo de acceso a documentación,
- *Documentcontentid*: Código del contenido del documento
- *Documenttype*: Tipo de documento
- *Exporationtime*: Tiempo de expiración
- *Faultaccesstype*: Tipo de acceso
- *Foultcontentid*: Código para identificar fallos
- *Foultname*: Nombre de fallo
- *Faulttype*: Tipo de fallo
- *Outputaccesstype*: Entero que identifica tipo de acceso a la salida
- *Outputcontentid*: Entero que identifica al contenido de salida
- *Outputtype*: Tipo de salida
- *Parentid*: Entero que identifica a la actividad padre
- *Previousstatus*: Estado anterior
- *Processid*: Proceso al que pertenece la tarea
- *Processinstanceid*: Instancia a la que pertenece la tarea
- *Processsessionid*: Sesión a la que pertenece la tarea
- *Skipable*: Boleando que permite identificar si la tarea se puede omitir
- *Status*: Estado de la tarea
- *Workitemid*: Entero que permite identificar al objeto de trabajo
- *Tasktype*: Tipo de actividad
- *Oplock*: Entero que permite saber estado de bloqueo de una actividad
- *Taskinitiator_id*: Nombre de la tarea que inicia la tarea actual

- *Actualowner_id*: Propietario actual de la tarea
- *Createdby_id*: Id del usuario que inicio la tarea
- *Description*: Descripción de la tarea
- *Name*: Nombre de la tarea
- **Taskevent**
 - *Id*: Entero que identifica al evento
 - *Logtime*: Hora del evento
 - *Processinstanceid*: Instancia del proceso
 - *Taskid*: Id de la actividad
 - *Userid*: Usuario que realizó el evento
 - *Oplock*: Entero que permite saber estado de bloqueo de una actividad
 - *Workitemid*: Entero que permite identificar al objeto de trabajo
- **Variableinstancelog**
 - *Id*: Entero que identifica registro
 - *Log_date*: Fecha del registro
 - *Externalid*: Id que identifica al proceso
 - *Oldvalue*: Valor anterior de la variable
 - *Processid*: Id del proceso
 - *Processinstanceid*: Id de la instancia del proceso
 - *Value*: Valor de la variable
 - *Variableid*: Id de la variable
 - *Variableinstanceid*: Id de la instancia de la variable
- **Bamtasksummary**
 - *Pk*: Entero que identifica el registro del resumen.
 - *Createdate*: Fecha de creación.
 - *Duration*: Duración
 - *Enddate*: Fecha de culminación.
 - *Processinstanceid*: Instancia del proceso a la que pertenece el resumen de la tarea.
 - *Startdate*: Fecha de inicio.

- *Status*: Estado
- *Taskid*: Id de la tarea.
- *Taskname*: Nombre de la tarea.
- *Userid*: Id del usuario.
- *Optlock*: Entero que permite saber estado de bloqueo de una actividad.
- **processintancelog**
 - *id*: Entero que identifica una instancia de proceso.
 - *duration*: Duración de la instancia.
 - *end_date*: Fecha de culminación del proceso
 - *externalid*: Identifica al proceso y versión de este.
 - *user_identity*: Usuario que inicia el proceso
 - *outcome*: Salida del proceso.
 - *Parentprocessinstanceid*: Instancia del proceso padre.
 - *Processid*: Id de proceso.
 - *Processintanceid*: Id de instancia de proceso.
 - *Processname*: Nombre del proceso.
 - *Processversion*: Versión del proceso.
 - *start_date*: Fecha de inicio.
 - *Status*: Estado del proceso.
- **Seuss_ursistem**
 - *Uss_identifica*: Usuario del sistema
 - *Cuni_id*: Id de unidad.
 - *Cui_cedula*: Cédula del usuario.
 - *Uss_password*: Clave del usuario.
 - *Uss_fecpas*: Fecha de creación del usuario.
 - *Uss_estado*: Estado de usuario.
 - *Código_hospital*: Hospital al que pertenece el usuario.
 - *Cargo*: Cargo al interior del Innovativa
 - *Rol_jBPM*: Rol al interior de jBPM.

Este campo es el único añadido a las bases de datos, para eso ejecutamos el siguiente script.

```
ALTER TABLE postgres.seuss_ursistem ADD COLUMN rol_jBPM character
varying(50);
ALTER TABLE postgres.seuss_ursistem ALTER COLUMN rol_jBPM SET
DEFAULT 'user'::character varying;
```

- **Aedph_hpaciente**

- *Dpa_hclinica*: Código que identifica la historia clínica.
- *Hdp_fcambio*: Fecha de última edición.
- *Hdp_ecivil*: Estado civil del paciente.
- *Hdp_ocupacion*: Código de ocupación del paciente.
- *Hdp_empresa*: Empresa a la que pertenece el paciente.
- *Hdp_seguro*: Código del seguro si el paciente posee uno.
- *Hdp_direccion*: Dirección del paciente
- *Hdp_barrio*: Barrio del paciente.
- *Hdp_zona*: Zona en la que vive el paciente.
- *Hdp_parroquia*: Parroquia en la que vive el paciente.
- *Hdp_canton*: Canto de la vivienda del paciente.
- *Hdp_provincia*: Provincia de residencia del paciente.
- *Hdp_telefono*: Teléfono del paciente.
- *Dpa_fnacimiento*: Fecha de nacimiento.
- *Dpa_sexo*: Sexo del paciente.
- *Dpa_nombres*: Nombre del paciente.
- *Dpa_apellidos*: Apellido del paciente.
- *Dpa_fechaereg*: Fecha del registro de la historia clínica.
- *Uss_identifica*: Usuario que generó la historia.

- **Aeseg_seguros**

- *Seg_codigo*: Código de seguro.
- *Seg_descripcion*: Descripción del seguro.
- *Seg_ruc*: Ruc de la aseguradora.
- *Seg_estado*: Estado de aseguradora.
- *Seg_procentaje*: Porcentaje asegurado.
- *Seg_tipo*: Tipo de seguro.

- **Cabecera_factura**

- *Código*: Código de la cabecera.
- *Número*: Número de factura.
- *Establecimiento*: Lugar en que se emitió la factura.
- *Código_documento*: Código que diferencia facturas de planillas.
- *Uss_identifica*: Usuario que emite el documento.
- *Fecha*: Fecha de emisión del documento.
- *Subtotal*: Subtotal a pagar por cliente.
- *Total*: Total que debe pagar el cliente.
- *Iva*: Iva cobrado.
- *Descuento*: Descuentos aplicables.
- *Estado*: Estado del documento.
- *Contabilizado*: Booleano que permite conocer si el documento fue contabilizado.
- *Contabilizadocosto*: Booleano que permite identificar si posee un costo por contabilización.
- *Justificativo*: Si se anula un documento es necesario detallar el motivo en este campo.
- *Numero_planilla*: Número de planilla
- *Usuario_anula*: Usuario que anula un documento.
- *Fecha_anula*: Fecha de anulación de un documento.
- *Lugar_descargo*: Lugar en que se realizó un descargo.
- *Dpa_hclinica*: Historia clínica relacionada con el paciente.
- *Med_cedula*: Cédula del médico.
- *Crédito*: Booleano que permite identificar si el pago del documento es mediante crédito.
- *Porcentaje_iva*: Porcentaje de IVA cobrado.
- *Paciente*: Código el paciente.
- *Código_seguro*: Código de seguro.
- *Código_convenio*: Código del convenio existente con la aseguradora.

- *Estado_planilla*: Estado de planilla, utilizado para realizar seguimiento del proceso de aprobación.
- *Cancelado_credito*: Identifica si el crédito ya fue cancelado.
- *Código_hospital*: Código del hospital que emite el documento.
- *Id_proceso*: Identifica al proceso al que pertenece una planilla
- **Cehcl_historia_clinica:**
 - *Dpa_hclinica*: Código de la historia clínica.
 - *Hcl_motconsulta*: Motivo de la consulta.
 - *Hcl_enfermedadactual*: Enfermedad actual.
 - *Hclo_estado*: Código del estado.
 - *Hcl_pmedica*: Prescripción médica.
 - *Hcl_fechainicio*: Fecha de inicio de la historia clínica.
 - *Código_turno*: Código del turno atendido.
 - *Hcl_antpersonales*: Antecedentes personales del paciente.
- **Tipo_cobertura**
 - *Código_tipocobertura*: Código que identifica al tipo de cobertura.
 - *Nombre*: Nombre o descripción del tipo de cobertura.
 - *Porcentaje*: Porcentaje cubierto.

3.5 Diseño del sistema

3.5.1 Diagrama de implementación:

El siguiente diagrama ilustra el funcionamiento y la comunicación existente entre los diferentes módulos de este proyecto.

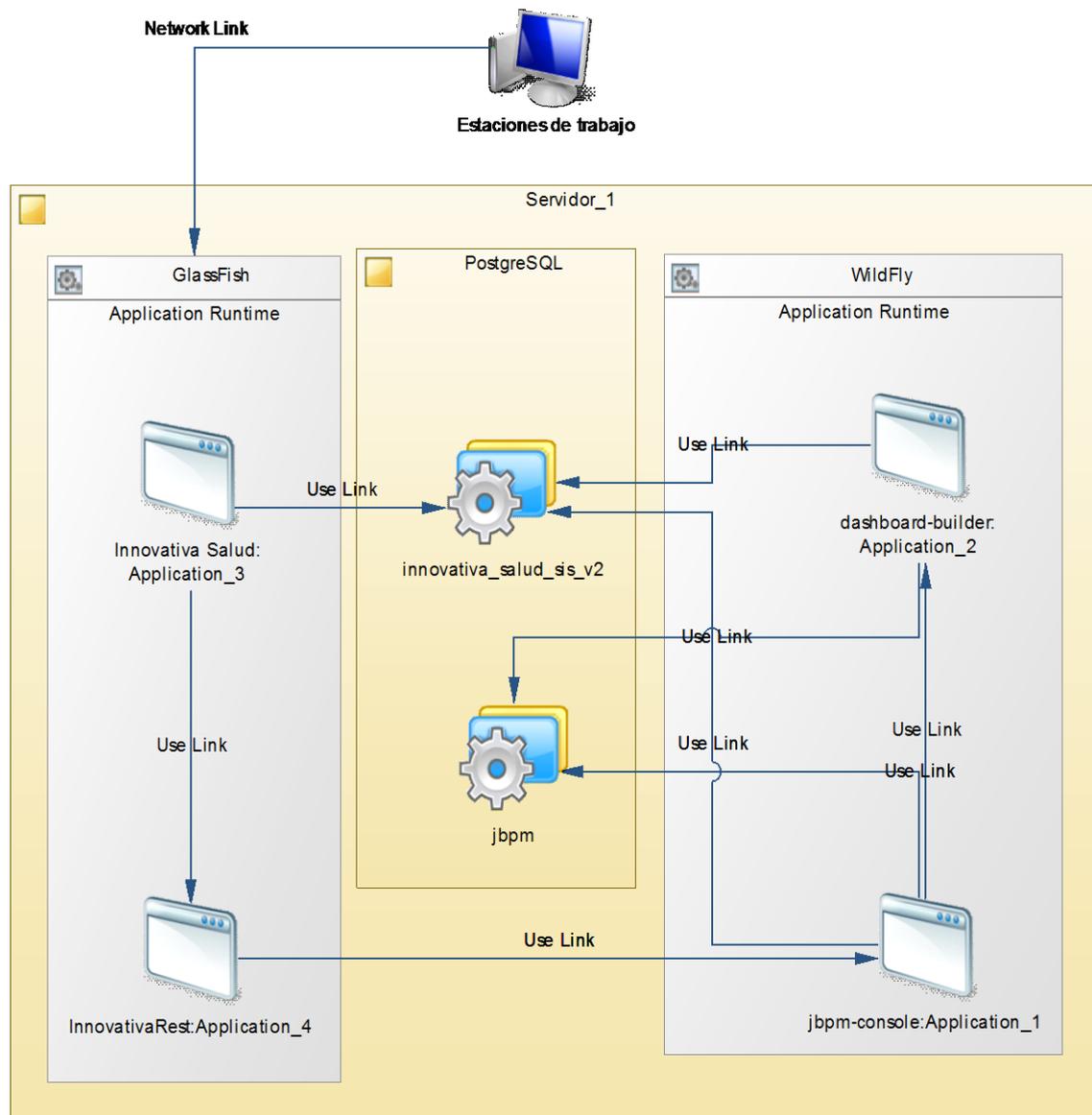


Diagrama 1 Implementación

3.5.2 Diseño del proceso

El proceso dentro del centro médico de la Universidad de las Fuerzas Armadas ESPE se encuentra definido pero no documentado, tras el levantamiento de proceso se definió que se realizaban las siguientes tareas con los tiempos aproximados que estas requieren:

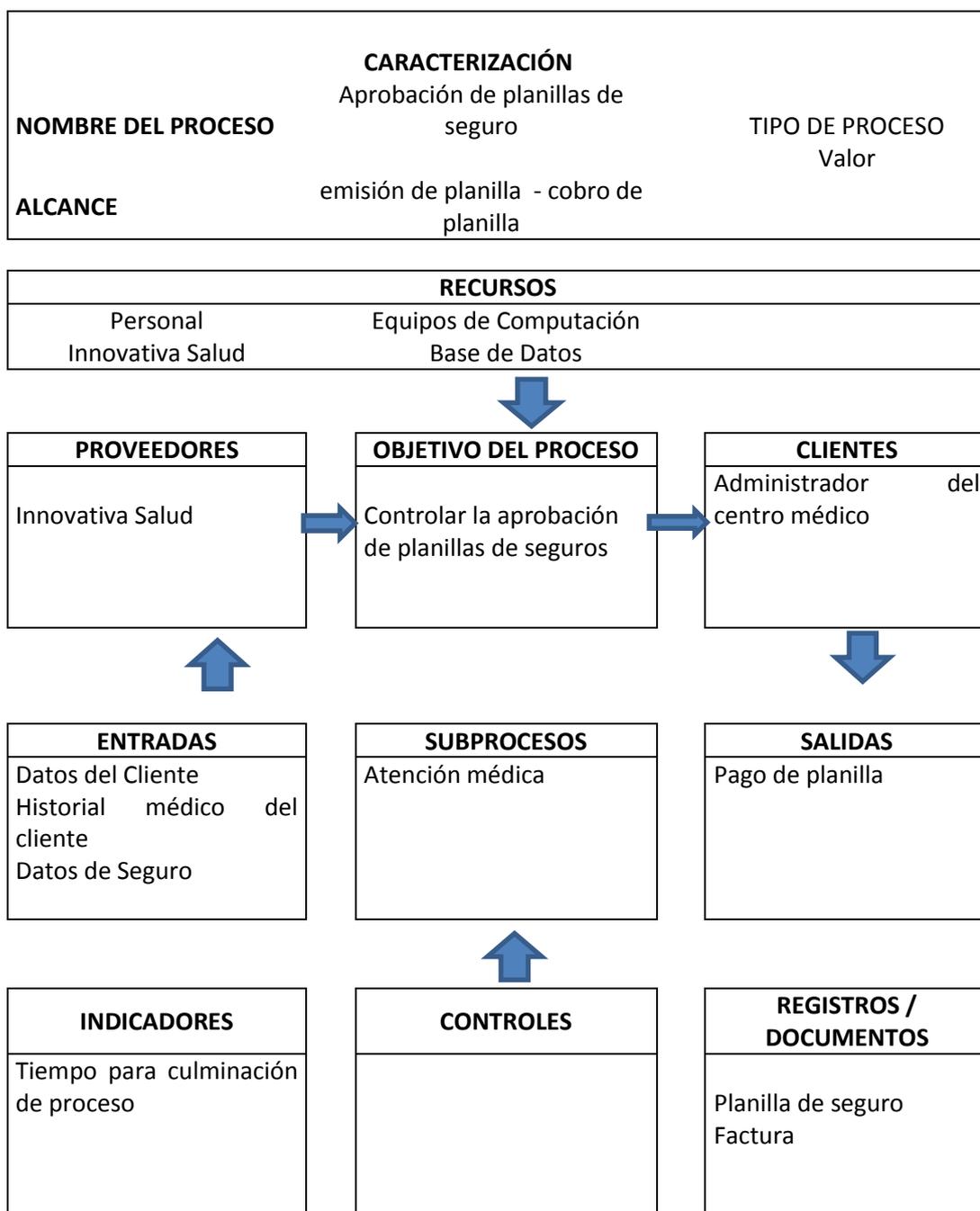


Diagrama 2 Caracterización del Proceso

		PROCESO: Aprobación de planillas de seguro LEVANTAMIENTO DE PROCESOS				
PROCESO	Aprobación de planillas de seguro	CÓDIGO	P001			
ENTRADA DEL PROCESO	Datos del cliente, Historial médico del cliente, Datos de Seguro	TIEMPO DE CICLO	170			
SALIDA DEL PROCESO	Pago de planilla	EFICIENCIA EN TIEMPO	53.35%			
NO.	ACTIVIDADES	DIAGRAMA DE FLUJO				
		Caja	Medico	Administrador	Tiempo Agrega Valor(min)	Tiempo no Agrega Valor (min)
1	Emitir de planilla				5	
2	Decidir servicio					1
3	Atender a paciente				15	
4	Completar planilla				5	
5	Revisar planilla					20
6	Enviar planilla					20
7	Receptar respuesta					15
8	Decidir según respuesta					5
9	Emitir factura				2	
10	Realizar cambios				60	
11	Incluir en cuentas por cobrar					10
12	Verificar pago				2	
13	Retirar de cuentas por cobrar					10
Total					89	81

Diagrama 3 Proceso Original de Aprobación de Planilla

Como podemos observar el tiempo que agrega valor es ligeramente mayor a aquel que no lo agrega por lo que se considera mejorar el proceso, parte de esta mejora se realiza gracias a la automatización de ciertas tareas con Innovativa Salud o integrando dos tareas como es la atención al paciente y el ingreso de datos a la planilla ya que se trata de los mismos datos que el médico ingresa en el historial clínico del paciente, con estas consideraciones se propone el control sobre el siguiente proceso.

		PROCESO: Aprobación de planillas de seguro LEVANTAMIENTO DE PROCESOS				
PROCESO	Aprobación de planillas de seguro	CÓDIGO	P001	TIEMPO DE CICLO	108	
ENTRADA DEL PROCESO	Datos del cliente, Historial médico del cliente, Datos de Seguro	EFICIENCIA EN TIEMPO	77.77%			
SALIDA DEL PROCESO	Pago de planilla					
NO.	ACTIVIDADES	DIAGRAMA DE FLUJO				
		Caja	Medico	Administrador	Tiempo Agrega Valor (min)	Tiempo no Agrega Valor (min)
1	Emitir de planilla				5	
2	Decidir servicio					1
3	Atender a paciente				15	
5	Revisar planilla					20
6	Auditoria Aseguradora					--
8	Decidir según respuesta					1
9	Emitir factura				2	
10	Realizar cambios				60	
11	Incluir en cuentas por cobrar					1
12	Verificar pago				2	
13	Retirar de cuentas por cobrar					1
Total					84	24

Diagrama 4 Proceso Recomendado

Con este proceso se pasa de una eficiencia de 53.35% al 77.77% aproximadamente gracias principalmente a la reducción de tiempo producto de automatizar tareas que no agregan valor al proceso.

3.6 Proceso en BPMN2

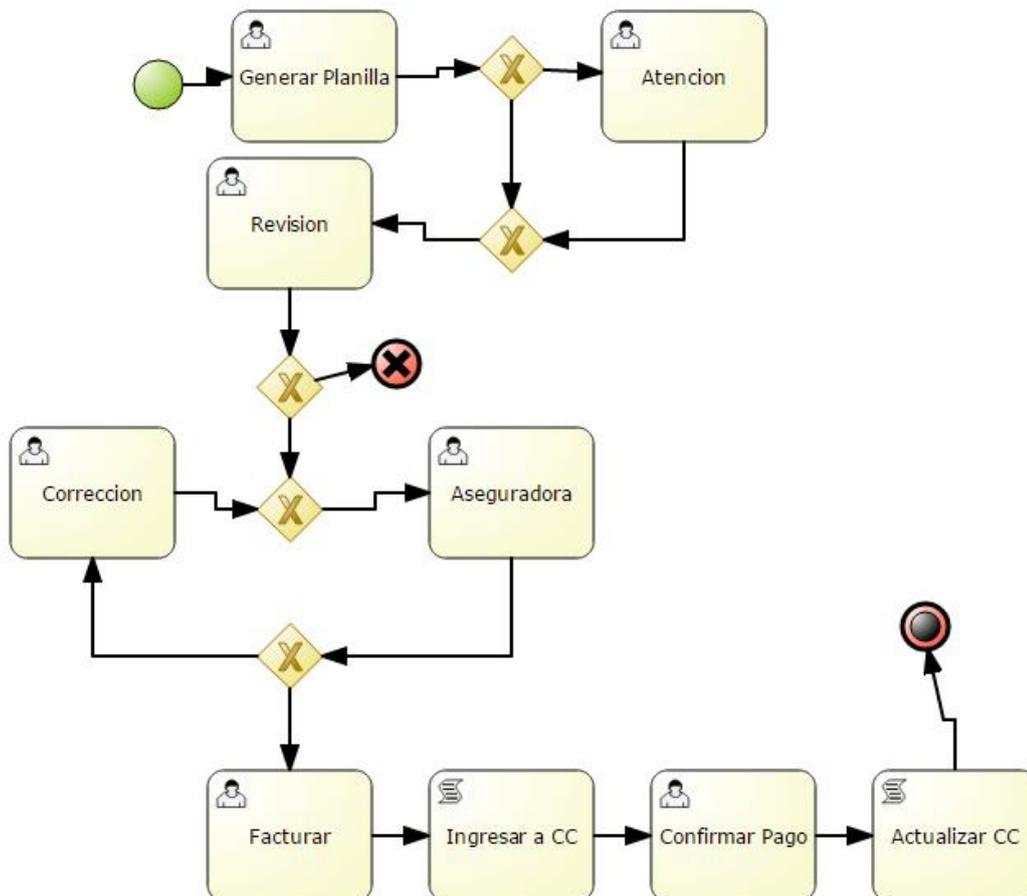


Diagrama 5 Proceso BPMN2

El código es autogenerado por jBPM en el momento de diseñar al proceso anterior mediante su interfaz gráfica, sin embargo y debido a que es necesario que el sistema sea multiempresa este código debe ser modificado para cada una de las empresas y esta tarea se realizará de manera manual directamente en el código.

La herramienta BPMS permite simular el proceso, instrumento que facilita la optimización tras el modelado, esta optimización puede darse tanto en tiempo como en costo de producción, en el proceso que nos compete la optimización se realiza en tiempo, específicamente en la eficiencia del tiempo en los gráficos 1, 2 y 3 podemos observar el resultado de las simulaciones realizadas.

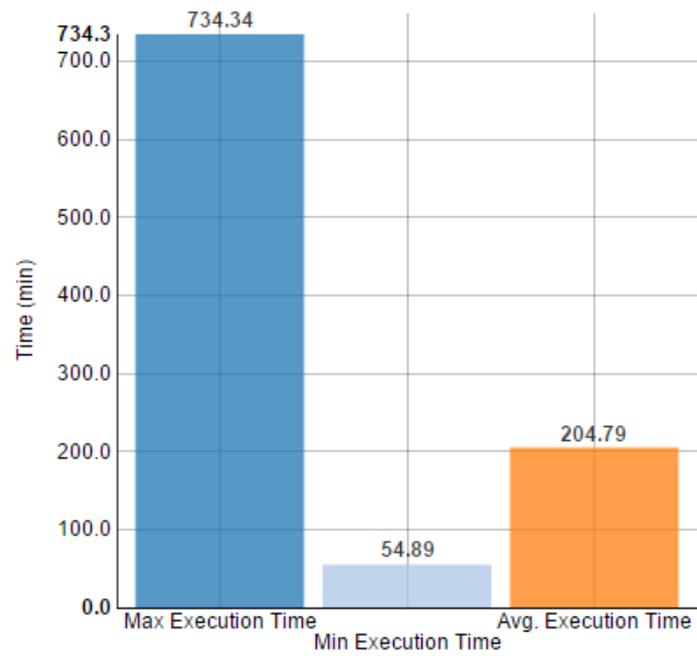


Gráfico 1 Tiempo simulación.

● Max Execution Time ● Min Execution Time
● Avg. Execution Time

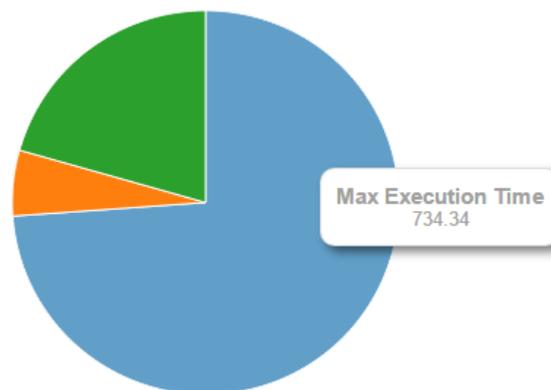


Gráfico 2 Tiempo simulación pastel

Veces ejecutadas		
Max Execution Time (min)	Min Execution Time (min)	Avg. Execution Time (min)
734.34	54.89	204.79

Instancias de actividad								
Facturar (#)	Correccion (#)	Atencion (#)	Revision (#)	Aseguradora (#)	Ingresar a CC (#)	Generar Planilla (#)	Confirmar Pago (#)	Actualizar CC (#)
67	50	38	50	67	67	50	67	67

Gráfico 3 Resumen Simulación

3.7 Diseño de reportes gerenciales

Los reportes se realizarán utilizando la herramienta de JBoss, Dashboard Builder. Lo que permite mayor flexibilidad en la información y en la presentación de datos, por lo que cada empresa podrá cambiar los gráficos y datos presentados según sus necesidades. En esta sección se mostraran la información que por defecto se entregara a cada centro médico.

Para la realización de los gráficos es necesario un proveedor de datos que además se utilizará como filtro para evitar que se mezclen los procesos de diferentes centros médicos. Esto se realiza en la pestaña Administrador/proveedores de datos, el tipo será una consulta sql, y el Data Source PostgresDS1 detallado en la sección 4.3 correspondiente a la integración de los reportes. El primer proveedor de datos permitirá generar gráficos, concederá el dato numérico de los procesos completados, activos, pendientes, suspendidos, abortados y el total mediante la siguiente sentencia sql.

```

select total.processname, coalesce(total.instances,0) total,
       Coalesce(active.instances_act,0) active,
       coalesce(completed.instances_compl,0) completed,
       coalesce(pending.instances_pend,0) pending,
       coalesce(suspended.instances_susp,0) suspended,
       coalesce(aborted.instances_abrt,0) aborted

from (select pi.processinstanceid as pId, pi.processname as
processname, count(*) as instances from processinstancelog pi group by
pi.processinstanceid,processname) as total left outer join

(select pi.processinstanceid as pId, count(*) as instances_act from
processinstancelog pi where pi.status=1 group by pi.processinstanceid)
as active on (total.pId=active.pId)

left outer join

(select pi.processinstanceid as pId, count(*) as instances_compl from
processinstancelog pi where pi.status=2 group by pi.processinstanceid)
as completed on (total.pId=completed.pId)

left outer join

(select pi.processinstanceid as pId, count(*) as instances_pend from
processinstancelog pi where pi.status=0 group by pi.processinstanceid)
as pending on (total.pId=pending.pId)

left outer join

(select pi.processinstanceid as pId, count(*) as instances_susp from
processinstancelog pi where pi.status=4 group by pi.processinstanceid)
as suspended

on (total.pId=suspended.pId)

left outer join

(select pi.processinstanceid as pId, count(*) as instances_abrt from
processinstancelog pi where pi.status=3 group by pi.processinstanceid)
as aborted on (total.pId=aborted.pId) where processname='Planillas_1'

```

El segundo proveedor de datos, nos brinda información de los procesos propiamente dichos, como su nombre, estado, fecha de inicio y culminación, versión y la duración total del proceso. Este proveedor de datos utiliza el mismo Data Source y maneja la siguiente consulta sql:

```
select processinstanceid,  
  
    processname,  
  
    status,  
  
    start_date,  
  
    end_date,  
  
    user_identity,  
  
    processversion,  
  
    duration  
  
from processinstancelog where processname='Planillas_1'
```

El tercer proveedor de datos, nos permite obtener información detallada de las diferentes tareas como nombre, nombre del proceso al que pertenece, fecha de creación y culminación, usuario, duración y estado. Con esta sentencia sql:

```
select ts.taskid,  
  
ts.processinstanceid,  
  
ps.processname,  
  
ps.processversion,  
  
ts.taskname,  
  
ts.createddate,  
  
ts.enddate,  
  
ts.userid,  
  
ts.duration,  
  
ts.status  
  
from bamtasksummary ts  
  
left join processinstancelog ps on  
(ts.processinstanceid=ps.processinstanceid) where  
ps.processname='Planillas_1'
```

La distribución de la información se la realizará según la Figura 1, se detallarán cada una de las secciones definidas a continuación:

1. Resumen representa al área en la cual se presentará al usuario datos generales del proceso como el número de tareas realizadas o el total de instancias de proceso generadas y en datos numéricos los estados de los procesos, permitiendo al usuario tener un acceso inmediato al funcionamiento general del proceso.

2. La sección filtros permite al usuario modificar la información que se presenta en cada uno de los gráficos que se analizan posteriormente, al igual que el resto del reporte esta sección es modificable según las necesidades de cada centro médico, los filtros con los que se entrega el proyecto permiten seleccionar una tarea, fecha de creación, fecha de culminación o duración específicas, facilitando así el análisis en tiempos determinados por el usuario. Estos filtros pueden ser combinados para facilitar la indagación de los datos.

3. El primer gráfico presentado corresponde a instancias por estado, este gráfico muestra la información de las instancias de proceso generadas, teniendo tres estados primordiales aquellas que han sido completadas con éxito, las instancias aun activas y aquellas que por diferentes razones fueron abortadas, este último dato sirve especialmente al desarrollador debido a que las instancias abortadas implican un mal funcionamiento y este puede ser del proceso, de la herramienta BPMS o de comunicación con el sistema médico pues incluso las cancelaciones se consideran parte del flujo normal del proceso. Los otros dos estados proporcionan información sobre el funcionamiento de la empresa, el número de planillas que se encuentran en algún punto de este proceso y aquellas que ya han sido cobradas a las aseguradoras. Se pretende facilitar la visualización en la relación existente entre estos estados utilizando un gráfico tipo pastel en el que se entiende que el número de instancias culminadas debería ser superior al existen en otros estados, caso contrario es necesario determinar la existencia de un problema en el proceso.

4. Ubicado a la derecha de instancias por estado se encuentran usuarios con tarea, este gráfico tal y como su nombre sugiere proporciona información de los diferentes usuarios del sistema y el número de tareas que estos han realizado, sin la aplicación de filtros se permitirá visualizar aquellos usuarios que poseen una mayor carga de trabajo dentro del proceso, si una misma tarea puede ser realizada por más de un usuario un filtro podrá facilitar el análisis y determinar cuál de los usuarios realiza una mayor cantidad de veces dicha tarea, dato útil si se desea motivar a los empleados.

5. En el siguiente panel se muestra 4 gráficos separados mediante pestañas, en la primera se tiene un gráfico de barras que representara al número de instancias de cada tarea generada, este gráfico facilita la búsqueda de tareas conflictivas que estén

deteniendo el normal funcionamiento del proceso. La segunda pestaña presenta un gráfico con una línea de tiempo, permite observar la relación entre las tareas que se han realizado y los días en que estas fueron iniciadas, esta información no representa el funcionamiento del proceso pero si puede ser relevante para realizar un análisis general del negocio, pues muestra los días en los que se realizó una mayor cantidad de tareas, esto no implica necesariamente que las tareas se realizarán en el mismo día por lo que la siguiente pestaña dentro de este panel se muestra las fechas en las que se culminaron las tareas, al igual que en el gráfico anterior se muestra el número de tareas en el eje x y la fecha en el eje y por lo que aquellas fechas en las que se culminó una mayor cantidad de tareas se reflejaran a manera de picos en el tiempo. Al igual que con otros panel es posible aplicar filtros que proporcionarán información más detallada como períodos específicos de tiempo o tareas particulares iniciadas o culminadas en un fecha.

6. La cuarta y última pestaña de este panel es un cuadro que resume el tiempo de las diferentes tareas, en este cuadro se muestra un listado de tareas junto a la duración máxima, promedio y mínima que han tenido dichas tareas, permitiendo al usuario una fácil comparación entre las diferentes tareas así como la diferencia entre los tres valores en una sola actividad.

7. El panel de instancias de proceso por estado será semejante al cuadro de resumen descrito al inicio, se decidió agregar este cuadro debido a la proyección del proyecto en la que se considera como proyectos futuros aumentar los procesos controlados desde la herramienta BPMS y el cuadro permite filtrar datos así como ordenarlos según se considere necesario.

8. El último panel posee dos pestañas que se asemejan a aquellas que permiten graficar las tareas iniciadas y culminadas en un período de tiempo, este panel se diferencia de los anteriores en que los datos presentados no corresponden a las tareas sino a procesos, por lo que se puede graficar la fecha de inicio y culminación de todo el proceso.

Resumen	Instancias por estado		Usuarios con tareas	
	Número de tareas	Tareas iniciadas	Tareas terminadas	Duración tareas
Filtros	Instancias por proceso por estado			
	Procesos iniciados por fecha		Procesos culminados por fecha	

Figura 1 Distribución de gráficos

CAPÍTULO 4 INTEGRACIÓN Y PRUEBAS

4.1 Proceso de integración del proceso con Innovativa Salud

El entorno de trabajo de jBPM contiene un servidor de ejecución, que permite a los usuarios y desarrolladores invocar a procesos y tareas a través de un API remoto como lo denomina la documentación propia del proyecto, como consecuencia de esto el motor del proceso se puede utilizar como un servicio e integrarlo de esta manera a otros proyectos.

Para el proyecto utilizaremos esta característica mediante servicios REST enviando las solicitudes desde Innovativa Salud en el momento en que cada tarea se realice en este aplicativo. Es posible realizar esta integración mediante JMS (Java Message Service) o mediante CDI (Context and Dependency Injection), CDI integra por completo el proyecto en jBPM al sistema, con lo que se pierde muchas de las ventajas de los servicios web, considerando que Innovativa Salud planea tener además una aplicación móvil, y más procesos en un futuro cercano esta integración dificultaría estas mejoras del sistema. (The JBoss jBPM team, 2014)

Es necesaria la utilización de servicios web, uno de los requerimientos restringe el proyecto a la creación de un bus de servicios que proporciones los servicios web necesarios para integrar al sistema medico con jBPM y que pueda a su vez ser usado en servicios futuros de Innovativa Salud.

Los servicios publicados dentro del bus para la integración serán desarrollados en Java en un proyecto denominado InnovativaRest haciendo referencia a la tecnología a ser utilizada, como medida de seguridad y para una correcta auditoria en caso de ser necesaria cada uno de los servicios publicados requerirá un usuario y contraseña que se verificarán con jBPM, el primer servicio permitirá preguntar si una actividad es la siguiente dentro del proceso para esto es necesario enviar los datos del id de proceso y el nombre de la tarea por la que se realiza la consulta y será publicado en "webresources/ActividadActual" y será accesible mediante solicitud get. En el código se realizara primero la búsqueda del proceso requerido así como la tarea que debe completarse (Ver Anexo 1)

El siguiente servicio necesario para la integración es aquel que permite informar al proceso que una planilla ha sido aprobada consintiendo así pasar a la etapa de facturación del proceso o en su defecto proceder a la modificación para una nueva auditoría, para esto el servicio requiere identificar a que procesos se hace referencia y la respuesta de la aseguradora, este servicio será publicado en "webresources/Aprobacion". (Ver Anexo 2)

Siguiendo el orden alfabético de los servicios publicados se tiene el servicio que permitirá a Innovativa Salud generar las planillas dentro del BPMS en este caso es necesario se envíe si es necesario o no una atención médica, es decir si el cliente está en el dispensario por consulta o si requiere exámenes, inyecciones o cualquier otro tipo de servicio médico que no requiera turno con un doctor, ésta información se recibe mediante un dato booleano que permita la toma de decisión dentro del proceso, este servicio será publicado en "webresources/Generar"(Ver Anexo 3)

Otro servicio necesario para esta integración es aquel que permita iniciar un proceso, es decir crear una nueva instancia del proceso dentro de jBPM en este caso y debido al requisito de ser multiempresa es necesario enviar un dato que permita decidir cuál es la empresa que requiere iniciar un nuevo proceso así como la información del proceso que se desea iniciar, en este caso y debido a que el proyecto se limita a un único proceso este segundo dato puede ser innecesario, este servicio será publicado en "webresources/IniciarProceso" el cual retorna un processInstance Id que permite continuar con el proceso y será almacenado como texto (Ver Anexo 4)

Un servicio que será utilizado para realizar pruebas de conectividad será a manera de ping, es decir se solicita el servicio y éste enviará una respuesta en caso de que lleguemos y se encuentre disponible, este servicio será publicado en "webresources/Ping" (Ver Anexo 5)

El siguiente servicio es uno de los de mayor importancia, permitirá informar a jBPM que una tarea ha sido realizada correctamente dentro de Innovativa Salud si dicha tarea no requiere informar de ningún dato adicional al BPMS para esos casos se tendrá servicios específicos que reciben el dato según sea el caso, por lo que el único

dato se requiere para este servicio es el instance ID proporcionado por el sistema médico este servicio se publicara en “webresources/RealizarTarea” (Ver Anexo 6)

Como se menciona en el servicio anterior, existen tareas que requieren información adicional para la correcta ejecución de la tarea dentro del administrador de procesos como es el caso del siguiente servicio en el que es necesario informar al BPMS si se desea continuar con el proceso, información que es recibida por el servicio como un dato booleano, este servicio se publicará en "webresources/Revision" (Ver Anexo 7)

A partir de este punto la integración se realizará directamente en Innovativa Salud para esto se creará un paquete llamado `com.jvc.medisys.bean.clienterest` donde se creará `ProcesoBean.java` clase desde la cual se realizarán las solicitudes al bus de servicios por lo que será necesario crear funciones que realicen estas solicitudes e informen al sistema la respuesta de estos servicios y si es el caso si el servicio no se encuentra disponible para así poder continuar con la tarea al interior de Innovativa Salud sin que sea un requisito la disponibilidad de jBPM, para esto se requiere una función capaz de analizar si el servicio responde con un número, dato que nos informará la correcta operación o la existencia de un mensaje de error. (Ver Anexo 8)

La integración de Innovativa Salud como interfaz de usuario se inicia con la emisión de la planilla desde la pantalla de facturación donde un cliente que posea convenio con una aseguradora no recibe una factura sea para una consulta médica o cualquier otro tipo de servicio.

CONSULTORIO GENERAL ESPE-LATACUNGA
ING. ALTAMIRANO VASCONEZ DIEGO DARIO
Fecha: 18/05/2015
Salir

Mantenimiento > Procesos > Reportes > Seguridades > Facturación >

Factura
Factura: 102

Datos del Paciente
Buscar Paciente
Nombres: Luis Fernando Guanoquiza Pallo
Dirección: c/ Marcelo Ayala s/n
Cédula: 0501304679
Historia Clínica: 1115
Teléfono: (02)2083-117

Datos del Cliente
Emitir a nombre de: Paciente [v] Buscar Cliente
Nombres: Luis Fernando Guanoquiza Pallo
Dirección: c/ Marcelo Ayala s/n
Cédula: 0501304679
Teléfono: (02)2083-117

Datos del Convenio
Buscar Convenio Eliminar
Nombre: ISSFFA
Fecha Inicio: 22/04/2014
Fecha Fin: 22/04/2020
Tipo de Cobertura: COBERTURA 100%
Tipo de Beneficiario: ASPIRANTE A SOLDADO
Porcentaje: 100.00

Buscar Facturas
Nuevo
Guardar

Figura 2 Pantalla de Facturación

La planilla se crea en la misma función que genera facturas dependiendo la existencia o no de un convenio decidiendo así que tipo de documento es necesario generar, una vez realizadas las validaciones dentro del sistema médico se genera una planilla de seguros y en este instante se inicia el proceso dentro del administrador de procesos de negocio.

Para el inicio del proceso se toma los datos del usuario que inicio sesión, este constará como la persona que inicia el proceso, como se observó la función en la clase `ProcesoBean` iniciar proceso permite solicitar este servicio y como respuesta se obtendrá un Id de proceso que permite a partir de este punto identificar de manera única a este proceso, esta información se almacenará dentro de los datos de la planilla en la Base de Datos, de igual manera se comprueba la respuesta desde `jBPM` para poder continuar con la emisión de planilla si no es posible iniciar el proceso.


```

public String enviarAseguradora() throws Exception
{
    SeussUrsistem usr = (MenuBarBean) FacesUtils.getManagedBean("menuBar").getUser();
    TripleDES des = new TripleDES();
    this.currentPlanilla.setEstadoPlanilla("P");
    this.currentPlanilla= planillasServicio.editar(currentPlanilla);
    ProcesoBean pb= new ProcesoBean();
    if(pb.ActividadActual("Revision", ""+currentPlanilla.getId_proceso(), usr.getUssIdentifica(), des.decrypt(usr.getUssPassword()))
    { if(!pb.RevisarPlanilla(""+currentPlanilla.getId_proceso(),"true",usr.getUssIdentifica(), des.decrypt(usr.getUssPassword())))
    {
        currentPlanilla.setId_proceso(0);
        planillasServicio.editar(currentPlanilla);
    }
    }else
    {
        if(!pb.Atencion( ""+currentPlanilla.getId_proceso(), usr.getUssIdentifica(), des.decrypt(usr.getUssPassword()))
        {
            currentPlanilla.setId_proceso(0);
            planillasServicio.editar(currentPlanilla);
        }
    }
    FacesUtils.addInfoMessage("La Planilla lista para envio.");
    actualizarBotones();
    return null;
}

```

Figura 6 Enviar Planilla

Una vez realizada esta tarea el sistema impide la modificación, facturación o cancelación de las planillas permitiendo únicamente el ingreso de la respuesta de la aseguradora después de la auditoría que considere adecuada, en este punto existe nuevamente una bifurcación del flujo de trabajo, en el supuesto en que una planilla sea rechazada la información será ingresada al BPMS mediante un botón que habilita nuevamente la edición dentro del sistema médico, y cambia el estado de la planilla dentro de la base de datos.

```

public String RespuestaAseguradoraF() throws Exception

ProcesoBean pb= new ProcesoBean();
if(!pb.Aseguradora(""+currentPlanilla.getId_proceso(),"false",usr.getUssIdentifica(), des.decrypt(usr.getUssPassword()))
{
    currentPlanilla.setId_proceso(0);
    planillasServicio.editar(currentPlanilla);
}

this.currentPlanilla.setEstadoPlanilla("K");
this.currentPlanilla= planillasServicio.editar(currentPlanilla);
actualizarBotones();
return null;

```

Figura 7 Planilla Requiere Cambios

Los cambios requeridos se realizan en la planilla y es posible enviar nuevamente a la aseguradora la planilla corregida, este proceso es posible realizarlo las veces que sean necesarias, hasta que la planilla sea finalmente aprobada, al igual que al realizar la tarea de revisión la llamada al servicio se realiza después de presionar

el botón de envío a la aseguradora que impide modificaciones hasta obtener respuesta y permite imprimir la planilla, el código para un nuevo envío se encuentra en la Figura 6.

Cuando una planilla ha sido aprobada se procede a la facturación, esta tarea se realiza desde la página de facturación, teniendo como cliente a la aseguradora, por lo general la facturación a una aseguradora se realiza al final de cada mes por lo que cada una de estas facturas incluyen varias planillas aprobadas, es necesario entonces tener en cuenta que el método de facturación es la misma que originalmente genera planillas y que la facturación como se muestra en la Figura 8, la solicitud del servicio necesario es posible realizarla con el método Atención pues no es necesario enviar ninguna información adicional.

```

}
if (detalle.getDetalle().startsWith("Planilla")) {
    ProcesoBean pb = new ProcesoBean();
    SeussUrsistem usr = ((MenuBarBean) FacesUtils.getManagedBean("menuBar")).getUser();
    TripleDES des = new TripleDES();
    if (!pb.Atencion("" + detalle.getPlanilla().getId_proceso(), usr.getUssIdentifica(), des.decrypt(usr.getUssPassword())))
    {
        detalle.getPlanilla().setId_proceso(0);
        planillasServicio.editar(detalle.getPlanilla());
    }
}

```

Figura 8 Facturación de planilla

El ingreso a las cuentas por cobrar de la factura recién creada se realiza internamente dentro de Innovativa Salud y así lo entiende jBPM, por lo que la tarea tipo script se realiza inmediatamente después de la facturación. Como última tarea realizada por un usuario se tiene la confirmación del pago, esto se realiza desde la página de pagos a créditos del sistema y se procede como con cualquier factura, teniendo en cuenta nuevamente que se trata de una factura que puede contener varias planillas por lo que es necesario una revisión a los detalles de la factura como se muestra en la Figura 9.

```

// |
for( DetalleFactura df : cabecera.getDetalleFacturaList())
{
    int num= this.obtenerNumPlanilla(df.getDetalle());
    int id=planillasServicio.buscarCodigo(num).getId_proceso();

    ProcesoBean pb = new ProcesoBean();
    SeussUrsistem usr = ((MenuBarBean) FacesUtils.getManagedBean("menuBar")).getUser();
    TripleDES des = new TripleDES();
    if(!pb.Atencion(""+id,usr.getUssIdentifica(), des.decrypt(usr.getUssPassword())))
    {
        cabecera.setId_proceso(0);
        facturasServicio.editar(cabecera);
    }

    }
    facturasServicio.editar(cabecera);
}
}

```

Figura 9 Pago de Factura

Al igual que con el anterior script en el proceso, jBPM entiende que la actualización de las cuentas por cobrar se realiza inmediatamente después de que el usuario ingrese el pago y con la ejecución de esta tarea al interior del sistema médico se entiende como concluido el proceso.

4.2 Problemas presentados en la integración.

Es necesario realizar varios cambios en la configuración de jBPM y del servidor de aplicaciones Wildfly para poder funcionar en armonía con el sistema médico, estos cambios se realizarán en archivos de configuración tanto de la aplicación como del servidor.

El primer inconveniente presentado al momento de integrar Innovativa Salud con el proceso en jBPM es debido a la utilización de dos servidores de aplicaciones que por defecto utilizan el mismo puerto, para evitar problemas y facilitar la integración se decidió cambiar el puerto a Wildfly pues el cambio en Glassfish podría generar problemas en un sistema en producción para esto se edita el archivo standalone-full.xml de la manera siguiente:

```

        <socket-binding-group name="standard-sockets" default-
interface="public" port-offset="{jboss.socket.binding.port-
offset:0}">
            <socket-binding name="management-http"
interface="management"
port="{jboss.management.http.port:9990}"/>
            <socket-binding name="management-https"
interface="management"
port="{jboss.management.https.port:9993}"/>
            <socket-binding name="ajp"
port="{jboss.ajp.port:8009}"/>
            <socket-binding name="http"
port="{jboss.http.port:8088}"/>
            <socket-binding name="https"
port="{jboss.https.port:8443}"/>

```

Figura 10 standalone-full.xml

De igual manera se cambió el puerto en build.xml esto para evitar el mensaje de error al momento de iniciar el servidor, sin embargo es posible levantar el servidor sin este cambio.

Para poder utilizar la base de datos Postgres utilizada por Innovativa en lugar de H2, base utilizada por defecto en jBPM es necesario entonces el registro de los drivers y data source esto se puede realizar desde la interfaz web o desde cli⁶ con los siguientes comandos:

```

module add --name=org.postgres --resources=tmp/postgresql-9.3-
1101.jdbc41.jar --dependencies=javax.api,javax.transaction.api

```

```

/subsystem=datasources/jdbc-driver=postgres:add(driver-
name="postgres",driver-module-name="org.postgres",driver-class-
name=org.postgresql.Driver)

```

Registra el driver, creando previamente una carpeta tmp dentro la cual se tiene el driver descargado.

```

data-source add --jndi-name=java:jboss/datasources/jBPMPostgresDS --
name=PostgresDS1 --connection-url=jdbc:postgresql://localhost:5432/jBPM --
driver-name=postgres --user-name=jBPM --password=jBPM

```

Para agregar el nuevo data source, que permite a jBPM trabajar desde postgres.

Gracias a estos cambios y ejecutando postgresql-jBPM-schema.sql en la base en para este caso se nombra jBPM es posible que el administrador de procesos corra

⁶ Comand Line Interface

en una base postgres, pero aún no es posible iniciar sesión con usuarios de Innovativa Salud, para esto es necesario editar el archivo.

\wildfly-8.1.0.Final\standalone\configuration\standalone-full.xml dentro del tag <subsystem xmlns="urn:jboss:domain:security:1.2"> agregar lo siguiente:

```

</subsystem>
<subsystem xmlns="urn:jboss:domain:resource-
adapters:2.0"/>
<subsystem xmlns="urn:jboss:domain:sar:1.0"/>
<subsystem xmlns="urn:jboss:domain:security:1.2">
  <security-domains>
    <security-domain name="DBAuthTest">
      <authentication>
        <login-module
code="com.jvc.medisys.security.CustomLoginModule"
flag="required"/>
      </authentication>
    </security-domain>
    <security-domain name="other" cache-
type="default">

```

Figura 11 Security Domain

Debido a que existen diferencias entre los roles disponibles en jBPM y aquellos existentes en el sistema médico es necesario además crear nuevos roles que posean los mismos permisos de usuario para diferenciar entre las diferentes empresas, estos cambios además deben ser reflejados en WEB-INF\web.xml

4.3 Integración de reportes gerenciales

La información requerida se presentará mediante JBoss Dashboard Builder, un proyecto separado de jBPM con soporte de JBoss que se integra de manera predeterminada a administrador de procesos, sin embargo es necesario realizar varios cambios en la configuración de este proyecto para que la información presentada sea la almacenada en la nueva base de datos, así como permitir usar los usuarios de Innovativa Salud para ingresar al proyecto.

Al interior de los despliegues del JBoss se modifica dashboard-builder.war. En WEB-INF/jBoss-web.xml se cambia las etiquetas <jndi-name> por java:jboss/datasources/jBPMPostgresDS de manera semejante a lo realizado para utilizar la base de datos Postgres en jBPM de igual manera se cambia <security-domain> a java:/jaas/DBAuthTest para utilizar el mismo security domain utilizado por

jBPM, en WEB-INF/lib se agrega la clase JBPM_LoginModule-ejb.jar desarrollada originalmente para el administrador de procesos. Con estos cambios en la configuración Dashboard Builder se puede considerar ya integrado al sistema, pero la data Source por defecto utiliza la base de datos H2.

Dentro de la administración del Dashboard Builder en la pestaña conexiones externas es posible añadir un nuevo Data Source, al que se nombra como PostgresDS1 y como ruta JNDI `java:jboss/datasources/jBPMPostgresDS` siendo este el mismo datasource utilizado por todo el sistema de esta manera, se tendrá ya disponible el datasource utilizado en el capítulo 3 sección 3.7 Diseño de reportes gerenciales.

Teniendo en cuenta que el sistema Innovativa Salud es multiempresa y al tener en cuenta que la solución propuesta para que el BPMS funcione de manera semejante, es necesario agregar los nuevos roles y permisos al proyecto, esto se realiza al interior de `web-inf/web.xml` agregando nuevos `security.roles` y agregando los nombres en el tag `<auth-constraint>` que permita a los nuevos roles de usuario ingresar al sistema.

4.4 Pruebas realizada

4.4.1 Requisito RF1:

Código	Requerimiento	Descripción
RF1	Proceso generado en BPMS	Proceso generado en BPMS funcional.

4.4.1.1 Caso de prueba 1. Ejecución proceso caso 1.

Caso de Prueba 001 Proceso generado en BPMS	
Código de Identificación:	CP001
Descripción	Ejecución con atención médica aprobado por administrador y aprobado por aseguradora.
Variables de Entrada (Inputs):	Atención médica, Continuar, Aprobado.
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Crear nueva instancia de proceso. 2. Emitir planilla 3. Atención médica. 4. Revisión. 5. Aseguradora. 6. Facturar. 7. Ingresar a cuentas por cobrar. 8. Confirmar pago. 9. Actualizar cuentas por cobrar.
Resultado esperado:	Proceso culminado y almacenado en base de datos.
Flujo alterno	Omitir Atención médica, cancelar el proceso o no ser aprobado por aseguradora.
Resultado alternativo esperado:	No se puede continuar con el proceso.
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

Registro de instancia del proceso	26/jun/15 15:25:10: 14 - EndNode
	26/jun/15 15:25:10: 13 - Actualizar CC (ActionNode)
	26/jun/15 15:25:05: 12 - Confirmar Pago (HumanTaskNode)
	26/jun/15 15:25:05: 11 - Ingresar a CC (ActionNode)
	26/jun/15 15:24:59: 10 - Facturar (HumanTaskNode)
	26/jun/15 15:24:59: 9 - Split
	26/jun/15 15:24:53: 8 - Aseguradora (HumanTaskNode)
	26/jun/15 15:24:53: 7 - Join
	26/jun/15 15:24:53: 6 - Split
	26/jun/15 15:24:46: 5 - Revision (HumanTaskNode)
	26/jun/15 15:24:46: 4 - Join
	26/jun/15 15:21:25: 3 - Atencion (HumanTaskNode)
	26/jun/15 15:21:25: 2 - Split
	26/jun/15 15:17:38: 1 - Generar Planilla (HumanTaskNode)
	26/jun/15 15:17:38: 0 - StartNode

Figura 12 Resultado CP001

4.4.1.2 Caso de prueba 2. Ejecución proceso caso 2.

Caso de Prueba 002	
Proceso generado en BPMS	
Código de Identificación:	CP002
Descripción	Ejecución sin atención médica, aprobado por administrador y aprobado por aseguradora.
Variables de Entrada (Inputs):	Atención médica, Continuar, Aprobado.
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Crear nueva instancia de proceso. 2. Emitir planilla 3. Revisión. 4. Aseguradora. 5. Facturar. 6. Ingresar a cuentas por cobrar. 7. Confirmar pago. 8. Actualizar cuentas por cobrar.
Resultado esperado:	Proceso culminado y almacenado en base de datos.
Flujo alterno	Atención médica, cancelar el proceso o no ser aprobado por aseguradora.

Resultado alternativo esperado:	No se puede continuar con el proceso.
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

Registro de instancia del proceso	26/jun/15 15:48:50: 17 - EndNode
	26/jun/15 15:48:50: 16 - Actualizar CC (ActionNode)
	26/jun/15 15:48:44: 15 - Confirmar Pago (HumanTaskNode)
	26/jun/15 15:48:44: 14 - Ingresar a CC (ActionNode)
	26/jun/15 15:48:35: 13 - Facturar (HumanTaskNode)
	26/jun/15 15:48:35: 12 - Split
	26/jun/15 15:48:27: 11 - Aseguradora (HumanTaskNode)
	26/jun/15 15:48:27: 10 - Join
	26/jun/15 15:48:22: 9 - Correccion (HumanTaskNode)
	26/jun/15 15:48:22: 8 - Split
	26/jun/15 15:48:02: 7 - Aseguradora (HumanTaskNode)
	26/jun/15 15:48:02: 6 - Join
	26/jun/15 15:48:02: 5 - Split
	26/jun/15 15:47:53: 4 - Revision (HumanTaskNode)
	26/jun/15 15:47:53: 3 - Join
	26/jun/15 15:47:53: 2 - Split
	26/jun/15 15:47:41: 1 - Generar Planilla (HumanTaskNode)
	26/jun/15 15:47:41: 0 - StartNode

Figura 13 Resultado CP002

4.4.1.3 Caso de prueba 3. Ejecución proceso caso 3.

Caso de Prueba 003	
Proceso generado en BPMS	
Código de Identificación:	CP003
Descripción:	Ejecución sin atención médica, aprobado por administrador y no fue aprobado por aseguradora.
Variables de Entrada (Inputs):	Atención médica, Continuar, Aprobado.
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Crear nueva instancia de proceso. 2. Emitir planilla 3. Revisión. 4. Aseguradora. 5. Correccion. 6. Aseguradora. 7. Facturar. 8. Ingresar a cuentas por cobrar 9. Confirmar pago. 10. Actualizar cuentas por cobrar.
Resultado esperado:	Proceso culminado y almacenado en base de datos.
Flujo alternativo	Atención médica, cancelar el proceso o no ser aprobado por aseguradora.
Resultado alternativo esperado:	No se puede continuar con el proceso.
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

Registro de instancia del proceso

- 26/jun/15 15:57:29: 17 - EndNode
- 26/jun/15 15:57:29: 16 - Actualizar CC (ActionNode)
- 26/jun/15 15:57:20: 15 - Confirmar Pago (HumanTaskNode)
- 26/jun/15 15:57:20: 14 - Ingresar a CC (ActionNode)
- 26/jun/15 15:57:14: 13 - Facturar (HumanTaskNode)
- 26/jun/15 15:57:14: 12 - Split
- 26/jun/15 15:57:08: 11 - Aseguradora (HumanTaskNode)
- 26/jun/15 15:57:08: 10 - Join
- 26/jun/15 15:57:02: 9 - Correccion (HumanTaskNode)
- 26/jun/15 15:57:02: 8 - Split
- 26/jun/15 15:56:56: 7 - Aseguradora (HumanTaskNode)
- 26/jun/15 15:56:56: 6 - Join
- 26/jun/15 15:56:56: 5 - Split
- 26/jun/15 15:56:51: 4 - Revision (HumanTaskNode)
- 26/jun/15 15:56:51: 3 - Join
- 26/jun/15 15:56:51: 2 - Split
- 26/jun/15 15:56:25: 1 - Generar Planilla (HumanTaskNode)
- 26/jun/15 15:56:25: 0 - StartNode

Figura 14 Resultado CP003

4.4.1.4 Caso de prueba 4. Ejecución proceso caso 4.

Caso de Prueba 004	
Proceso generado en BPMS	
Código de Identificación:	CP004
Descripción	Ejecución con atención médica cancelado por administrador
Variables de Entrada (Inputs):	Atención médica, Continuar, Aprobado.
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Crear nueva instancia de proceso. 2. Emitir planilla 3. Revisión.
Resultado esperado:	Proceso culminado y almacenado en base de datos.
Flujo alterno	Omitir atención médica, continuar con proceso.
Resultado alternativo esperado:	Continúa el proceso
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

Registro de instancia	26/jun/15 16:05:13: 6 - EndNode
del proceso	26/jun/15 16:05:13: 5 - Split
	26/jun/15 16:05:08: 4 - Revision (HumanTaskNode)
	26/jun/15 16:05:08: 3 - Join
	26/jun/15 16:05:08: 2 - Split
	26/jun/15 15:51:20: 1 - Generar Planilla (HumanTaskNode)
	26/jun/15 15:51:20: 0 - StartNode

Figura 15 Resultado CP004

4.4.2 Requisito RF2:

Código	Requerimiento	Descripción
RF2	Integrar proceso al Front-End de Innovativa Salud	Proceso generado deberá ejecutarse utilizando Innovativa Salud como interfaz de usuario.

4.4.2.1 Caso de prueba 5. Ejecución proceso caso 1.

Caso de Prueba 005	
Integrar proceso al Front-End de Innovativa Salud	
Código de Identificación:	CP005
Descripción	Ejecución con atención médica aprobado por administrador y aprobado por aseguradora.
Variables de Entrada (Inputs):	Atención médica, Continuar, Aprobado.
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Crear nueva instancia de proceso. 2. Emitir planilla 3. Atención médica. 4. Revisión. 5. Aseguradora. 6. Facturar. 7. Ingresar a cuentas por cobrar. 8. Confirmar pago. 9. Actualizar cuentas por cobrar.
Resultado esperado:	Proceso culminado y almacenado en base de datos.
Flujo alterno	Omitir Atención médica, cancelar el proceso o no ser aprobado por aseguradora.
Resultado alternativo esperado:	No se puede continuar con el proceso.
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

Registro de instancia del proceso	
	29/jun/15 11:36:35: 14 - EndNode
	29/jun/15 11:36:35: 13 - Actualizar CC (ActionNode)
	29/jun/15 11:35:06: 12 - Confirmar Pago (HumanTaskNode)
	29/jun/15 11:35:06: 11 - Ingresar a CC (ActionNode)
	29/jun/15 11:34:00: 10 - Facturar (HumanTaskNode)
	29/jun/15 11:34:00: 9 - Split
	29/jun/15 11:33:55: 8 - Aseguradora (HumanTaskNode)
	29/jun/15 11:33:55: 7 - Join
	29/jun/15 11:33:55: 6 - Split
	29/jun/15 11:32:43: 5 - Revision (HumanTaskNode)
	29/jun/15 11:32:43: 4 - Join
	29/jun/15 11:31:11: 3 - Atencion (HumanTaskNode)
	29/jun/15 11:31:11: 2 - Split
	29/jun/15 11:31:10: 1 - Generar Planilla (HumanTaskNode)
	29/jun/15 11:31:10: 0 - StartNode

Figura 16 Resultado CP005

4.4.2.2 Caso de prueba 6. Ejecución proceso caso 2.

Caso de Prueba 006	
Integrar proceso al Front-End de Innovativa Salud	
Código de Identificación:	CP006
Descripción	Ejecución sin atención médica, aprobado por administrador y aprobado por aseguradora.
Variables de Entrada (Inputs):	Atención médica, Continuar, Aprobado.
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Crear nueva instancia de proceso. 2. Emitir planilla 3. Revisión. 4. Aseguradora. 5. Facturar. 6. Ingresar a cuentas por cobrar. 7. Confirmar pago. 8. Actualizar cuentas por cobrar.
Resultado esperado:	Proceso culminado y almacenado en base de datos.
Flujo alterno	Atención médica, cancelar el proceso o no ser aprobado por aseguradora.
Resultado alternativo esperado:	No se puede continuar con el proceso.
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

Registro de instancia del proceso	29/jun/15 12:10:01: 13 - EndNode
	29/jun/15 12:10:01: 12 - Actualizar CC (ActionNode)
	29/jun/15 12:08:55: 11 - Confirmar Pago (HumanTaskNode)
	29/jun/15 12:08:55: 10 - Ingresar a CC (ActionNode)
	29/jun/15 12:08:27: 9 - Facturar (HumanTaskNode)
	29/jun/15 12:08:27: 8 - Split
	29/jun/15 12:08:20: 7 - Aseguradora (HumanTaskNode)
	29/jun/15 12:08:20: 6 - Join
	29/jun/15 12:08:20: 5 - Split
	29/jun/15 12:07:39: 4 - Revision (HumanTaskNode)
	29/jun/15 12:07:39: 3 - Join
	29/jun/15 12:07:39: 2 - Split
	29/jun/15 12:07:39: 1 - Generar Planilla (HumanTaskNode)
	29/jun/15 12:07:39: 0 - StartNode

Figura 17 Resultado CP006

4.4.2.3 *Caso de prueba 7. Ejecución proceso caso 3.*

Caso de Prueba 007	
Integrar proceso al Front-End de Innovativa Salud	
Código de Identificación:	CP007
Descripción	Ejecución sin atención médica, aprobado por administrador y no fue aprobado por aseguradora.
Variables de Entrada (Inputs):	Atención médica, Continuar, Aprobado.
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Crear nueva instancia de proceso. 2. Emitir planilla 3. Revisión. 4. Aseguradora. 5. Correccion. 6. Aseguradora. 7. Facturar. 8. Ingresar a cuentas por cobrar. 9. Confirmar pago. 10. Actualizar cuentas por cobrar.
Resultado esperado:	Proceso culminado y almacenado en base de datos.
Flujo alterno	Atención médica, cancelar el proceso o no ser aprobado por aseguradora.
Resultado alternativo esperado:	No se puede continuar con el proceso.
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

Registro de instancia del proceso	29/jun/15 12:01:43: 17 - EndNode 29/jun/15 12:01:43: 16 - Actualizar CC (ActionNode) 29/jun/15 12:01:09: 15 - Confirmar Pago (HumanTaskNode) 29/jun/15 12:01:09: 14 - Ingresar a CC (ActionNode) 29/jun/15 12:00:15: 13 - Facturar (HumanTaskNode) 29/jun/15 12:00:15: 12 - Split 29/jun/15 12:00:12: 11 - Aseguradora (HumanTaskNode) 29/jun/15 12:00:12: 10 - Join 29/jun/15 12:00:06: 9 - Correccion (HumanTaskNode) 29/jun/15 12:00:06: 8 - Split 29/jun/15 11:59:01: 7 - Aseguradora (HumanTaskNode) 29/jun/15 11:59:01: 6 - Join 29/jun/15 11:59:01: 5 - Split 29/jun/15 11:42:52: 4 - Revision (HumanTaskNode) 29/jun/15 11:42:52: 3 - Join 29/jun/15 11:42:52: 2 - Split 29/jun/15 11:42:50: 1 - Generar Planilla (HumanTaskNode) 29/jun/15 11:42:50: 0 - StartNode
--	--

Figura 18 Resultado CP007

4.4.2.4 *Caso de prueba 8. Ejecución proceso caso 4.*

Caso de Prueba 008	
Integrar proceso al Front-End de Innovativa Salud	
Código de Identificación:	CP008
Descripción	Ejecución con atención médica cancelado por administrador
Variables de Entrada (Inputs):	Atención médica, Continuar, Aprobado.
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Crear nueva instancia de proceso. 2. Emitir planilla 3. Revisión.
Resultado esperado:	Proceso culminado y almacenado en base de datos.
Flujo alternativo	Omitir atención médica, continuar con proceso.
Resultado alternativo esperado:	Continúa el proceso
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

Registro de instancia del proceso	29/jun/15 12:12:21: 7 - EndNode
	29/jun/15 12:12:21: 6 - Split
	29/jun/15 12:12:21: 5 - Revision (HumanTaskNode)
	29/jun/15 12:12:21: 4 - Join
	29/jun/15 12:11:59: 3 - Atencion (HumanTaskNode)
	29/jun/15 12:11:59: 2 - Split
	29/jun/15 12:11:59: 1 - Generar Planilla (HumanTaskNode)
	29/jun/15 12:11:59: 0 - StartNode

Figura 19 Resultado CP008

4.4.3 Requisito RF3:

Código	Requerimiento	Descripción
RF3	Utilizar usuarios de Innovativa Salud en BPMS	El acceso al BPMS debe realizarse utilizando la base de datos con usuarios de Innovativa Salud

4.4.3.1 Caso de prueba 9. Ingreso con usuario en Innovativa Salud

Caso de Prueba 009 Utilizar usuarios de Innovativa Salud en BPMS	
Código de Identificación:	CP009
Descripción	Ingresar a jBPM utilizando un usuario con permisos para generar planillas
Variables de Entrada (Inputs):	Usuario, Contraseña
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Ingreso de Usuario 2. Ingreso de Contraseña. 3. Ingreso al sistema.
Resultado esperado:	Ingreso a jBPM
Flujo alternativo	Usuario o contraseña incorrectos
Resultado alternativo esperado:	Mensaje que indica el error
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

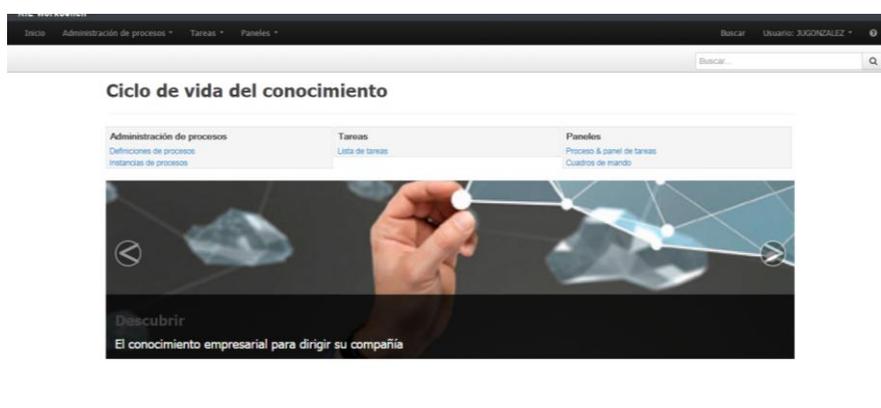


Figura 20 Resultado CP009

Resultado Alternativo:



The image shows a login interface for KIE IDE. At the top, the logo consists of the letters "KIE" in a bold, black, serif font, followed by a yellow circle containing the letters "IDE" in a black, sans-serif font. Below the logo, a black horizontal bar contains the text "Inicio de sesión fallido: no autorizado" in white. Underneath this bar, there are two white input fields: the first is labeled "Nombre del usuario" and the second is labeled "Contraseña". At the bottom right of the form area, there is a green button with the text "Inicio de sesión" in white.

Figura 21 Resultado Alternativo CP009

4.4.3.2 Caso de prueba 10. Ingreso con médico en Innovativa Salud

Caso de Prueba 010		
Ingreso con médico en Innovativa Salud		
Identificación:	Código de	CP010
Descripción	Ingresar a jBPM utilizando un médico con permisos para generar planillas	
(Inputs):	Variables de Entrada	Usuario, Contraseña
Flujo normal del proceso		<ol style="list-style-type: none"> 1. Ingreso de Usuario 2. Ingreso de Contraseña. 3. Ingreso al sistema.
Resultado esperado:	Ingreso a jBPM	
Flujo alternativo	Usuario o contraseña incorrectos	
Resultado alternativo esperado:	Mensaje que indica el error	
Errores encontrados	N/A	
Sugerencias de corrección	N/A	

Captura resultado esperado:

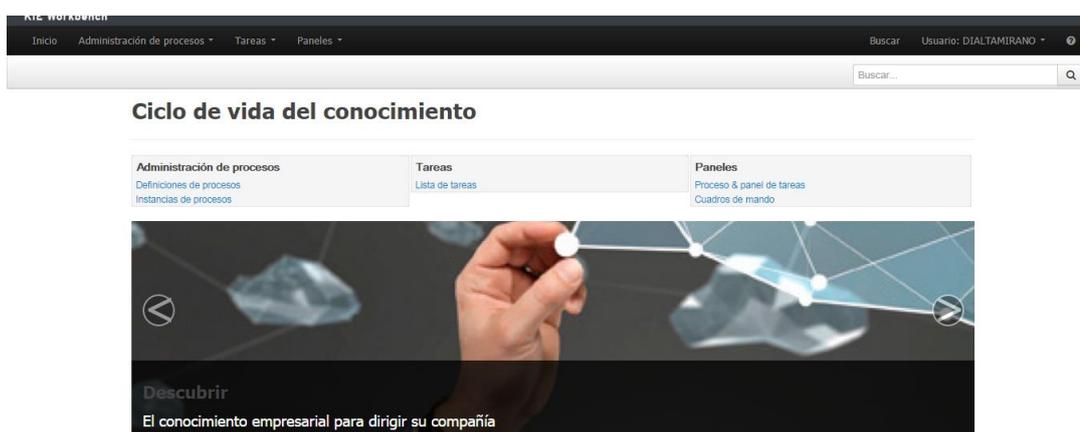


Figura 22 Resultado CP010

Resultado alternativo:



The image shows a web interface for KIE IDE. At the top, the logo consists of the letters 'KIE' in a bold, black, serif font, followed by a yellow circle containing the letters 'IDE' in a black, sans-serif font. Below the logo, the text 'Inicio de sesión fallido: no autorizado' is displayed in white on a dark background. Underneath this message are two white input fields: the first is labeled 'Nombre del usuario' and the second is labeled 'Contraseña'. At the bottom right of the form area is a green button with the text 'Inicio de sesión' in white.

Figura 23 Resultado Alternativo CP010

4.4.4 Requisito RF5:

Código	Requerimiento	Descripción
RF5	Generación de reportes del proceso	El sistema debe presentar en pantalla el estado del proceso con datos actualizados

4.4.4.1 Caso de prueba 11. Visualización de reportes.

Caso de Prueba 011	
Generación de reportes del proceso	
Código de Identificación:	CP011
Descripción	Ingreso a reportes y correcta visualización de datos.
Variables de Entrada (Inputs):	Usuario
Flujo normal del proceso	1. Ingreso a DashBoard
Resultado esperado:	Visualización de reporte
Flujo alterno	N/A
Resultado alternativo esperado:	N/A
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

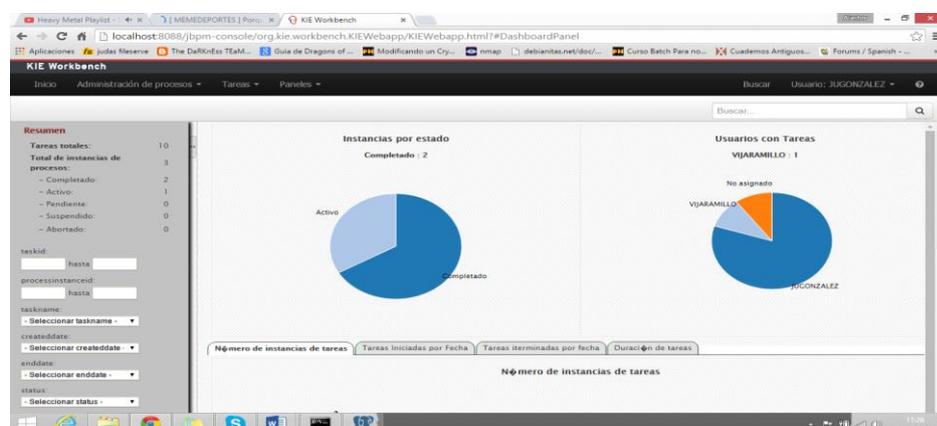


Figura 24 Resultado CP011

4.4.4.2 Caso de prueba 12. Visualización de reportes actualizados.

Caso de Prueba 012		
Generación de reportes del proceso		
Identificación:	Código de	CP012
Descripción	Ingreso a reportes y correcta visualización de datos actualizados.	
(Inputs):	Variables de Entrada	Usuario
Flujo normal del proceso		<ol style="list-style-type: none"> 1. Ejecución de tarea. 2. Ingreso a DashBoard 3. Visualización de datos.
Resultado esperado:	Visualización de reporte actualizado	
Flujo alternativo	N/A	
Resultado alternativo esperado:	N/A	
Errores encontrados	N/A	
Sugerencias de corrección	N/A	

Captura resultado esperado:

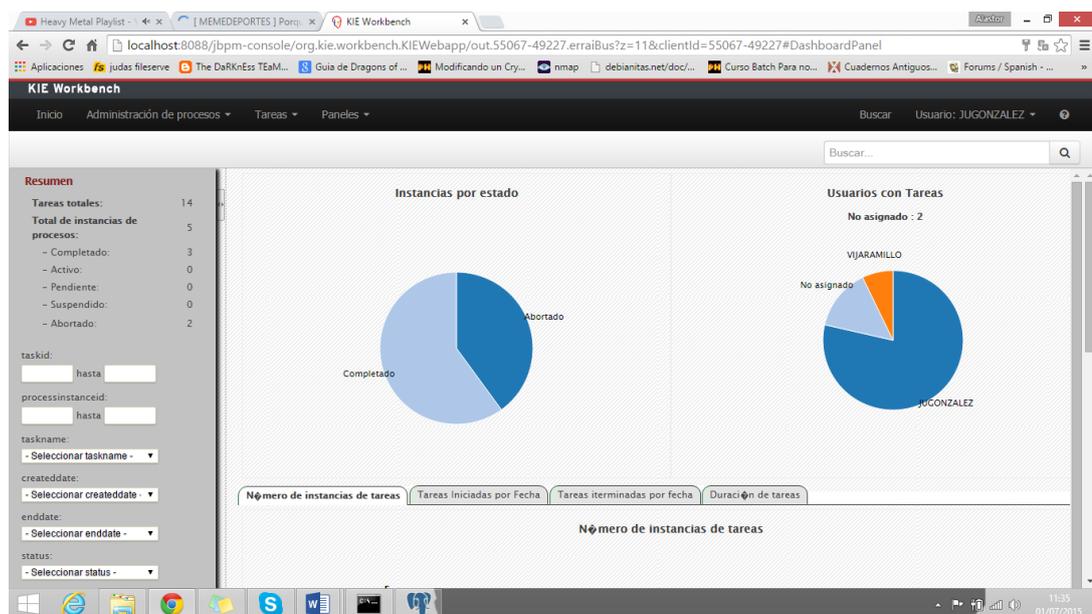


Figura 25 Resultado CP012

4.4.5 Requisito RF7:

Código	Requerimiento	Descripción
RF7	La facturación final se realiza en por grupos de planillas pero el proceso es individual para cada una	En actividades que se las realiza a grupos de planillas cada una debe continuar con su proceso de manera individual

4.4.5.1 *Caso de prueba 14. Seguimiento individual de planillas, en facturación.*

Caso de Prueba 014	
La facturación final se realiza en por grupos de planillas pero el proceso es individual para cada una	
Código de Identificación:	CP014
Descripción	Facturación de 3 planillas de seguros.
Variables de Entrada (Inputs):	planilla
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Seleccionar cliente (Aseguradora). 2. Seleccionar planillas. 3. Emitir factura.
Resultado esperado:	Cada planilla actualiza su estado en jBPM
Flujo alterno	Facturación de planilla única.
Resultado alternativo esperado:	Planilla actualiza su estado en jBPM
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

Tiene Convenio de Seguro	Detalle	Cantidad Existente	Cantidad	PVP	Caducidad	Descuento	Cobertura Seguro	Subtotal	IVA	Editar	Eliminar
<input type="checkbox"/>	Planilla: 326 Fecha: 30/06/15 Empresa: ISSFFA Paciente Cl:		1	4.00		0	0	4.00	<input type="checkbox"/>		Eliminar
<input type="checkbox"/>	Planilla: 327 Fecha: 30/06/15 Empresa: ISSFFA Paciente Cl:		1	4.00		0	0	4.00	<input type="checkbox"/>		Eliminar
<input type="checkbox"/>	Planilla: 328 Fecha: 30/06/15 Empresa: ISSFFA Paciente Cl:		1	19.75		0	0	19.75	<input type="checkbox"/>		Eliminar

Figura 26 Resultado 1 CP014

Id	Planillas	admin	1.0	Activo	29/06/2015 11:21		
51	Planillas	admin	1.0	Activo	29/06/2015 11:21		
52	Planillas	JUGONZALEZ	1.0	Activo	29/06/2015 11:22		
55	Planillas	JUGONZALEZ	1.0	Activo	29/06/2015 12:06		
61	Planillas	JUGONZALEZ	1.0	Activo	30/06/2015 11:05		
62	Planillas	JUGONZALEZ	1.0	Activo	30/06/2015 11:09		
63	Planillas	JUGONZALEZ	1.0	Activo	30/06/2015 11:11		

1-6 of 6

Actividades actuales
30/jun/15 11:14:23: 11 - Confirmar Pago (HumanTaskNode)

Registro de instancia del proceso
30/jun/15 11:14:23: 11 - Confirmar Pago (HumanTaskNode)
30/jun/15 11:14:23: 10 - Ingresar a CC (ActionNode)
30/jun/15 11:06:16: 9 - Facturar (HumanTaskNode)
30/jun/15 11:06:16: 8 - Split (HumanTaskNode)
30/jun/15 11:06:13: 7 - Aseguradora (HumanTaskNode)
30/jun/15 11:06:13: 6 - Join (HumanTaskNode)
30/jun/15 11:06:13: 5 - Split (HumanTaskNode)
30/jun/15 11:05:49: 4 - Revision (HumanTaskNode)
30/jun/15 11:05:49: 3 - Join (HumanTaskNode)
30/jun/15 11:05:49: 2 - Split (HumanTaskNode)
30/jun/15 11:05:49: 1 - Generar Planilla (HumanTaskNode)
30/jun/15 11:05:49: 0 - StartNode

Figura 27 Resultado 2 CP014

Id	Nombre	Iniciador	Versión	Estado	Fecha de inicio	Acciones
51	Planillas	admin	1.0	Activo	29/06/2015 11:21	
52	Planillas	JUGONZALEZ	1.0	Activo	29/06/2015 11:22	
55	Planillas	JUGONZALEZ	1.0	Activo	29/06/2015 12:06	
61	Planillas	JUGONZALEZ	1.0	Activo	30/06/2015 11:05	
62	Planillas	JUGONZALEZ	1.0	Activo	30/06/2015 11:09	
63	Planillas	JUGONZALEZ	1.0	Activo	30/06/2015 11:11	

1-6 of 6

Estado de la instancia del proceso
Active

Actividades actuales
30/jun/15 11:14:23: 11 - Confirmar Pago (HumanTaskNode)

Registro de instancia del proceso
30/jun/15 11:14:23: 11 - Confirmar Pago (HumanTaskNode)
30/jun/15 11:14:23: 10 - Ingresar a CC (ActionNode)
30/jun/15 11:10:09: 9 - Facturar (HumanTaskNode)
30/jun/15 11:10:09: 8 - Split (HumanTaskNode)
30/jun/15 11:09:55: 7 - Aseguradora (HumanTaskNode)
30/jun/15 11:09:55: 6 - Join (HumanTaskNode)
30/jun/15 11:09:55: 5 - Split (HumanTaskNode)
30/jun/15 11:09:27: 4 - Revision (HumanTaskNode)
30/jun/15 11:09:27: 3 - Join (HumanTaskNode)
30/jun/15 11:09:27: 2 - Split (HumanTaskNode)
30/jun/15 11:09:27: 1 - Generar Planilla (HumanTaskNode)
30/jun/15 11:09:27: 0 - StartNode

Figura 28 Resultado 3 CP014

Instancias de procesos							Acciones en masa	Actualizar	X	-	+
Mostrando:		Activo	Relacionado conmigo	Completado	Abortado						
Id	Nombre	Iniciador	Versión	Estado	Fecha de inicio	Acciones					
51	Planillas	admin	1.0	Activo	29/06/2015 11:21	Q ⊕					
52	Planillas	JUGONZALEZ	1.0	Activo	29/06/2015 11:22	Q ⊕					
56	Planillas	JUGONZALEZ	1.0	Activo	29/06/2015 12:06	Q ⊕					
61	Planillas	JUGONZALEZ	1.0	Activo	30/06/2015 11:05	Q ⊕					
62	Planillas	JUGONZALEZ	1.0	Activo	30/06/2015 11:09	Q ⊕					
63	Planillas	JUGONZALEZ	1.0	Activo	30/06/2015 11:11	Q ⊕					

Detalles de la instancia del ...		Acciones	Vistas	Actualizar	Actualizar	X	-	+
Versión de definición del proceso		1.0						
Estado de la instancia del proceso		Active						
Actividades actuales		30/jun/15 11:14:23: 11 - Confirmar Pago (HumanTaskNode)						
Registro de instancia del proceso		30/jun/15 11:14:23: 11 - Confirmar Pago (HumanTaskNode) 30/jun/15 11:14:23: 10 - Ingresar a CC (ActionNode) 30/jun/15 11:11:55: 9 - Facturar (HumanTaskNode) 30/jun/15 11:11:55: 8 - Split 30/jun/15 11:11:52: 7 - Aseguradora (HumanTaskNode) 30/jun/15 11:11:52: 6 - Join 30/jun/15 11:11:52: 5 - Split 30/jun/15 11:11:35: 4 - Revision (HumanTaskNode) 30/jun/15 11:11:35: 3 - Join 30/jun/15 11:11:35: 2 - Split 30/jun/15 11:11:35: 1 - Generar Planilla (HumanTaskNode) 30/jun/15 11:11:35: 0 - StartNode						

Figura 29 Resultado 4 CP014

4.4.5.2 Caso de prueba 15. Seguimiento individual de planillas, en cobro.

Caso de Prueba 015	
La facturación final se realiza en por grupos de planillas pero el proceso es individual para cada una	
Código de Identificación:	CP015
Descripción	Cobro de 3 planillas de seguros.
Variables de Entrada (Inputs):	factura
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Seleccionar factura. 2. Fijar monto de pagado. 3. Registrar cobro
Resultado esperado:	Finalización del proceso para cada planilla
Flujo alterno	<ul style="list-style-type: none"> • No se realiza el pago completo de la planilla • Cobro de planilla unica
Resultado alternativo esperado:	<ul style="list-style-type: none"> • Actividad de cobro no realizada en jBPM • Proceso correspondiente a planilla culminado
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:



Figura 30 Resultado 1 CP015

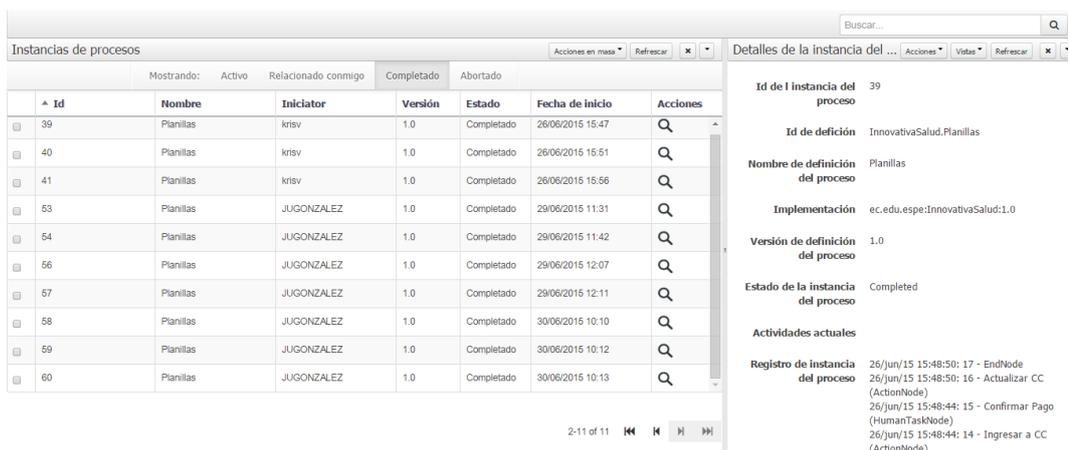


Figura 31 Resultado 2 CP015

4.4.6 Requisito RF6:

Código	Requerimiento	Descripción
RF6	Seguimiento de planillas de manera individual	Cada planilla emitida deberá tener su correspondiente instancia de proceso.

4.4.6.1 Caso de prueba 16. Seguimiento individual de planillas, en facturación.

Caso de Prueba 016	
Seguimiento de planillas de manera individual	
Identificación:	Código de CP016
Descripción	Se comprobara que tras la emisión de planillas se inicia una nueva instancia de proceso
Variables de Entrada (Inputs):	Datos cliente, detalle de planilla, datos de convenio.
Flujo normal del proceso	1. Completar datos de factura. 2. Emitir planilla
Resultado esperado:	Nueva instancia de proceso en jBPM
Flujo alterno	Cliente no asegurado, emisión de factura.
Resultado alternativo esperado:	Emisión de factura sin instancia en jBPM
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

The screenshot shows a web application interface for process management. On the left, there is a table titled 'instancias de procesos' with columns: Id, Nombre, Iniciador, Versión, Estado, Fecha de inicio, and Acciones. The table lists several instances of 'Planillas' created by 'admin' and 'JUGONZALEZ' in an 'Activo' state. On the right, a 'Detalles de la instancia del proceso' panel shows the 'Estado de la instancia del proceso' as 'Activo' and a 'Registro de instancia del proceso' with a list of activities and timestamps, including 'Confirmar Pago', 'Ingresar a CC', 'Facturar', 'Aseguradora', 'Revision', and 'Generar Planilla'.

Figura 32 Resultado 1 CP016

	historico_nombres_apellidos f) character varying(200)	historico_telefono character varying(12)	historico_direccion character varying(200)	codigo_hospital integer	codigo_lote_facturas_electronicas integer	id_proceso integer
534	ISSFFA	(02) 3966000	IHAQUITO	1		118
535				1		0
536	ISSFFA	(02) 3966000	IHAQUITO	1		0
537				1		120
538				1		127
539	ISSFFA	(02) 3966000	IHAQUITO	1		0
540				1		121
541				1		122
542	ISSFFA	(02) 3966000	IHAQUITO	1		0
543				1		127
544				1		0
545				1		0
546				1		0
547				1		53
548	ISSFFA	(02) 3966000	IHAQUITO	1		54
549				1		
550	ISSFFA	(02) 3966000	IHAQUITO	1		56
551	HUGO FABIAN ARMENDARIZ MOLINA	**	**	1		
552				1		56
553	ISSFFA	(02) 3966000	IHAQUITO	1		
554				1		57
555				1		58
556				1		59
557				1		60
558	ISSFFA	(02) 3966000	IHAQUITO	1		0
559				1		61
560				1		62
561				1		63
*						

id_proceso integer
0
118
0
0
120
0
121
122
0
127
0
0
0
0
53
54
56
57
58
59
60
0
61
62
63

Figura 33 Resultado 2 CP016

4.4.7 Requisito RF8:

Código	Requerimiento	Descripción
RF8	En caso en que el bpms no pueda ejecutarse, Innovativa Salud debe poder seguir operando	Si el BPMs no puede ejecutarse o se presenta algún error en comunicaciones Innovativa Salud debe continuar con su funcionamiento normal

4.4.7.1 *Caso de prueba 17. Seguimiento individual de planillas, en facturación.*

Caso de Prueba 017	
En caso en que el bpms no pueda ejecutarse, innovativa Salud debe poder seguir operando	
Código de Identificación:	CP017
Descripción	Se detendrá al servidor Glassfish para simular errores en jBPM, sin embargo Innovativa Salud permanecerá en ejecución.
Variables de Entrada (Inputs):	N/A
Flujo normal del proceso	N/A
Resultado esperado:	Innovativa Salud emite planillas con normalidad.
Flujo alterno	N/A
Resultado alternativo esperado:	N/A
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

El Documento se guardo exitosamente.

No se creo ninguna factura por favor revise las planillas.

Buscar Facturas
Nuevo
Guardar

Version 1.0.1 Copyright CTT

Figura 34 Resultado CP017

4.4.8 Requisito RF10:

Código	Requerimiento	Descripción
RF10	Multiempresa	Innovativa Salud es concebido como sistema multiempresa por lo que el proceso debe funcionar de igual manera

4.4.8.1 Caso de prueba 18. Ejecución con ESPE-Sangolqui.

Caso de Prueba 018 Multiempresa	
Código de Identificación:	CP018
Descripción	Se realizaran una ejecución completa de proceso con usuarios de ESPE-Sangolqui.
Variables de Entrada (Inputs):	N/A
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Crear nueva instancia de proceso. 2. Emitir planilla 3. Atención médica. 4. Revisión. 5. Aseguradora. 6. Facturar. 7. Ingresar a cuentas por cobrar.. 8. Confirmar pago. 9. Actualizar cuentas por cobrar..
Resultado esperado:	Innovativa Salud emite planillas con normalidad.
Flujo alterno	N/A
Resultado alternativo esperado:	N/A
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

69	Planillas_1	JUGONZALEZ	1.0	Completado	30/06/2016 12:27	Q
----	-------------	------------	-----	------------	------------------	---

Registro de instancia del proceso

30/jun/15 12:31:25: 14 - EndNode
 30/jun/15 12:31:25: 13 - Actualizar CC (ActionNode)
 30/jun/15 12:30:50: 12 - Confirmar Pago (HumanTaskNode)
 30/jun/15 12:30:50: 11 - Ingresar a CC (ActionNode)
 30/jun/15 12:30:13: 10 - Facturar (HumanTaskNode)
 30/jun/15 12:30:13: 9 - Split
 30/jun/15 12:30:11: 8 - Aseguradora (HumanTaskNode)
 30/jun/15 12:30:11: 7 - Join
 30/jun/15 12:30:11: 6 - Split
 30/jun/15 12:28:50: 5 - Revision (HumanTaskNode)
 30/jun/15 12:28:50: 4 - Join
 30/jun/15 12:27:43: 3 - Atencion (HumanTaskNode)
 30/jun/15 12:27:43: 2 - Split
 30/jun/15 12:27:43: 1 - Generar Planilla (HumanTaskNode)
 30/jun/15 12:27:43: 0 - StartNode

Figura 35 Resultado CP018

4.4.8.2 Caso de prueba 19. Ejecución con ESPE-Latacunga.

Caso de Prueba 019 Multiempresa	
Código de Identificación:	CP019
Descripción	Se realizaran una ejecución completa de proceso con usuarios de ESPE-Latacunga.
Variables de Entrada (Inputs):	N/A
Flujo normal del proceso	<ol style="list-style-type: none"> 1. Crear nueva instancia de proceso. 2. Emitir planilla 3. Atención médica. 4. Revisión. 5. Aseguradora. 6. Facturar. 7. Ingresar a cuentas por cobrar.. 8. Confirmar pago. 9. Actualizar cuentas por cobrar..
Resultado esperado:	Innovativa Salud emite planillas con normalidad.
Flujo alterno	N/A
Resultado alternativo esperado:	N/A
Errores encontrados	N/A
Sugerencias de corrección	N/A

Captura resultado esperado:

ID	Nombre	Usuario	Valor	Completado	Fecha de Inicio	
70	Planillas_5	DIALTAMIRANO	1.0	Completado	30/06/2015 12:34	🔍

Registro de instancia del proceso	Detalle
30/jun/15 12:36:46: 13 - EndNode	
30/jun/15 12:36:46: 12 - Actualizar CC (ActionNode)	
30/jun/15 12:35:53: 11 - Confirmar Pago (HumanTaskNode)	
30/jun/15 12:35:53: 10 - Ingresar a CC (ActionNode)	
30/jun/15 12:35:18: 9 - Facturar (HumanTaskNode)	
30/jun/15 12:35:18: 8 - Split	
30/jun/15 12:35:15: 7 - Aseguradora (HumanTaskNode)	
30/jun/15 12:35:15: 6 - Join	
30/jun/15 12:35:15: 5 - Split	
30/jun/15 12:34:49: 4 - Revision (HumanTaskNode)	
30/jun/15 12:34:49: 3 - Join	
30/jun/15 12:34:49: 2 - Split	
30/jun/15 12:34:49: 1 - Generar Planilla (HumanTaskNode)	
30/jun/15 12:34:49: 0 - StartNode	

Figura 36 Resultado CP019

CAPÍTULO 5.- Conclusiones y recomendaciones

5.1 Conclusiones

- Existe variedad entre los BPMS libres en el mercado, y la mayoría poseen todas las características básicas para un proyecto como el planteado, sin embargo por las características de lenguaje, documentación, soporte e integración jBPM se seleccionó como la mejor opción para este proyecto.

- La interfaz gráfica de jBPM facilita el desarrollo de procesos y el mantenimiento de los mismos, además la notación BPMN proporciona a los usuarios no técnicos una idea clara de cómo opera el proceso.

- La metodología para procesos no posee un enfoque de Ingeniería de Software incluso en BPMS, debido a que procesos es una herramienta administrativa y los BPMS heredan el ciclo de vida de estos como metodología de mejora continua.

- jBPM ofrece varias opciones para integrar el administrador de procesos a otros proyectos, se determinó que la manera apropiada de realizar esta tarea es mediante servicios en este caso utilizando la tecnología REST que requiere menores recursos para su ejecución.

- DashBoard proyecto generado por JBoss puede ser utilizado fácilmente como complemento de jBPM facilitando la presentación de informes en pantalla y permite la aplicación de filtros en estos informes. Para utilizar otros proyectos como Innovativa Salud es necesario realizar configuraciones de seguridad en su código.

5.2 Recomendaciones

- Se recomienda la utilización de la versión 6.1.0 de jBPM, debido a la estabilidad de la versión así como la existencia de importantes modificaciones entre las diferentes versiones del proyecto.
- Existe un bug dentro del proyecto en su versión 6.1.0 por el cual no se realiza el correcto mapeo de variables dentro de la compuerta lógica divergente, se recomienda utilizar el plugin de Eclipse para realizar de manera adecuado el mapeo de variables.
- Debido a la gran cantidad de librerías necesarias para la correcta integración del sistema se recomienda utilizar Maven para facilitar la descarga de estos recursos.
- Durante la integración, se presentaron problemas con las versiones de ciertas librerías descargadas por Maven, por lo que se recomienda utilizar las versiones manejadas en el archivo pom.xml de proyecto InnovativaRest. (ver Anexo 9)
- Se recomienda la utilización de bpms para la gestión de procesos debido a que el ciclo de vida con estas herramientas posee una etapa de simulación que permite analizar si un determinado cambio al proceso tendrá un impacto favorable en la producción.

CAPÍTULO 6.- ANEXOS

6.1 Código Fuente

6.1.1 Anexo 1: Actividad Actual.

```

package com.espe.ctt.innovativarest;

import java.net.MalformedURLException;
import java.util.List;
import javax.ejb.Stateless;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import org.kie.api.runtime.KieSession;
import org.kie.api.task.TaskService;
import org.kie.api.task.model.TaskSummary;
import org.kie.services.client.api.RemoteRestRuntimeFactory;
import org.kie.services.client.api.command.RemoteRuntimeEngine;

/**
 *
 * @author ashweizer
 */

@Stateless
@Path("webresources/ActividadActual")
public class ActividadActual extends GenericTask{
    @GET
    @Path("get")
    public String ActividadActual(@QueryParam("Nombre") String
nombre,@QueryParam("InstanceID") long
InstanceID,@QueryParam("Usuario") String
usuario,@QueryParam("Contrasena") String contrasena) throws
MalformedURLException
    {
        super.setUser(usuario);
        super.setPassword(contrasena);

        RemoteRestRuntimeFactory restSessionFactory = new
RemoteRestRuntimeFactory(super.getDeploymentId(),super.getAppUrl(),s
uper.getUser(),super.getPassword());
        RemoteRuntimeEngine engine =
restSessionFactory.newRuntimeEngine();

        KieSession ksession = engine.getKieSession();
        TaskService taskService = engine.getTaskService();
        List<TaskSummary> tasks =
taskService.getTasksAssignedAsPotentialOwner(usuario, "en-UK");

        Boolean actual=Boolean.FALSE;

```

```
        for(TaskSummary tasksummary : tasks) {
            if (tasksummary.getName().matches(nombre) &&
tasksummary.getProcessInstanceId()==InstanceID) {
                actual=Boolean.TRUE;
            }
        }
        if(actual)
            return "true";
        else
            return "false";
    }
}
```

6.1.2 Anexo 2: Aprobacion:

```

package com.espe.ctt.innovativarest;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.ejb.Stateless;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import org.kie.api.runtime.KieSession;
import org.kie.api.task.TaskService;
import org.kie.api.task.model.TaskSummary;
import org.kie.services.client.api.RemoteRestRuntimeFactory;
import org.kie.services.client.api.command.RemoteRuntimeEngine;

/**
 *
 * @author ashweizer
 */

@Stateless
@Path("webresources/Aprobacion")

public class Aprobacion extends GenericTask{

    @GET
    @Path("get")

    public String Aprobacion(@QueryParam("InstanceID") long
InstanceID,@QueryParam("Aprobado") boolean aprobado,
@QueryParam("Usuario") String usuario,@QueryParam("Contrasena")
String contrasena) throws MalformedURLException
    {

        super.setUser(usuario);
        super.setPassword(contrasena);

        RemoteRestRuntimeFactory restSessionFactory = new
RemoteRestRuntimeFactory(super.getDeploymentId(),super.getAppUrl(),s
uper.getUser(),super.getPassword());
        RemoteRuntimeEngine engine =
restSessionFactory.newRuntimeEngine();

        KieSession ksession = engine.getKieSession();
        TaskService taskService = engine.getTaskService();
        List<TaskSummary> tasks =
taskService.getTasksAssignedAsPotentialOwner(usuario, "en-UK");
        TaskSummary task = tasks.get(0);

```

```
        for(TaskSummary tasksummary : tasks) {
            if (tasksummary.getProcessInstanceId() == InstanceID)
            {
                task = tasksummary;
                break;
            }
        }

        // kcontext.setVariable(variableName, value);
        taskService.start(task.getId(), usuario);

        Map results = new HashMap();
        results.put("aprobado", aprobado);
        taskService.complete(task.getId(), usuario, results);
        return "1";
    }
}
```

6.1.3 Anexo 3: Generar:

```

package com.espe.ctt.innovativarest;

import java.io.ObjectOutput;
import java.io.ObjectOutputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.ejb.Stateless;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import org.kie.api.runtime.KieSession;
import org.kie.api.task.TaskService;
import org.kie.api.task.model.TaskSummary;
import org.kie.services.client.api.RemoteRestRuntimeFactory;
import org.kie.services.client.api.command.RemoteRuntimeEngine;

/**
 *
 * @author ashweizer
 */

@Stateless
@Path("/webresources/Generar")

public class Generar extends GenericTask{

    @GET
    @Path("get")

    public String Generar(@QueryParam("InstanceID") long
InstanceID,@QueryParam("Atencion") boolean atencion,
@QueryParam("Usuario") String usuario,@QueryParam("Contrasena")
String contrasena) throws MalformedURLException
    {
        super.setUser(usuario);
        super.setPassword(contrasena);

        RemoteRestRuntimeFactory restSessionFactory = new
RemoteRestRuntimeFactory(super.getDeploymentId(),super.getAppUrl(),s
uper.getUser(),super.getPassword());
        RemoteRuntimeEngine engine =
restSessionFactory.newRuntimeEngine();

        KieSession ksession = engine.getKieSession();

```

```
TaskService taskService = engine.getTaskService();
List<TaskSummary> tasks =
taskService.getTasksAssignedAsPotentialOwner(usuario, "en-UK");
TaskSummary task = tasks.get(0);
for(TaskSummary tasksummary : tasks) {
    if (tasksummary.getProcessInstanceId() == InstanceID)
{
        task = tasksummary;
        break;
    }
}

// kcontext.setVariable(variableName, value);
taskService.start(task.getId(), usuario);

Map results = new HashMap();
results.put("atencion",atencion);
taskService.complete(task.getId(), usuario, results);
return "1";
}
}
```

6.1.4 Anexo 4: Iniciar Proceso:

```

package com.espe.ctt.innovativarest;

import java.net.MalformedURLException;
import javax.ejb.Stateless;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import org.kie.api.runtime.KieSession;
import org.kie.api.runtime.process.ProcessInstance;
import org.kie.services.client.api.RemoteRestRuntimeFactory;
import org.kie.services.client.api.command.RemoteRuntimeEngine;

/**
 *
 * @author ashweizer
 */
@Stateless
@Path("/webresources/IniciarProceso")

public class IniciarProceso extends GenericTask{

    @GET
    @Path("get")
    public String Inicio(@QueryParam("Usuario") String
usuario,@QueryParam("Contrasena") String
contrasena,@QueryParam("Hospital") String hospital) throws
MalformedURLException
    {

        super.setUser(usuario);
        super.setPassword(contrasena);

        RemoteRestRuntimeFactory restSessionFactory;
        restSessionFactory = new
RemoteRestRuntimeFactory(super.getDeploymentId(),super.getAppUrl(),s
uper.getUser(),super.getPassword());
        RemoteRuntimeEngine engine =
restSessionFactory.newRuntimeEngine();

        KieSession ksession = engine.getKieSession();
        ProcessInstance processInstance=

ksession.startProcess("InnovativaSalud.Planillas_"+hospital);

        return ""+processInstance.getId();
    }

}

```

6.1.5 Anexo 5: Ping:

```

package com.espe.ctt.innovativarest;

import javax.ws.rs.core.Context;
import javax.ws.rs.core.UriInfo;
import javax.ws.rs.Consumes;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.GET;

/**
 * REST Web Service
 *
 * @author ashweizer
 */
@Path("webresources/Ping")
public class Ping {

    @Context
    private UriInfo context;

    /**
     * Creates a new instance of PingResource
     */
    public Ping() {
    }

    /**
     * Retrieves representation of an instance of
     com.espe.ctt.innovativarest.Ping
     * @return an instance of java.lang.String
     */
    @GET
    @Path("get")
    public String getXml() {
        //TODO return proper representation object
        return "<html><body><h1>Hello World123!</body></h1></html>";
    }

    /**
     * PUT method for updating or creating an instance of Ping
     * @param content representation for the resource
     * @return an HTTP response with content of the updated or
     created resource.
     */
    @PUT
    @Consumes("application/xml")
    public void putXml(String content) {
    }
}

```


6.1.6 Anexo 6: Realizar Tarea:

```

package com.espe.ctt.innovativarest;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.ejb.Stateless;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import org.kie.api.runtime.KieSession;
import org.kie.api.task.TaskService;
import org.kie.api.task.model.TaskSummary;
import org.kie.services.client.api.RemoteRestRuntimeFactory;
import org.kie.services.client.api.command.RemoteRuntimeEngine;

/**
 *
 * @author ashweizer
 */

@Stateless
@Path("/webresources/RealizarTarea")

public class RealizarTarea extends GenericTask {

    @GET
    @Path("get")

    public String Correccion(@QueryParam("InstanceID") long
InstanceID,@QueryParam("Usuario") String
usuario,@QueryParam("Contrasena") String contrasena) throws
MalformedURLException
    {
        super.setUser(usuario);
        super.setPassword(contrasena);

        RemoteRestRuntimeFactory restSessionFactory = new
RemoteRestRuntimeFactory(super.getDeploymentId(),super.getAppUrl(),s
uper.getUser(),super.getPassword());
        RemoteRuntimeEngine engine =
restSessionFactory.newRuntimeEngine();

        KieSession ksession = engine.getKieSession();
        TaskService taskService = engine.getTaskService();
        List<TaskSummary> tasks =
taskService.getTasksAssignedAsPotentialOwner(usuario, "en-UK");
        TaskSummary task = tasks.get(0);
        for(TaskSummary tasksummary : tasks) {

```

```
        if (tasksummary.getProcessInstanceId() == InstanceID)
    {
        task = tasksummary;
        break;
    }

    taskService.start(task.getId(), usuario);

    Map results = new HashMap();
    results.put("atencion", "true");
    taskService.complete(task.getId(), usuario, results);
    return "1";
}
}
```

6.1.7 Anexo 7: Revision:

```

package com.espe.ctt.innovativarest;

import java.net.MalformedURLException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import javax.ejb.Stateless;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.QueryParam;
import org.kie.api.runtime.KieSession;
import org.kie.api.task.TaskService;
import org.kie.api.task.model.TaskSummary;
import org.kie.services.client.api.RemoteRestRuntimeFactory;
import org.kie.services.client.api.command.RemoteRuntimeEngine;

/**
 *
 * @author ashweizer
 */
@Stateless
@Path("webresources/Revision")
public class Revision extends GenericTask {
    @GET
    @Path("get")
    public String Generar(@QueryParam("InstanceID") long
InstanceID,@QueryParam("Continuar") boolean continuar,
@QueryParam("Usuario") String usuario,@QueryParam("Contrasena")
String contrasena) throws MalformedURLException
    {
        super.setUser(usuario);
        super.setPassword(contrasena);

        RemoteRestRuntimeFactory restSessionFactory = new
RemoteRestRuntimeFactory(super.getDeploymentId(),super.getAppUrl(),s
uper.getId(),super.getPassword());
        RemoteRuntimeEngine engine =
restSessionFactory.newRuntimeEngine();

        KieSession ksession = engine.getKieSession();
        TaskService taskService = engine.getTaskService();
        List<TaskSummary> tasks =
taskService.getTasksAssignedAsPotentialOwner(usuario, "en-UK");
        TaskSummary task = tasks.get(0);
        for(TaskSummary tasksummary : tasks) {
            if (tasksummary.getProcessInstanceId() == InstanceID)
{

```

```
        task = tasksummary;
        break;
    }
}

// kcontext.setVariable(variableName, value);
taskService.start(task.getId(), usuario);

Map results = new HashMap();
results.put("continuar",continuar);
taskService.complete(task.getId(), usuario, results);
return "1";
}
}
```

6.1.8 Anexo 8: Proceso Bean:

```

package com.jvc.medisys.bean.clienterest;

import com.jvc.medisys.clienterest.ActividadActual;
import com.jvc.medisys.clienterest.Atencion;
import com.jvc.medisys.clienterest.GenerarPlanilla;
import com.jvc.medisys.clienterest.IniciarProceso;
import com.jvc.medisys.clienterest.Pendientes;
import com.jvc.medisys.clienterest.RespuestaAseguradora;
import com.jvc.medisys.clienterest.RevisarPlanilla;

/**
 *
 * @author ashweizer
 */
public class ProcesoBean {

    String ID;

    public ProcesoBean() {

    }

    public ProcesoBean(String ID) {
        this.ID = ID;
    }

    public void IniciarProceso(String usuario, String contrasena,
String hospital)
    {
        IniciarProceso client = new IniciarProceso();
        try {

            Object response = client.Inicio(usuario,contrasena,hospital);
            // return "Inicio.xhtml";
            this.ID=response.toString();
        } catch (Exception e) {

        }

    }

    public Boolean GenerarPlanilla(String InstanceID,String
atencion,String Usuario, String Contrasena) throws Exception
    {
        try {
            GenerarPlanilla client = new GenerarPlanilla();
            Object response = client.Generar(InstanceID, atencion, Usuario,
Contrasena);
            if(response.equals("1"))
            {
                return true;
            }else{

```

```

return false;}

    } catch (Exception e) {
        return false;
    }
}

    public Boolean Atencion(String InstanceID,String Usuario, String
Contrasena) throws Exception
    {
        try {
            Atencion client = new Atencion();
            Object response = client.Atencion(InstanceID, Usuario,
Contrasena);
            if(response.equals("1"))
        {
            return true;
        }else{
            return false;}

        } catch (Exception e) {
            return false;
        }
    }

    public Boolean RevisarPlanilla(String InstanceID,String
Continuar,String Usuario, String Contrasena) throws Exception
    {
        try {
            RevisarPlanilla client = new RevisarPlanilla();
            Object response =
client.Revision(InstanceID,Continuar,Usuario, Contrasena);
            if(response.equals("1"))
        {
            return true;
        }else{
            return false;}

        } catch (Exception e) {
            return false;
        }
    }

    public Boolean ActividadActual(String nombre, String InstanceID,
String Usuario, String Contrasena) throws Exception
    {
        try {
            ActividadActual client = new ActividadActual();
            Boolean response = client.ActividadActual(nombre,
InstanceID,Usuario, Contrasena);

            return response;
        } catch (Exception e) {
            return false;
        }
    }

```

```

    }
}

public Boolean Aseguradora(String ID, String Aprobado, String
usuario, String Contraseña) throws Exception
{
    try {
        RespuestaAseguradora client = new
RespuestaAseguradora();
        Object response = client.Aprobacion(ID, Aprobado, usuario,
Contraseña);
        if(response.equals("1"))
    {
        return true;
    }else{
        return false;}
        } catch (Exception e) {
            return false;
        }
    }

public boolean isNumeric(String cadena){
    try {
        Integer.parseInt(cadena);
        return true;
    } catch (NumberFormatException nfe){
        return false;
    }
}

public String getID() {
    return ID;
}

public void setID(String ID) {
    this.ID = ID;
}

public void Aseguradora(String string, String toString) {
    throw new UnsupportedOperationException("Not supported
yet."); //To change body of generated methods, choose Tools |
Templates.
}
}

```

6.1.9 Anexo 9: Pom.xml de InnovativaRest:

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.espe.ctt</groupId>
  <artifactId>InnovativaRest</artifactId>
  <version>1</version>
  <packaging>war</packaging>

  <name>InnovativaRest</name>

  <properties>

  <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
    <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.jboss.resteasy</groupId>
      <artifactId>resteasy-jaxrs</artifactId>
      <version>3.0.6.Final</version>
    </dependency>
    <dependency>
      <groupId>org.kie.remote</groupId>
      <artifactId>kie-services-client</artifactId>
      <version>6.0.1.Final</version>
      <type>jar</type>

    </dependency>
    <dependency>
      <groupId>javax</groupId>
      <artifactId>javaee-web-api</artifactId>
      <version>6.0</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.3.2</version>
        <configuration>
          <source>1.6</source>
          <target>1.6</target>
          <compilerArguments>
            <endorseddirs>${endorsed.dir}</endorseddirs>
          </compilerArguments>
        </configuration>
      </plugin>
    </plugins>
  </build>

```

```

        </configuration>
    </plugin>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.1.1</version>
        <configuration>
            <failOnMissingWebXml>false</failOnMissingWebXml>
        </configuration>
    </plugin>
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-dependency-plugin</artifactId>
        <version>2.1</version>
        <executions>
            <execution>
                <phase>validate</phase>
                <goals>
                    <goal>copy</goal>
                </goals>
                <configuration>
                    <outputDirectory>${endorsed.dir}</outputDirectory>
                    <silent>true</silent>
                    <artifactItems>
                        <artifactItem>
                            <groupId>javax</groupId>
                            <artifactId>javaee-endorsed-
api</artifactId>
                            <version>6.0</version>
                            <type>jar</type>
                        </artifactItem>
                    </artifactItems>
                </configuration>
            </execution>
        </executions>
    </plugin>
</plugins>
</build>

</project>

```

6.2 Glosario de Términos

- actividad
Conjunto de operaciones o tareas propias de una persona o entidad, 13, 28, 29, 40, 42
- BonitaSoft
BPMS, 11, 14, 114
- BPMN
Business Process Modeling Notation, 10, 11, 13, 20, 75, 77
- BPMN2
Segunda versión de la notación para BPM, 7, 13, 36, 115
- bpms
Siglas de Business Process Management Software, 6, 21, 25, 39, 40, 51, 52, 71
- Business Process Management, 6, 10, 21
Metodología que permite a las empresas modelizar, implementar y ejecutar conjuntos de actividades interrelacionadas., 6
- código
Sistema de signos y de reglas que permite formular y comprender un mensaje., 10, 11, 12, 14, 15, 16, 17, 36, 42, 46, 47, 75
- código abierto
software o hardware distribuido y desarrollado libremente. Se focaliza más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto se destacan en el software libre., 10, 11, 14, 15, 16
- comunicación
Transmisión de señales mediante un código común al emisor y al receptor., 18, 19, 39
- Dashboard Builder
Herramienta de JBOSS para presentación de reportes en pantalla, 36, 51
- Data Source
Fuente de datos, 37, 38, 51
- drag-n-drop
Acción de arrastrado y pegado, 11
- eclipse
IDE de desarrollo que proporciona servicios integrales para facilitarle al programador de computadora el desarrollo de software., 11
- filtro
Sistema de selección en un proceso según criterios previamente establecidos., 36, 40
- gráficos
Representación de datos numéricos por medio de una o varias líneas que hacen visible la relación que esos datos guardan entre sí., 36, 37, 39, 40, 41
- HTML
Texto que contiene elementos a partir de los cuales se puede acceder a otra información., 9, 19
- información
Comunicación o adquisición de conocimientos que permiten ampliar o precisar los que se poseen sobre una materia determinada., 7, 9, 10, 12, 17, 22, 27, 36, 38, 39, 40, 43, 44, 45, 46, 47, 48, 51, 115
- infraestructura
Conjunto de elementos o servicios que se consideran necesarios para la creación y funcionamiento de una organización cualquiera., 18
- Innovativa Salud
Sistema de Gestión hospitalaria desarrollado por el centro de

partir del acrónimo de la expresión
inglesa eXtensible Markup
Language., 18

BIBLIOGRAFÍA

- Activiti. (17 de 09 de 2015). *Activiti*. Obtenido de <http://www.activiti.org/userguide/>
- Aguilar Mosquera, O., & Chico Macias, G. (2014). *APLICACIÓN MÓVIL, PARA EMITIR SEÑALES DE AUXILIO INMEDIATO A TRAVÉS DE LA TECNOLOGÍA ANDROID PARA EL UPC DE LA CABECERA CANTONAL DE LA CIUDAD DE BABA*. Babahoyo.
- AuraPortal. (12 de 07 de 2015). *auraportal.com*. Obtenido de <http://www.auraportal.com/es/ciclo-vida-procesos-bpm-workflow-software>
- Barbero, P. C. (s.f.). *Plataformas de integración. Servicios web basados en rest y soap*.
- Bizagui. (2014). *BPMN2* .
- BonitaSoft S.A. (2012). *Introduction Tutorial*.
- Innovativa. (17 de 09 de 2015). *CTT ESPE - CECAI*. Obtenido de CTT ESPE - CECAI
- Java. (09 de 17 de 2015). *Java*. Obtenido de <https://www.java.com/es/download/faq/techinfo.xml>
- Kian Garimella, M. L. (s.f.). *BPM (GERENCIA DE PROCESOS)*.
- Koftikian, J. (s.f.). *Simple Object Access Protocol*. Hamburg.
- Navarro Marsler, R. (08 de 02 de 2015). <http://users.dsic.upv.es/>. Obtenido de <http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf>
- NetBeans. (17 de 09 de 2015). *NetBeans*. Obtenido de <https://netbeans.org/kb/docs/java/quickstart.html>

Oracle. (08 de 02 de 2015). Obtenido de oracle.com:
<http://www.oracle.com/technetwork/java/javaee/overview-140548.html>

Oracle. (08 de 02 de 2015). *oracle.com*. Obtenido de
<http://www.oracle.com/technetwork/java/overview-138580.html>

Papazoglou, M. P. (08 de 02 de 2015). *cs.colorado.edu*. Obtenido de
http://www.cs.colorado.edu/~kena/classes/7818/f08/lectures/lecture_3_soap.pdf

Pérez, J. D. (s.f.). *Notaciones y lenguajes de procesos. Una visión global*. Sevilla.

ProcessMaker. (07 de 02 de 2015). *processmaker.com*. Obtenido de
<http://www.processmaker.com/es/key-features>

Real Academia Española. (06 de 02 de 2015). *rae*. Obtenido de
<http://lema.rae.es/drae/?val=hipertexto>

Serra Manchado, D. (Junio de 2010). Estudio del servidor de. Barcelona, España.

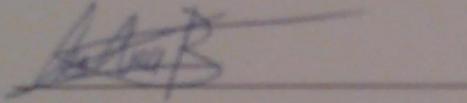
The JBoss jBPM team. (2014). *jBPM User Guide 6.2.0.Final*.

w3. (06 de 02 de 2015). *w3*. Obtenido de <http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html>

WildFly. (17 de 09 de 2015). *docs.jboss*. Obtenido de
[https://docs.jboss.org/author/display/WFLY8/Getting+Started+Guide#GettingSt
artedGuide-GettingStartedwithWildFly8](https://docs.jboss.org/author/display/WFLY8/Getting+Started+Guide#GettingStartedGuide-GettingStartedwithWildFly8)

HOJA DE LEGALIZACIÓN DE FIRMAS

ELABORADO POR



SHWEIZER BONILLA ARIE

DIRECTOR DE CARRERA



ING. MAURICIO CAMPAÑA

