UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS

TEMA:

DESARROLLO DE COMPONENTES PARA LA PLATAFORMA BONITA BPM (BUSINESS PROCESS MANAGEMENT) UTILIZANDO LA METODOLOGÍA CBD (COMPONENT BASED DEVELOPMENT).

Trabajo de Graduación Modalidad: Proyecto de Investigación, presentado previo la obtención del título de Ingeniero en Sistemas Computacionales e Informáticos

SUBLÍNEA DE INVESTIGACIÓN: Métodos avanzados de producción de Software.

AUTOR: Darwin Stalin Ramírez Supe
TUTOR: Ing. Álvarez Mayorga Édison Homero, Mg

Ambato - Ecuador

FEBRERO, 2016

APROBACIÓN DEL TUTOR

En mi calidad de Tutor del Trabajo de Investigación sobre el Tema:

"DESARROLLO DE COMPONENTES PARA LA PLATAFORMA BONITA BPM (BUSINESS PROCESS MANAGEMENT) UTILIZANDO LA METODOLOGÍA CBD (COMPONENT BASED DEVELOPMENT).", del señor Darwin Stalin Ramírez Supe, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, considero que el informe investigativo reúne los requisitos suficientes para que continúe con los trámites y consiguiente aprobación de conformidad con el numeral 7.2 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ambato, febrero del 2016

EL TUTOR

Ing. Mg. Álvarez Mayorga Edison Homero.

AUTORÍA

El presente trabajo de investigación titulado: **DESARROLLO DE COMPONENTES PARA LA PLATAFORMA BONITA BPM (BUSINESS PROCESS MANAGEMENT) UTILIZANDO LA METODOLOGÍA CBD (COMPONENT BASED DEVELOPMENT)**. Es absolutamente original, auténtico y personal, en tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, febrero del 2016

Darwin Stalin Ramírez Supe

CC: 1804608774

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.
Cedo los derechos de mi Trabajo de Titulación, con fines de difusión pública, además autorizo su reproducción dentro de las regulaciones de la Universidad.

Ambato, febrero del 2016

Darwin Stalin Ramírez Supe

CC: 1804608774

APROBACIÓN COMISIÓN CALIFICADORES

La Comisión Calificadora del presente trabajo conformada por los señores docentes Ing. Franklin Mayorga, Ing. Renato Urvina, revisó y aprobó el Informe Final del trabajo de graduación titulado **DESARROLLO DE COMPONENTES PARA LA PLATAFORMA BONITA BPM(BUSINESS PROCESS MANAGEMENT) UTILIZANDO LA METODOLOGÍA CBD(COMPONENT BASED DEVELOPMENT)**, presentado por el señor Darwin Stalin Ramírez Supe de acuerdo al numeral 9.1 de los Lineamientos Generales para la aplicación de Instructivos de las Modalidades de Titulación de las Facultades de la Universidad Técnica de Ambato.

Ing Vicente Morales, Mg.

PRESIDENTE DEL TRIBUNAL

Ing. Franklin Mayorga, Mg	Ing. Renato Urvina, Mg
DOCENTE CALIFICADOR	DOCENTE CALIFICADOR

DEDICATORIA

Este proyecto así como el resultado de todos estos años de estudio van dedicados únicamente a la persona que hizo capaz que yo lograra tener éxito en la vida. A la persona quien con tanto esfuerzo me sacó adelante, quien luchó y lucha incansablemente por mí, quien me enseño los valores que he adquirido con el pasar de los años, quien me ha dado el ejemplo de ser un luchador en la vida y jamás rendirme, quien me enseño a convertir los muros, en obstáculos y sobrepasar cada una de las pruebas que la vida me ha puesto y quien me enseño a levantar la cabeza para no perder el rumbo de mis metas.
Gracias a ella hoy soy lo que soy, esto va dedicado a ti madre mía.

Stalin Ramírez

AGRADECIMIENTO

Debo agradecer de manera muy especial al Doctor José María Lavín por haberme guiado en una etapa importante de mi vida, por sus sabios consejos, por hacerme ver mis errores, quien me ha entregado su amistad de manera desinteresada. José es lo más cercano a un padre que he tenido por todo lo mencionado se ha ganado todo mi cariño y respeto.

Agradezco a mis tutores: Ing. Franklin Mayorga e Ing. Édison Álvarez, quienes me dieron cabida en el proyecto y siempre estuvieron presentes para con su ayuda.

Agradezco a mis amigos: Cristian Gallardo, Renato Toasa, Francisco Galora, mi primo Byron Palate que ha sido como mi hermano y muchos otros amigos tanto como de la infancia, colegio y en mi vida universitaria, quienes me han enseñado el valor de la verdadera amistad.

Stalin Ramírez

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

La necesidad de incorporar nuevos componentes en ediciones de software libres (community) ha sido un inconveniente para los programadores al momento de desarrollar aplicaciones y verse restringidos en el desarrollo del software. Además, los problemas económicos en empresas tanto privadas como públicas han obligado a que se usen versiones libres obligando a su equipo de programadores a tener un desarrollo de software de una manera no convencional.

La metodología CBSE proporciona un modelo de desarrollo especializado en componentes para la especificación de requisitos, desarrollo, pruebas y liberación de los mismos. Con esta metodología, los programadores podrán desarrollar e integrar componentes adicionales al software community con la finalidad de construir un software de calidad.

El proyecto SEGIC, de la Universidad Técnica de Ambato, ha adoptado dicha metodología para integrar nuevos componentes en Bonita BPM que es una plataforma especializada en workflow para la gestión de procesos, más concretamente de los procesos de acreditación de las carreras. Estos componentes a integrar son imprescindibles para que las versiones community de este tipo de software puedan incorporar más utilidades. En el caso que nos ocupa fue la posibilidad de subir y mostrar múltiples documentos simultáneamente a Bonita BPM.

ABSTRACT

The need to incorporate new software components on issues of community has been inconvenient for programmers when developing applications and be restricted in software development. In addition, economic problems in both private and public companies that have forced free versions are used forcing his team of programmers to have a software development in an unconventional way.

The CBSE methodology provides a development model that specializes in components for requirements specification, development, testing and release them. With this methodology, developers can develop and integrate additional software components to the community in order to build quality software.

The SEGIC project of the Universidad Técnica de Ambato, has adopted this methodology to integrate new components in Bonita BPM that specializes in workflow management process, specifically the process of accreditation of racing platform. These components are essential to integrate the community versions of this type of software can bring more profits. In the case before us was the ability to upload and display multiple documents simultaneously Bonita BPM.

INTRODUCCIÓN

Actualmente, las empresas privadas como públicas operan en un entorno global que cambia rápidamente. Todas ellas deben responder a las nuevas oportunidades y mercados, condiciones económicas cambiantes y a la aparición de productos y servicios competitivos. El software se ha convertido en parte esencial de casi todas las operaciones de negocio, por lo que es fundamental que el mismo se desarrolle rápidamente, para aprovechar las nuevas oportunidades y responder a la presión del mercado [1]. El Desarrollo de Software Basado en Componentes (CBSE, en sus siglas en inglés) juega un papel vital en el aumento de la productividad de una organización ya que evita retrasos en el desarrollo del software.
Los componentes de software almacenados se utilizaran para construir software sin tener que empezar desde cero, estas dos no solamente incrementan la calidad del producto sino que optimizan el tiempo [2].

La Universidad Técnica de Ambato (UTA) ha puesto en marcha un proyecto para la automatización del proceso de recolección de datos y evidencias para la evaluación de carreras establecido por el concejo de Evaluación, Acreditación y Aseguramiento de la Calidad de la Educación Superior (CEEACES). Este proyecto ha sido desarrollado mediante la aplicación de una herramienta de software libre especializada en Business Process Management denominado (BPM)[3], Esta herramienta permite el modelado e implementación de todo el proceso de recolección de evidencias necesarias para la acreditación de todas las carreras de la UTA.

Por razones económicas, el proyecto se realiza con la edición libre *(community)* de la plataforma Bonita Studio [4]. Esta versión incluye características muy reducidas en comparación con las ediciones *TeamWork, Efficiency y Performance* [4]. Esto ha generado un problema al momento de diseñar las interfaces web de algunos procesos en el cual se solicita y se visualizan múltiples archivos. BonitaStudio viene con un componente para dicha acción pero está limitado a solicitar un único

archivo, haciendo de estos procesos una tarea recursiva y repetitiva ya que siendo la petición de archivos una tarea de sistema, se ha acoplado a un proceso de negocio y esto genera una sobrecarga de trabajo tanto en el servidor como para los usuarios. Además, el ambiente de desarrollo integrado (IDE, en sus siglas en inglés) de BonitaStudio es muy restrictivo para programar un componente de solución integrada.

CAPÍTULO 1

El problema

1.1. Tema de Investigación

"Desarrollo de componentes para la plataforma Bonita BPM (Business Process Management) utilizando la metodología CBD (Component Based Development)".

1.2. Planteamiento del problema

La creciente necesidad de realizar sistemas complejos en cortos periodos de tiempo, a la vez con menores esfuerzos tanto humanos como económicos, está favoreciendo el avance de lo que se conoce como Desarrollo de Software Basado en Componentes. Esta nueva disciplina se apoya en componentes de software ya desarrollados, que son combinados adecuadamente para satisfacer los requisitos del sistema. Construir una aplicación se convierte por tanto en la búsqueda y ensamblaje de piezas prefabricadas, desarrolladas en su mayoría por terceras personas. Bajo este nuevo planteamiento, cobran especial interés los procesos de búsqueda y selección de los componentes apropiados para construir las aplicaciones. En este sentido, la tecnología de componentes ofrece soluciones robustas para muchos de los problemas que se plantean en la construcción de grandes sistemas de software[5].

Así el desarrollo de software basado en componentes se ha convertido en uno de los mecanismos más efectivos para la construcción de grandes sistemas y aplicaciones de software. Una vez que la mayor parte de los aspectos funcionales de esta disciplina comienzan a estar bien definidos, la atención de la comunidad de desarrolladores comienza a centrarse en los aspectos extra-funcionales y de calidad, como un paso hacia una verdadera ingeniería [5].
Por otra parte en la Unidad Operativa de la Dirección de Investigación de Desarrollo de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial(UODIDE), se

realiza un proyecto BPM(Business Process Management) denominado Sistema Electrónico de Garantía Interna de Calidad (SEGIC), el cual se lo está desarrollando en la plataforma Bonita BPM versión community, la cual contiene características muy reducidas y no satisfacen las necesidades el proyecto en su totalidad, los componentes faltantes hacen que el modelo sea mucho más complejo haciendo que el proceso de recolección de evidencias sea redundante y así produciendo sobrecargas en el sistema.

1.3. Delimitación

Área Académica: Software
Línea de Investigación: Ingeniería del software.
Sub Línea de Investigación: Métodos avanzados de producción de Software.
Delimitación Espacial: La presente investigación se realizará en la Universidad Técnica de Ambato, en la provincia de Tungurahua, cantón Ambato, predios Huachi Av. Los Chasquis.
Delimitación Temporal: La presente investigación se desarrollará, en los 6 meses posteriores a la aprobación del proyecto por parte del Honorable Consejo Directivo.

1.4. Justificación

El objetivo principal de este proyecto se da por la necesidad de integrar nuevos componentes que realicen tareas específicas inexistentes en la plataforma Bonita BPM, para desarrollar el proyecto que se lleva a cabo en la Unidad Operativa de la Dirección de Investigación y Desarrollo de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

Así, el tema a investigar es de gran importancia para el departamento de investigación, ya que con éste coadyuva el proyecto SEGIC y deja abierta la posibilidad para la integración de nuevos componentes apoyándose en la metodología a investigar en el presente proyecto.

Los beneficiarios directos con la implementación de este proyecto será el equipo de investigación que desarrollan el sistema SEGIC para la Universidad Técnica de Ambato ya que esta metodología ayudará a desarrollar e integrar nuevos componentes que ayudarán a diseñar un flujo BPM más comprensivo y optimizar tareas que no

lleven a redundancias ni sobrecargas en el sistema.

Además la Unidad Operativa de la Dirección de Investigación de Desarrollo de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial prestará facilidades necesarias para la realización del presente proyecto de graduación, el mismo que simplificará tareas específicas del proyecto que se lleva a cabo en dicho departamento.

1.5. Objetivos

1.5.1. General

Desarrollar componentes para la plataforma Bonita BPM (Business Process Management) utilizando la metodología CBD (Component Based Development).

1.5.2. Específicos

- Determinar los componentes a desarrollarse para el proyecto SEGIC.
- Describir la metodología a utilizar.
- Desarrollar e integrar los componentes en la plataforma Bonita BPM.
- Implantar los componentes creados en el sistema SEGIC.

CAPÍTULO 2

Marco Teórico

2.1. Antecendentes Investigativos

Laura Andrade describe que el desarrollo de software basado en componentes es un proceso que concede particular importancia al diseño y la construcción de sistemas basados en computadoras que utilizan componentes de software reutilizables. Es una tecnología que ofrece ventajas en tiempo de desarrollo, también reducción de costos en el proceso de desarrollo de software, además describe, construye, utiliza técnicas software para la elaboración de sistemas abiertos y distribuidos mediante el ensamblaje de partes software[6].

Mariela Farfán señala que las nuevas tendencias en desarrollo de software es simplificar las tareas que se realizan en el afán de llegar a completar en el menor tiempo posible el adelanto de un proyecto, con lo cual los ingenieros en sistemas optan por el uso de ciertos componentes en sus proyectos, tal es el caso de las páginas web en las cuales se unen diversos elementos encontrados como: menús, diversas validaciones, animaciones, etc, los cuales funcionan por si solos y pueden ser reutilizados en diversos casos. Por lo que hemos decido realizar un estudio de la ISBC (Ingeniería de software basada en componentes) que supone la utilización de “partes de software“ bien especificadas, con interfaces descritas como contratos, con arquitecturas de software y estándares para hacer el diseño genérico de una aplicación para el ingreso de notas dirigida a un centro educativo[7].

I-Ling Yen, pone especial énfasis en el rápido crecimiento de la demanda de los sistemas y el aumento de la complejidad de software embebido, plantean una necesidad urgente de técnicas de desarrollo de software integrado y avanzado. La tecnología está cambiando hacia un proceso semiautomatizado de generación de código e integración de sistemas de componentes. Por ende en el Desarrollo Basado en Componentes (CDB), existen técnicas que pueden reducir significativamente el tiempo y costo para el desarrollo de software. Sin embargo, hay algunos problemas difíciles con el enfoque del CDB. Como los componentes de identificación y recuperación. Los diseñadores necesitan pasar por una curva de aprendizaje con el fin de componer efectivamente una SystemOut de componentes disponibles. En este trabajo se discute un mecanismo integrado para el desarrollo basado en componentes de software embebido, para facilitar la gestión de componentes y recuperación. El repositorio utiliza una ontología para facilitar la navegación y una eficaz búsqueda [8].

Luiz Fernando Capretz, infiere que el desarrollo de software basado en componentes (CBD) se esfuerza por lograr un conjunto de software estandarizada componentes pre-construidos disponibles para adaptarse a un estilo arquitectónico específico para algunos dominios de aplicaciones; el tiempo de desarrollo frente a las fases del ciclo de vida del software, es una evaluación importante del modelo de desarrollo basado en componentes propuesto [9].

S. Mahamood, concluye que debido a los extensos usos de los componentes, el desarrollo de software basado en componentes es bastante diferente de la metodología en cascada de enfoque tradicional. CBSE no sólo requiere centrarse en el sistema, especificación y desarrollo, sino también requiere consideración adicional para el contexto general del sistema, las propiedades de componentes individuales y proceso de integración. El desarrollo de software basado en componentes (CDB), puede ser referido como el proceso para la construcción de un sistema que utiliza componentes. El ciclo de vida del CDB consiste en un conjunto de fases, a saber, la identificación y selección de componentes sobre la base de las partes interesadas, requisitos, la integración y el montaje de los componentes seleccionados, por último actualizar el sistema como componentes a evolucionar en el tiempo con las nuevas versiones. Este trabajo presenta un estudio de la literatura indicativo de técnicas propuestas para las diferentes fases del ciclo de vida CDB. El objetivo de esta encuesta es ayudar proporcionar una mejor comprensión de las diferentes técnicas del CDB para cada

una de estas áreas [10].

Meng Shang, describe que el desarrollo de software basado en componentes es un enfoque eficaz, eficiente para mejorar la productividad y calidad del desarrollo de software. Hasta el momento, casi todas las tecnologías de componentes de software se centran en el componente, descripción y funcionamiento. Sin embargo, la arquitectura de software que toma como componente la unidad básica, tiene un gran progreso, y proporciona un proceso de desarrollo de software de arriba hacia abajo a través de la descripción de su estructura y características. A fin de mejorar la eficiencia y calidad de la información de gestión sistema (MIS), la tecnología de componentes reutilizables ha sido introducido en MIS. A través de la descripción de la estructura y las características, los componentes reutilizables se utilizan en la estructura y funciones de SIG en el proceso de desarrollo de tales como el análisis de la demanda, diseño de sistemas, implementación de sistemas, las pruebas del sistema, y el mantenimiento del sistema. El resultado muestra que apoyar y aplicar el conjunto de MIS bajo la plataforma de componente, funcionamiento, ha mejorado la estabilidad, calidad y desarrollo de la eficiencia del MIS [11].

Tien N.Nguyen propone que la Ingeniería del software basada en componentes es una tecnología clave para el desarrollo y mantenimiento sistemas de software de gran escala en un entorno de outsourcing. Estos componentes de software tienden a ser desarrollado al mismo tiempo en diferentes lugares con el ciclo de vida asíncrono. En la práctica, las nuevas versiones de componentes de software a menudo pueden cambiar sus interfaces. Para explorar espacios de soluciones de software basado en componentes, actualización y evolución, hemos desarrollado un formalismo que es capaz de modelar el proceso de actualización de software en un entorno de desarrollo concurrente basado en componentes. El formalismo se puede utilizar para determinar la solidez de un proceso de actualización de software basado en componentes, es decir, para hacer seguro que los componentes se actualicen en el orden correcto [12].

2.2. Fundamentación teórica

Para el presente proyecto, es necesario tener en claro conceptos relacionados sobre CBD (Component-Based Development).

Reutilización de software.- La reutilización de software es un proceso de la ingeniería de software que conlleva al uso recurrente de funciones de software, análisis, diseño, implementación y pruebas de una aplicación o sistema de software [13].

Sistema de aplicación.- Un conjunto de herramientas que permiten la creación e interconexión de componentes software, junto con una colección de servicios para facilitar las labores de los componentes que residen y se ejecutan en él [14].

Desarrollo de Software Basado en Componentes.- El desarrollo de software basado en componentes puede verse como una extensión natural de la programación orientada a objetos dentro del ámbito de los sistemas abiertos y distribuidos. Este paradigma se basa en el uso de los componentes software como entidades básicas del modelo, entendiendo por componente "una unidad de composición de aplicaciones software que posee un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes, de forma independiente en tiempo y espacio"[15].
Características y utilidades:
Es utilizado para reducir los costos, tiempo y esfuerzos de desarrollo del software, y de esta manera incrementar el nivel de productividad de los grupos desarrolladores y minimizar los riesgos; a su vez ayuda a optimizar la fiabilidad, flexibilidad y la reutilización de la aplicación final. De esta manera, las pequeñas empresas pueden tener una mayor confiabilidad a la hora de realizar una inversión tecnológica.

La modularidad, la reusabilidad y compatibilidad son características muy relevantes de la tecnología de programación basada en componentes, en las cuales coincide con la tecnología orientada a objetos de la que puede considerarse una evolución. No obstante en esta tecnología también se requiere robustez debido a que los componentes deben operar en entornos muchos más heterogéneos.

El DSBC, se corresponde al paradigma de programación de sistemas abiertos, los cuales son extensibles y tienen una interacción con componentes heterogéneos que se integran o abandonan el sistema de manera dinámica, o sea, que los componentes pueden ser substituidos por otros componentes, independientemente de su arquitectura y desarrollo.

Otro aspecto a tener en cuenta en el DBC, es el poder integrar lo mejor de las tecnologías para realizar aplicaciones personalizadas, ajustadas a los requisitos de los clientes; lo cual le permite a los desarrolladores y/o a la empresa adquirir las tecnologías que más se adapten a sus particularidades, sin incurrir en gastos de licenciamiento o soporte y actualización de grandes soluciones, aunque muchas de estas tecnologías se encuentran bajo la premisa de Freeware y GNU (General Public License), lo cual añade otra gran ventaja [16].

Beneficios del DSBC:
Este enfoque de desarrollo además de que posibilita alcanzar un alto nivel de reutilización de software, también ofrece otros beneficios que no son menos importantes, como por ejemplo:
Simplifica las pruebas. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados [16].
Simplifica el mantenimiento del sistema. Cuando existe un débil acoplamiento entre sus componentes, el desarrollador puede actualizar y/o adicionar componentes según sea requerido, sin afectar otras partes del sistema [16].
Mayor calidad. Dado que un componente puede ser construido y luego optimizado continuamente por un experto u organización, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo [16].
De la misma manera, el hecho de comprar componentes de terceros en lugar de desarrollarlos, posee algunas ventajas:
Ciclos de desarrollo más cortos. La adición de una pieza dada de funcionalidad tomará días en lugar de tardar meses o incluso años [16].
Mejor ROI. Usando correctamente esta estrategia, el retorno sobre la inversión puede ser más favorable que desarrollando los componentes uno mismo [16].
Funcionalidad mejorada. Para usar un componente que contenga una pieza de funcionalidad, solo se necesita entender su naturaleza, más no sus detalles internos. Así, una funcionalidad que sería impráctica de implementar en la empresa, se vuelve ahora completamente asequible [16].

Sistema Abierto.- Un sistema es abierto si es concurrente, independientemente extensible, y permite a componentes heterogéneos ingresar o abandonar el sistema de

forma dinámica. Estas condiciones implican que los sistemas abiertos son inherente mente evolutivos, y que la vida de sus componentes es más corta que la del propio sistema[17].

Composición tardía.- Composición que se realiza en un tiempo posterior al de la compilación del componente, como puede ser durante su enlazado, carga o ejecución, y por alguien ajeno a su desarrollo, es decir, que sólo conoce al componente por su interfaz o contrato, sin necesidad de conocer detalles de implementación, ni la forma en que fue creado [17].

Eventos.- Mecanismo de comunicación por el que se pueden propagar las situaciones que ocurren en un sistema de forma asíncrona. Emitidos para avisar a los componentes de su entorno de cambios en su estado [17].

Reutilización.- Habilidad de un componente software de ser utilizado en contextos distintos a aquellos para los que fue diseñado (reutilizar no significa usar más de una vez) [17].

Contratos.- Especificación que se añade a la interfaz de un componente y que establece las condiciones de uso e implementación que ligan a los clientes y proveedores del componente. Los contratos cubren aspectos tanto funcionales (semántica de las operaciones de la interfaz) como no funcionales (calidad de servicio, prestaciones, fiabilidad o seguridad)[17].

Polimorfismo.- Habilidad de un mismo componente de mostrarse de diferentes formas, dependiendo del contexto; o bien la capacidad de distintos componentes de mostrar un mismo comportamiento en un contexto dado. En POC muestra tres nuevas posibilidades: La posibilidad de reemplazar (Inclusión), o capacidad de un componente de reemplazar a otro en una aplicación, sin romper los contratos con sus clientes. El polimorfismo paramétrico, o implementación genérica de un componente. Este no se resuelve en tiempo de compilación (generando la típica explosión de código) sino en tiempo de ejecución. Por último, el polimorfismo acotado, para indicar restricciones sobre los tipos sobre los que se puede parametrizar un componente[17].

Seguridad.- Garantía que debe ofrecer un componente de respetar sus interfaces y contratos. -Seguridad a nivel de tipos: Asegura que las invocaciones usen los parámetros adecuados (o súper tipos) y que los valores que devuelven son del tipo adecuado (o subtipos). -Seguridad a nivel de módulo: Asegura que solo se utilizan los servicios ajenos al componente que se han declarado. Reflexión: Habilidad para conocer y modificar el estado de un componente [17].

Frameworks.- Un framework es una estructura conceptual y tecnológica compuesta por módulos de software concretos, que sirven de base para organizar y desarrollar nuevos proyectos de software. Los Frameworks incluyen soporte para programas, bibliotecas y distintos programas para ayudar a desarrollar y unir los distintos componentes de un proyecto [17].

Arquitectura de software.- Entendemos por Arquitectura Software la representación de alto nivel de la estructura de un sistema o aplicación, que describe las partes que la integran, las interacciones entre ellas, los patrones que supervisan su composición, y las restricciones a la hora de aplicar esos patrones.
En general, la arquitectura software nace como una herramienta de alto nivel para cubrir distintos objetivos:
1. Comprender y manejar la estructura de las aplicaciones complejas.
2. Reutilizar dicha estructura (o partes de ella) para resolver problemas similares.
3. Planificar la evolución de la aplicación, identificando sus partes mutables e inmutables, así como los costes de los posibles cambios.
4. Analizar la corrección de la aplicación, y su grado de cumplimiento respecto a los requisitos iniciales
5. Permitir el estudio de alguna propiedad específica del dominio.

Programación Orientada a Componentes (POC).- La POC nace con el objetivo de construir un mercado global de componentes software, cuyos usuarios son los propios desarrolladores de aplicaciones que necesitan reutilizar componentes ya hechos y probados para construir sus aplicaciones de forma más rápida y robusta [15].

Componente.- Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que

ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio[15].

Tecnologías de componentes estandarizadas -CORBA (Common Object Request Broker Architecture — arquitectura común de intermediarios en peticiones a objetos): Es un estándar que establece una plataforma de desarrollo de sistemas distribuidos facilitando la invocación de métodos remotos bajo un paradigma orientado a objetos.
-DCOM (Distributed Component Object Model- Modelo de Objetos de Componentes Distribuidos): Es una tecnología propietaria de Microsoft para desarrollar componentes software distribuidos sobre varios ordenadores y que se comunican entre sí.
-.NET es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones.
-Enterprise JavaBeans: Modelo de componentes basado en arquitectura cliente servidor. Esta plataforma ofrece una solución multiplataforma, de fácil reutilización, integración universal con otros componentes además de la máquina virtual de java. La tecnología java ha estado a la vanguardia del DSBC y constituye una referencia clave en este tema.
-La programación de aplicaciones distribuidas se basa en un conjunto de servicios que proporcionan a los componentes el acceso a los recursos compartidos de una forma segura y eficiente. Estos servicios suelen englobarse en las siguientes categorías básicas:

- Comunicaciones Remotas
- Servicios de Directorio
- Seguridad
- Transacciones
- Gestión [15].

Bonita BPM Es una solución de código abierto para la Gestión de Procesos de Negocio permite conectar personas (actores), sistemas y tareas diarias utilizando un conjunto de aplicaciones BPM. Tiene un avanzado modelador BPMN 2.0, permite la implementación de varios conectores y utiliza un motor Java ultra escalable[18].

Edición Community Esta versión fue creada para el desarrollador que desea adquirir la competencia profesional para crear potentes aplicaciones de procesos de negocio [18].

Suscriptores de las ediciones de Bonita BPM BonitaSoft ha creado diferentes versiones de su plataforma para equipos de desarrolladores que colaboran para resolver problemas técnicos o de negocio mejorando sus aplicaciones de procesos de negocio [18].

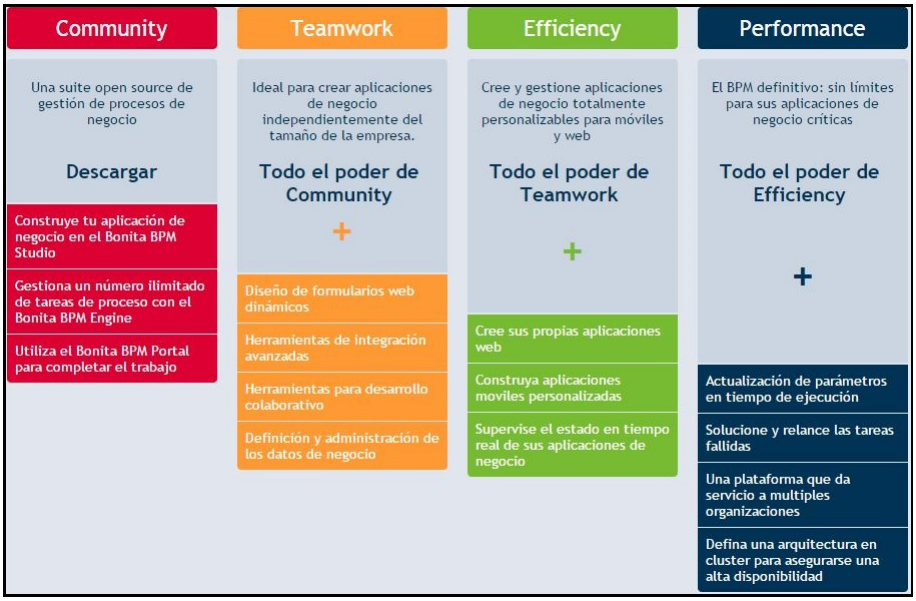


Figura 2.1: Suscripciones Bonita BPM
Fuente: "BonitaSoft", http://es.bonitasoft.com/

Comparación de ediciones:

Monitorización y elaboración de informes Controla la evolución de sus procesos y crear informes de Business Intelligence [18].

Figura 2.2: Monitorización y elaboración de informes
Fuente: "BonitaSoft", http://es.bonitasoft.com/

Simulación y optimización Utiliza datos de los procesos para establecer los ajustes necesarios para optimizar su eficiencia. Antes de implementar los cambios, se puede construir y probar modelos de proceso alternativos [18].

Figura 2.3: Simulación y optimización
Fuente: "BonitaSoft", http://es.bonitasoft.com/

Test, ejecución e implementación La nueva arquitectura basada en servicios mejora el rendimiento de los procesos y permite afrontar procesos todavía más grandes y complejos [18].

Figura 2.4: Test, ejecución e implementación
Fuente: "BonitaSoft", http://es.bonitasoft.com/

Modelado y desarrollo Diseña fácilmente un diagrama de proceso utilizando elementos gráficos de tipo "arrastrar y soltar" en BPMN2. Los diseñadores de los procesos pueden incluso empezar con un diagrama simple y posteriormente colaborar con los desarrolladores para ampliar y completar el proceso de negocio [18].

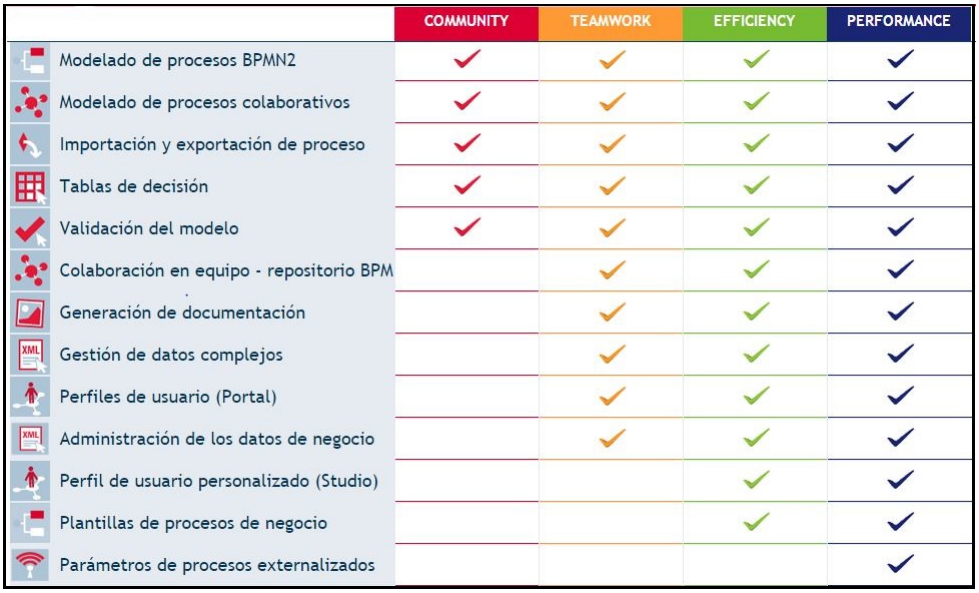


Figura 2.5: Modelado y Desarrollo
Fuente: "BonitaSoft", http://es.bonitasoft.com/

Conexión con sistemas de información Conecta procesos con bases de datos y otros sistemas de información ya existentes. Dispone de asistentes fáciles de utilizar para actualizar sistemas externos, recuperar información y agregar el retorno al proceso [18].

Figura 2.6: Conexión con sistemas de información
Fuente: "BonitaSoft", http://es.bonitasoft.com/

Diseño de la organización y actores implicados Define la estructura organizacional de las personas involucradas en gestionar y realizar los procesos [18].

Figura 2.7: Diseño de la organización y actores implicados
Fuente: "BonitaSoft", http://es.bonitasoft.com/

Diseño de formularios y aplicaciones Crea formularios web para el usuario final mediante widgets de tipo "arrastrar y soltar". Define el manejo de los datos, la validación de la entrada de datos, etc. [18].

Figura 2.8: Diseño de formularios y aplicaciones
Fuente: "BonitaSoft", http://es.bonitasoft.com/

Aplicaciones en producción Una vez desarrolladas, las aplicaciones despliega de manera remota en su servidor de aplicaciones y mostrar los formularios a los usuarios finales [18].

Figura 2.9: Aplicaciones en producción
Fuente: "BonitaSoft", http://es.bonitasoft.com/

2.3. Propuesta de Solución

Con la realización de este proyecto se logrará desarrollar e integrar nuevos componentes para la plataforma Bonita BPM que beneficien y satisfagan los requerimientos del proyecto SEGIC, además se deja abierta la posibilidad para futuras integraciones de nuevos componentes en base a la metodología propuesta, según los requerimientos vayan surgiendo.

CAPÍTULO 3

Metodología

3.1. Modalidad Básica de la investigación

Investigación aplicada
Se realizara una investigación aplicada ya que se empleará y aplicará todo lo estudiado durante la carrera para alcanzar a satisfacer las necesidades del desarrollo del presente proyecto.
Investigación Bibliográfica-Documental
Se realizará una investigación bibliográfica - documental para poder recopilar información sobre el desarrollo basado en componentes que servirá de apoyo para la realización del proyecto.

3.2. Recolección de información

La información que se necesitará para la realización de este proyecto se obtendrá mediante la biblioteca virtual y los repositorios de la Universidad Técnica de Ambato, además se necesitará datos sobre la plataforma Bonita BPM que se la obtendrá mediante el repositorio de la organización desarrolladora de la misma, por otra parte se necesitará información y detalles de los componentes a desarrollar para el presente proyecto que será detallada por los coordinadores del proyecto SEGIC de la UODIDE.

3.3. Procesamiento y análisis de datos

Lo primero que se realizará es recopilar la información sobre la metodología a emplearse, del mismo modo se analizará los componentes a desarrollarse, seguidamente se procederá a investigar el funcionamiento de la plataforma Bonita BPM, para así poder cumplir con los objetivos planteados anteriormente.

3.4. Desarrollo del Proyecto

- Descripción de los componentes.
- Búsqueda de información sobre la metodología.
- Descripción de la metodología.
- Obtención del código fuente de la Plataforma Bonita BPM.
- Análisis del funcionamiento de los componentes de la plataforma Bonita BPM.
- Desarrollo de los componentes propuestos.
- Integración de los componentes en la plataforma.
- Análisis del modelo BPM del SEGIC.
- Rediseño del modelo con los nuevos componentes.
- Evaluación del funcionamiento de desempeño.

CAPÍTULO 4

Desarrollo de la propuesta

4.1. Descripción de los componentes.

Mediante entrevista realizada a los coordinadores del proyecto SEGIC del Departamento de Investigación de la UTA, se propuso el desarrollo de componentes para la plataforma Bonita BPM los cuales deben cumplir ciertos requerimientos para mejorar la gestión de documentos en la plataforma.
Los componentes a desarrollar son:

- MultiFilesUpload: Se encargará de la subida de múltiples archivos hacia el servidor de aplicaciones.
- ViewUploadFiles: Se encargará de la visualización de los archivos subidos.

Los componentes deben cumplir las siguientes características:

- Interfaz intuitiva.
- Transparencia al usuario final.
- Mejoras con respecto al componente original de Bonita Studio.
- Soporte para JBoss5.10GA de Bonita BPM.
- Escalable.
- Solo admitirá documentos de tipo pdf y/o rar/zip.
- El tamaño de cada documento no deberá exceder los 20MB.

4.2. Búsqueda de información sobre la metodología.

Mediante investigación y consultas en los repositorios de la Universidad Técnica de Ambato y demás, se ha encontrado los siguientes tipos de desarrollo de componentes para desarrollar el proyecto:

- Desarrollo de componentes impulsados por la arquitectura.- Este tipo de desarrollo se basa en el ciclo de vida del software (tomando como referencia el modelo en cascada), trabaja en las cuatro primeras fases del ciclo (requerimientos, diseño, implementación, integración), como se aprecia en la Figura 4.1.

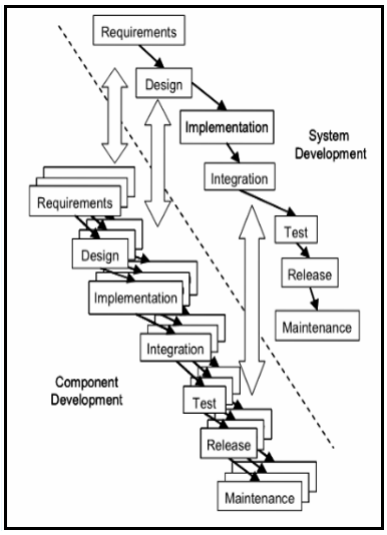


Figura 4.1: Desarrollo por componentes impulsados por la arquitectura
Fuente: "Ingeniería de Software Basada en Componentes", https://prezi.com/

- Desarrollo de línea de productos.- Este tipo de desarrollo es muy similar al desarrollo de componentes impulsados por la arquitectura con la particularidad de existir un repositorio de los componentes desarrollados como se lo puede ver en la Figura 4.2.

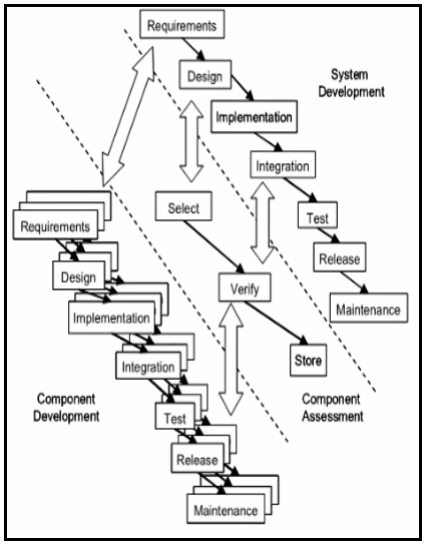


Figura 4.2: Desarrollo por línea de productos
Fuente: "Ingeniería de Software Basada en Componentes", https://prezi.com/

- Desarrollo basado en COTS(*).- Son componentes de software ya desarrollados y de índole comercial.

En la Tabla 4.1 se realizó una comparación de las diferentes características más sobresalientes que debe reunir el desarrollo de un componente, y se ha optado por el desarrollo por línea de productos.

Tipos de desarrollo de componentes			
Características	Impulsados por la arquitectura	Línea de productos	COTS
Reutilización	si	si	si
Evolutivo	si	si	no
Acceso a código	si	si	no
Confiable	si	si	no
Estandarizados	si	si	no
Independiente	si	si	si
Desplegable	si	si	si
Documentado	si	si	no
Repositorio	no	si	no

Tabla 4.1: Comparación de tipos de desarrollo de componentes.

4.3. Descripción de la metodología.

La metodología a utilizar es Component Based Development (CBD, en sus siglas en inglés). Ésta propone el desarrollo de componentes mediante la programación

orientada a componentes (POC).

Esta metodología permite la reutilización de código y es indistinta al lenguaje de programación a utilizar ya que los componentes pueden ser desarrollados en cualquier lenguaje.

La fase de requisitos es completamente corta por que se especifican los componentes y las funcionalidades individuales.
Se puede obtener una mayor calidad de productos POC porque el componente puede seguir mejorando y puede ser adaptable según las necesidades vayan creciendo.

La fase de pruebas de los componentes es corta, los componentes se prueban individualmente y no interfieren con el resto de componentes de la plataforma.

El mantenimiento del sistema se simplifica, algunos componentes cambian con el paso del tiempo pero no todos.
La fase de desarrollo se vuelve más rápida ya que los componentes son individuales y se puede compartimentar el trabajo de desarrollo.

4.4. Descarga del código fuente de la Plataforma Bonita BPM

El código fuente de la plataforma Bonita BPM se encuentra en el siguiente repositorio online de GitHub: `https://github.com/Bonitasoft-Community/Build-Bonita-BPM`, aquí se encuentra los scripts para construir las versiones de Bonita BPM 6.2.x , 6.3.6 y 6.3.8 respectivamente como se puede observar en la Figura 4.3.

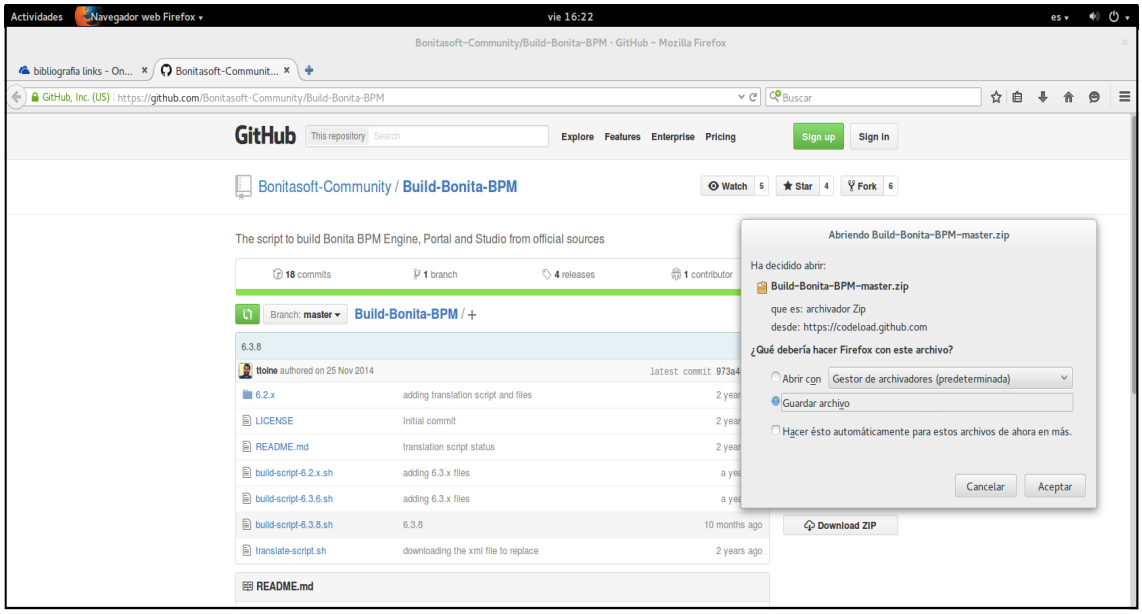


Figura 4.3: Repositorio online BonitaSoft-Community

En el archivo README.md del repositorio se observa el ambiente en el que ha sido construido Bonita BPM Figura 4.4, para construir el entorno de Bonita BPM, se necesita ciertas dependencias las cuales se tendrán que instalar posteriormente.

README.md

Build-Bonita-BPM

The script to build Bonita BPM Engine, Portal and Studio from official sources

This script has been tested on Ubuntu 12.04 and 13.10, with Open JDk 7, and latest available version of Git and Maven. At the beginning, it tests that java, mvn, git, and unzip are installed.

Around 4 Gb of dependencies will be downloaded (sources, target sources archive, maven, ...). You will need a good Internet connection.

Place this script in a disk partition with more than 15 Gb free space. Of course, you can build Bonita BPM in your /home.

Then, run 'bash build-script.sh' in a terminal.

Once finished, you will find a working build of Bonita BPM in:

BonitaBPM-build/bonita-studio/all-in-one/target/BonitaBPMCommunity-6.2.1

Figura 4.4: Especificaciones de Bonita BPM
Fuente: "GitHub: Bonitasoft-Community/Build-Bonita-BPM",
`https://github.com/Bonitasoft-Community/Build-Bonita-BPM`

4.4.1. Instalación de dependencias

4.4.1.1. Instalación de Ubuntu

El primer requisito para construir el motor Bonita BPM es instalar Ubuntu, el cual está disponible para su descarga en la página oficial `https://Ubuntugnome.org/download/`. Después de realizar la descarga, se procede a instalar en una máquina virtual o en instalación limpia en el disco duro, para este caso se lo ha hecho en una máquina virtual.

La máquina virtual a utilizar es VirtualBox, igualmente está disponible en la página oficial del software `https://www.virtualbox.org/`, en la siguiente Figura 4.5 se observa el inicio de la instalación de Ubuntu, al finalizar la instalación del sistema operativo es recomendable hacer una actualización de repositorios y de sistema con los siguientes comandos:

```
sudo apt-get update -y
```

```
sudo apt-get upgrade -y
```

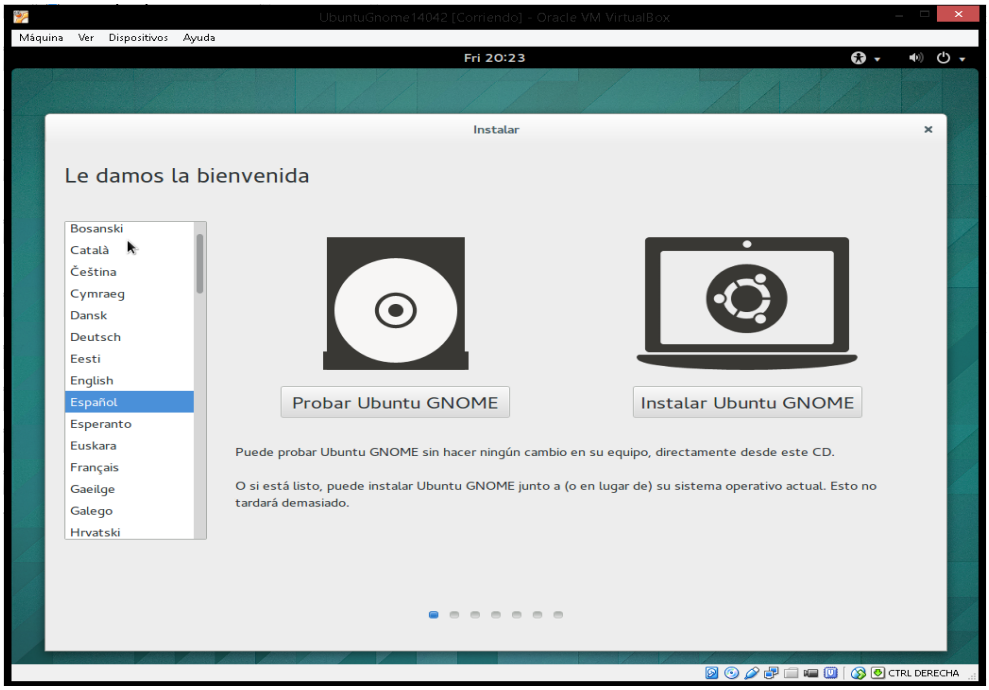


Figura 4.5: Instalación Ubuntu 14.04
Elaborado por el autor

4.4.1.2. Instalación y configuración de Java JDK

Como segundo requisito es necesario instalar Java JDK 7, en la Figura 4.6 se muestra la ejecución del siguiente comando:

```
$ sudo apt-get install openjdk-7-jdk
```

Para configurar la variable de entorno de Java, se edita el archivo *.bashrc* con el siguiente comando:

```
$ sudo vim /home/user/.bashrc
```

Al final del archivo se agrega el PATH que es la ruta en donde se encuentra instalado Java, en la Figura 4.11 se observa la configuración que se realizó para este caso.

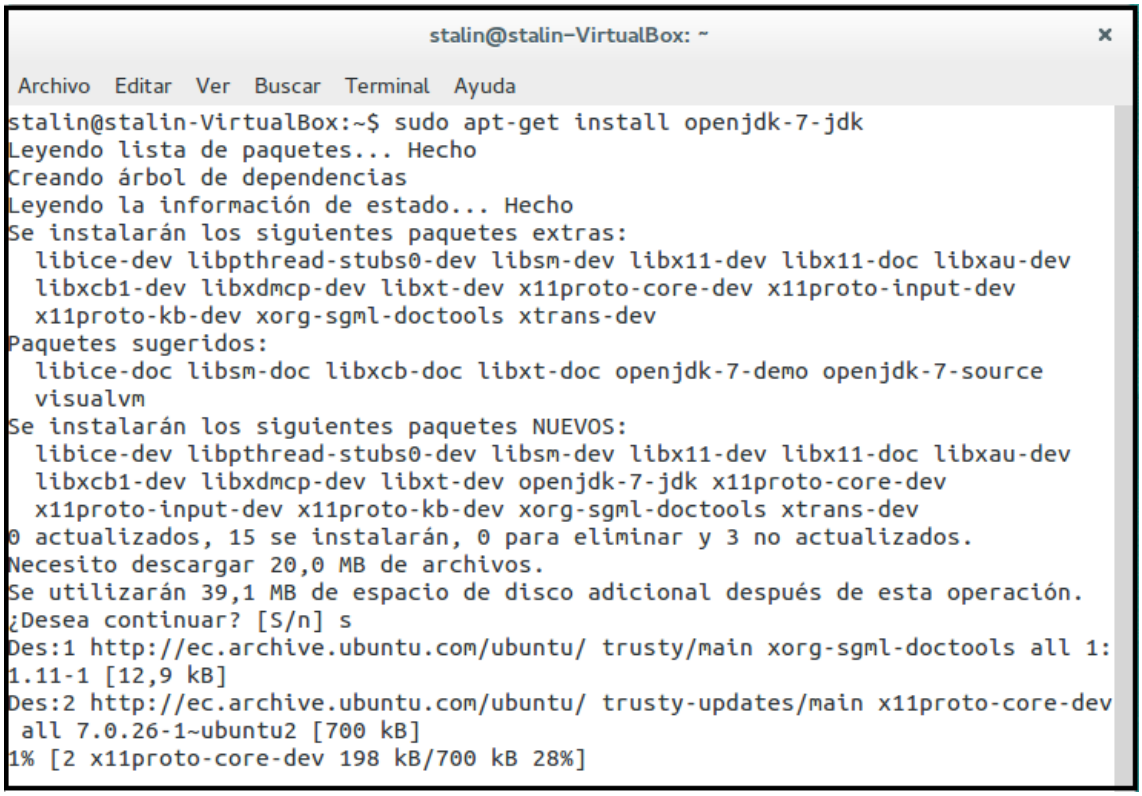


Figura 4.6: Instalación Open JDK 7
Elaborado por el autor

4.4.1.3. Instalación y configuración GitHub

Para tener acceso al repositorio de GitHub, es necesario tener una cuenta en la página oficial:`https://github.com/`, una vez creada la cuenta se procede a la instalación en el sistema operativo desde los repositorios de Ubuntu con el siguiente comando:

```
$ sudo apt-get install git
```

Así se contempla en la Figura 4.7.

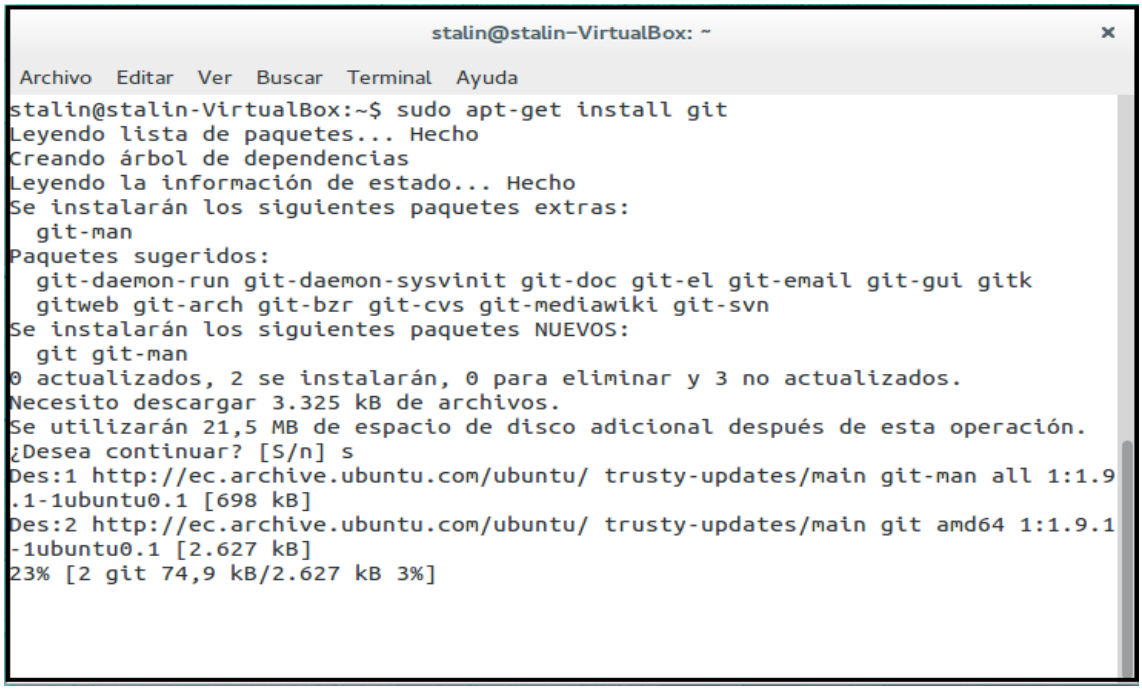


Figura 4.7: Instalación GitHub
Elaborado por el autor

Después de haber instalado Git, es importante configurar con los datos de la cuenta del usuario antes creada, aquí se especifíca el usuario y e-mail, datos que se utilizó para crear la cuenta, posteriormente generamos una clave la cual se enlaza con la cuenta de GitHub, Figura 4.8.

Figura 4.8: Configuración GitHub
Elaborado por el autor

En la opción *settings* > *SSH Keys* de nuestra cuenta GitHub se copia la clave generada en nuestro ordenador para su enlace con el mismo, agregamos un nombre y presionamos la opción *Add Key,* Figura 4.9

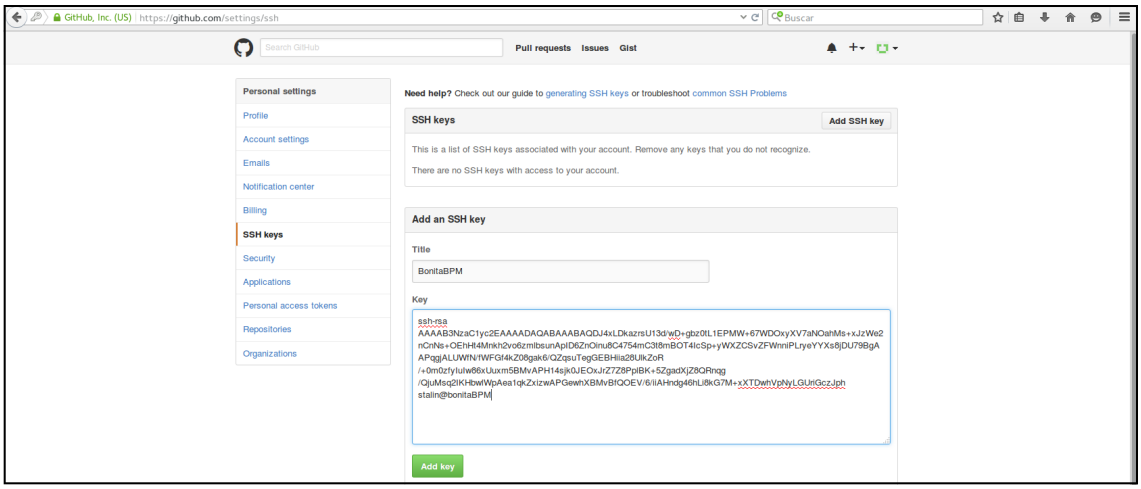


Figura 4.9: Clave Git Generada
Elaborado por el autor

4.4.1.4. Descarga y configuración de MAVEN

La descarga de Maven está disponible a través de la página oficial `http://maven.apache.org/download.cgi`, se elige la primera opción para descargar y guardamos en el computador.

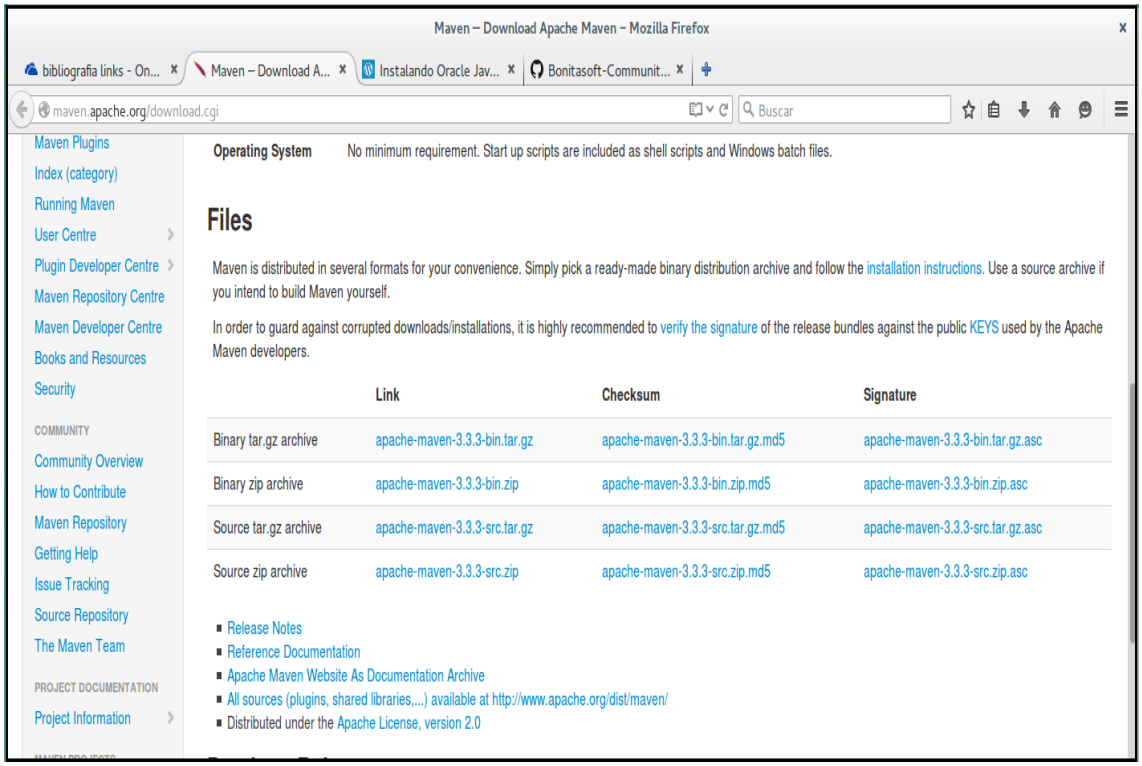


Figura 4.10: Descarga Maven
Elaborado por el autor

Para la configuración de maven se descomprime el archivo descargado en el punto anterior, después se copia en cualquier ruta del ordenador, en este caso se ha elegido */usr/local/apache-maven-3.3.3,* editamos el archivo *.bashrc* que se encuentra en */home/usuario/.bashrc,* se añaden las siguientes líneas como en la Figura 4.11. Al final es necesario reiniciar máquina para que los cambios se surtan efecto.

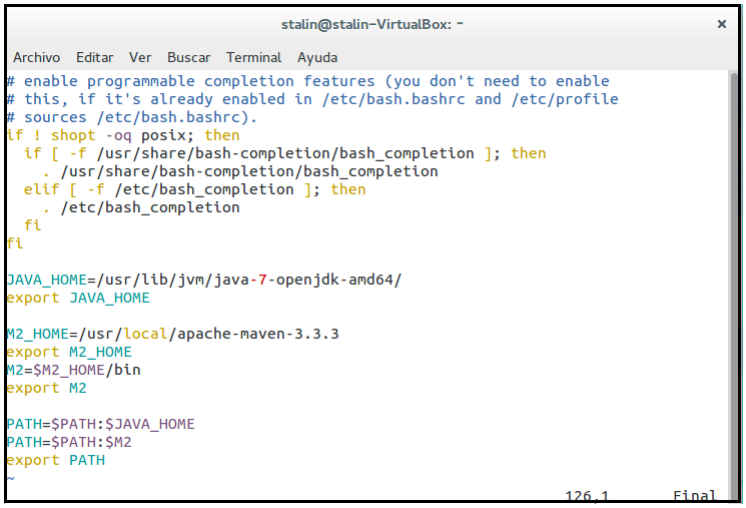


Figura 4.11: Configuración de las variables de entorno de Maven y Java
Elaborado por el autor

Luego de haber reiniciado el sistema operativo (S.O.) hay que verificar el PATH de la instalación tanto de Java como de Maven, para ello en el terminal ingresamos los siguientes comandos:

```
$ java --version
```

```
$ mvn --version
```

Si la configuración es aceptada el S.O. visualizará un mensaje con la versión de los programas instalados como se ve en la Figura 4.12.

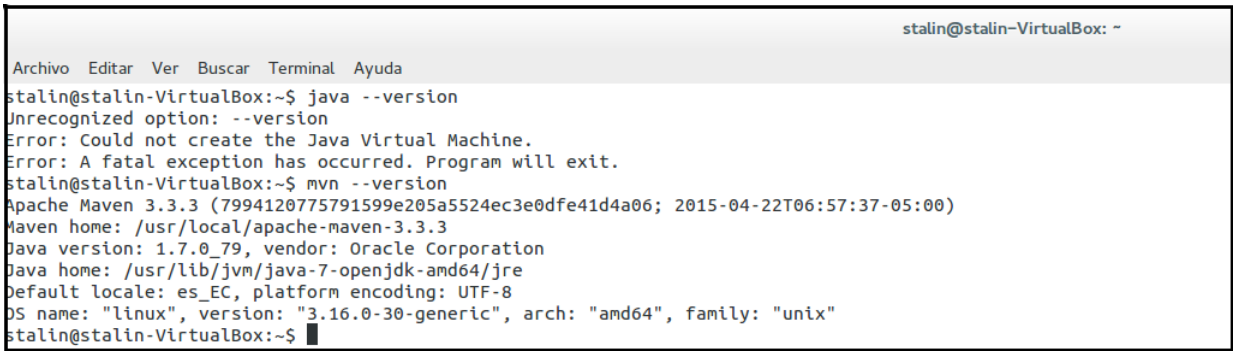


Figura 4.12: Verificación de la instalación de Java y Maven
Elaborado por el autor

4.4.1.5. Cambios en el Script de Bonita

A continuación de haber instalado y configurado todos los requisitos de Bonita BPM, creamos una carpeta en la siguiente dirección */home/user/* con el nombre de Bonita

BPM o cualquier otro. Dentro de esta carpeta se agregan los scripts descargados del repositorio GitHub, los scripts anteriores se encuentran para construir las versiones 6.2.x, 6.3.6, 6.3.8, como ya se ha hablado posteriormente. Así, modificamos el script de la versión 6.3.6 a la versión 6.3.0, ya que esta versión es en la que se desarrolla el proyecto SEGIC, en la sección Anexo A.1 se contempla el resultado final del script a compilar.

4.4.1.6. Construcción de Bonita BPM

En seguida de haber acoplado el script para la versión 6.3.0 se procede a otorgar permisos de ejecución con el siguiente comando:

```
$ sudo chmod +x bonita.sh
```

Por último se compila el script con el comando:

```
$ bash bonita.sh
```

La descarga dependerá de la velocidad del internet, son alrededor de 4Gb de código fuente. Así es necesario tener una buena conexión, cuando finalice la descarga se mostrará un mensaje de *succesfull* en el terminal. En la siguiente Figura 4.13 se muestra la compilación del script de Bonita BPM.

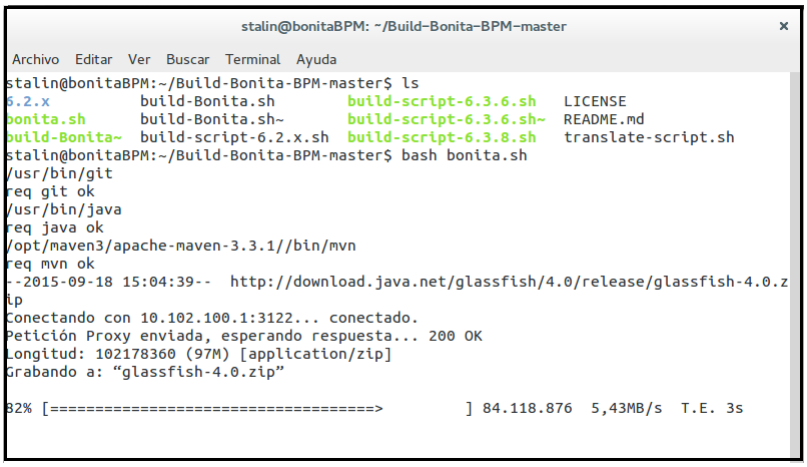


Figura 4.13: Ejecución del script de Bonita BPM
Elaborado por el autor

Al final de la descarga, dentro de la carpeta Bonita BPM-build-6.3.0 se generará todo un conjunto de carpetas que pertenecen al código fuente de Bonita BPM, Figura 4.14.

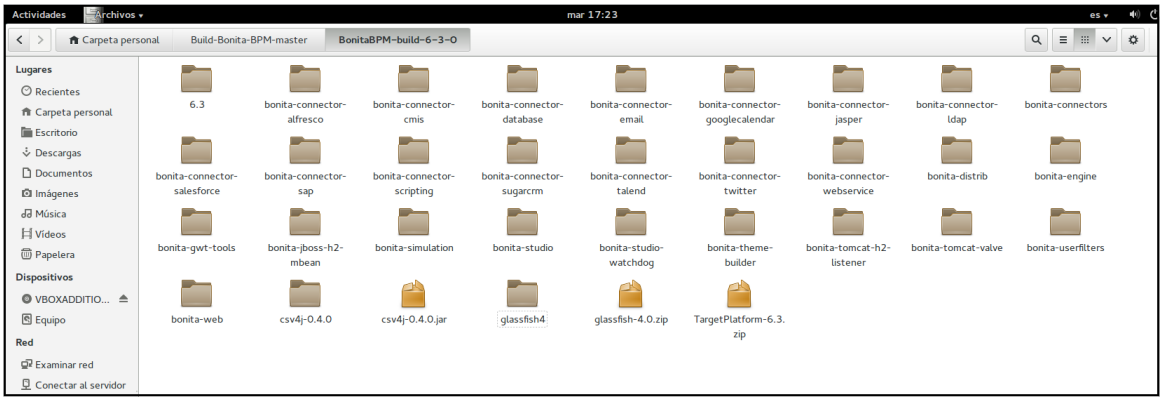


Figura 4.14: Código Fuente de Bonita BPM
Elaborado por el autor

4.5. Análisis del funcionamiento de los componentes de la plataforma Bonita BPM.

El componente *Widget File* de Bonita Studio funciona de la siguiente manera: Cuando se elige el documento del componente, llama al servlet FileUpload este realiza un copia del archivo en el servidor. El servidor devuelve como parámetro desde el servlet hacia la interfaz web la ruta del archivo, en la interfaz se oculta el componente de subir archivos y se muestra un link de descarga para el archivo desde el servidor.

Cuando ha finalizado la tarea, el archivo es guardado en dos tablas *document_mapping y document_content* pertenecientes a la base de datos de Bonita BPM, con los siguientes datos Figura 4.15, una vez finalizado el proceso, los datos del archivo de la tabla *document_mapping* pasan a la tabla *arch_document_mapping,* dejando la tabla anterior vacía.

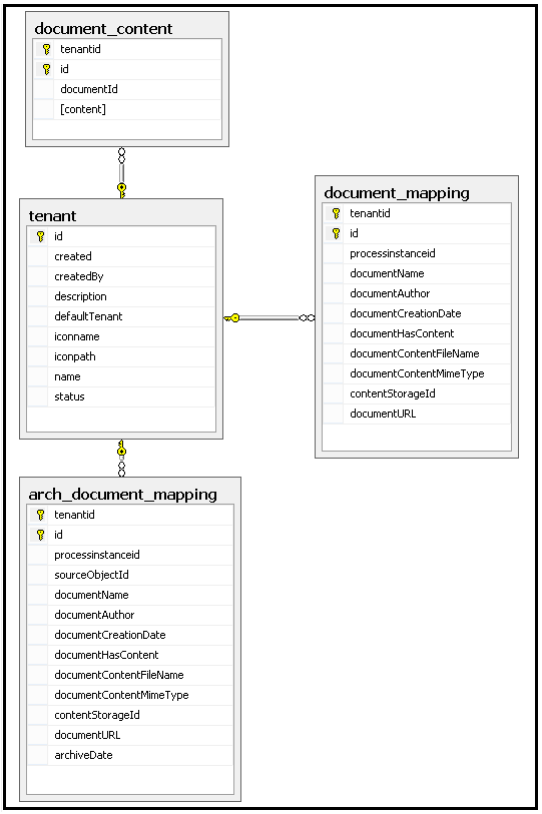


Figura 4.15: Almacenamiento de archivos en la Base de Datos de Bonita BPM.
Elaborado por el autor

Bonita BPM edición *community* es muy restrictivo al momento de programar los componentes en especial el componente para la subida de archivos, las propiedades del componente no son flexibles en la Figura 4.16 se puede observar que se puede elegir cualquier tipo de archivo y no específicos por ejemplo: *pdf, zip,rar*. Bonita BPM no trae una opción para seleccionar que tipos de archivos sean visibles, Figura 4.16.

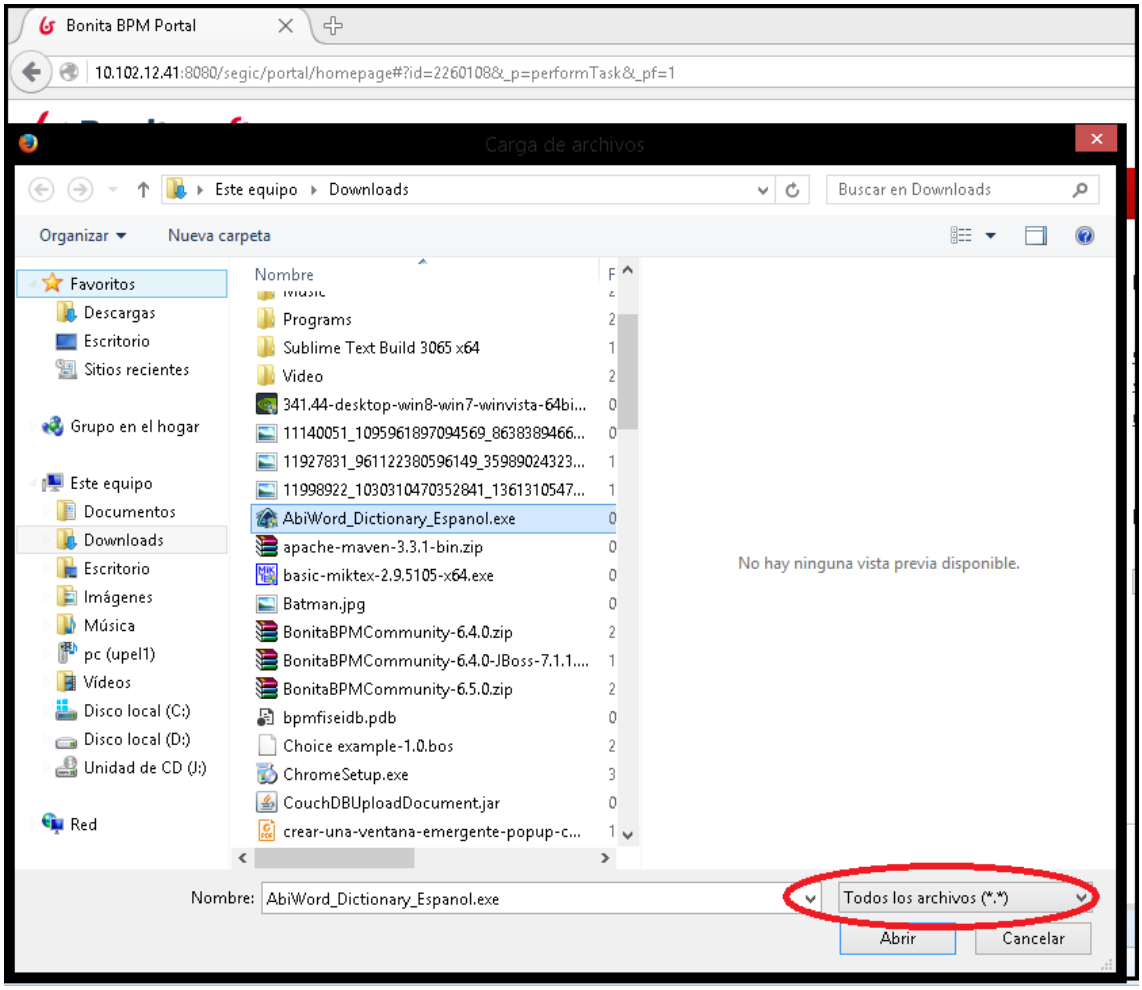
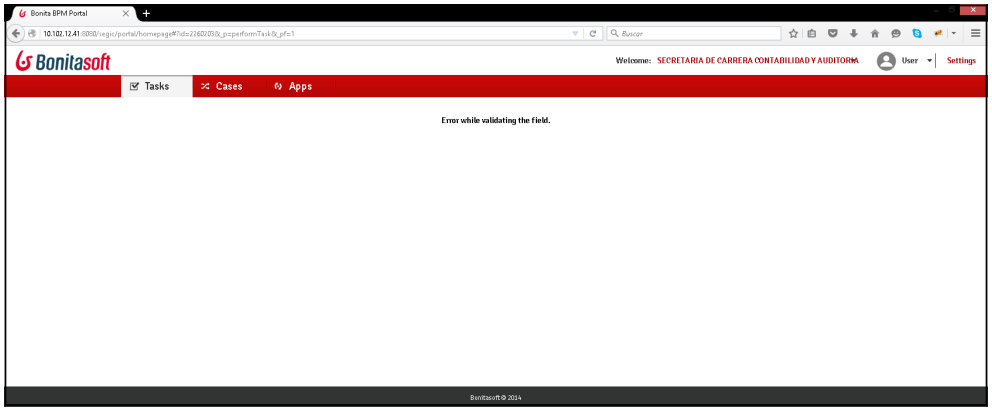
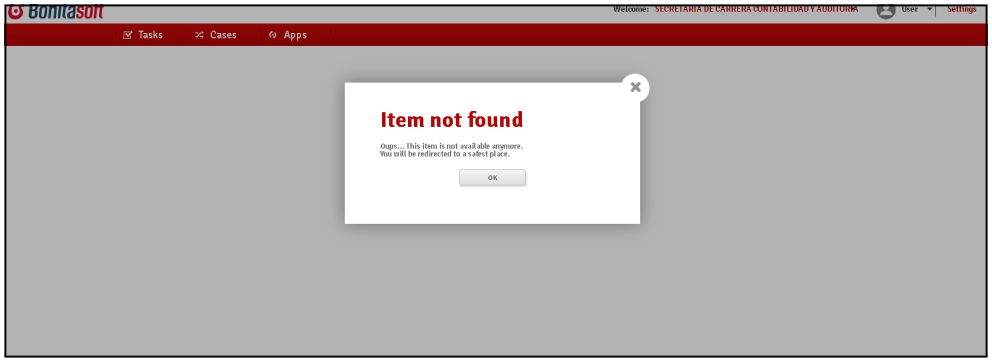


Figura 4.16: Selección de archivos en Bonita BPM
Elaborado por el autor

En la Figura 4.17 se muestran errores comunes al sobrepasar el tamaño del archivo configurado previamente en el servidor de aplicaciones, estos errores se vuelve incontrolables ya que en el componente no trae alguna opción para controlar el tamaño, ni mucho menos se puede programar en GROOVY [19]un método que lo realice con éxito.



(a) Error 1: Componente Files Bonita



(b) Error 2: Componente Files Bonita

Figura 4.17: Errores comunes de Bonita BPM en el manejo de archivos
Elaborado por el autor

Otro inconveniente que se encontró fue que para validar un tipo de archivo Bonita BPM primero carga el archivo al servidor para posteriormente hacer la validación de la extensión, este proceso de sistema se realiza con mucha demora cuando el archivo es muy pesado por lo que tarda en validar. Figura 4.18.

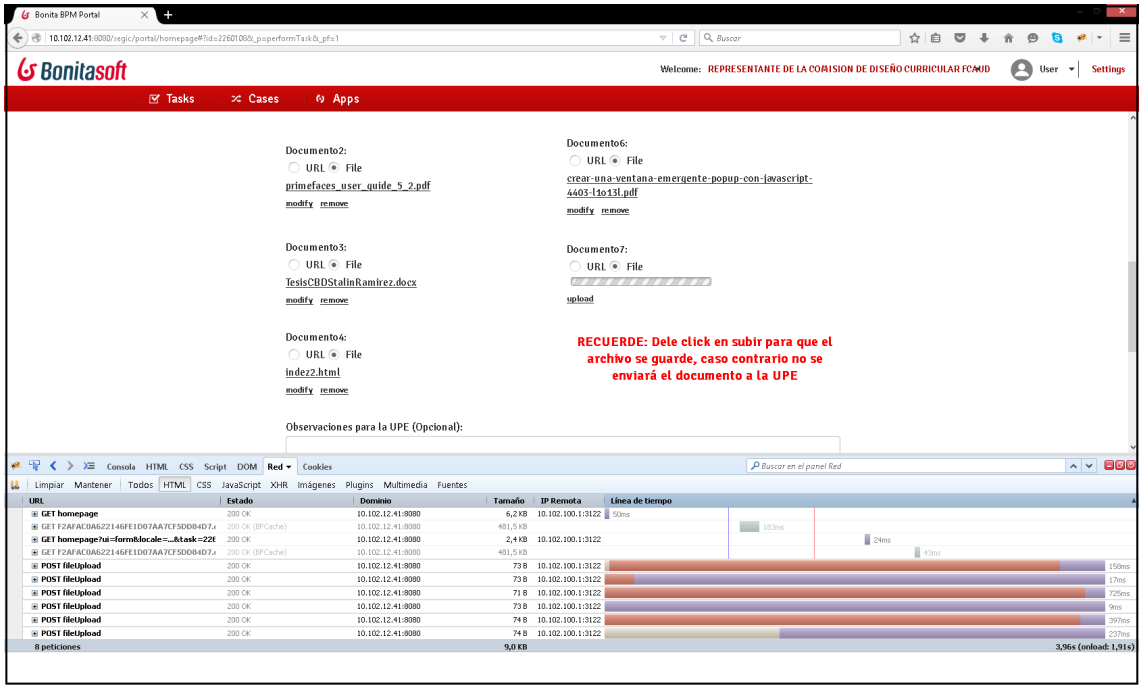


Figura 4.18: Validación en Bonita BPM
Elaborado por el autor

4.6. Desarrollo de los componentes propuestos

Los componentes que se a propuesto desarrollar y ser integrados en la plataforma Bonita BPM serán denominados MultiFilesUpload y ViewFilesUpload. Estos componentes deberán resolver el problema de peticiones múltiples de archivos y su correspondiente visualización. Además, el componente MultiFilesUpload debe soportar la carga de n archivos independientes de formato pdf. Su tamaño no debe sobrepasar los 20 MB. El componente ViewFilesUpload debe tener la funcionalidad de descargar todos los archivos simultaneamente dentro de un paquete zip.

4.6.1. Especificación de los componentes

Tomando en cuenta que la metodología CBSE propone un desarrollo independiente de la plataforma y del lenguaje de programación [20], por consecuencia los componentes se desarrollaron de manera externa e independiente a la plataforma Bonita BPM.

Anteriormente se realizó un análisis para verificar cómo Bonita BPM gestiona los archivos y observar cómo interactúan los datos con el servidor de aplicaciones,

interfaz web y BD, según este análisis se procedió a desarrollar los componentes propuestos y con las siguientes herramientas:
Lenguaje de programación: Java, JavaScript, Groovy, HTML.
Tecnologías: Servlet, Ajax, Jsp.
IDE: Eclipse.
Versión de JDK: 7.
Librerías alternas: jquery-1.11.3.min.js, AjaxFileUpload.js.
Servidor de Aplicaciones: JBoss5.10GA.

4.6.1.1. Modelo Ontológico

Como se puede observar en la Figura 4.19 (a), representa el modelo Ontológico del componente MultiFilesUploady la Figura 4.19 (b) representa el modelo ontológico del componente ViewFilesUploadrespectivamente, Para definir los modelos ontológicos se ha utilizado una notación extendida de los casos de uso que nos permite apreciar una acción abstracta a partir de la cual se inicia el proceso de refinamiento del componente y la interacción con el usuario.

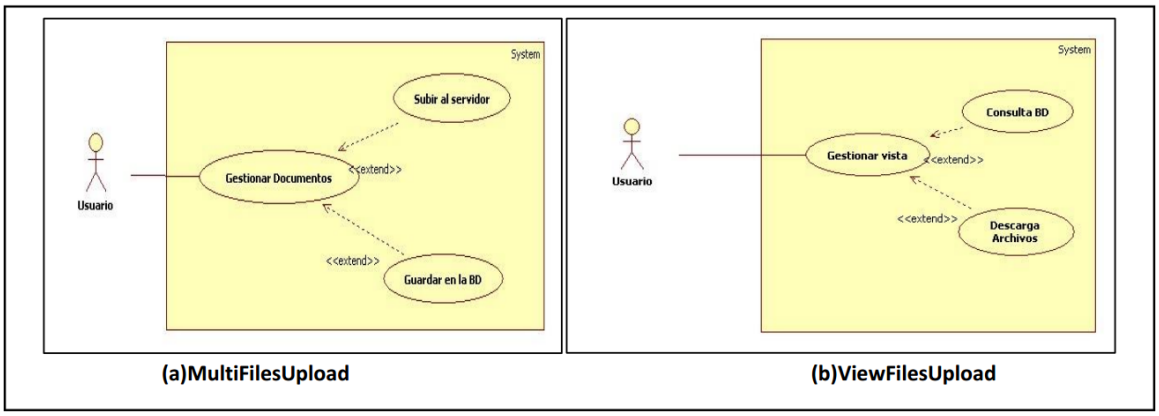


Figura 4.19: Modelo Ontológico
Elaborado por el autor

4.6.1.2. Modelo de Colaboraciones

La dinámica de los componentes se describe a través del modelo de colaboraciones, en la Figura 4.20 (a) y (b), se puede apreciar los diagramas de secuencia de los componentes MultiFilesUploady ViewFilesUploadrespectivamente. Para definir el modelo de colaboraciones se ha utilizado un diagrama de secuencias. En primer lugar la Figura 4.20 (a) representa el escenario en el que el usuario inicia la petición

de subir las evidencias hacia el servidor desde la plataforma BPM. Éste encapsula los datos para posteriormente enviarlos hacia el componente, este los gestiona mediante las instancias (servlets) y permite una vista al usuario de la evidencia/archivo subido, finalmente se guardaran todos los archivos en la BD de la plataforma Bonita BPM. En la Figura 4.20 (b) se observa el escenario del componente ViewUploadFiles, en cual el usuario desde la plataforma BPM solicita ver todos los documentos subidos, este envía los parámetros necesarios al componente, realiza la consulta a la BD y muestra todos los documentos almacenados en la plataforma permitiendo descargarlos.

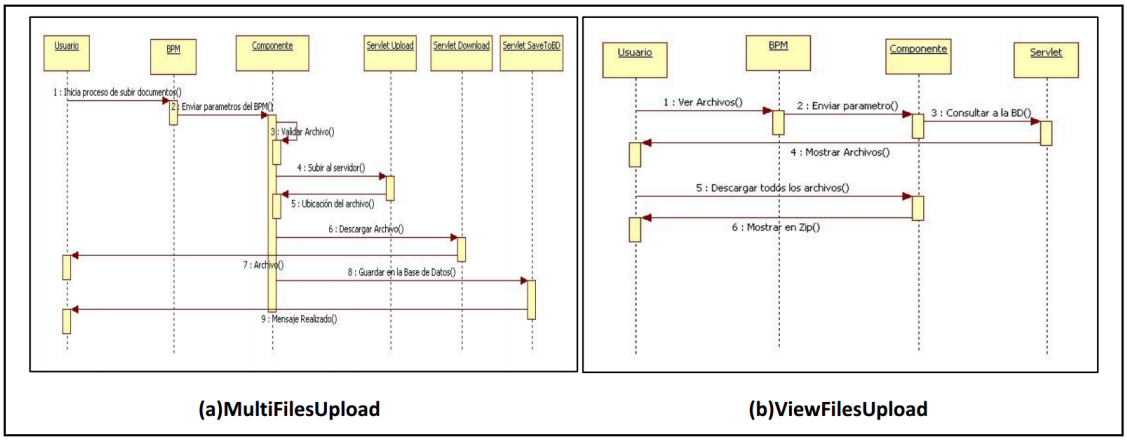


Figura 4.20: Modelo de Colaboraciones
Elaborado por el autor

4.6.2. Desarrollo del componente MultiFilesUpload

4.6.2.1. Diseño y Programación del Front-End

4.6.2.2. Hojas de Estilo (CSS)

El siguiente código *CSS* se aplica en la interfaz del componente MultiFilesUpload, éstos estilos son similares a los estilos de Bonita BPM. Todo esto para que el componente tenga una apariencia similar a la plataforma.

Estas hojas de estilo están programadas en archivos independientes separando la programación HTML y Javascript, estas hojas de estilo permiten cambiar: colores, fondos, márgenes, bordes y tipos de letra empleados en el desarrollo de la interfaz del componente, permitiendo así controlar el estilo y formato de las mismas.

```
1  .cls_add_btn, .cls_save_btn, .cls_download_link_Zip{
2   -moz-user-select: none;
3   background: rgba(0, 0, 0, 0)
```

```
4     linear-gradient(to bottom, #3d3d3d 0%, #272727 100%) repeat scroll 0 0;
5    border: 1px solid #b9b9b9;
6    border-radius: 3px;
7    box-shadow: 2px 2px 2px #d8d8d8;
8    color: #ffffff;
9    cursor: pointer;
10   display: inline-block;
11   float: left;
12   font-family: "Signika-Semibold";
13   font-size: 75%;
14   line-height: 25px !important;
15   min-width: 90px;
16   opacity: 1;
17   outline-color: #f9f9f9;
18   padding: 0 2px;
19   font-weight: bold;
20   text-align: center;
21   text-decoration: none;
22   text-shadow: 0 0 0 transparent, -1px -1px 0 transparent, 1px -1px 0
23    transparent, -1px 1px 0 transparent, 1px 1px 0 transparent;
24   text-transform: uppercase;
25   vertical-align: middle;
26   white-space: nowrap;
27  }
28  .cls_upload_btn, .remove_field {
29   color: #333333;
30   cursor: pointer;
31   font-size: 12px;
32   font-weight: bold;
33   left: 100px;
34   text-decoration: underline;
35   text-align: left;
36   float:left;
37   }
38  .cls_label, .cls_label {
39   box-sizing: border-box;
40   color: #333333;
41   font-size: 13px;
42   float: left;
43   margin-bottom: 8px;
44   margin-top: 8px;
45   overflow: hidden;
46   white-space: normal;
47  }
48
49  a.cls_download_link {
50   color: #52574D;
51   float: left;
52   display: block;
53   margin-top: 3px;
54   outline: medium none;
55   text-decoration: underline;
56  }
```

```
57
58 a.cls_download_link:visited {
59  color: Blue;
60  text-decoration: underline;
61 }
62
63 a.cls_download_link:hover {
64  color: black;
65  float: left;
66  text-decoration: underline;
67 }
68 .classLink{
69  position:relative;
70   clear:both;
71 }
72
73 // Código elaborado por el autor
```

4.6.2.3. Lenguaje de marcado HTML

El siguiente código muestra la estructura del HTML que fue empleado para crear el componente MultiUploadFiles, el componente consta de 3 partes básicas: Cabecera, Cuerpo y Pie de página.

En la cabecera se puede observar las funciones JavaScript que se han utilizado, así como las hojas de estilo (*CSS*).

El *body* está conformado por *divs* anidados para agrupar las secciones de las vistas por capas.

Dentro de la clase *bonita_form_container* se encuentra un formulario *(form)* padre el cual contendrá formularios hijos que a su vez contendrán los componentes de tipo *file*, estos serán creador dinámicamente por la función de JavaScript *appendFiles*. Además dentro de este formulario padre se encuentra dos botones *AÑADIR, GUARDAR y SALIR.*

Por último en el pie de página se hace referencia a la Unidad Educativa.

En el siguiente código se observa la estructura general del componente:

```
1 <!DOCTYPE HTML>
2 <HTML>
```

```
3  <head>
4  <link href="CSS/applicationResource.CSS" type="text/CSS" rel="stylesheet">
5  <link href="CSS/CSS2.CSS" type="text/CSS" rel="stylesheet">
6  <script src="funcionesJs/jquery-1.11.3.min.js"></script>
7  <script src="funcionesJs/appendFiles.js"></script>
8  <script src="funcionesJs/fileUploadAjax.js"></script>
9  <script src="funcionesJs/sendParams.js"></script>
10 <title>Subir Adjuntos</title>
11 </head>
12 <body class="bonita-body">
13  <div id="footerpusher">
14   <div id="main">
15    <div id="static_application">
16     <div id="bonita_process_label" class="bonita_process_label">
17      <div class="gwt-HTML">Documentos</div>
18     </div>
19     <div id="bonita-banneraltbas"></div>
20     <div id="bonita_form">
21      <div>
22       <div id="bonita_form_page_label" class="bonita_form_page_label">
23        <div class="gwt-HTML">Subir Adjuntos</div>
24       </div>
25       <div class="bonita_form_container" overflow:auto;>
26        <form  method="post"  enctype="multipart/form-data" >
27         <div id="contenido_UploadFiles">
28          <div class="cls_label">
29           Documentos <br />
30           <button class="cls_add_btn">AÑADIR</button>
31           <input class="cls_save_btn" type="button" value="GUARDAR">
32           <br />
33           <br />
34          </div>
35         </div>
36        </form>
37       </div>
38      </br>
39      </br>
40     </div>
41    </div>
42   </div>
43  </div>
44  </div>
45  <div class="footer">
46  <table>
47  <tr>
48  <td width="500%"><center><span id="footer" >Universidad Técnica de Ambato</span></
        center></td>
49  <td><a href="JavaScript:window.close()"><FONT COLOR="#FF0000">SALIR</FONT></a></td
        >
50  </tr>
51  </table>
52  </div>
53 </body>
```

```
54 </HTML>
55
56 // Código elaborado por el autor
```

La estructura del mensaje que se utilizará para mostrar al usuario cuando los documentos se hayan guardado no difiere mucho de la anterior, se utilizó las mismas hojas de estilo imprimiendo el siguiente mensaje: *"Archivos Subidos Exitosamente..."*.

```
1  <!DOCTYPE HTML>
2  <HTML>
3  <head>
4  <meta http-equiv="Content-Type" content="text/HTML; charset=ISO-8859-1">
5  <link href="CSS/applicationResource.CSS" type="text/CSS" rel="stylesheet">
6  <link href="CSS/CSS2.CSS" type="text/CSS" rel="stylesheet">
7  <title>Subir Adjuntos</title>
8  </head>
9
10 <body class="bonita-body">
11  <div id="footerpusher">
12   <div id="main">
13    <div id="static_application">
14     <div id="bonita_process_label" class="bonita_process_label">
15      <div class="gwt-HTML">Documentos</div>
16     </div>
17     <div id="bonita-banneraltbas"></div>
18     <div id="bonita_form">
19      <div>
20       <div id="bonita_form_page_label" class="bonita_form_page_label">
21        <div class="gwt-HTML">Archivos Subidos Exitosamente..</div>
22       </div>
23      </div>
24     </div>
25    </div>
26   </div>
27  </div>
28
29  <div class="footer">
30   <span id="footer">Universidad Tecnica de Ambato <a
31    href="JavaScript:window.close()">Salir</a></span>
32  </div>
33 </body>
34 </HTML>
35
36 // Código elaborado por el autor
```

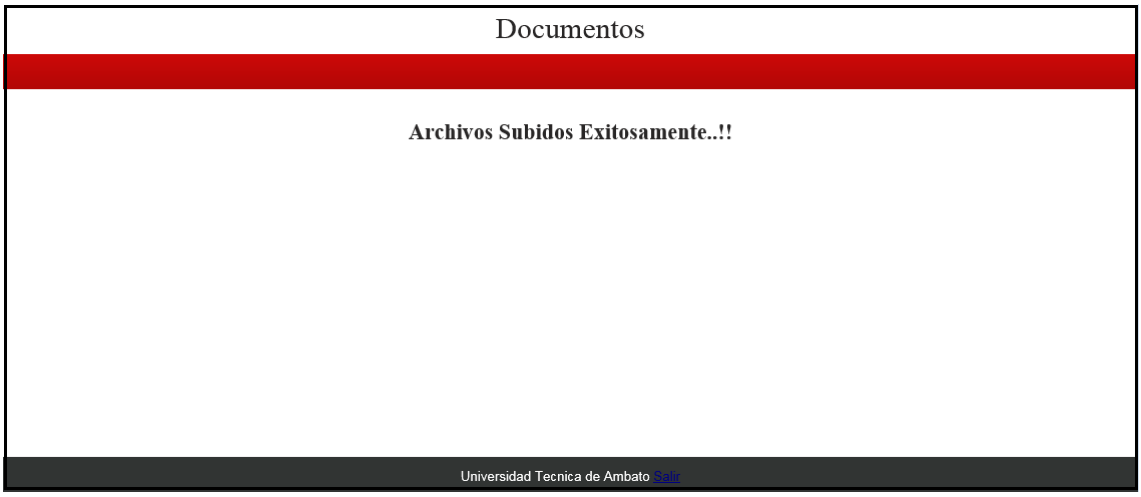


Figura 4.21: Mensaje de confirmación
Elaborado por el autor

4.6.2.4. JavaScript

Con la ayuda de código JavaScript se añaden todos los formularios hijos que contendrán los componentes para la carga y descarga de archivos.

La estructura de los formularios hijos es la siguiente:
Una campo de tipo tabla contiene el formulario hijo y este a su vez contiene un campo de tipo file, en el formulario asignamos las propiedades *action, method, enctype.* En el componente *file* asignamos las propiedades *name, file, id, accept.*

action: Hace referencia al servidor de aplicaciones en donde está el servlet.

method: Indica la forma en que serán enviados los datos del formulario hacia el servidor en este caso se enviara mediante *POST.*

enctype: Especifíca el protocolo que se enviarán los archivos.

accept: Indica que el archivo a subir es de formato *pdf.* Así la ventana de diálogo solo aceptara tipos de dicha extensión.

Además, la tabla contiene divs adicionales que se mostrarán solo cuando el archivo se haya subido al servidor éstos *divs* son:

classLink: Muestra un link que hace referencia al archivo, que permitirá la descarga inmediatamente desde el servidor.

remove_field: Botón que removerá los componentes de tipo *files* que se hayan creado o que el usuario decida eliminarlos.

cls_upload_btn: Botón permitirá seleccionar nuevamente un nuevo archivo.

En el siguiente fragmento de código se puede observar la estructura del formulario hijo.

```
1  $(wr).append ('<TABLE id="documento" class="cls_form_entry">'+
2          '<TR>'+
3          '<TH COLSPAN="2">'+
4      '<div class="classFormHide" style="" aria-hidden="false">'+
5       '<form id="forma" action="http://localhost:8080/Archivos/UploadFiles" method="
            post" enctype="multipart/form-data">'+
6        '<input type="file" name="file" id="file" accept="application/pdf" />'+
7       '</form>'+
8      '</div>'+
9          '</TH>'+
10         '</TR>'+
11         '<TR>'+
12         '<TH COLSPAN="2">'+
13          '<div class="upload" style="display: none;">Uploading....</div>'+
14         '</TH>'+
15         '</TR>'+
16         '<TR>'+
17         '<TH COLSPAN="2">'+
18     '<div class="classLink" style="display: none;" aria-hidden="true" >'+
19      '<a class="cls_download_link"></a>'+
20     '</div>'+
21         '</TH>'+
22         '</TR>'+
23         '<TR>'+
24         '<TD >'+
25         '<a href="" class="remove_field">Remover</a>'+
26         '</TD>'+
27         '<TD>'+
28         '<div class="cls_upload_btn" title="cambiar el adjunto" style="display:
               none;" aria-hidden="true">Cambio</div>'+
29         '</TD>'+
30         '</TR>'+
31    '</TABLE>')
32
33 // Código elaborado por el autor
```

En la Figura 4.22 se pueden analizar los componentes de tipo *file* creados dinámicamente con JavaScript y Jquery.

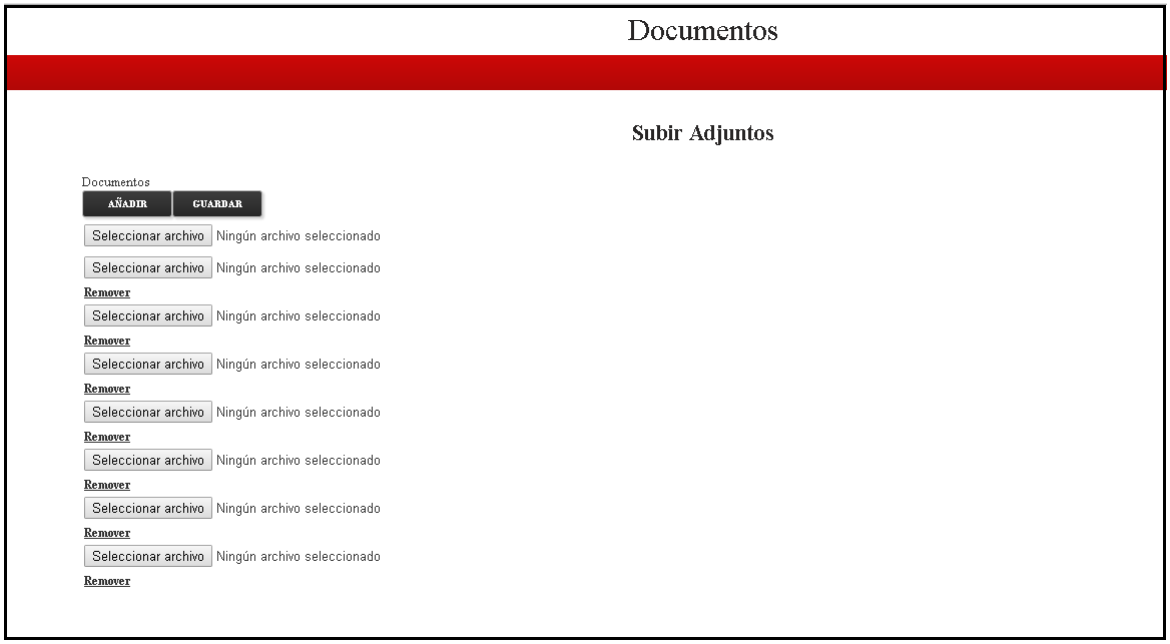


Figura 4.22: Función añadir MultiUploadFiles
Elaborado por el autor

Para validar un archivo lo hacemos del lado del cliente. Así permitiremos que el servidor tenga un mejor desempeño en la transacción de archivos.

El componente debe aceptar solo archivos de tipo *pdf* y su tamaño no debe sobrepasar las 20MB según requisitos de la sección 4.1. Estos datos son variantes que pueden cambiar a futuro dependiendo la necesidad.

Si el archivo no cumple con los requisitos, el componente mostrará una alerta en el navegador impidiendo la carga del archivo hacia el servidor como en la Figura 4.21.

```
1  $(wr).on("click",".classFormHide", function(){
2    var parentTag = null;
3    var filesValue = null;
4    var extension = null;
5    var tamBits = null;
6    var bytes = null;
7    $(this).children().children('input[type="file"]').on('change', function() {
8     try{
9      extension = (this.value.substring(this.value.lastIndexOf("."))).toLowerCase();
10     tamBits = ((this.files[0].size/1024)/1024).toFixed(3);
11     if(extension!='.pdf' || tamBits >= 20){
```

```
12        alert("ERROR: Su archivo puede contener los siguientes errores:"+
13          " \nNo es un archivo pdf o, \nSu tamaño excede las 20 MB\nNOTA:"+
14          " \nSu tamaño real es: "+ tamBits +" MB \nSu extensión real es: " +
                extension );
15        $(this).parent('#forma').get(0).reset();
16       }
17      }
18      catch(err){
19       tamBits = null;
20      }
21     });
22
23  // Código elaborado por el autor
```

Como resultado de la validación se procedió a probar con archivos que no cumplan con los requisitos establecidos dando como resultado la validación correcta como se puede observar en la siguiente Figura 4.23.

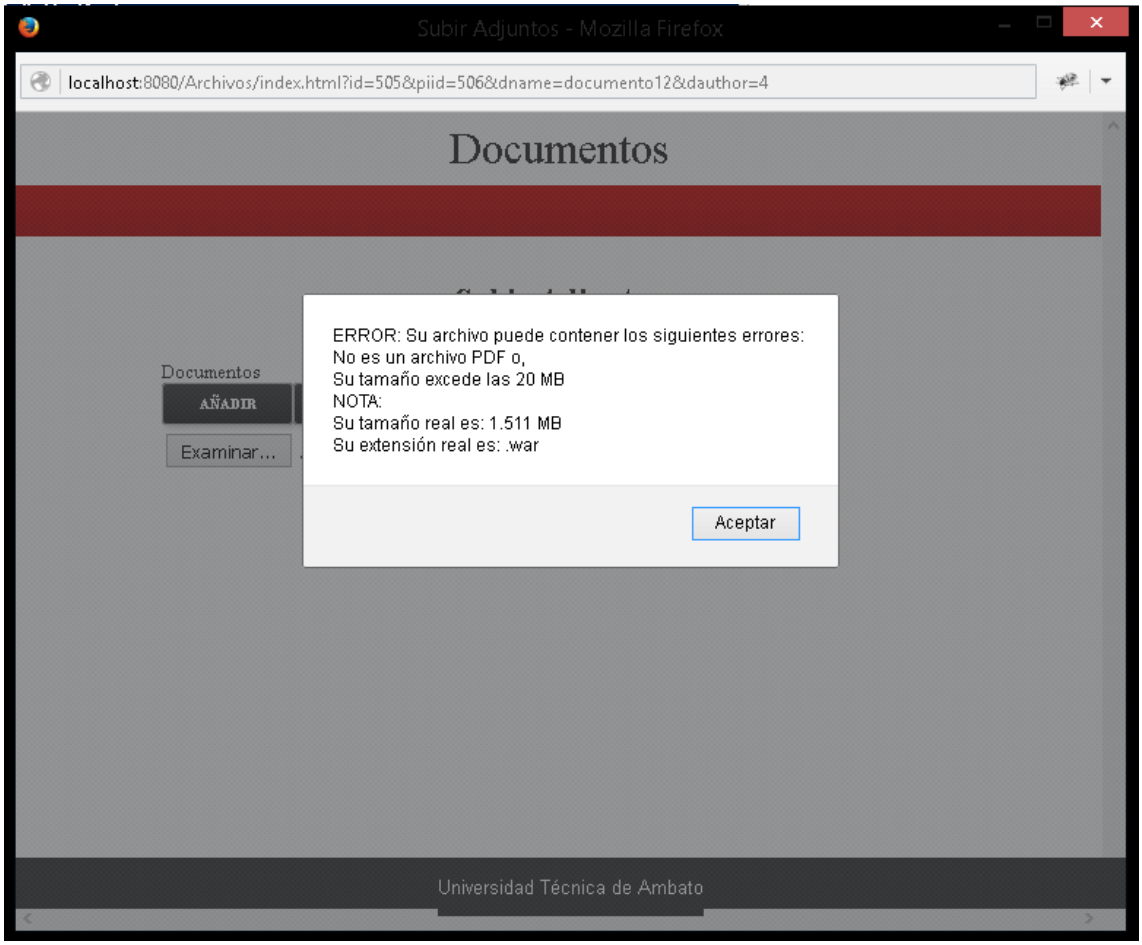


Figura 4.23: Validación mediante JavaScript
Elaborado por el autor

En el siguiente código, cuando el usuario requiera remover un componente *file* del DOM, esta función lo eliminará completamente permitiendo así enviar formularios con datos limpios al servidor de aplicaciones de Bonita BPM.

```
1  //Remover field
2   $(wr).on("click",".remove_field", function(e){
3    e.preventDefault();
4    $(this).parent().parent().parent().parent(".cls_form_entry").remove();
5    x--;
6   });
7
8  // Código elaborado por el autor
```

La siguiente función permite mostrar y ocultar elementos del DOM, es muy útil cuando el archivo ya está cargado en el servidor porque nos permitirá visualizar los campos ocultos de la tabla que pertenecen al componente como las etiquetas de *link* de descarga del archivo y la etiqueta de *cambio,* a su vez oculta el campo *form* para impedir la subida de otro archivo.

```
1    (function ( $ ) {
2     $.fn.myShow = function(){
3      return this.each(function() {
4       $(this).attr('aria-hidden', 'false').show();
5      });
6     };
7     $.fn.myHide = function(){
8      return this.each(function() {
9       $(this).attr('aria-hidden', 'true').hide();
10     });
11    };
12   }(jQuery));
13
14 // Código elaborado por el autor
```

La próxima función permite enviar datos hacia el servidor utilizando la librería *ajax-fileupload* al momento que se seleccione el archivo en la interfaz, inmediatamente se aplicaran las validaciones correspondientes y si los datos son válidos, se enviara el archivo al servidor, la librería *ajaxfileupload* permite la transacción de datos del cliente al servidor mediante ajax para hacer un *refresh* solo del formulario del que se envió el archivo.

Cuando el traspaso de datos hacia el servidor sea exitoso, éste como respuesta devolverá la ruta en donde se encuentra alojado dicho archivo.

A continuación se procede a ocultar la clase *classFormHide* que contiene el formulario de carga de archivos. Además se muestra los *divs* ocultos *classLink, cls_upload_btn, cls_download_link* en esta última etiqueta mediante Jquery se adjuntan las siguientes propiedades *href* que contiene la ruta del archivo con referencia al servelet para poder descargarlo, y *appendChild* para mostrar el nombre del archivo a descargarse.

```
1   $(this).children().children('input[type="file"]').ajaxfileupload({
2    'action': 'http://localhost:8080/Archivos/UploadFiles',
3    'onComplete': function(response) {
4     var ruta = response.substring(0 ,response.lastIndexOf('\\'));
5     $(this).parent().parent('.classFormHide').myHide();
6     $(this).parent().parent().parent().parent().siblings().siblings().children().
          children('.classLink').myShow();
7     $(this).parent().parent().parent().parent().siblings().siblings().siblings().
          children().children('.cls_upload_btn').myShow();
8     $(this).parent().parent().parent().parent().siblings().children().children('.
          upload').hide();
9     var nombre = (response.substring(response.lastIndexOf('\\')+1));
10    var link = document.createTextNode(nombre);
11    var elemento = $(this).parent().parent().parent().parent().siblings().siblings
          ().children().children().children('.cls_download_link')[0];
12    var dirServer = "http://localhost:8080/Archivos/DownloadFiles";
13    var FilePath =ruta;
14    var FileName =nombre;
15    elemento.href=dirServer+"?FilePath="+FilePath+"&FileName="+FileName;
16    elemento.appendChild(link);
17   },'onStart': function() {
18    $(this).parent().parent().parent().parent().siblings().children().children('.
          upload').show();
19   }
20  });
21
22  // Código elaborado por el autor
```

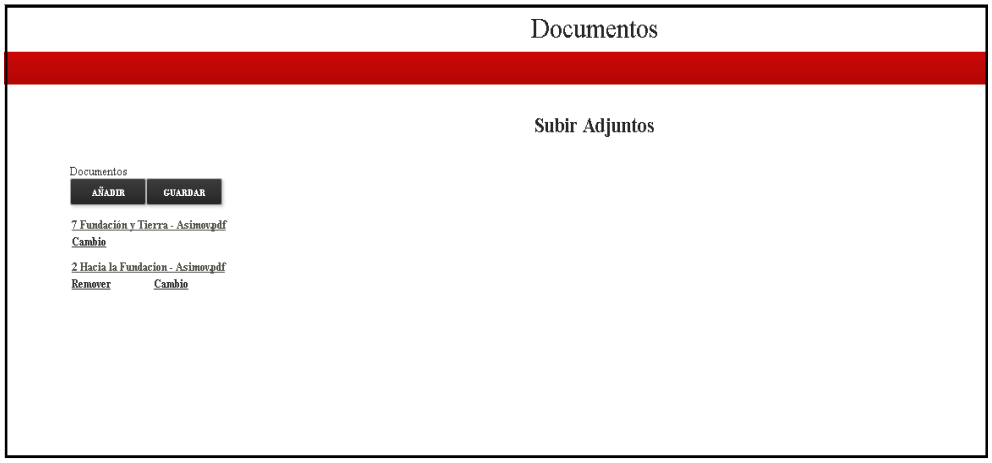


Figura 4.24: Componente MultiFilesUpload
Elaborado por el autor

Esta función se ejecuta al hacer click en la etiqueta *cambio*, formatea todos los atributos de los campos visibles y no visibles dejando nuevamente en un estado inicial para la subida de archivos.

```
1   $(this).parent().parent().siblings().siblings().siblings().children().children('.
        cls_upload_btn').click(function(i){
2    $(this).parent().parent().siblings().children().children().children('.
        cls_download_link')[0].href=null;
3    $(this).parent().parent().siblings().children().children().children('.
        cls_download_link').empty();
4    $(this).parent().parent().siblings().siblings().siblings().children().children
        ('.classFormHide').myShow();
5    $(this).parent().parent().siblings().children().children('.classLink').myShow();
6    $(this).myHide();
7    $(this).parent().parent().siblings().siblings().siblings().children().children
        ('.classFormHide').children("#forma").get(0).reset();
8   });
9
10  // Código elaborado por el autor
```

Esta función permite validar todos los archivos cargados y que estén listos para ser enviados a la base de Bonita BPM. El proceso de esta función es realizar un barrido de todos los formularios existentes verificando si existe un formulario con un *file* vacío. Además guarda en una variable los nombres y direcciones de los archivos subidos en ese instante para guardarlos y enviarlos al servidor. Cuando la respuesta sea completada exitosamente se mostrará un mensaje de finalización.

```
1  $(document).ready(function() {
2   var misVariablesGet = getVarsUrl();
3   var verif = new Array();
4   var nomFile = new Array();
5   var nomReal = new Array();
6   var aux = null;
7
8   $('.cls_save_btn').click(function(){
9    $('#contenido_UploadFiles #documento').each(function() {
10    var elementoSelect =  $(this).attr("id");
11    var table = document.getElementById("documento");
12    var componente = (table.rows[0].cells[0].childNodes[0].children[0].children[0]);
13    if(componente.files.length == 0){
14     alert("No se ha seleccionado un documento");
15     return false;
16    }else{
17     $('#contenido_UploadFiles .cls_download_link').each(function() {
18           var elemento= this;
19           nomFile.push(elemento.innerHTML);
20           aux = elemento.href;
21           nomReal.push(aux.substring(aux.lastIndexOf('\\')+1,aux.lastIndexOf('&') )
                 );
22     });
23     $.post( "http://localhost:8080/Archivos/DirFilesUpload", { nomFile:nomFile,
           nomReal:nomReal,pId:misVariablesGet.var1, pNameVarDoc:misVariablesGet.var2}
            )
24     .done(function() {
25      document.location.href="http://localhost:8080/Archivos/mensaje.HTML";
26      });
27    }
28   });
29  });
30
31  function getVarsUrl(){
32      var url= location.search.replace("?", "");
33      var arrUrl = url.split("&");
34      var urlObj={};
35      for(var i=0; i<arrUrl.length; i++){
36          var x= arrUrl[i].split("=");
37          urlObj[x[0]]=x[1]
38      }
39      return urlObj;
40  }
41 });
42
43 // Código elaborado por el autor
```

4.6.2.5. Programación del Back-End

4.6.2.6. Servlets

DownloadFiles

El siguiente fragmento de código de servlet permite la descarga del archivo seleccionado en la interfaz y que éste alojado en el servidor. Éste servlet recibe dos parámetros, el nombre del archivo y la dirección del archivo, después de verificar el archivo y leerlo, éste servlet envía como respuesta el archivo completo, produciendo como resultado la descarga del mismo.

```
1  protected void doGet(HttpServletRequest request, HttpServletResponse response)
       throws ServletException, IOException {
2    try {
3     String fileName = request.getParameter("FileName");
4     String filePath = request.getParameter("FilePath");
5     response.setContentType("APPLICATION/OCTET-STREAM");
6     response.setHeader("Content-Disposition","attachment; filename=\"" + fileName +
          "\"");
7     OutputStream out = response.getOutputStream();
8     FileInputStream in = new FileInputStream(filePath);
9     byte[] buffer = new byte[4096];
10    int length;
11    while ((length = in.read(buffer)) > 0){
12        out.write(buffer, 0, length);
13    }
14    in.close();
15    out.flush();
16   }catch(Exception e){
17    System.out.println(e);
18   }
19  }
20
21 // Código elaborado por el autor
```

UploadFiles

El siguiente servlet permite la subida de archivos al servidor, éste servlet recibe el archivo enviado desde la interfaz HTML por medio de Jquery Ajax. Del archivo recibido se obtiene el nombre y a extensión, seguidamente se genera un nuevo nombre para el archivo que tendrá un formato como el siguiente "tpm_**********.ext" siendo los asteriscos un numero generado aleatoriamente, todo esto para evitar duplicidades de un mismo nombre, a continuación se verifica la dirección en donde se almacenará en el sistema de archivos. Se guarda el archivo con el nuevo nombre generado, y como respuesta hacia la interfaz HTML se envía el nombre físico y

el nombre real del archivo, teniendo en cuenta que el nombre físico del archivo es el nombre generado por el servlet y el nombre real es el nombre verdadero del archivo.

```
1  protected void doPost(HttpServletRequest request, HttpServletResponse response)
       throws ServletException, IOException {
2    boolean isMultipart = ServletFileUpload.isMultipartContent(request);
3          if (isMultipart) {
4
5           FileItemFactory factory = new DiskFileItemFactory();
6
7           ServletFileUpload upload = new ServletFileUpload(factory);
8           List items = null;
9              try {
10              items = upload.parseRequest(request);
11              Iterator iterator = items.iterator();
12                 while (iterator.hasNext()) {
13                     FileItem item = (FileItem) iterator.next();
14                     if (!item.isFormField()) {
15
16                      //Obtengo el nombre
17                         String fileName = item.getName();
18
19                         //Obtengo la extension(.ext)
20                         String extension = fileName.substring( fileName.lastIndexOf
                               ( "." ) );
21
22        //Genero un nombre para guardar en el servidor
23                         long x = (long) (Math.random () * (1000000000000000000L -
                               1) + 1);
24                         String tpmName = ("tpm_"+ String.valueOf(x)+extension);
25
26                         //Crea la direccion si no existe
27                         File path = new File("/uploads");
28                         if (!path.exists()) {
29                          boolean status = path.mkdirs();
30                         }
31                         //Guarda con el nombre long
32                         File uploadedFile = new File(path + "/" + tpmName);
33                         item.write(uploadedFile);
34
35                         response.setContentType("text/HTML");
36                       response.setCharacterEncoding("UTF-8");
37                       response.getWriter().write(uploadedFile.getAbsolutePath()
                             +"\\"+fileName);
38                     }
39                 }
40             } catch (FileUploadException e) {
41                 e.printStackTrace();
42             } catch (Exception e) {
43                 e.printStackTrace();
44             }
```

```
45          }
46  }
47
48  // Código elaborado por el autor
```

DirFilesUpload

El siguiente servlet recibe como parámetros los nombres físicos y los nombres reales del archivo. También recibe los parámetros enviados desde la plataforma Bonita BPM.

Además, el servlet realiza la acción de guardar uno por uno cada archivo con los siguientes métodos *saveToBonitaDM, saveToBonitaDC.*

```
1  protected void doPost(HttpServletRequest request, HttpServletResponse response)
       throws ServletException, IOException {
2    String[] nomEta = request.getParameterValues("nomFile[]");
3    String[] nomReal = request.getParameterValues("nomReal[]");
4    String idBonita = request.getParameter("pid");
5    String piidBonita = request.getParameter("ppiid");
6    String dnameBonita = request.getParameter("pdname");
7    String dauthorBonita = request.getParameter("pdauthor");
8
9
10   for(String i:nomEta) {
11    System.out.println(i);
12   }
13   for(String j:nomReal) {
14    System.out.println(j);
15   }
16   System.out.println(idBonita);
17   System.out.println(piidBonita);
18   System.out.println(dnameBonita);
19   System.out.println(dauthorBonita);
20
21   for(int i=0; i<nomReal.length; i++){
22     documentFileName = nomEta[i];
23     long x = (long) (Math.random () * (1000000000000000000L - 1) + 1);
24     storageId = Long.toString(x*(-1));
25     Path path = Paths.get("C:\\uploads\\"+nomReal[i]);
26     byte[] docs = Files.readAllBytes(path);
27
28    saveToBonitaDM(Integer.parseInt(idBonita),Integer.parseInt(piidBonita),
          dnameBonita,Integer.parseInt(dauthorBonita),documentFileName,storageId);
29    saveToBonitaDC(Integer.parseInt(idBonita),docs,storageId);
30    }
31  }
32
33  // Código elaborado por el autor
```

4.6.2.7. Java

La siguiente función escrita en código Java, permite guardar los archivos subidos desde el cliente en la BD de Bonita BPM.

La función *saveToBonitaDM* recibe 6 parámetros importantes para su guardado en la tabla *arch_document_mapping* de Bonita BPM. Estos parámetros son: *id, processinstanceid, documentName, documentAuthor, documentContentFileName, contentStorageId*.

Para realizar la conexión a la BD es importante conocer el nombre del DataSource creado en el servidor de aplicaciones de Bonita BPM para referenciarlo en el componente.

```
1  public static void saveToBonitaDM(int cidBonita,int cpiidBonita,String cdnameBonita
       ,int cdauthorBonita,String cdocumentFileName,String cstorageId) {
2      try {
3       ic = new InitialContext();
4       ds = (DataSource)ic.lookup("java:/bonitaDS");
5       conn = ds.getConnection();
6
7       int success=0;
8       String sql1="INSERT INTO [Bonita BPM].[dbo].[arch_document_mapping]("
9         + "[tenantid],"
10        + "[id],"
11        + "[processinstanceid],"
12        + "[sourceObjectId],"
13        + "[documentName],"
14        + "[documentAuthor],"
15        + "[documentCreationDate],"
16        + "[documentHasContent],"
17        + "[documentContentFileName],"
18        + "[documentContentMimeType],"
19        + "[contentStorageId],"
20        + "[documentURL],"
21        + "[archiveDate])"
22      +"VALUES ("
23        + "1,"
24        + "?,"
25        + "?,"
26        + "null,"
27        + "?,"
28        + "?,"
29        + "(select CAST(convert(varchar,getdate(),112) as int)),"
30        + "1,"
31        + "?,"
```

```
32        + "null,"
33        + "?,"
34        + "null,"
35        + "(select CAST(convert(varchar,getdate(),112) as int)));";
36      ps = conn.prepareStatement(sql1);
37      ps.setInt(1, cidBonita);
38      ps.setInt(2, cpiidBonita);
39      ps.setString(3, cdnameBonita);
40      ps.setInt(4, cdauthorBonita);
41      ps.setString(5, cdocumentFileName);
42      ps.setString(6, cstorageId);
43
44      success = ps.executeUpdate();
45      if(success>=1)
46       System.out.println("Table arch_documment_mapping Stored");
47      ps.close();
48
49      conn.close();
50
51     }catch(Exception e){
52      System.out.println("ERROR: "+e);
53     }finally{
54               if (ps != null) {
55                   try {
56                       ps.close();
57                   } catch (SQLException sqlex) {
58                    sqlex.printStackTrace();
59                   }
60                   ps = null;
61               }
62
63     }
64    }
65
66  // Código elaborado por el autor
```

La función *saveToBonitaDC* sirve para almacenar los datos en la tabla *document_content* de Bonita BPM. Esta función recibe los siguientes parámetros: *id, documentId, content.* El parámetro *documentId* será igual a *contentStorageId* de la función *saveToBonitaDM.* También se crea el enlace al nombre del DataSource de la plataforma de Bonita BPM para referenciar y poder crear una conexión.

```
1    public static void saveToBonitaDC(int c_id, byte[] c_docs, String c_storageId) {
2     try {
3      ic = new InitialContext();
4      ds = (DataSource)ic.lookup("java:/bonitaDS");
5      conn = ds.getConnection();
6
7      int success2=0;
8      String sql2 = "INSERT INTO [Bonita BPM].[dbo].[document_content]("
```

```
9         + "[tenantid],"
10        + "[id],"
11        + "[documentId],"
12        + "[content])"
13       + "VALUES("
14        + "1,"
15        + "?,"
16        + "?,"
17        + "?);";
18      ps2 = conn.prepareStatement(sql2);
19      ps2.setInt(1, c_id);
20      ps2.setString(2, c_storageId);
21      ps2.setBytes(3, c_docs);
22
23      success2 = ps2.executeUpdate();
24      if (success2>=1)
25       System.out.println("Table Document_Content Stored");
26      ps2.close();
27      //Cerrar la conexion.
28      conn.close();
29
30     }catch(Exception e){
31      System.out.println("Error: "+e);
32     }finally{
33               if (ps2 != null) {
34                   try {
35                       ps2.close();
36                   } catch (SQLException sqlex) {
37                    sqlex.printStackTrace();
38                   }
39                   ps2 = null;
40               }
41
42     }
43    }
44
45  // Código elaborado por el autor
```

4.6.3. Desarrollo del componente ViewUploadFiles

4.6.3.1. Diseño y Programación del Front-End

4.6.3.2. Lenguaje de marcado HTML

La parte del diseño *HTML* del componente *ViewFilesUploade*s igual al que el componente *MultiFilesUpload,* consta de una cabecera que referencia a las funciones *JavaScript, CSS*. Además, contiene un cuerpo en donde está escrito código *JSP* este código se ejecuta del lado del servidor, Figura 4.25.

```
1  <HTML>
```

```
2  <head>
3  <link href="CSS/applicationResource.CSS" type="text/CSS" rel="stylesheet">
4  <link href="CSS/CSS2.CSS" type="text/CSS" rel="stylesheet">
5  <script src="funcionesJs/jquery-1.11.3.min.js"></script>
6  <title>Lista Adjuntos</title>
7  </head>
8  <body class="bonita-body">
9   <div id="footerpusher">
10   <div id="main">
11    <div id="static_application">
12     <div id="bonita_process_label" class="bonita_process_label">
13      <div class="gwt-HTML">Documentos</div>
14     </div>
15     <div id="bonita-banneraltbas"></div>
16     <div id="bonita_form">
17      <div id="bonita_form_page_label" class="bonita_form_page_label">
18       <div class="gwt-HTML">Lista Adjuntos</div>
19      </div>
20      <div class="bonita_form_container" >
21       <form method="get" action="http://localhost:8080/Archivos/
             GenerarZipDownload" >
22        <div id="contenido_UploadFiles">
23        <button class="cls_download_link_Zip" >DESCAGAR TODO</button>
24
25        </div>
26       </form>
27      </div>
28     </div>
29    </div>
30   </div>
31  </div>
32  <div class="footer">
33   <span id="footer">Universidad Técnica de Ambato <a
34    href="JavaScript:window.close()">Salir</a></span>
35  </div>
36 </body>
37 </HTML>
38
39 // Código elaborado por el autor
```

Figura 4.25: Componente ViewUploadFiles
Elaborado por el autor

4.6.3.3. JavaScript

La siguiente función JavaScript envía todos nombres de los archivos visualizados y los envía al servidor. Además recibe como respuesta el archivo de tipo *zip* que se genera en el servidor.

```
1  $(document).ready(function() {
2   var nomFile = new Array();
3   var nomReal = new Array();
4   var aux = null;
5
6   $('.cls_download_link_Zip').click(function(){
7        $('#contenido_UploadFiles .cls_download_link').each(function() {
8          var elemento= this;
9          nomFile.push(elemento.innerHTML);
10         aux = elemento.href;
11         nomReal.push(aux.substring(aux.lastIndexOf('\\')+1,aux.lastIndexOf('&') ));
12   });
13
14     $.ajax({
15          type: 'get',
16          url: 'http://localhost:8080/Archivos/GenerarZipDownload',
17          data: { nomFile:nomFile,nomReal:nomReal },
18          success: function(data){
19          var rutaZip = data;
20          var NomZip = (data.substring(data.lastIndexOf('\\')+1))
21          document.location.href="http://localhost:8080/Archivos/DownloadFiles?
                FilePath="+rutaZip+"&FileName="+NomZip;
22          }
23        }).fail(function(jqXHR, textStatus, errorThrown){
24          console.log(jqXHR.statusText);
```

```
25            console.log(textStatus);
26            console.log(errorThrown);
27              });
28      return false;
29
30   });
31
32 });
33
34 // Código elaborado por el autor
```

4.6.3.4. Programación del Back-End

4.6.3.5. JSP

La función de siguiente código *JSP* es visualizar los archivos almacenados en la BD de Bonita BPM en la interfaz del componente.

Para éste proceso la página *JSP* recibe 4 parámetros desde la plataforma Bonita BPM, realiza una conexión haciendo referencia al DataSource de la plataforma, posteriormente ejecuta una consulta SQL con los parámetros recibidos, después crea un directorio temporal en el sistema de archivos del sistema operativo, enseguida dentro de un ciclo repetitivo se genera un número aleatorio de 10 cifras y este servirá para el nuevo nombre físico del archivo que se guardará en el directorio temporal creado, después convierte el documento guardado en la BD como hexadecimal a archivo por consiguiente el nombre del archivo lo reemplaza con el nombre generado y escribe el archivo en el sistema, enseguida imprime etiquetas *HTML* desde *JSP,* la etiqueta generada es de tipo *link* en donde las propiedades a asignarse con los datos generados son *href* que contendrá la dirección del archivo para la descarga, y la propiedad *appendChild* que contendrá el nombre real del archivo almacenado en la BD de la plataforma Bonita BPM.

```
1 <%@page contentType="text/HTML"
2   import="java.util.*, javax.sql.DataSource,javax.naming.*, java.sql.*,java.io.
        OutputStream,java.io.FileOutputStream,java.io.File;"%>
3 <%
4             //pid:misVariablesGet.id,ppiid:misVariablesGet.piid,pdname:misVariablesGet
                  .dname,pdauthor:misVariablesGet.dauthor
5             String pid=request.getParameter("id");
6             String ppiid=request.getParameter("piid");
7             String pdname=request.getParameter("dname");
8             String pdauthor=request.getParameter("dauthor");
```

```
9
10              DataSource ds = null;
11              Connection conn = null;
12              PreparedStatement ps = null;
13              InitialContext ic;
14
15              byte[] fileBytes;
16              String query;
17              String paramDocumento = null;
18
19              //Recuperar Archivos de la BD
20              try {
21               ic = new InitialContext();
22               ds = (DataSource) ic.lookup("java:/bonitaDS");
23               conn = ds.getConnection();
24
25               String sql = " select a.documentContentFileName, b.content "
26                       +" from [Bonita BPM].[dbo].[arch_document_mapping] a ,[Bonita
                             BPM].[dbo].[document_content] b "
27                    +" where a.id = b.id "
28                    +" and a.contentStorageId = b.documentId "
29                    +" and a.documentAuthor = "+pdauthor
30                    +" and a.documentName = " +"\'"+pdname +"\'"
31                    +" and a.processinstanceid = "+ppiid;
32
33               ps = conn.prepareStatement(sql);
34               ResultSet rs = ps.executeQuery();
35
36               File path = new File("C://seeFilesBonita//");
37                                 if (!path.exists()) {
38                                  boolean status = path.mkdirs();
39                                 }
40
41               while (rs.next()) {
42
43                long x = (long) (Math.random () * (1000000000000000000L - 1) + 1);
44
45                String nomArch = rs.getString(1);
46                String extension = nomArch.substring( nomArch.lastIndexOf( "." ) );
47                String nombGen = "tpm_"+x+extension;
48                fileBytes = rs.getBytes(2);
49
50                OutputStream targetFile = new FileOutputStream( path +"\\"+ nombGen);
51
52                targetFile.write(fileBytes);
53                targetFile.close();
54
55                out.println("<div class=\"classLink\">");
56                out.println("<a class=\"cls_download_link\" href=\"http://localhost
                        :8080/Archivos/DownloadFiles?FilePath=C:\\seeFilesBonita\\"
57                  + nombGen
58                  + "&FileName="
59                  + rs.getString(1)
```

```
60                + "\">" + rs.getString(1) + "</a>");
61               out.println("</div>");
62
63              }
64
65              rs.close();
66              ps.close();
67              conn.close();
68
69             } catch (Exception e) {
70              out.println("ERROR:" + e);
71             }
72            %>
73
74  // Código elaborado por el autor
```

4.6.3.6. Servlet

La función del siguiente servlet sirve para descargar como archivo zip todos los documentos generados en la vista del componente ViewUploadFiles.

El servlet recibe los nombres tanto físico como el nombre real desde la interfaz del componente, se genera un nombre aleatorio que tendrá como formato Adjuntos**********.zip los asteriscos representan un nombre aleatorio, posteriormente se realiza una copia de los archivos entre el nombre físico y el nombre real del archivo, después se genera un archivo zip añadiendo cada uno de los archivos duplicados con el nombre verdadero del documento, enseguida se elimina los archivos duplicados, y por último se envía como respuesta a la interfaz la dirección del archivo zip, el mismo que contendrá todos los documentos generados en la vista del componente.

```
1   protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException, NullPointerException {
2    String [] fileName  = request.getParameterValues("nomFile[]");
3    String [] realName  = request.getParameterValues("nomReal[]");
4
5    //Generar Nombre Para Zip
6     long x = (long) (Math.random () * (1000000000000000000L - 1) + 1);
7     String nomZip = ("Adjuntos"+x+".zip");
8     String filePath = ("C:\\seeFilesBonita\\"+nomZip);
9
10   //Copiar con otro nombre
11   try{
12    Integer n = 0;
13      n=fileName.length;
14    for( int i=0; i<n;i++){
```

```
15      Path FROM = Paths.get("C:\\seeFilesBonita\\"+realName[i]);
16            Path TO = Paths.get("C:\\seeFilesBonita\\"+fileName[i]);
17            CopyOption[] options = new CopyOption[]{
18              StandardCopyOption.REPLACE_EXISTING,
19              StandardCopyOption.COPY_ATTRIBUTES
20            };
21            Files.copy(FROM, TO, options);
22     }
23    }catch(Exception e){
24     System.out.println(e);
25    }
26
27    //Generar Zip
28      try {
29        ZipFile zipFile = new ZipFile(filePath);
30        ArrayList filesToAdd = new ArrayList();
31
32         for(int i=0; i < realName.length; i++){
33          filesToAdd.add(new File("C:\\seeFilesBonita\\"+fileName[i]));
34         }
35
36        ZipParameters parameters = new ZipParameters();
37        parameters.setCompressionMethod(Zip4jConstants.COMP_DEFLATE);
38        parameters.setCompressionLevel(Zip4jConstants.DEFLATE_LEVEL_NORMAL);
39        zipFile.addFiles(filesToAdd, parameters);
40         } catch (ZipException e) {
41          e.printStackTrace();
42         }
43
44    //Borrar Archivos cambiados el nombre
45      try{
46       for (int j=0; j<fileName.length; j++){
47       File deleteFile = new File("C:\\seeFilesBonita\\"+fileName[j]) ;
48       if( deleteFile.exists() )
49        deleteFile.delete() ;
50       }
51      }
52      catch(Exception e){
53       e.printStackTrace();
54      }
55
56      response.setContentType("text/HTML");
57                response.setCharacterEncoding("UTF-8");
58                response.getWriter().write(filePath);
59   }
60
61  // Código elaborado por el autor
```

En la siguiente Figura 4.26 se puede observar la función del botón *DESCARGAR TODO,* éste botón descarga todos los archivos empaquetados en un solo archivo de tipo zip.

Figura 4.26: Función Descargar Todo
Elaborado por el autor

4.7. Integración de los componentes en la plataforma Bonita BPM

Los componentes se alojarán en el servidor de aplicaciones JBoss 5.10 GA. Para ello se debe abrir una ventana de comandos en Windows (cmd), nos dirigimos a la carpeta raíz de Bonita_Home en este caso

```
#cd c:\Bonita BPMCommunity-6.3.0-JBoss-5.1.0.GA\bin
```

Aquí arrancamos el servidor ejecutando el siguiente código, como se muestra en la Figura 4.27.

```
# run.bat -b 0.0.0.0
```

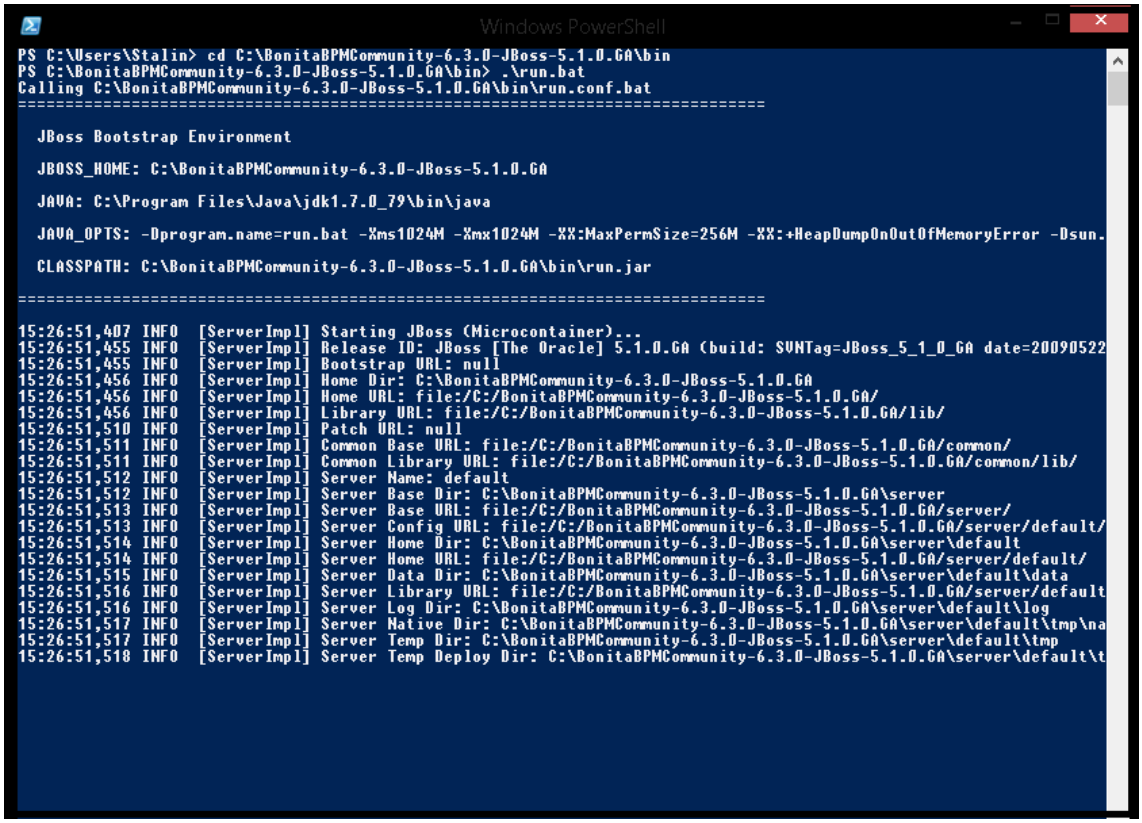


Figura 4.27: Inicio de la plataforma Bonita BPM
Elaborado por el autor

Una vez iniciado el servidor JBoss y con cualquier navegador introducimos la siguiente dirección *localhost:8080,* y aparecerá la siguiente ventana del panel del servidor como se aprecia en la Figura 4.28

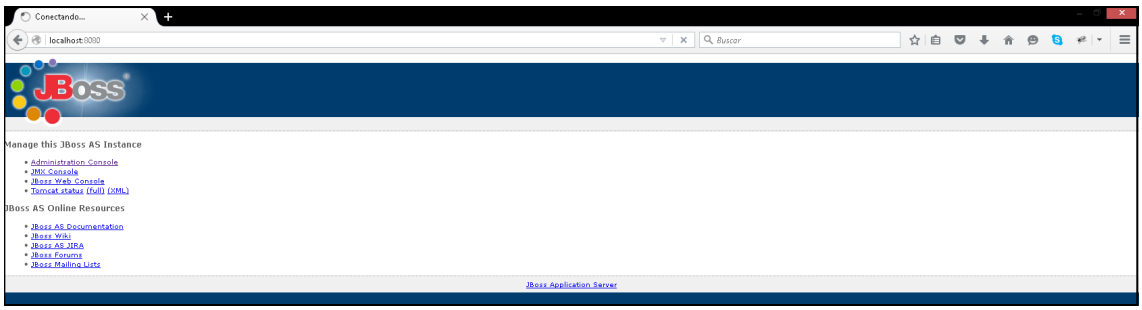


Figura 4.28: Panel Jboss5.10 GA
Elaborado por el autor

Al entrar en la pestaña *Administration Console*, ésta página solicitará una contraseña, la misma será proporcionada por el administrador del proyecto. Figura 4.29.

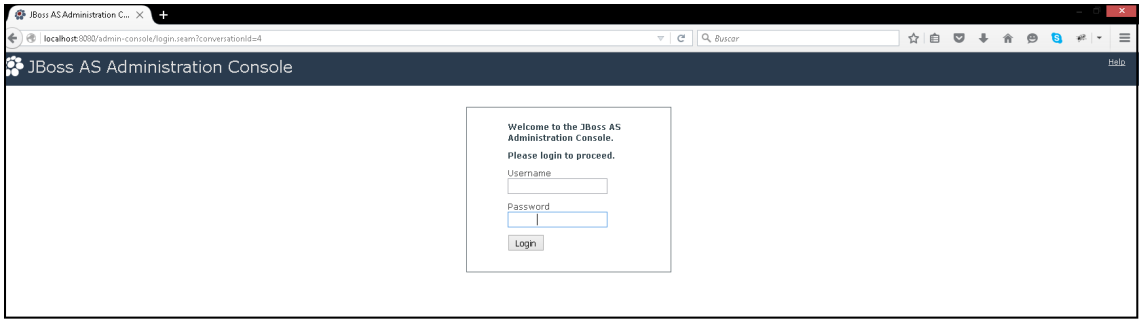


Figura 4.29: Login JBoss
Elaborado por el autor

Una vez introducido las credenciales, el servidor mostrará la siguiente ventana, aquí nos dirigimos a la opción *Web Application,* al hacer click en la opción *examinar* podemos elegir nuestros conponentes empaquetados en formato *war.* Figura 4.30.

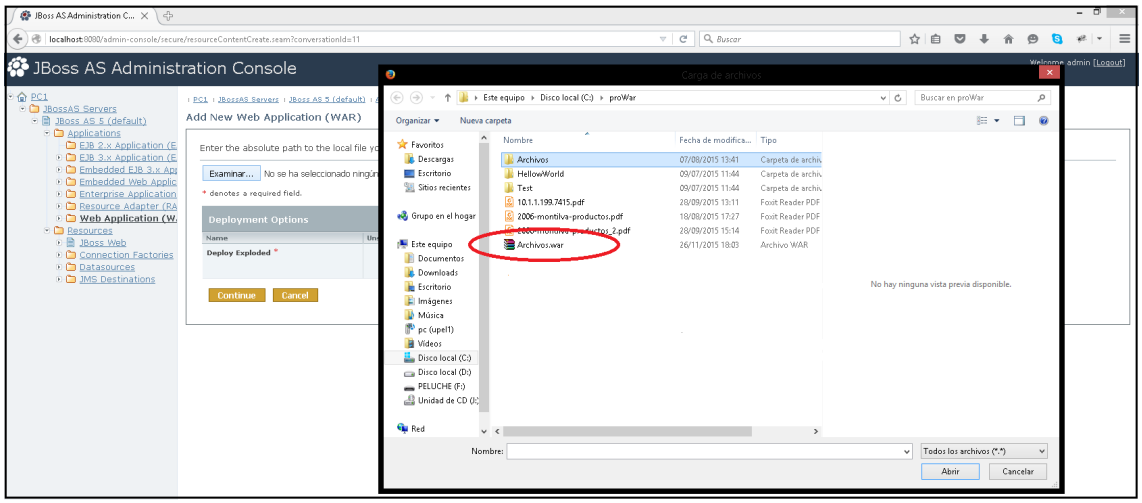


Figura 4.30: Selección de los componentes
Elaborado por el autor

Cuando haya finalizado la subida de nuestros componentes, se notará el siguiente mensaje: *Resource Archivos.war created successfully!,* en este caso los componentes están empaquetados en un solo archivo por lo que se muestra solo un empaquetado como se nota en la Figura 4.31.

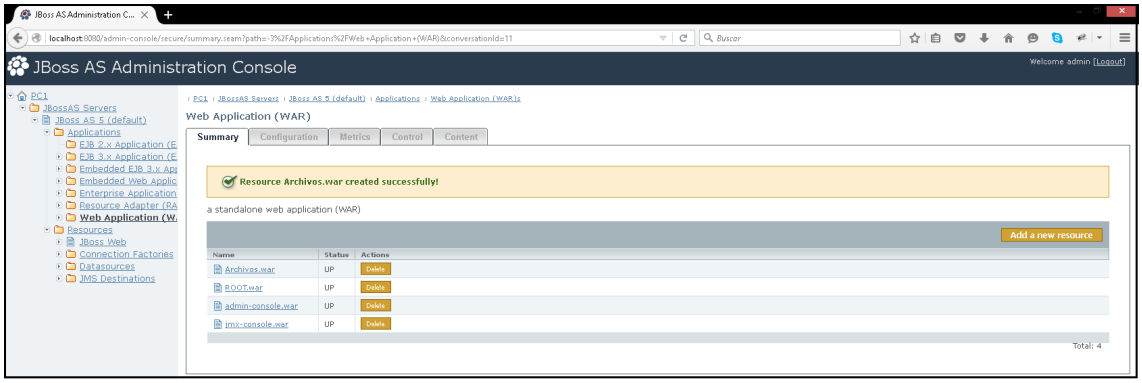


Figura 4.31: Componentes alojados en el servidor
Elaborado por el autor

En la Figura 4.32 se muestran los DataSources configurados en Bonita BPM, esto es muy importante porque deben estar bien identificados ya que se harán referencias desde los componentes a éstos nombres JNDI.

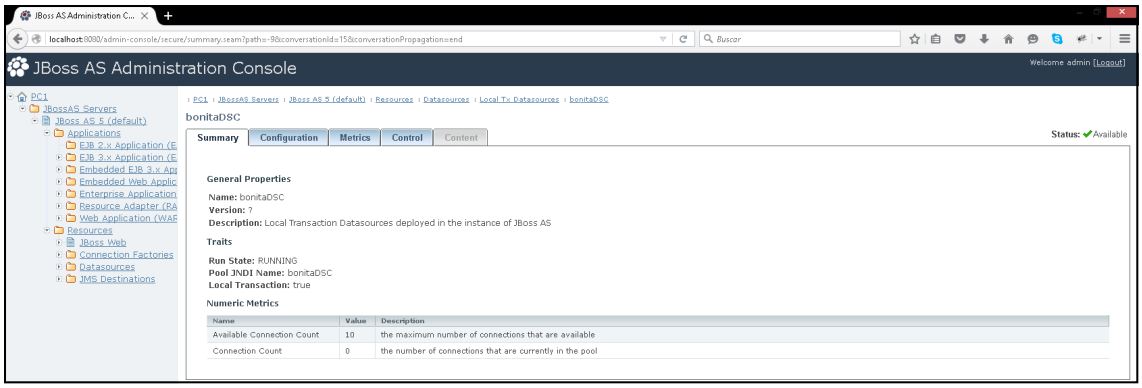


Figura 4.32: Verificación de JNDI
Elaborado por el autor

4.8. Análisis del modelo BPM del proyecto SEGIC.

A continuación se analiza el modelo BPM del proyecto SEGIC Anexo A.1 en donde intervienen componentes que se necesita cambiar por los componentes creados especialmente para este proyecto.

4.8.1. Tareas del proceso

Tarea *Subir Evidencias,* se solicitan al usuario las primeras evidencias que se subirán para ser evaluadas.

Como se puede ver en la Figura 4.33, existe un elemento BPMN gateway (compuerta) de tipo XOR después de la tarea *subir evidencias*, es aquí en donde se emplea una decisión para combinar una variable de tipo *boolean* y preguntar si desea subir más documentos. En esta sección del proceso es la que se vuelve recursiva y repetitiva por que combina acciones de proceso de negocio y de sistema haciendo que el servidor y el proceso como tal se desenvuelvan con mayor lentitud.

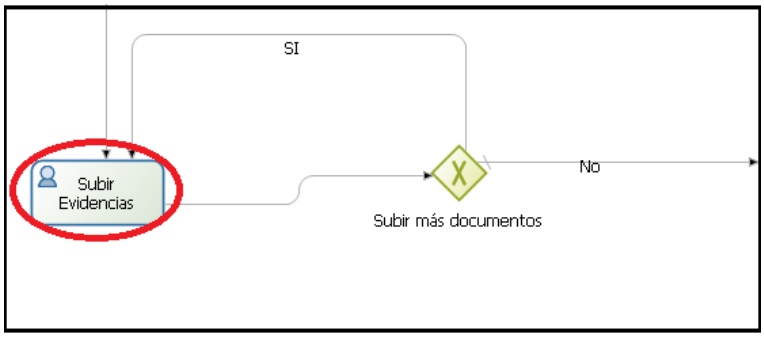


Figura 4.33: Proceso SEGIC
Fuente: Diagrama del proyecto SEGIC.

Tareas: *Evidencia enviada por el responsable, Corrección de la información de la evidencia, notificación de rechazo de la evidencia.* Figura 4.34.
En la primera tarea se utiliza un componente para visualizar los archivos subidos al servidor. En este proceso se visualiza las evidencias que hayan sido enviadas anteriormente por los responsables.
En la segunda tarea se visualiza y se solicita nuevamente un documento faltante, aquí se hacen correcciones en las evidencias subidas para que sean evaluadas por las distintas comisiones.
En la tercera tarea, si las evidencias han sido rechazadas se notifica a cada responsable que evidencias fueron negadas por las comisiones y se termina el proceso.

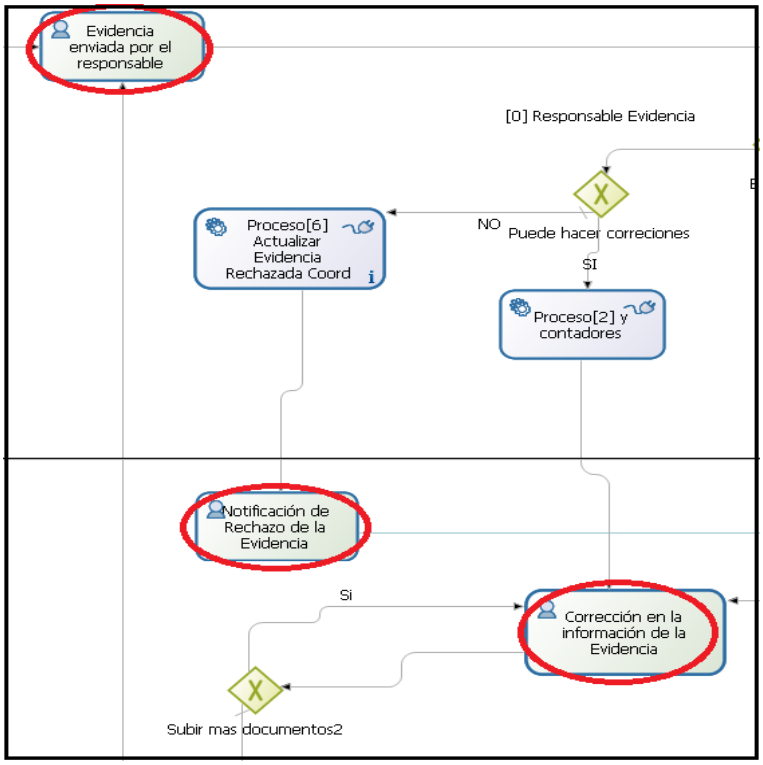


Figura 4.34: Parte del Proceso SEGIC
Fuente: Diagrama del proyecto SEGIC.

En la tarea *Validación de Evidencias,* se envía a la Unidad de planificación y evaluación (UPE) para que los encargados de ésta corrijan y validen las evidencias. Figura 4.35.

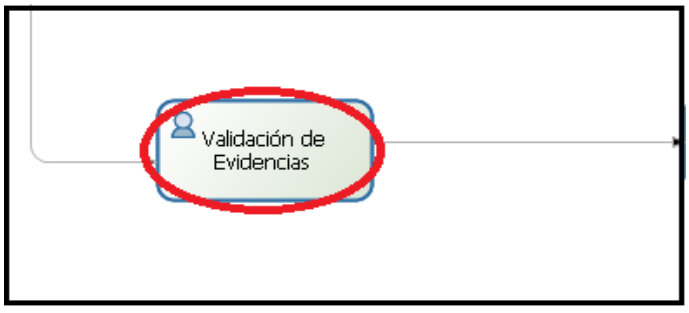


Figura 4.35: Validación de Evidencias
Fuente: Diagrama del proyecto SEGIC

Tareas: *Revisión de resultados de la UPE, Notificación de la Rechazo de la UPE, Notificación de Rechazo de la Evidencia UPE.* Figura 4.36

En la primera tarea se validan todas las evidencias por la UPE y ésta unidad determina si finalmente se guardan los documentos y termina el proceso o, se

rechazan por completo para que nuevamente sean solicitadas.
En la segunda tarea la UPE envía al responsable las evidencias que fueron rechazadas.
En la tercera tarea el responsable recibe la notificación por la UPE y recibe todos los documentos que fueron rechazados, terminando así el proceso por completo.

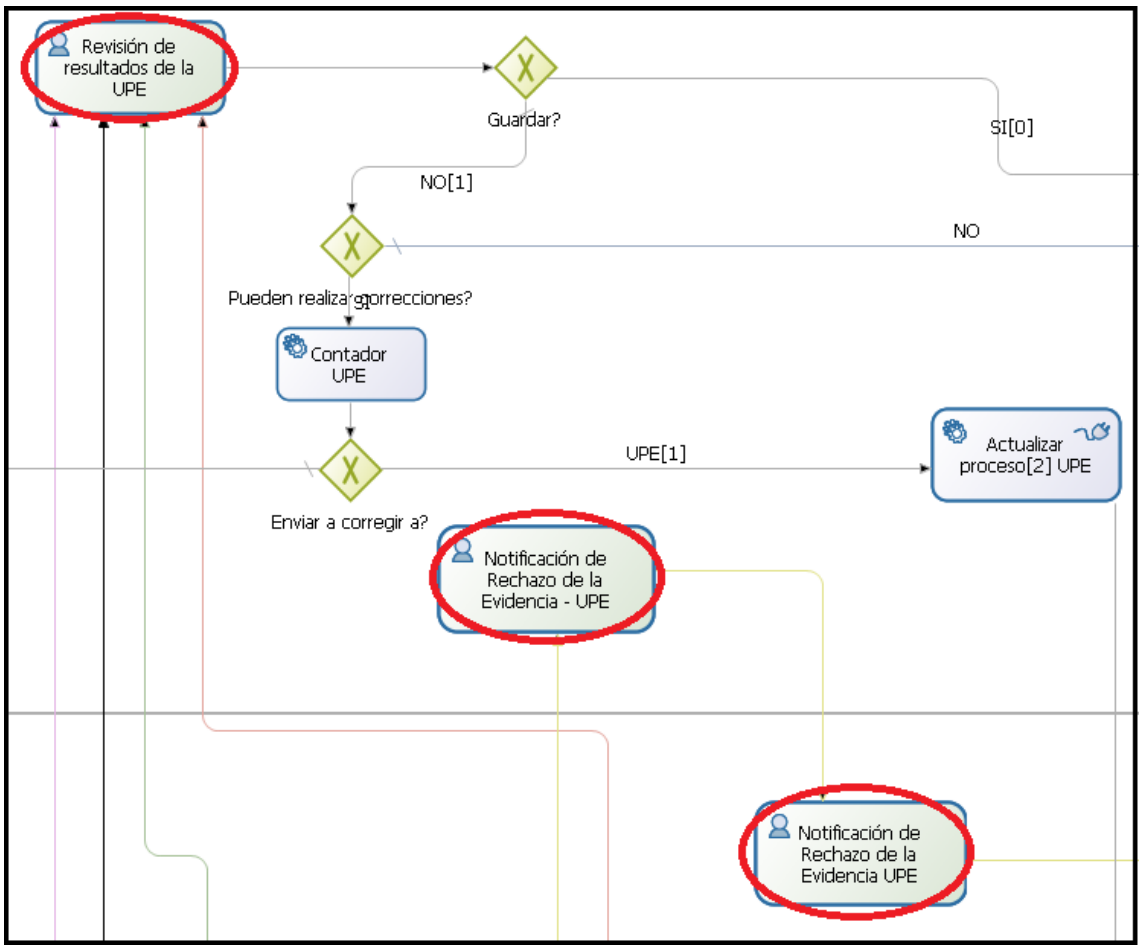


Figura 4.36: Parte del Proceso SEGIC
Fuente: Diagrama del proyecto SEGIC

Tareas: *Corrección de la validación de evidencias, Notificación de rechazo de evidencia Coord. , Notificación de rechazo de la Evidencia - Coord.* Figura 4.37.
En la primera tarea se vuelven a corrigen por segunda vez las evidencias enviadas.
En la segunda tarea la UPE notifica a cada coordinador que han sido rechazadas las evidencias.
En la tercera tarea cada coordinador recibe el mensaje de rechazo con las observaciones.

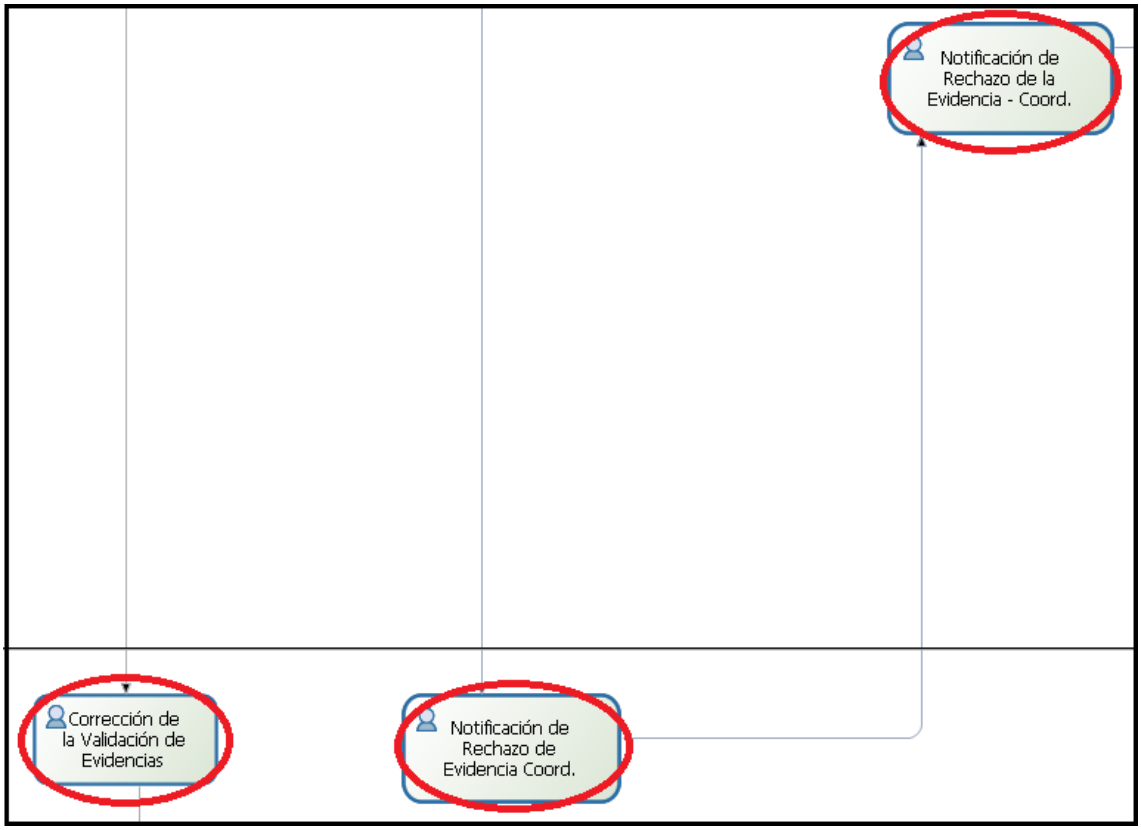


Figura 4.37: Parte del Proceso SEGIC
Fuente: Diagrama del proyecto SEGIC.

4.8.2. Formularios de las Tareas

Todos los formularios tienen el mismo diseño tanto para la visualización y la subida de archivos.

Éstos componentes o *widgets* son los que trae la versión libre o *community* de Bonita BPM, estos no vienen con una opción para solicitar múltiples documentos, es por lo que el diseño original se desarrolló con 7 *widgets files* independientes y estáticos limitando así la carga de archivos y demora en el proceso del *workflow,* Figura 4.38.

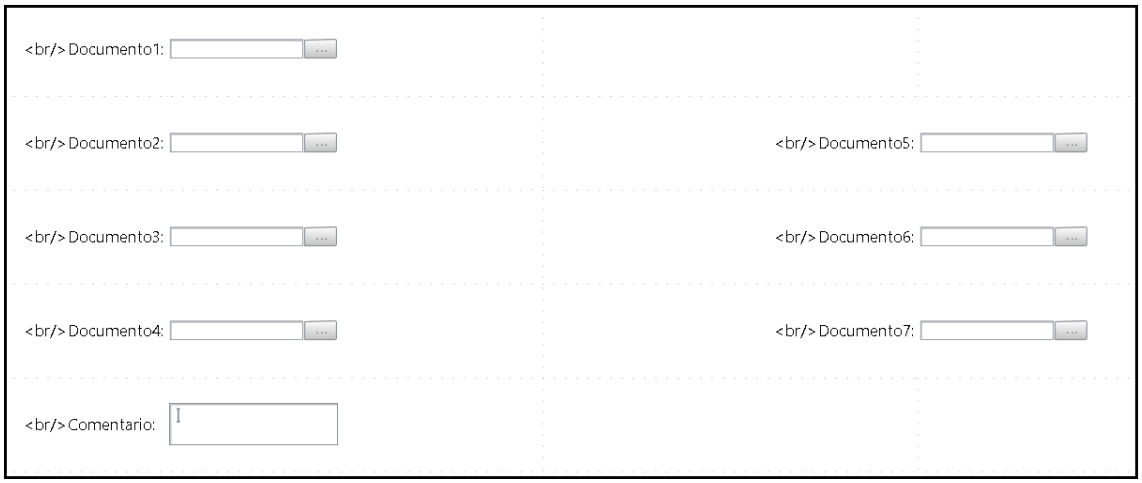


Figura 4.38: Formularios de las tareas.
Fuente: Interfaz del proceso SEGIC.

Interfaz web del diseño final Figura 4.39, esta se muestra en todas las tareas que se solicitan archivos, aquí se aprecia 7 *widgets* de tipo *file* con la información necesaria como el tipo y tamaño de archivo a subir. En esta figura se puede ver de una manera más completa la limitación de Bonita BPM sobre la gestión de archivos. Figura 4.39.

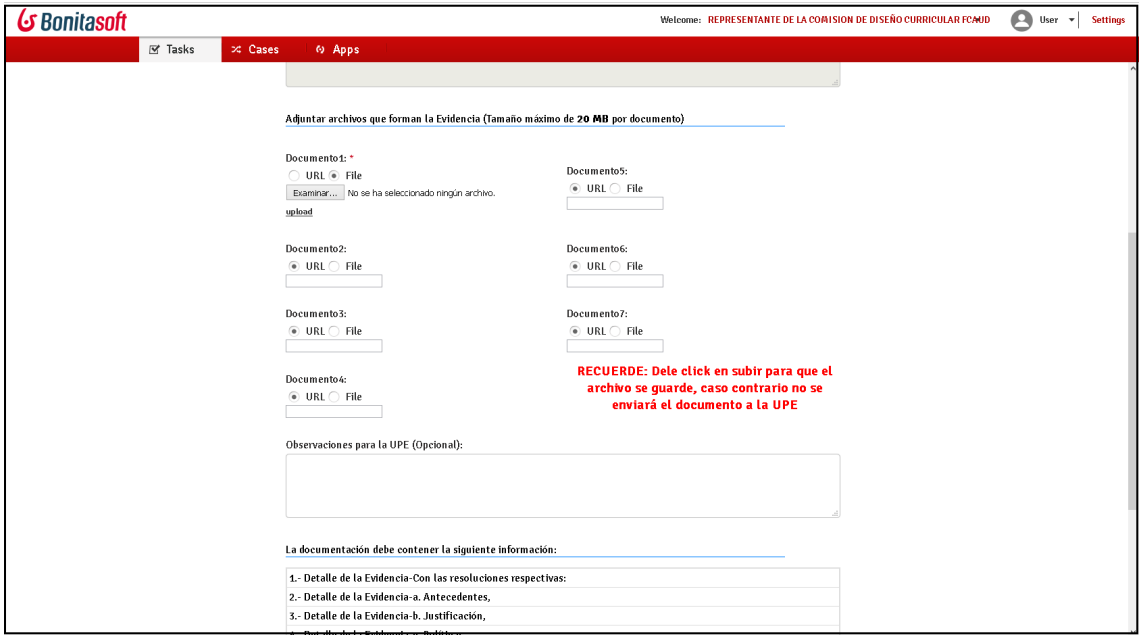


Figura 4.39: Diseño de la Interfaz Web
Fuente: Diseño web Proyecto SEGIC.

4.9. Rediseño del modelo con los nuevos componentes.

Para el rediseño del modelo y el intercambio de datos con los componentes es importante obtener los siguientes parámetros:

- AuthorId.- Es código numérico que identifica al autor del proceso y de la tarea.
- CaseId.- Es el código del caso del proceso.
- DocumentName.- Es el nombre de la variable tipo documento declarada.
- ProcessInsId.- Es el código del proceso activo en el cual se realizan las acciones de las tareas.

En cada *pool* del proceso del entorno de Bonita Studio se declaran cuatro variables de tipo *texto* así como lo muestra la Figura 4.40. Aquí se almacenarán los códigos que se necesita para enviar a los componentes.

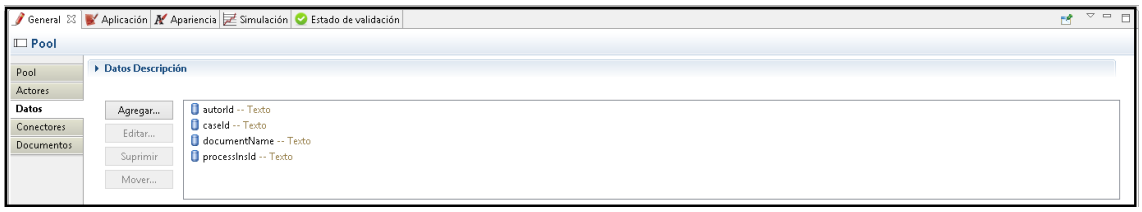


Figura 4.40: Variables Bonita
Elaborado por el autor

Al inicio del proceso es necesario insertar un elemento BPMN de la paleta de controles, el elemento a utilizar es de tipo *script* este elemento será el encargado de recolectar y almacenar datos en las variables anteriormente declaradas en el pool Figura 4.41.

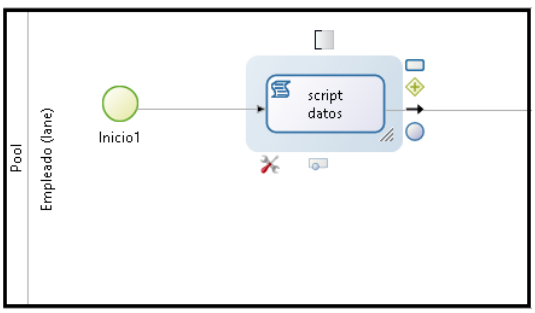


Figura 4.41: Elemento BPMN de tipo Script
Elaborado por el autor

En el diseño de la interfaz web, arrastramos un elemento de tipo *Widget HTML,* éste componente servirá para enlazar tanto el componente MultiFilesUpload como el componente ViewFilesUpload. Figura 4.42.

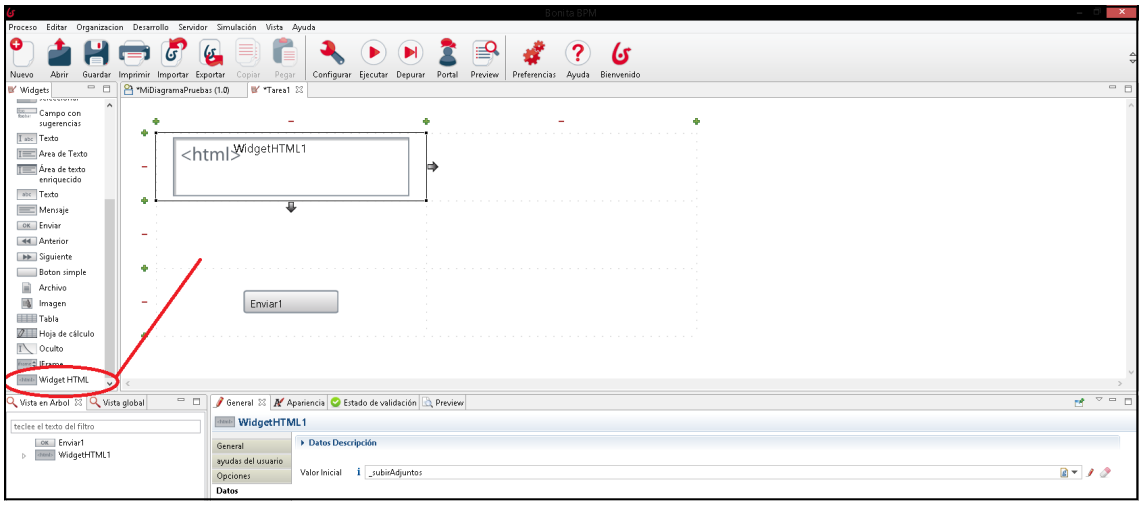


Figura 4.42: Componente Widget HTML
Elaborado por el autor

4.9.1. Inserción del componente MultiUploadFiles

Mediante programación GROOVY insertamos código HTML, CSS y JavaScript diseñando una etiqueta con el mismo estilo de la interfaz web de Bonita BPM, esta etiqueta invocará mediante URL al componente *MultiFilesUpload* y enviará los parámetros de las variables declaradas en Bonita Studio.

```
1  StringBuilder sb=new StringBuilder("<HTML");
2  sb.append("<head>");
3  sb.append("<style type=\"text/CSS\">");
4  sb.append(".subirAdjuntos {");
5  sb.append("-moz-user-select: none;background: rgba(0, 0, 0, 0)linear-gradient(to
       bottom, #3d3d3d 0%, #272727 100%) repeat scroll 0 0;");
6  sb.append("border: 1px solid #b9b9b9;border-radius: 3px;box-shadow: 2px 2px 2px #
       d8d8d8;color: #ffffff;cursor: pointer;display: inline-block;float: left;");
7  sb.append("font-family: \"Signika-Semibold\";font-size: 75%;line-height: 25px !
       important;min-width: 90px;opacity: 1;outline-color: #f9f9f9; padding: 0 2px;
       font-weight: bold;");
8  sb.append("text-align: center;text-decoration: none;text-shadow: 0 0 0 transparent,
        -1px -1px 0 transparent, 1px -1px 0 transparent, -1px 1px 0 transparent, 1px 1
       px 0 transparent;");
9  sb.append("vertical-align: middle;white-space: nowrap;");
10 sb.append("}");
11 sb.append(".subirAdjuntos:active { border-top-color: #fcfcfc; background: #fcfcfc
       ; color: #030303; }");
```

```
12 sb.append("</style>");
13 sb.append("<script>");
14 sb.append("\$(document).ready(function(){");
15 sb.append("\$('#eta_mostrar').hide();");
16 sb.append("\$(\"#eta_ocultar\").on( \"click\", function() {");
17 sb.append("\$('.subirAdjuntos').hide();");
18 sb.append("\$('#eta_mostrar').show();");
19 sb.append(" });");
20 sb.append("});");
21 sb.append("</script>");
22 sb.append("</head>");
23 sb.append("<body>");
24 sb.append("<div id=\"eta_mostrar\"><strong>Archivos Subidos</strong></div>");
25 sb.append("<a id=\"eta_ocultar\" class=\"subirAdjuntos\" onclick=\"window.open('
       http://localhost:8080/Archivos/index.HTML?id="++"&piid="++"&dname="++"&dauthor
       ="++"', '', 'status=yes,top=0,left=0,width=760,height=560', false); return true
       ;\">Subir Documentos</a>");
26 sb.append("</body>");
27 sb.append("</HTML>");
28 sb.toString();
29
30 // Código elaborado por el autor
```

Con el código anterior se puede observar el diseño de la interfaz web de la etiqueta para invocar al componente *MultiFilesUpload.* Figura 4.43.

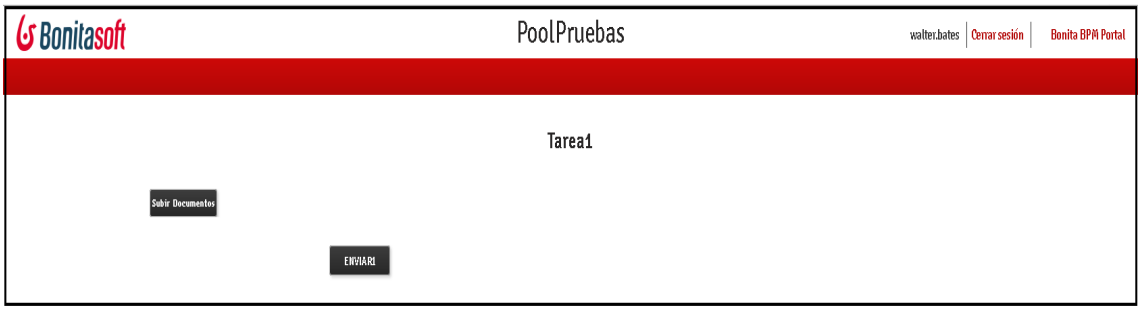


Figura 4.43: Etiqueta para invocar al componente MultiUploadFiles
Elaborado por el autor

4.9.2. Inserción del componente ViewUploadFiles

El proceso del diseño para el componente ViewFilesUploades similar al componente MultiFilesUpload, mediante programación GROOVY se diseña una interfaz de una etiqueta con los mismos estilos CSS de la plataforma web de Bonita BPM y por medio de url se envía los parámetros al componente.

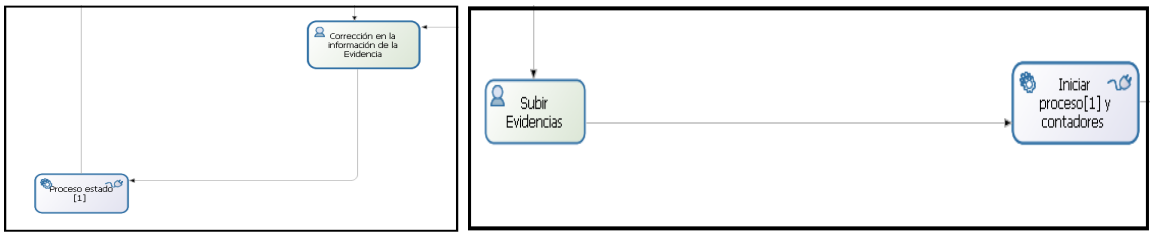
```
1 StringBuilder sb=new StringBuilder("<HTML>");
2 sb.append("<style type=\"text/CSS\">");
```

```
3  sb.append(".subirAdjuntos {-moz-user-select: none; background: rgba(0, 0, 0, 0)
       linear-gradient(to bottom, #3d3d3d 0%, #272727 100%) repeat scroll 0 0; border:
        1px solid #b9b9b9;");
4  sb.append(" border-radius: 3px; box-shadow: 2px 2px 2px #d8d8d8;color: #ffffff;
       cursor: pointer;display: inline-block;float: left;font-family: \"Signika-
       Semibold\";");
5  sb.append(" font-size: 75%; line-height: 25px !important;min-width: 90px;opacity:
       1;outline-color: #f9f9f9; padding: 0 2px; font-weight: bold;");
6  sb.append(" text-align: center; text-decoration: none;text-shadow: 0 0 0
       transparent, -1px -1px 0 transparent, 1px -1px 0");
7  sb.append(" transparent, -1px 1px 0 transparent, 1px 1px 0 transparent; vertical-
       align: middle; white-space: nowrap;   }");
8  sb.append(".subirAdjuntos:active { border-top-color: #fcfcfc;  background: #fcfcfc;
        color: #030303; }</style>");
9  sb.append("<script>\$(document).ready(function(){\$(\"#eta_ocultar\").on( \"click
       \", function() {\$('.subirAdjuntos').hide(); }); }); </script>");
10 sb.append("<a id=\"eta_ocultar\" class=\"subirAdjuntos\" onclick=\"window.open('
       http://localhost:8080/Archivos/VerDocumentos.jsp?id="+caseId+"&piid="+
       processInsId+"&dname="+documentName+"&dauthor="+autorId+"', '', 'status=yes,top
       =0,left=0,width=760,height=560', false); return true;\" >Ver Documentos</a>");
11 sb.append("<body>");
12 sb.append("</body>");
13 sb.append("</HTML>");
14 sb.toString();
15
16 // Código elaborado por el autor
```

En la siguiente Figura 4.44 se puede observar el diseño de la etiqueta para invocar al componente ViewUploadFiles.

Figura 4.44: Etiqueta para invocar al componente ViewUploadFiles
Elaborado por el autor

Con la inserción de los nuevos componentes en el modelo SEGIC se omiten los gateways de decisión y de repetición en las tareas de petición de documentos, separando así el proceso de negocio con el proceso del sistema Figura 4.45.



(a) Nuevo modelo SEGIC **(b)** Nuevo modelo SEGIC

Figura 4.45: Diseño de las tareas del proceso
Fuente: Diagrama Proyecto SEGIC.

4.10. Evaluación del funcionamiento del desempeño del servidor.

Para probar el funcionamiento del rendimiento del servidor ante la carga de archivos, se ha elegido 35 documentos de diferentes tamaños y se ha tomado el tiempo de respuesta del envió hacia el servidor.

El tiempo se ha probado tanto con los componentes originales de Bonita BPM como con los componentes MultiFilesUpload y ViewFilesUpload anteriormente desarrollados en la Tabla 4.2 y 4.3 del Anexo A.4 y A.5 respectivamente se aprecia el tamaño de cada documentos con sus respectivos tiempos arrojados desde el servidor.

Ante estos datos se ha hecho un análisis estadístico para comparar tiempos en el anexo A.2 y A.3 se observa las tablas de distribución de frecuencia por cada caso.

Hipótesis: en donde μ = media (Bonita BPM), μ_0 = media (componentes), z = estadígrafo de contraste, $\propto$= nivel de significación
$\mu = 0.520$
$\sigma = 0.137$
$\mu_0 = 0.10$

$$H_0 \mu \succ \mu_0$$

$$H_1 \mu \leqslant \mu_0$$

$$z = \frac{x - \mu_0}{\frac{\sigma}{\sqrt{n}}}$$

$$z \geq z_{\propto}$$

$$6,6 \geqq 1{,}96$$

Mediante un contraste de hipótesis se ha demostrado que la hipótesis es válida, aceptamos $H0$ y rechazamos $H1$.
Al final se ha obtenido un análisis estadígrafo entre todos los tiempos y se demuestra que con los componentes integrados se obtienen mejores tiempos y resultados.

CAPÍTULO 5

Conclusiones y Recomendaciones

5.1. CONCLUSIONES

Se ha logrado desarrollar e integrar componentes que realicen nuevas funcionalidades en la plataforma Bonita BPM en base a la metodología CBSE. Además, gracias a que los componentes fueron desarrollados específicamente para el presente proyecto, se tienen un control total del código fuente, lo que permite la constante evolución de los mismos componentes según sean solicitada por los requisitos. También asegura la confiabilidad y fiabilidad del origen de los componentes sin tomar riesgos de código malicioso que puedan afectar la seguridad de los datos.

Se consiguió que la plataforma Bonita BPM, versión *community*, adopte componentes extras e independientes a la misma, posibilitando que estos componentes desarrollados funcionen de manera cohesiva con el motor, plataforma y base de datos de Bonita BPM, permitiendo así reutilizar en peticiones y visualizaciones de archivos.

A nivel de la aplicación se realizaron sustanciales mejoras con respecto al tiempo/respuesta desde y hacia el servidor de Bonita BPM, permitiendo que la plataforma gestione mucho más rápidamente los datos validados en el Front-End de los componentes.

Se comprobó que la herramienta BPM BonitaStudio versión libre no es tan accesible en el código, ni manejable para agregar más opciones como validación de archivos y del tamaño de los mismos, entre otras, ya que posee muchas limitaciones no solo en la gestión de archivos sino también en el servidor de aplicaciones.

5.2. RECOMENDACIONES

Es necesario verificar minuciosamente los requisitos de la plataforma, así como la versión que se está utilizando de Bonita BPM y BonitaStudio. La empresa BonitaSoft actualiza constantemente las versiones de sus programas como el motor de aplicaciones, Bases de datos y BonitaEngine, lo que conlleva cambios significativos que pueden afectar y/o cambiar totalmente el funcionamiento de los componentes desarrollados.

Es también imprescindible poseer un conocimiento profundo de las herramientas que se van a utilizar. En este caso fue necesaria la investigación sobre Desarrollo Web (*Servlets, Ajax, JavaScript, HTML, CSS, Jquery, Groovy, Maven, SQL*) en Java y el entorno de trabajo Eclipse.

Para la integración de los componentes realizados con Bonita BPM, hay que tener en cuenta el funcionamiento de los elementos BPMN de la paleta de recursos de BonitaStudio. Además, se deben tener amplios conocimientos sobre gestión de procesos, especialmente la diagramación del o workflow, ya que los nuevos componentes deberán integrarse dentro del proceso sin que aparezcan cambios sustanciales o errores.

En la descarga y/o construcción del código fuente de BonitaBPM se deberá tener mucha precaución con los requisitos, ya que deben ser instalados y configurados correctamente porque pueden dar error en cualquier momento de la descarga. Asimismo se deberá ejecutar un *log* para poder observar al final del proceso, si existieron o no errores tras compilar las dependencias.

No es aconsejable utilizar la versión libre de Bonita Studio para una producción de desarrollo software en sistemas donde se requiera alta concurrencia de usuarios porque el servidor de aplicaciones JBoss que viene integrado en la plataforma BPM, no soporta múltiples conexiones simultaneas de usuarios al mismo tiempo conllevando esto a problemas con la aplicación.

Bibliografia

[1] I. Sommerville, *Software Engineering: Seventh Edition,Ingeniería del Software Basada en componentes*. Miguel Martín-Romo, 2004.

[2] N. Basha and S. Moiz, "Component based software development: A state of art," in *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, pp. 599–604, 2012.

[3] Bonita. [En línea] http://es.bonitasoft.com/productos-servicios

[4] B. Suscripcion. [En linea] www.bonitasoft.com/products-v2/versions-subscription.

[5] M. F. Bertoa, J. M. Troya, and A. Vallecillo, "Aspectos de calidad en el desarrollo de software basado en componentes," *IEEE Conference*, vol. 1, no. 2, p. 3, 2002.

[6] L. A. Romero, "CreaciÓn de un componente personalizado bajo la plataforma netbeans como estrategia de optimizaciÓn en el desarrollo de sistemas informÁticos.," Abril 2013.

[7] M. S. F. Torres, "Investigación del proceso de ingeniería de software basado en componentes y su aplicacion a un caso practico," Enero 2008.

[8] I.-L. Yen, J. Goluguri, F. Bastani, L. Khan, and J. Linn, "A component-based approach for embedded software development," in *Object-Oriented Real-Time Distributed Computing, 2002.(ISORC 2002). Proceedings. Fifth IEEE International Symposium on*, IEEE, 2002.

[9] L. F. Capretz and D. Li, "Component-based software development," in *Industrial Electronics Society, 2001. IECON'01. The 27th Annual Conference of the IEEE*, vol. 3, IEEE, 2001.

[10] S. Mahmood, R. Lai, and Y. S. Kim, "Survey of component-based software development," *Software, IET*, vol. 1, no. 2, p. 5, 2007.

[11] M. Shang, H. Wang, and L. Jiang, "The development process of component-based application software," in *Information Technology, Computer Engineering and Management Sciences (ICM), 2011 International Conference on*, vol. 1, IEEE, 2011.

[12] T. N. Nguyen, "Component-based software update process in collaborative software development," in *Software Engineering Conference, 2008. APSEC'08. 15th Asia-Pacific*, IEEE, 2008.

[13] R. Pressman, "Ingeniería de software: un enfoque práctico," *Mc Graw Hill*, vol. 1, no. 2, p. 5, 2002.

[14] M. F. Bertoa and A. Vallecillo, "Medidas de usabilidad de componentes software.," in *JISBD*, 2005.

[15] C. Szyperski, *Component software: beyond object-oriented programming*. Pearson Education, 2002.

[16] J. C. Terreros, "Desarrollo de software basado en componentes." [En línea] http://msdn.microsoft.com/es-es/library/bb972268.aspx.

[17] E. Fotán, "Metodología de desarrollo de software basada en componentes," Abril 2010.

[18] BonitaSoft, "Bonita bpm." [En línea] http://es.bonitasoft.com/productos-servicios

[19] K. Barclay and J. Savage, *Groovy programming: an introduction for Java developers*. Morgan Kaufmann, 2010.

[20] J. Q. Ning, "A component-based software development model," in *compsac*, p. 0389, IEEE, 1996.

Anexos y Apéndices

Anexo A

Diagrama Proceso SEGIC Completo

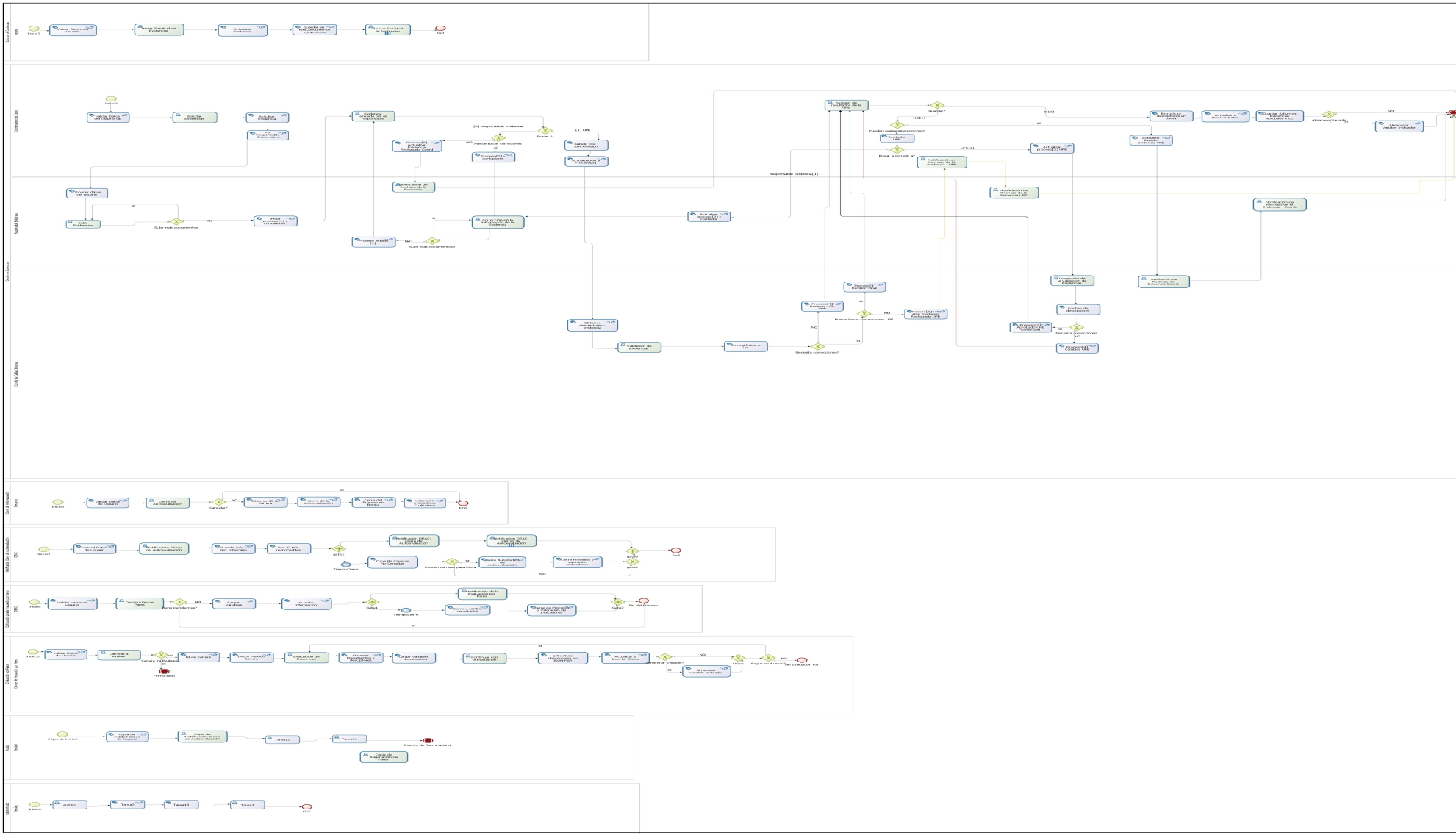


Figura A.1: Diagrama Proceso SEGIC Completo

A.1. Script Build-Bonita-BPM 6.3.0

```
1  #!/bin/bash
2  #inicio
3  #Verificar que java, git and maven están instalados
4  for req in git java mvn
5  do
6          which $req
7          if [ $? -eq 0 ]
8          then
9                  echo "req $req ok"
10         else
11                 echo "req $req Ok, por favor instalar $req"
12                 exit 1
13         fi
14 done
15
16 #Crear Carpeta para los archivos
17 mkdir Bonita BPM-build-6-3-0
18 cd Bonita BPM-build-6-3-0
19
20 # jms.jar Needed at least for Bonita Engine
21 wget http://download.java.net/glassfish/4.0/release/glassfish-4.0.zip
22 unzip glassfish-4.0.zip
23 mvn install:install-file -Dfile=glassfish4/mq/lib/jms.jar -DgroupId=javax.jms -
       DartifactId=jms -Dversion=1.1 -Dpackaging=jar -DgeneratePom=true
24
25 #**** BUILD ENGINE MODULE ****
26 git clone https://github.com/bonitasoft/bonita-engine.git
27 cd bonita-engine
28 # Replace by the tag you want to build
29 git checkout 6.3.0
30 mvn clean install -DskipTests=true -P package
31 cd ..
32
33 #**** BUILD USERFILTERS MODULE ****
34 git clone https://github.com/bonitasoft/bonita-userfilters.git
35 cd bonita-userfilters
36 # Replace by the tag you want to build
37 git checkout 6.3.0
38 mvn clean install -DskipTests
39 cd ..
40
41 #**** BUILD CONNECTORS MODULE ****
42 git clone https://github.com/bonitasoft/bonita-connectors.git
43 cd bonita-connectors
44 # Replace by the tag you want to build
45 # Connectors might have different version of this module as a dependency
46 # so we need to build many versions
47 git checkout bonita-connectors-6.1.1
48 mvn clean install -DskipTests
49 git checkout bonita-connectors-1.0.0
```

```
50 mvn clean install -DskipTests
51 cd ..
52
53 #**** BUILD ALFRESCO CONNECTOR MODULE ****
54 git clone https://github.com/bonitasoft/bonita-connector-alfresco.git
55 cd bonita-connector-alfresco
56 # Replace by the tag you want to build
57 git checkout bonita-connector-alfresco-1.1.3
58 cd bonita-connector-alfresco34-uploadfile-impl
59 sed -i "s/1.1.1/1.1.3/g" pom.xml
60 cd ..
61 mvn clean install -DskipTests
62 cd ..
63
64 #**** BUILD CMIS CONNECTOR MODULE ****
65 git clone https://github.com/bonitasoft/bonita-connector-cmis.git
66 cd bonita-connector-cmis
67 # Replace by the tag you want to build
68 git checkout main
69 mvn clean install -DskipTests
70 cd ..
71
72 #**** BUILD DATABASE CONNECTOR MODULE ****
73 git clone https://github.com/bonitasoft/bonita-connector-database.git
74 cd bonita-connector-database
75 # Replace by the tag you want to build
76 git checkout bonita-connector-database-datasource-1.0.12
77 mvn clean install -DskipTests
78 cd ..
79
80 #**** BUILD EMAIL CONNECTOR MODULE ****
81 git clone https://github.com/bonitasoft/bonita-connector-email.git
82 cd bonita-connector-email
83 # Replace by the tag you want to build
84 git checkout bonita-connector-email-1.0.0
85 mvn clean install -DskipTests
86 cd ..
87
88 #**** BUILD GOOGLE CALENDAR CONNECTOR MODULE ****
89 git clone https://github.com/bonitasoft/bonita-connector-googlecalendar.git
90 cd bonita-connector-googlecalendar
91 # Replace by the tag you want to build
92 git checkout bonita-connector-googlecalendar-1.0.0
93 # add Google gdata .jar files to Maven
94 mvn install:install-file -Dfile=bonita-connector-googlecalendar-common/lib/gdata-
       calendar-2.0.jar -DgroupId=com.google.gdata -DartifactId=gdata-calendar -
       Dversion=2.0 -Dpackaging=jar -DgeneratePom=true
95 mvn install:install-file -Dfile=bonita-connector-googlecalendar-common/lib/gdata-
       core-1.0.jar -DgroupId=com.google.gdata -DartifactId=gdata-core -Dversion=1.0 -
       Dpackaging=jar -DgeneratePom=true
96 mvn install:install-file -Dfile=bonita-connector-googlecalendar-common/lib/gdata-
       client-1.0.jar -DgroupId=com.google.gdata -DartifactId=gdata-client -Dversion
       =1.0 -Dpackaging=jar -DgeneratePom=true
```

```
97  mvn install:install-file -Dfile=bonita-connector-googlecalendar-common/lib/google-
        collect-1.0-rc1.jar -DgroupId=com.google.common -DartifactId=google-collect -
        Dversion=1.0-rc1 -Dpackaging=jar -DgeneratePom=true
98  mvn clean install -DskipTests
99  cd ..
100
101 #**** BUILD JASPER CONNECTOR MODULE ****
102 git clone https://github.com/bonitasoft/bonita-connector-jasper.git
103 cd bonita-connector-jasper
104 # Replace by the tag you want to build
105 git checkout bonita-connector-jasper-1.0.0
106 mvn clean install -DskipTests
107 cd ..
108
109 #**** BUILD LDAP CONNECTOR MODULE ****
110 git clone https://github.com/bonitasoft/bonita-connector-ldap.git
111 cd bonita-connector-ldap
112 # Replace by the tag you want to build
113 git checkout bonita-connector-ldap-1.0.0
114 mvn clean install -DskipTests
115 cd ..
116
117 #**** BUILD SALESFORCE CONNECTOR MODULE ****
118 git clone https://github.com/bonitasoft/bonita-connector-salesforce.git
119 cd bonita-connector-salesforce
120 # Replace by the tag you want to build
121 git checkout update_querysobjects_output_name_1.0.14
122 mvn clean install -DskipTests
123 cd ..
124
125 #**** BUILD SAP CONNECTOR MODULE ****
126 git clone https://github.com/bonitasoft/bonita-connector-sap.git
127 cd bonita-connector-sap
128 # add sapjco.jar file to Maven
129 # Replace by the tag you want to build
130 git checkout jco2-callfunction-update-1.0.3
131 cp bonita-connector-sap-jco2-impl/lib/com/sap/sapjco/1.0/sapjco-1.0.jar sapjco-1.0.
        jar
132 mvn install:install-file -Dfile=sapjco-1.0.jar  -DgroupId=com.sap -DartifactId=
        sapjco -Dversion=2.1.9 -Dpackaging=jar -DgeneratePom=true
133 mvn clean install -DskipTests
134 cd ..
135
136 #**** BUILD SCRIPTING CONNECTOR MODULE ****
137 git clone https://github.com/bonitasoft/bonita-connector-scripting.git
138 cd bonita-connector-scripting
139 # Replace by the tag you want to build
140 git checkout bonita-connector-scripting-1.0.0
141 mvn clean install -DskipTests
142 cd ..
143
144 #**** BUILD SUGARCRM CONNECTOR MODULE ****
145 git clone https://github.com/bonitasoft/bonita-connector-sugarcrm.git
```

```
cd bonita-connector-sugarcrm
# Replace by the tag you want to build
git checkout bonita-connector-sugarcrm-1.0.0
mvn clean install -DskipTests
cd ..

#**** BUILD TALEND CONNECTOR MODULE ****
git clone https://github.com/bonitasoft/bonita-connector-talend.git
cd bonita-connector-talend
# Replace by the tag you want to build
git checkout update-joblauncher-impl-1.0.2
mvn clean install -DskipTests
cd ..

#**** BUILD TWITTER CONNECTOR MODULE ****
git clone https://github.com/bonitasoft/bonita-connector-twitter.git
cd bonita-connector-twitter
# Replace by the tag you want to build you want to build
# At the moment, you need to build 1.0.0 before building master...
git checkout bonita-connector-twitter-1.0.0
mvn clean install -DskipTests
git checkout master
mvn clean install -DskipTests
cd ..

#**** BUILD WEBSERVICE CONNECTOR MODULE ****
git clone https://github.com/bonitasoft/bonita-connector-webservice.git
cd bonita-connector-webservice
# Replace by the tag you want to build
git checkout master
mvn clean install -DskipTests
cd ..

#**** BUILD THEME BUILDER MODULE ****
git clone https://github.com/bonitasoft/bonita-theme-builder.git
cd bonita-theme-builder
# Replace by the tag you want to build you want to build
git checkout 6.1.0
mvn clean install -DskipTests
cd ..

#**** BUILD TOMCAT H2 LISTENER MODULE ****
git clone https://github.com/bonitasoft/bonita-tomcat-h2-listener.git
cd bonita-tomcat-h2-listener
# Replace by the tag you want to build
git checkout bonita-tomcat-h2-listener-1.0.1
mvn clean install -DskipTests
cd ..

#**** BUILD JBOSS H2 BEAN LISTENER MODULE ****
git clone https://github.com/bonitasoft/bonita-jboss-h2-mbean.git
cd bonita-jboss-h2-mbean
# Replace by the tag you want to build
```

```
199 git checkout bonita-jboss-h2-mbean-1.0.0
200 mvn clean install -DskipTests
201 cd ..
202
203 #**** BUILD TOMCAT VALVE MODULE ****
204 git clone https://github.com/bonitasoft/bonita-tomcat-valve.git
205 cd bonita-tomcat-valve
206 # Replace by the tag you want to build
207 git checkout 6.0.35
208 #replace 1.6 by the version of your JDK.
209 mvn clean install -DskipTests -Dmaven.compiler.source=1.6 -Dmaven.compiler.target
      =1.6
210 cd ..
211
212 #**** BUILD STUDIO WATCHDOG MODULE ****
213 git clone https://github.com/bonitasoft/bonita-studio-watchdog.git
214 cd bonita-studio-watchdog
215 # Replace by the tag you want to build
216 git checkout studio-watchdog-6.0.1
217 mvn clean install -DskipTests
218 cd ..
219
220 #****  BUILD GWT TOOLS MODULE ****
221 git clone https://github.com/bonitasoft/bonita-gwt-tools.git
222 cd bonita-gwt-tools
223 # Replace by the tag you want to build
224 git checkout master
225 mvn clean install -DskipTests
226 cd ..
227
228 #**** BUILD WEB MODULE ****
229 git clone https://github.com/bonitasoft/bonita-web.git
230 cd bonita-web
231 # Replace by the tag you want to build
232 git checkout 6.3.0
233 mvn clean install -DskipTests
234 cd ..
235
236 #**** BUILD BONITA DISTRIB MODULE ****
237 git clone https://github.com/bonitasoft/bonita-distrib.git
238 cd  bonita-distrib
239 # Replace by the tag you want to build
240 git checkout 6.3.0
241 mvn clean install -DskipTests
242 cd ..
243
244 #**** BUILD SIMULATION MODULE ****
245 #get csv4j-0.4.0.jar and add to maven
246 wget http://csvobjectmapper.sourceforge.net/maven2/net/sf/csv4j/0.4.0/csv4j-0.4.0.
      jar
247 mvn install:install-file -Dfile=csv4j-0.4.0.jar  -DgroupId=net.sf.csv4j -
      DartifactId=csv4j -Dversion=0.4.0 -Dpackaging=jar -DgeneratePom=true
248 git clone https://github.com/bonitasoft/bonita-simulation.git
```

```
249 cd bonita-simulation
250 # Replace by the tag you want to build
251 git checkout bos-simulation-6.1.0
252 mvn clean install -DskipTests
253 cd ..
254
255 #****BUILD STUDIO - COMMON ACTION ****
256 # Prerequisite step for building any of the Studio modules listed below
257
258 # You must have a correct target platform, available at http://download.forge.ow2.
        org/bonita/TargetPlatform-6.1.zip
259 wget http://download.forge.ow2.org/bonita/TargetPlatform-6.3.zip
260 unzip TargetPlatform-6.3.zip
261 DIR=`pwd`
262
263 # You might need to increase max perm size of maven
264 export MAVEN_OPTS="-XX:MaxPermSize=256m"
265 git clone https://github.com/bonitasoft/bonita-studio.git
266 cd bonita-studio
267 # Replace by the tag you want to build
268 git checkout bos-studio-6.3.0-201411051715
269 #**** BUILD STUDIO PLATFORM MODULE ****
270 cd  platform
271 # Replace '6.3' by the version you want to build
272 mvn clean install -Pmirrored -Dp2MirrorUrl=file://$DIR/6.3/
273 cd ..
274
275 #**** BUILD STUDIO PATCHED PLUGINS MODULE ****
276 cd patched-plugins
277 mvn clean install -DskipTests
278 cd ..
279
280 #**** GENERATE STUDIO MODELS SOURCES ****
281 cd  bundles/plugins/org.bonitasoft.studio-models/
282 # Replace '6.3' by the version you want to build
283 mvn clean initialize -Pgenerate -Dp2MirrorUrl=file://$DIR/6.3/
284 cd ../../..
285
286 #**** BUILD TEST-DEPENDENCIES ****
287 cd tests-dependencies
288 mvn clean install -DskipTests
289 cd ..
290
291 #**** BUILD STUDIO BUNDLES MODULE ****
292 cd bundles
293 mvn clean install -DskipTests
294 cd ..
295
296 #**** BUILD STUDIO I18N BUNDLES MODULE ****
297 cd  translations
298 mvn clean install -DskipTests
299 cd ..
300
```

```
301 #**** BUILD STUDIO ALL-IN-ONE MODULE ****
302 cd all-in-one
303 # If you also want to build the installers, you need to provide a correct
        installation of Bitrock and
304 # set the BITROCK_HOME system property
305 mvn clean package -DskipTests
306 cd ../..
307
308 echo '-------------------------------'
309 echo ''
310 echo 'You just finished to build Bonita BPM. Congratulations !!!'
311 echo 'You you can find it in:'
312 echo ''
313 echo 'Bonita BPM-build/bonita-studio/all-in-one/target/Bonita BPMCommunity-6.3.6'
314 echo ''
315 echo '-------------------------------'
316 echo ''
317 #end
```

A.2. Tabla distribución de frecuencias (Tiempos con Bonita BPM)

Tabla de distribución de frecuencias (Bonita BPM)						
	intervalos		marca de clase(xi)	frecuencia absoluta(fi)	Xi * fi	((xi-med)^2)*fi
1	0,0000066667	0,173	0,086	13	1,121	2,445
2	0,173	0,345	0,259	1	0,259	0,068
3	0,345	0,518	0,431	2	0,863	0,016
4	0,518	0,690	0,604	3	1,811	0,021
5	0,690	0,863	0,776	6	4,658	0,394
6	0,863	1,035	0,949	10	9,488	1,839
				35	0,520	0,137
				n	**media**	**varianza**

Tabla A.1: Distribución de frecuencias (Bonita BPM)

A.3. Tabla distribución de frecuencias (Tiempos con Componentes desarrollados)

Tabla de distribución de frecuencias (Componentes desarrollados)						
	intervalos		marca de clase(xi)	frecuencia absoluta(fi)	Xi * fi	((xi-med)^2)*fi
1	0,000000069	0,042	0,0211667299167	19	0,402	0,129
2	0,042	0,085	0,0635000517500	2	0,127	0,003
3	0,085	0,127	0,1058333735833	1	0,106	0,000
4	0,127	0,169	0,1481666954167	0	0,000	0,000
5	0,169	0,212	0,1905000172500	1	0,191	0,008
6	0,212	0,254	0,2328333390833	12	2,794	0,201
				35	0,103	0,010
				n	**media**	**varianza**

Tabla A.2: Distribución de frecuencias (Componentes desarrollados)

A.4. Tabla de valores de medición de tiempos (Bonita BPM)

MEDIDAS DE TIEMPO CON BONITA BPM		
	Tam. Archivo (MB)	tiempo de subida y validación(min)
1	0,01	0,0000066667
2	0,15	0,000045
3	0,17	0,000059
4	0,20	0,000068
5	0,25	0,000069
6	0,42	0,00012
7	0,59	0,00015
8	0,91	0,00032
9	1,50	0,00276
10	1,90	0,00381
11	2,43	0,00499
12	3,12	0,0281
13	5,30	0,0514
14	5,70	0,181
15	6,38	0,383
16	7,84	0,402
17	8,30	0,584
18	8,60	0,601
19	9,20	0,624
20	10,90	0,692
21	11,10	0,733
22	13,33	0,741
23	13,50	0,785
24	14,24	0,811
25	14,80	0,814
26	15,01	0,945
27	16,20	0,979
28	16,60	0,986
29	17,40	0,987
30	17,45	0,988
31	18,36	1,001
32	18,37	1,002
33	19,36	1,007
34	19,70	1,009
35	19,94	1,035

Tabla A.3: Tiempos con Bonita BPM

A.5. Tabla de valores de medición de tiempos (MultiFilesUpload, ViewFilesUpload)

MEDIDAS DE TIEMPO CON COMPONENTES		
	Tam. Archivo (MB)	tiempo de subida y validación(min)
1	0,01	0,000000069
2	0,15	0,00000055
3	0,17	0,00000057
4	0,20	0,00000062
5	0,25	0,00000063
6	0,42	0,0000086
7	0,59	0,000085
8	0,91	0,00032
9	1,50	0,00049
10	1,90	0,00051
11	2,43	0,00073
12	3,12	0,00084
13	5,30	0,00098
14	5,70	0,0011
15	6,38	0,003
16	7,84	0,005
17	8,30	0,008
18	8,60	0,012
19	9,20	0,024
20	10,90	0,047
21	11,10	0,054
22	13,33	0,111
23	13,50	0,19
24	14,24	0,224
25	14,80	0,229
26	15,01	0,237
27	16,20	0,237
28	16,60	0,238
29	17,40	0,241
30	17,45	0,241
31	18,36	0,241
32	18,36	0,249
33	19,36	0,254
34	19,70	0,254
35	19,94	0,254

Tabla A.4: Tiempos con los componentes desarrollados