Sistema Interoperable para compartir InformaciÃ³n entre Sistemas Web con Autocompletado de Formularios

Generado por Doxygen 1.5.8

Sun Apr 10 19:42:03 2011

Índice general

Capítulo 1

Indice de namespaces

1.1. Lista de Paquetes

Aquí van los paquetes con una breve descripción (si etá disponible):

Capítulo 2

Índice de clases

2.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Capítulo 3

Indice de archivos

3.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

Capítulo 4

Documentación de namespaces

4.1. Paquetes applet

Clases

- class **prueba4_en_jdom**

4.2. Paquetes com

Paquetes

- package **pt2**

4.3. Paquetes com.pt2

Paquetes

- package **pt2Web**

4.4. Paquetes com.pt2.pt2Web

Paquetes

- package **classes**
- package **controller**

4.5. Paquetes com.pt2.pt2Web.classes

Clases

- class **FileManager**

4.5.1. Descripción detallada

Created on 07/03/2011

4.6. Paquetes com.pt2.pt2Web.controller

Clases

- class **FormularioTestPaperServlet**
- class **FormularioTestServlet**

4.7. Paquetes para_cifrar_p4

Clases

- class **Decrypt**
- class **Encrypt**

Capítulo 5

Documentación de las clases

5.1. Referencia de la Clase para_cifrar_p4.Decrypt

Métodos públicos estáticos

- static void **main** (String args[ ]) throws Exception

Funciones Estáticas del Paquete

- **[static initializer]**

5.1.1. Descripción detallada

La clase **Decrypt** (p. 15) lee un archivo cifrado de disco, descifra el contenido del archivo con una clave previamente almacenada.

5.1.2. Documentación de las funciones miembro

5.1.2.1. para_cifrar_p4.Decrypt.[static initializer] () `[static, package]`

5.1.2.2. static void para_cifrar_p4.Decrypt.main (String *args*[ ]) throws Exception `[static]`

La documentación para esta clase fue generada a partir del siguiente fichero:

- /home/itza/ProyectoTerminal/**Decrypt.java**

5.2. Referencia de la Clase para_cifrar_p4.Encrypt

Métodos públicos estáticos

- static Document **parseFile** (String fileName) throws Exception
- static SecretKey **GenerateKeyEncryptionKey** () throws Exception
- static void **storeKeyFile** (Key keyEncryptKey) throws IOException
- static SecretKey **GenerateSymmetricKey** () throws Exception
- static void **writeEncryptedDocToFile** (Document doc, String fileName) throws Exception
- static void **usage** ()
- static void **main** (String args[ ]) throws Exception

Funciones Estáticas del Paquete

- **[static initializer]**

5.2.1. Descripción detallada

La clase **Encrypt** (p. 16) lee la entrada de un archivo, cifra el contenido del archivo y, a continuacion, almacena el archivo cifrado en el disco. Con el fin de lograr esto, la herramienta utiliza el marco de Apache XML para crear dos claves simétricas para los siguientes fines: 1) para cifrar los datos reales del archivo XML 2) para cifrar la clave utilizada para cifrar los datos del archivo XML

Los datos cifrados se escriben en el disco y la clave utilizada para cifrar la clave de encriptacion de datos tambien se almacena en el disco.

5.2.2. Documentación de las funciones miembro

5.2.2.1. para_cifrar_p4.Encrypt.[static initializer] () `[static, package]`

5.2.2.2. static SecretKey para_cifrar_p4.Encrypt.GenerateKeyEncryptionKey () throws Exception `[static]`

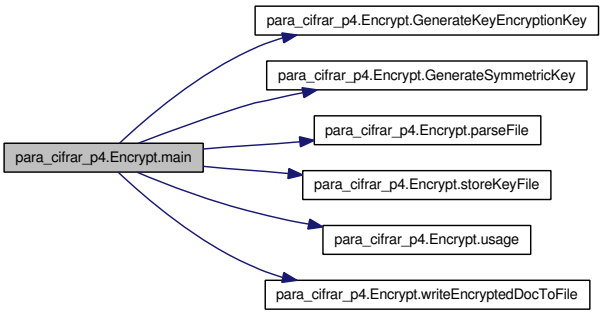
Gráfico de llamadas a esta función:

5.2.2.3. static SecretKey para_cifrar_p4.Encrypt.GenerateSymmetricKey () throws Exception `[static]`

Gráfico de llamadas a esta función:

5.2.2.4. static void para_cifrar_p4.Encrypt.main (String *args*[ ]) throws Exception [static]

Gráfico de llamadas para esta función:

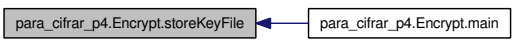


5.2.2.5. static Document para_cifrar_p4.Encrypt.parseFile (String *fileName*) throws Exception [static]

Gráfico de llamadas a esta función:

5.2.2.6. static void para_cifrar_p4.Encrypt.storeKeyFile (Key *keyEncryptKey*) throws IOException [static]

Gráfico de llamadas a esta función:



5.2.2.7. static void para_cifrar_p4.Encrypt.usage () [static]

Gráfico de llamadas a esta función:

5.2.2.8. static void para_cifrar_p4.Encrypt.writeEncryptedDocToFile (Document *doc*, String *fileName*) throws Exception [static]

Gráfico de llamadas a esta función:

La documentación para esta clase fue generada a partir del siguiente fichero:

- /home/itza/ProyectoTerminal/**Encrypt.java**

5.3. Referencia de la Clase com.pt2.pt2Web.classes.FileManager

Métodos públicos

- void **FileManager** ()
- StringBuffer **readFile** (String filename)
- void **saveFile** (String filename, String dataToWrite, boolean append)
- String **replaceValues** (String path, String[ ] valuesToSearch, String[ ] valuesToReplace)
- String **poner_acentos** (String texto)

5.3.1. Descripción detallada

Title: **FileManager** (p. 19)

Description: Manejo de archivos de texto

Autor:

Maritza Hernández Arias

Versión:

3.0

5.3.2. Documentación del constructor y destructor

5.3.2.1. void com.pt2.pt2Web.classes.FileManager.FileManager ()

5.3.3. Documentación de las funciones miembro

5.3.3.1. String com.pt2.pt2Web.classes.FileManager.poner_acentos (String *texto*)

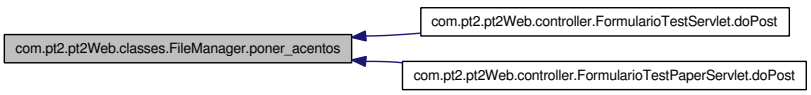
El siguiente método, recibe un String, cualquiera que el usuario haya ingresado desde el formulario, este texto es procesado por si se encuentra algún acento o algún carácter que es especial para latex, pueda sustituirlo de tal forma que Latex lo pueda procesar posteriormente.

Parámetros:

texto

Devuelve:

Gráfico de llamadas a esta función:



5.3.3.2. StringBuffer com.pt2.pt2Web.classes.FileManager.readFile (String *filename*)

El metodo readFile lee un archivo de texto y retorna su contenido en formato de StringBuffer

Parámetros:

filename String

Devuelve:

StringBuffer

Aqui creamos un objeto File que representa el archivo de texto que queremos leer

Variable temporal que usaremos para leer cada una de las lineas del archivo de texto

BufferReader - Es el encargado de leer el archivo de texto. El constructor recibe como parametro un objeto FileReader, que a s vez recibe el objeto File creado precedentemente.

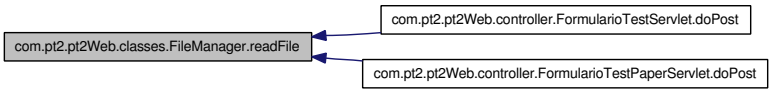
A través de este ciclo el BufferReader lee todo el archivo, y lo va acumulando (sb.append) en un StringBuffer

Al final de la lectura cerramos el objeto

Si damos un nombre de archivo que no existe el sistema genera automaticamente un error.

Se ha producido un error durante la lectura del archivo

Gráfico de llamadas a esta función:



5.3.3.3. String com.pt2.pt2Web.classes.FileManager.replaceValues (String *path*, String[ ] *valuesToSearch*, String[ ] *valuesToReplace*)

Esta función permite, dado un archivo en particular, buscar dentro el mismo determinados valores y cambiarlos por una serie de nuevos valores dados, generando un objeto de tipo String con el resultado

Parámetros:

path String

valuesToSearch String[] Ejemplo {"NOMRE", "APELLIDO"}

valuesToReplace String[] Ejemplo {"Fernando Augusto", "Arturi"}

Devuelve:

String

Lectura del archivo de texto dado

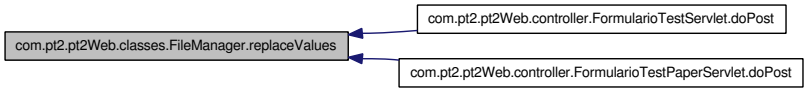
Si damos un nombre de archivo que no existe el sistema genera automaticamente un error.

Se ha producido un error durante la lectura/escritura del archivo

Una vez completada la fase de lectura del archivo, pasamos a la búsqueda y reemplazo de los valores datos. Para esto generamos un ciclo que recorremos tantas veces como valores que tenemos que procesar.

búsqueda y reemplazo de la cadena.

Gráfico de llamadas a esta función:



5.3.3.4. void com.pt2.pt2Web.classes.FileManager.saveFile (String *filename*, String *dataToWrite*, boolean *append*)

Este método permite, dada una cadena de caracteres determinada, salvar la misma como un archivo de texto, o agregarla a un archivo ya existente

Parámetros:

filename String

dataToWrite String

append boolean

Creación del objeto FileWriter dado un nombre de archivo determinado El segundo parametro (append) contiene un valore booleano que indica si la informacion recibida debe ser agregada el final del archivo o, en caso contrario, reemplazar la información ya existente.

Escritura de la informacion en el archivo

Se cierra el archivo

Se ha producido un error durante la lectura/escritura del archivo

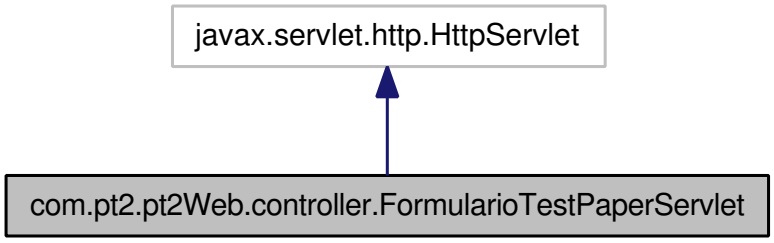
La documentación para esta clase fue generada a partir del siguiente fichero:

- /home/itza/ProyectoTerminal/**FileManager.java**

5.4. Referencia de la Clase com.pt2.pt2Web.controller.FormularioTestPaperServle

Herencias javax::servlet::http::HttpServlet.

Diagrama de colaboración para com.pt2.pt2Web.controller.FormularioTestPaperServlet:



Métodos protegidos

- void **doPost** (HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

5.4.1. Descripción detallada

Title: **FormularioTestPaperServlet.java** (p. 32)

Description: Servlet utilizado para generar un paper de un Proyecto Terminal en formato PDF

Autor:

Maritza Hernández Arias

Versión:

1.0 Servlet implementation class **FormularioTestPaperServlet** (p. 22)

5.4.2. Documentación de las funciones miembro

5.4.2.1. void com.pt2.pt2Web.controller.FormularioTestPaperServlet.doPost (HttpServletRequest *request*, HttpServletResponse *response*) throws ServletException, IOException [protected]

Ver también:

HttpServlet.doPost(HttpServletRequest request, HttpServletResponse response)

Se recuperan los valores ingresados por el usuario desde el Formulario

Los textos recibidos son procesados para que Latex pueda comprender ciertos caracteres, como lo son las letras con acentos o símbolos como pueden ser : '\', '', '_', etc.

La plantilla en Latex es leida. Posteriormente se buscan los valores a reemplazar (p. ej.) y son reemplados por los textos que ingreso el usuario en los apartados correspondientes

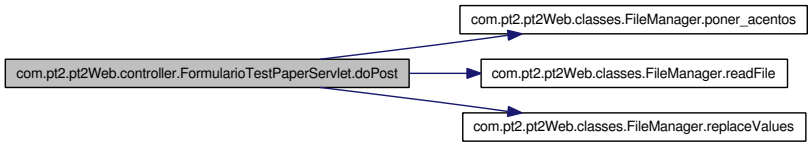
Se crea el archivo .tex fisicamente

Se compila el archivo .tex y se genera el archivo PDF

Se abre el documento PDF para mostrarlo al usuario

Redireccion a la pagina principal

Gráfico de llamadas para esta función:



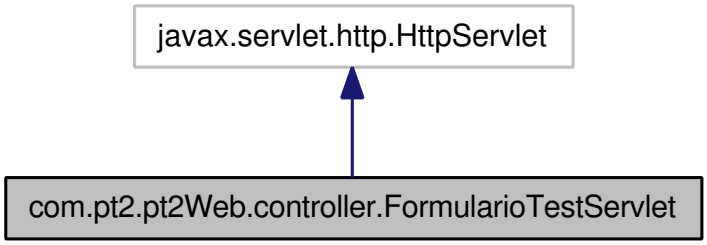
La documentación para esta clase fue generada a partir del siguiente fichero:

- /home/itza/ProyectoTerminal/**FormularioTestPaperServlet.java**

5.5. Referencia de la Clase com.pt2.pt2Web.controller.FormularioTestServlet

Herencias javax::servlet::http::HttpServlet.

Diagrama de colaboración para com.pt2.pt2Web.controller.FormularioTestServlet:



Métodos protegidos

- void **doPost** (HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

5.5.1. Descripción detallada

Title: **FormularioTestServlet.java** (p. 33)

Description: Servlet utilizado para generar un Reporte de Proyecto Terminal en formato PDF

Autor:

Maritza Hernández Arias

Versión:

1.0 Servlet implementation class **FormularioTestServlet** (p. 24)

5.5.2. Documentación de las funciones miembro

5.5.2.1. void com.pt2.pt2Web.controller.FormularioTestServlet.doPost (HttpServletRequest *request*, HttpServletResponse *response*) throws ServletException, IOException [protected]

Ver también:

HttpServlet.doPost(HttpServletRequest request, HttpServletResponse response)

Se recuperan los valores ingresados por el usuario desde el Formulario

Los textos recibidos son procesados para que Latex pueda comprender ciertos caracteres, como lo son las letras con acentos o símbolos como pueden ser : '\', '', '_', etc.

La plantilla en Latex es leida. Posteriormente se buscan los valores a reemplazar (p. ej.) y son reemplados por los textos que ingreso el usuario en los apartados correspondientes

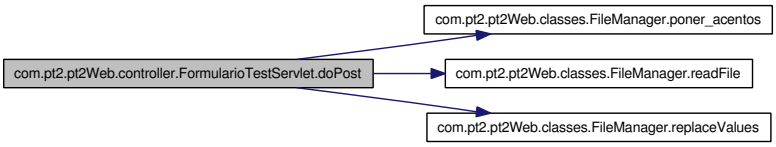
Se crea el archivo .tex fisicamente

Se compila el archivo .tex y se genera el archivo PDF

Se abre el documento PDF para mostrarlo al usuario

Redireccion a la páagina principal

Gráfico de llamadas para esta función:



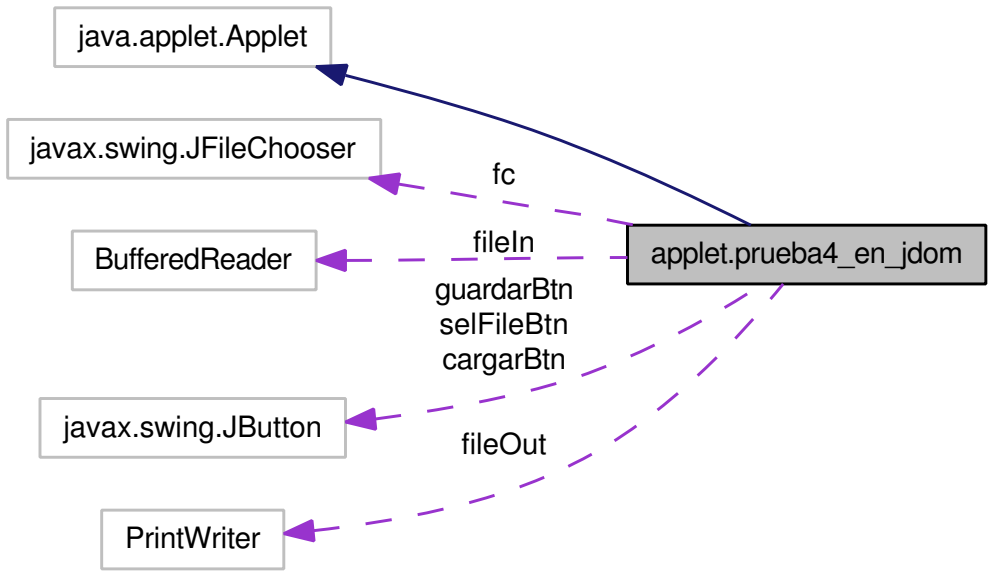
La documentación para esta clase fue generada a partir del siguiente fichero:

- /home/itza/ProyectoTerminal/**FormularioTestServlet.java**

5.6. Referencia de la Clase applet.prueba4_en_jdom

Herencias java::applet::Applet.

Diagrama de colaboración para applet.prueba4_en_jdom:



Clases

- class **Listener**

Métodos públicos

- **prueba4_en_jdom** ()
- void **init** ()
- void **setText** (String s)
- void **setParametrosDeFormulario** (String nombre_proyecto, String alumno, String fecha, String pal_clave, String obj_grales, String obj_part, String antecedentes, String justificacion, String desc_tec, String esp_tec, String desarrollo, String resultados, String biblio) throws ParserConfigurationException, Error, TransformerException

Atributos del Paquete

- BufferedReader **fileIn**
- PrintWriter **fileOut**
- Label **label1** = new Label("Seleccione XML")

5.6.1. Documentación del constructor y destructor

5.6.1.1. applet.prueba4_en_jdom.prueba4_en_jdom ()

5.6.2. Documentación de las funciones miembro

5.6.2.1. void applet.prueba4_en_jdom.init ()

5.6.2.2. void applet.prueba4_en_jdom.setParametrosDeFormulario (String *nombre_proyecto*, String *alumno*, String *fecha*, String *pal_clave*, String *obj_grales*, String *obj_part*, String *antecedentes*, String *justificacion*, String *desc_tec*, String *esp_tec*, String *desarrollo*, String *resultados*, String *biblio*) throws ParserConfigurationException, Error, TransformerException

5.6.2.3. void applet.prueba4_en_jdom.setText (String *s*)

5.6.3. Documentación de los datos miembro

5.6.3.1. BufferedReader applet.prueba4_en_jdom.fileIn [package]

5.6.3.2. PrintWriter applet.prueba4_en_jdom.fileOut [package]

5.6.3.3. Label applet.prueba4_en_jdom.label1 = new Label("Seleccione XML") [package]

La documentación para esta clase fue generada a partir del siguiente fichero:

- /home/itza/ProyectoTerminal/**prueba4_en_jdom.java**

5.7. Referencia de la Clase applet.prueba4_en_jdom.prueba4_en_-jdom.Listener

Métodos públicos

- void **actionPerformed** (ActionEvent e)

5.7.1. Documentación de las funciones miembro

5.7.1.1. void applet.prueba4_en_jdom.prueba4_en_jdom.Listener.actionPerformed (ActionEvent *e*)

La documentación para esta clase fue generada a partir del siguiente fichero:

- /home/itza/ProyectoTerminal/**prueba4_en_jdom.java**

Capítulo 6

Documentación de archivos

6.1. Referencia del Archivo /home/itza/ProyectoTerminal/Decrypt.java

Clases

- class **para_cifrar_p4.Decrypt**

Paquetes

- package **para_cifrar_p4**

6.2. Referencia del Archivo /home/itza/ProyectoTerminal/Encrypt.java

Clases

- class **para_cifrar_p4.Encrypt**

Paquetes

- package **para_cifrar_p4**

6.3. Referencia del Archivo /home/itza/ProyectoTerminal/FileManager.java

Clases

- class **com.pt2.pt2Web.classes.FileManager**

Paquetes

- package **com.pt2.pt2Web.classes**

6.4. Referencia del Archivo /home/itza/ProyectoTerminal/FormularioTestPaperSe

Clases

- class **com.pt2.pt2Web.controller.FormularioTestPaperServlet**

Paquetes

- package **com.pt2.pt2Web.controller**

6.5. Referencia del Archivo /home/itza/ProyectoTerminal/FormularioTestServlet.j

Clases

- class **com.pt2.pt2Web.controller.FormularioTestServlet**

Paquetes

- package **com.pt2.pt2Web.controller**

6.6. Referencia del Archivo /home/itza/ProyectoTerminal/prueba4_-en_jdom.java

Clases

- class **applet.prueba4_en_jdom**
- class **applet.prueba4_en_jdom.prueba4_en_jdom.Listener**

Paquetes

- package **applet**

Índice alfabético

Sistema interoperable para compartir información entre sistemas web con autocompletado de formularios

Maritza Hernández Arias

2011-03-27

Objetivos

Objetivos Generales

- Desarrollar una aplicación web que facilite la interoperabilidad de aplicaciones e intercambio de información entre sistemas web a través del lenguaje de intercambio de datos XML.
- Desarrollar dos sistemas web de prueba para mostrar su interoperabilidad.

Objetivos Particulares

- Desarrollar dos sistemas web que admitan la captura de información en formularios web y crear documentos con tipografía y estilo especificados en plantillas en formato Estos sistemas serán capaces de presentar la salida en formato portable PDF.
- Construir dos plantillas en formato LATEX para dos sistemas web que compartan información. La primera plantilla permitirá generar un reporte de proyecto terminal en formato PDF y la segunda un artículo (paper). Ambos tendrán información compartida.
- Desarrollar un módulo (Applet) ejecutable con soporte de entrada y salida de datos, capaz de almacenar información en un archivo XML de forma local.
- Integrar el módulo applet a los sistemas web para facilitar el almacenamiento y recuperación de información compartida en el formato de intercambio de datos XML.
- Desarrollar módulos (Servlets) encargados de procesar los formularios web y de generar los reportes correspondientes en formato PDF.
- Elaborar la documentación requerida para el sistema.

Antecedentes

Los navegadores web vistos como clientes en un modelo cliente-servidor basados en el protocolo HTTP que implementan las últimas capas de los modelos TCP/IP o de referencia OSI, tienen distintas funciones. Una de ellas es la de interpretar el formato de los documentos HTML y proveer la presentación de los datos; otra la de proveer una interfaz con el usuario que desea capturar información en formularios web. Dichos datos son codificados para ser enviados a través del protocolo http mediante los métodos GET y POST (Métodos de petición de información).

El cliente web tiene otras funcionalidades pero también limitaciones determinadas por la arquitectura misma, una de ellas es que es un programa de aplicación tipo ”View only”, donde la prioridad es la presentación (salida) de la información pero no la captura (entrada) de datos, las reglas de ”sandbox”imponen restricciones de seguridad bajo este esquema. Así, el cliente web en su forma más elemental, no posee capacidad de procesamiento, pero ante la creciente necesidad de procesar datos, se han provisto varias tecnologías para dotar a los navegadores de nuevas funcionalidades.

Los mecanismos básicos para soporte de sesiones son los cookies que son archivos almacenados localmente en la computadora cliente y las sesiones que almacenan información del cliente del lado del servidor. Estas tecnologías son provistas por los navegadores en sus configuraciones por defecto. Existen otros módulos configurables conocidos como de .ªutocompletado de formularios”que administran claves de usuario y datos de formularios que facilitan la captura de información y que también son parte de los navegadores. Estos módulos no son compatibles entre navegadores.

Existen productos como RoboForm que es un administrador sofisticado de contraseñas y complementador de formularios que ayuda a generar contraseñas y que permiten almacenar la información en memoria secundaria (archivos) y que permite compartir información entre distintos navegadores. Sin embargo, estas aplicaciones no emplean formatos estándar para intercambio de datos (como XML) o no permiten exportar la información a otros formatos como Portable Document File (PDF), lenguajes de marcas HTML o LATEX, ODT, DOC, etc., con los cuales se podrían generar documentos o reportes sin necesidad de conectarse al servidor de web para descargar el formulario.

Se propone construir una aplicación ejecutable del lado del cliente que almacene los datos encriptados en forma local en formato XML, detecte los campos de los formularios y los llene en forma automática y que trabaje en conjunción con aplicaciones del lado del servidor los cuales pueden exportar la información en formatos como PDF y LATEX. Esta arquitectura permite integrar aplicaciones web que comparten información capturada en formularios web, lo cual previene el tener que volver a capturarla una y otra vez exponiéndola a posibles virus, intrusos, keyloggers y malware en general.

La arquitectura web modificada, propuesta en este trabajo, no ha sido explorada anteriormente y la compartición de información se ha planteado como uno de los desafíos de la ”Integración de Aplicaciones Empresariales”, EAI por sus siglas en inglés.

Justificación

Existen varios sistemas web, como ejemplos se mencionan los portales de PROMEP (Programa de Mejoramiento del Profesorado) para proyectos de investigación y de CONACYT (Consejo Nacional de Ciencia y Tecnología) para el Sistema Nacional de Investigadores, en los cuales se debe capturar una gran cantidad de información en formularios web que, en el mejor caso, exportan al formato DOC pero a partir de ese documento no es posible llenar en forma automática cualquier otro sistema que requiera la misma información. Una posible solución, es que entre las instituciones involucradas compartan e integren directamente la información, asumiendo que pueden empatarla. Puesto que esta alternativa aún no ha sido implementada, no sólo entre estas dos instituciones, sino entre muchas más, la información específica de cada institución (portal web) debe ser capturada por separado. Cabe mencionar que los sistemas anteriores no se usarán en el presente trabajo.

Concretamente, se propone usar dos sistemas de prueba los cuales comparten información que se capturará mediante formularios web. El primer sistema de información almacenará los datos en forma local en memoria secundaria, donde se ejecuta el cliente web, con la finalidad de que el segundo sistema de información al solicitar un subconjunto de los datos (compartidos) previamente capturados, puedan ser llenados en forma automática. El resto de los datos que hagan falta para el segundo sistema de información deben ser completados por el usuario. El formato para intercambio de datos será XML (Extensible Markup Language Â«lenguaje de marcas ampliableÂ»). La figura 1 ilustra lo anterior y se aprecia que existe un traslape de información que es común a ambos sistemas, a manera de ejemplo se comenta un documento de proyecto terminal que tiene un título al igual que un artículo científico que tiene el mismo título.

Una posible continuación a este proyecto sería que el documento XML tenga acceso restringido a ciertos datos, hablando entonces de niveles de seguridad de información personal. En un primer nivel, podrían estar los datos más comunes (no confidenciales) tales como nombre, apellido, email. En un segundo nivel, se tendrían datos como fecha de nacimiento, dirección postal, etc. Y así, sucesivamente. De tal forma que el usuario pueda dar permiso para abrir los diferentes niveles de seguridad en los datos con una llave de encriptamiento distinta para cada nivel.

Este proyecto no podría ser concluido satisfactoriamente por una persona que no tuviera un perfil de un Ingeniero en computación ya que en este proyecto se conjunta una serie de conocimientos que se entrelazan para poder realizar una serie de actividades, como lo son el agregar capacidad de almacenamiento a un sistema web a través de un applet, el interoperar distintos sistemas, utilizar algoritmos de encriptado, entre otros temas que fueron admitidos durante la formación universitaria en base a práctica y experiencia.

Descripción Técnica

El ”Sistema interoperable para compartir información entre sistemas web con autocompletado de formularios”será utilizado por medio de una interfaz web con la finalidad de ayudar al usuario a ingresar información personalizada. De tal forma que cuando un usuario normalmente, tiene que ingresar su información una y otra vez (volviendo el proceso de captura engorroso), sólo tendría que insertar su información a través de un archivo.

El sistema propuesto se utilizará en sistemas que admiten información en formularios web, así, en este sistema se insertará el módulo (applet) capaz de leer y escribir información en un archivo con formato XML.
El sistema contará con los siguientes módulos para su adecuado funcionamiento:

Obtención de información: La información se obtendrá a partir de formularios web y a partir de estos datos se creará el archivo XML. Para este módulo es necesario crear los sistemas de prueba que, en este caso, admitirán información que se requiere en un reporte de Proyecto Terminal e información para crear un paper del mismo tema.

Procesador de la información: Este módulo comprende la implementación del applet que permitirá la lectura y escritura en un archivo con formato XML, para lo cual el usuario deberá permitir al navegador la descarga del applet en la computadora. La escritura se realizará al momento en que el usuario envíe la información (a través de un botón tipo submit), entonces los datos enviados se escribirán en un archivo XML. La lectura de los datos se llevará a cabo cuando el sistema muestre un formulario en blanco y el usuario, a través del applet, busqué el archivo XML que contiene su información, es así como los campos del formulario serán completados según la información que contenga el archivo XML.

Almacenamiento de la información: Este módulo estará dado por un archivo XML, en él se escribirán los datos que el usuario ingrese desde el formulario web del sistema.

El uso del archivo se describe a continuación:

Cuando un usuario envíe datos de un primer formulario, éstos serán guardados en el archivo XML cuya ubicación es el ordenador. Así, al ingresar al segundo sistema cuyo formulario tenga un subconjunto de datos comunes al primero; el usuario, a través del applet buscará el archivo y los campos coincidentes serán ingresados automáticamente. Los campos que queden vacíos tendrán que ser ingresados por el usuario, nuevamente al enviar la información a través del submit se procesa la información y los nuevos datos serán ingresados al mismo archivo XML.

De tal forma que el archivo puede considerarse como un recolector”de información, permitiendo al usuario que entre a un nuevo sistema, llenar el menor número de campos posible sino es que ninguno; sólo bastará con cargar su información del archivo.

Presentación de la información: Los sistemas tendrán la opción de presentar las salidas en archivos con formato PDF, éstos se presentarán a partir de plantillas prediseñadas.

Especificaciones Técnicas

El desarrollo de este proyecto se realizará bajo la plataforma Linux.

Del lado del cliente. Se incluye al navegador con formularios web en lenguaje HTML, con un programa applet en lenguaje Java que almacena y recupera los datos capturados en los formularios mediante JavaScript para llevarlos a archivos encriptados con Data Encryption Standar (DES) y procesarlos en XML usando las bibliotecas de Java JDOM.

Contenedor de servlets. Existen varios contenedores, uno de los más ampliamente usados es Tomcat y se usará en este proyecto.

Del lado del servidor. Se desarrollará un programa en lenguaje Java tipo servlet distinto para cada uno de esos sistemas de información. En general se pueden tener tantos sistemas de información como se desee siempre que sea conveniente que compartan información. Los programas servlet generarán archivos en formato LATEX a través del sistema java.io que serán procesados con el programa ”GNU/latex”para obtener archivos en formato DVI a partir del cual se genera el documento PDF distinto para cada sistema de información para lo cual se usará el programa ”GNU/dvipdfm”.

El primer sistema admitirá información que se requiere en un reporte de Proyecto Terminal y el segundo admitirá información requerida para crear un borrador de un paper del mismo tema que el proyecto terminal 1

La comunicación entre los clientes y servidores web se hace a través de HTTP.

Desarrollo

Para un desarrollo controlado del proyecto, éste se divide en 5 etapas. Las dos primeras corresponden al periodo del primer trimestre del Proyecto Terminal y las siguientes 3 etapas se desarrollarán durante el segundo trimestre del Proyecto Terminal.

La primera etapa consistió de las siguientes actividades o tareas:

a. Analizar cuáles son los requerimientos apropiados de información para el reporte de un Proyecto Terminal.

b. Desarrollar un sistema web que admita información pertinente a un reporte de Proyecto Terminal por medio de un formulario.

c. Analizar cuáles son los requerimientos básicos de información para la elaboración de un paper enfocado a la publicación de un Proyecto Terminal.

d. Desarrollar un sistema web que admita información pertinente a un paper, con contenido referente a la publicación de un Proyecto Terminal por medio de un formulario.

La segunda etapa consistió en realizar las siguientes actividades o tareas:

a. Investigar la presentación que debe tener un reporte de Proyecto Terminal.

b. Realizar una plantilla en formato LATEX que genere la presentación de un reporte de proyecto Terminal.

c. Investigar la presentación que debe tener un paper enfocado a la publicación de un Proyecto Terminal.

d. Realizar una plantilla en formato LATEX que genere la presentación de un paper referente a la publicación de un proyecto Terminal.

e. Documentación de las etapas realizadas en Proyecto Terminal 1.

La tercera etapa se llevó acabo por medio de las siguientes tareas o actividades:

a. Implementación de un algoritmo cifrado que encriptará la información ingresada por el usuario.

b. Implementación de un algoritmo que permita descifrar la información del archivo XML, haciendo posible obtener la información real del usuario.

La cuarta etapa consistirá en realizar las siguientes actividades:

a. Desarrollo del módulo applet con soporte de entrada y salida de datos capaz de almacenar información en un archivo XML (encriptado).

b. Integrar el módulo applet a los sistemas de información para el intercambio de datos.

La quinta etapa consistiré en realizar las siguientes tareas o actividades:

a. Desarrollar el módulo servlet responsable de generar los archivos latex en base a las plantillas prediseñadas para los sistemas correspondientes.

b. Convertir los archivos latex a archivos en formato DVI para su posterior salida en archivos PDF.

c. Documentación de las etapas realizadas en Proyecto Terminal 2.

Resultados

Se generaron 2 Sistemas Web de prueba, el primero presenta un formulario con información pertinente a un Reporte de Proyecto Terminal, el segundo Sistema presenta otro formulario con información pertinente a un paper enfocado a la publicación de un Reporte de Proyecto Terminal se requerían formularios con información en común. En los formularios mencionados anteriormente se debe incrustar el código que contiene al applet, se le indica mediante el tag <applet> </applet>, como se muestra a continuación:

```
<APPLET
    code = " applet.prueba4_en_jdom.class"
    archive = "prueba_en_jdom.jar,jdom.jar,commons-logging.jar,serializer.jar,xalan.jar,xercesImpl.jar,
              xml-apis.jar,xmlsec-1.4.3.jar"
    name = "applet"
    width = "700"
    height = "220"
    align = "middle">
</APPLET>
```

Donde:

code : Indica el nombre de la clase que contiene el applet.

archive : Indica los jars que debe de cargar al navegador, ej. archive = " uno.jar, dos.jar, . . . "

name : Indica el nombre del applet.

width : Indica el ancho que ocupará el applet en la página.

height : Indica el alto que ocupará el applet en la página

align : Indica la alineación que tendrá el applet.

Asímismo, se generaron 2 plantillas en formato latex para la posterior generación de reportes de ambos Sistemas antes mencionados.

Estas plantillas contienen el fomato de report y article según corresponda y se insertaron palabras clave para posteriormente ser reemplazadas por el texto que ingresa el usuario.

Posteriormente se elaboró la etapa 4, antes de la etapa 3 por conveniencia, ya que la etapa 3 habla de cifrado de información, información ingresada a través del applet, el cual es mencionado en la etapa 4.

La cuarta etapa consistió en implementar el applet.
El IDE utilizado para la elaboración del applet es Eclipse una vez creado el proyecto se debe generar un archivo **.jar**, en Eclipse se realiza de la siguiente manera:

file / Export. . . / JAR File / botón Next / Elegir ruta y poner nombre al archivo jar

Una vez creado se debe de firmar, así cuando se ejecute en el navegador, saldrá un mensaje preguntando al cliente si da permiso al applet para ser ejecutado.

Se firmó de la siguiente manera:

Posicionarse en la carpeta bin del jdk.

Escribir en una ventana de comandos:

keytool -genkey -alias *mi_alias*

Este comando genera una clave que será utilizada para firmar el .jar, donde se pedirá como clave lo que escribimos como textitmi_alias
Después se desplegará un cuestionario, debe ser contestado y finalmente indicar si/yes

Finalmente, posicionarse donde se encuentra el .jar y escribir el siguiente comando:

jarsigner archivo.jar *mi_alias*

Pedirá una vez más la clave keystore (*mi_alias*), insértarla nuevamente, y listo, el archivo .jar ha sido firmado.

La descripción del código se muestra y explica a continuación:

```
1  package applet;
2
3  import java.applet.Applet;
4  import java.io.BufferedReader;
5  import java.io.File;
6  import java.io.FileInputStream;
7  import java.io.FileOutputStream;
8  import java.io.InputStream;
9  import java.io.PrintWriter;
10
11 import org.apache.xml.security.encryption.EncryptedData;
12 import org.apache.xml.security.encryption.EncryptedKey;
13 import org.apache.xml.security.encryption.XMLCipher;
14 import org.apache.xml.security.keys.KeyInfo;
15 import org.jdom.Document;
16 import org.jdom.Element;
17 import org.jdom.input.SAXBuilder;
18 import org.jdom.output.Format;
19 import org.jdom.output.XMLOutputter;
20 import org.w3c.dom.Node;
```

```
21
22 import java.awt.*;
23 import java.awt.event.*;
24
25 import java.net.URL;
26 import java.security.Key;
27
28 import javax.swing.ImageIcon;
29 import javax.swing.JFileChooser;
30 import javax.swing.JOptionPane;
31 import javax.swing.JButton;
32
33 import javax.xml.parsers.ParserConfigurationException;
34 import javax.xml.transform.OutputKeys;
35 import javax.xml.transform.Transformer;
36 import javax.xml.transform.TransformerException;
37 import javax.xml.transform.TransformerFactory;
38 import javax.xml.transform.dom.DOMSource;
39 import javax.xml.transform.stream.StreamResult;
40
41 import para_cifrar_p4.Encrypt;
42 import para_cifrar_p4.Decrypt;
43
44
45 public class prueba4_en_jdom extends Applet {
46
47   public prueba4_en_jdom() {
48
49   }
50
51   private GridBagLayout gbl = new GridBagLayout();
52   private GridBagConstraints restriccion = new GridBagConstraints();
53
54   private JButton selFileBtn = new JButton("Seleccionar Archivo");
55
56   private TextField nomFileTxt = new TextField("", 20);
57
58   private JButton cargarBtn = new JButton("Cargar Datos");
59
60   private JButton guardarBtn = new JButton("Guardar Datos");
61   private TextArea areaTxt = new TextArea();
62
63   private JFileChooser fc = new JFileChooser();
64
65   BufferedReader fileIn;
66   PrintWriter fileOut;
67
68   Label label1 = new Label("Seleccione XML");
69
70   public void init() {
71     /* Se define el gestor de disposicion */
72     setLayout(gbl);
73
74     /* *** FILA 1 (2 COLUMNAS) *** */
75     /* Componente 1 (Seleccionar archivo)*/
76     restriccion.weightx = 1.0;
77     restriccion.fill = GridBagConstraints.BOTH;
78     gbl.setConstraints(selFileBtn, restriccion);
79     add(selFileBtn);
80     selFileBtn.addActionListener(new Listener());
81
```

```
82      // Componente 2 (Campo de texto. Nombre del archivo)
83      restriccion.gridwidth = GridBagConstraints.REMAINDER;
84      gbl.setConstraints(nomFileTxt, restriccion);
85      add(nomFileTxt);
86
87      // *** FILA 2 (2 COLUMNAS) ***
88      // Componente 3 (Cargar datos en el area de texto)
89      restriccion.weightx = 1.0;
90      restriccion.fill = GridBagConstraints.BOTH;
91      restriccion.gridwidth = 1;
92      gbl.setConstraints(cargarBtn, restriccion);
93      add(cargarBtn);
94      cargarBtn.addActionListener(new Listener());
95
96      // Componente 4 (Guardar Datos)
97      restriccion.gridwidth = GridBagConstraints.REMAINDER;
98      gbl.setConstraints(guardarBtn, restriccion);
99      add(guardarBtn);
100     guardarBtn.addActionListener(new Listener());
101
102     // *** FILA 3 (1 columna) ***
103     // Componente 5
104     restriccion.weightx = 0.0;
105     gbl.setConstraints(areaTxt, restriccion);
106     add(areaTxt);
107
108 }
109
110 public void setText(String s) {
111   nomFileTxt.setText(s);
112 }
113
114 public void setParametrosDeFormulario(String nombre_proyecto, String alumno,
115         String fecha, String pal_clave, String obj_grales, String obj_part, String
              antecedentes,
116         String justificacion, String desc_tec, String esp_tec, String desarrollo,
117         String resultados, String biblio)
118     throws ParserConfigurationException, Error, TransformerException
119 {
120
121         String datosRecibidos = "nombre_proyecto: " +  nombre_proyecto +
122                 ", alumno: " +  alumno + ", fecha: " + fecha + ", pal_clave:"+
                      pal_clave +
123                 ", obj_grales: " +  obj_grales +
124                 ", obj_part: " + obj_part + ", antecedentes: " + antecedentes +
125                 ", justificacion: " +  justificacion + ", desc_tec: " + desc_tec +
126                 ", esp_tec: " + esp_tec + ", desarrollo: " +  desarrollo +
127                 ", resultados: " + resultados + ", biblio: " + biblio;
128
129
130         System.out.println("Datos: " + datosRecibidos);
131         //JOptionPane.showInputDialog(datosRecibidos);
132         areaTxt.setText(datosRecibidos);
133
134
135         //Comienza creacion del archivo
136         Document newDoc = new Document();
137
138     Element eRoot = new Element("proyecto");
139     newDoc.addContent(eRoot);
140
```

```
141         Element nombre_proyecto_aux = new Element("nombre_proyecto").setText(
                nombre_proyecto);
142         eRoot.addContent(nombre_proyecto_aux);
143 
144         Element alumno_aux = new Element("alumno").setText(alumno);
145         eRoot.addContent(alumno_aux);
146 
147         Element fecha_aux = new Element("fecha").setText(fecha);
148         eRoot.addContent(fecha_aux);
149 
150         Element pal_clave_aux = new Element("pal_clave").setText(pal_clave);
151         eRoot.addContent(pal_clave_aux);
152 
153         Element obj_grales_aux = new Element("obj_grales").setText(obj_grales);
154         eRoot.addContent(obj_grales_aux);
155 
156         Element obj_part_aux = new Element("obj_part").setText(obj_part);
157         eRoot.addContent(obj_part_aux);
158 
159         Element antecedentes_aux = new Element("antecedentes").setText(antecedentes);
160         eRoot.addContent(antecedentes_aux);
161 
162         Element justificacion_aux = new Element("justificacion").setText(justificacion
                );
163         eRoot.addContent(justificacion_aux);
164 
165         Element desc_tec_aux = new Element("desc_tec").setText(desc_tec);
166         eRoot.addContent(desc_tec_aux);
167 
168         Element esp_tec_aux = new Element("esp_tec").setText(esp_tec);
169         eRoot.addContent(esp_tec_aux);
170 
171         Element desarrollo_aux = new Element("desarrollo").setText(desarrollo);
172         eRoot.addContent(desarrollo_aux);
173 
174         Element resultados_aux = new Element("resultados").setText(resultados);
175         eRoot.addContent(resultados_aux);
176 
177         Element biblio_aux = new Element("biblio").setText(biblio);
178         eRoot.addContent(biblio_aux);
179 
180 
181         Format format =Format.getPrettyFormat();
182           try{
183                 XMLOutputter out = new XMLOutputter(format);
184                 File file = new java.io.File(nomFileTxt.getText());
185 
186                 FileOutputStream file_out=new FileOutputStream(file);
187                 out.output(newDoc,file_out);
188                 file_out.flush();
189                 file_out.close();
190                 out.output(newDoc,System.out);
191                 /*
                        ------------------------------------------------------------------------
                        */
192                 JOptionPane.showMessageDialog(null, " El archivo XML ha sido creado ")
                        ;
193 
194                 String name_file = nomFileTxt.getText();
195 
196                 JOptionPane.showMessageDialog(null, "el nombre es "+name_file);
```

```
197
198
199             String name_file_encrypt;
200
201             if(name_file.contains("_encrypt.xml"))
202             {
203                name_file_encrypt = name_file; //nomFileTxt.getText().replace("
                      _encrypt.xml", "_encrypt.xml");
204             }
205             else
206             {
207                name_file_encrypt = nomFileTxt.getText().replace(".xml", "_encrypt.
                      xml");
208             }
209             JOptionPane.showMessageDialog(null, "el nombre2 es "+name_file_encrypt
                   );
210
211             String [] args = {name_file, name_file_encrypt};
212
213             Encrypt.main(args);
214
215             JOptionPane.showMessageDialog(null, "ya se encripto ");
216
217             if(name_file.equals(name_file_encrypt))
218             {
219               ;//file.delete();
220             }
221             else
222             {
223               ;//file.delete();
224             }
225
226
227             //JOptionPane.showMessageDialog(null, "Eliminado archivo xml simple ")
                   ;
228             /*
                   ----------------------------------------------------------------------
                   */
229       }
230         catch(Exception e1){
231           e1.printStackTrace();
232         }
233
234
235 }
236
237
238
239 public class Listener implements ActionListener{
240
241   public void actionPerformed(ActionEvent e){
242
243     if (e.getSource() == selFileBtn) {
244
245       int returnVal = fc.showOpenDialog(prueba4_en_jdom.this);
246
247       if (returnVal == JFileChooser.APPROVE_OPTION) {
248         File file = fc.getSelectedFile();
249         nomFileTxt.setText("" + file);
250       } else {
251         ;
```

```
252         }
253
254       } else if (e.getSource() == cargarBtn) {
255         try {
256
257           if (!nomFileTxt.getText().equals("")) {
258
259             File file = fc.getSelectedFile();
260
261             String nombre_actual_archivo = nomFileTxt.getText();
262             String nombre_archivo_descifrado = nomFileTxt.getText().replace("
                  _encrypt.xml", "_decrypt.xml");
263
264             /* Aqui debe descifrar el archivo para poder utilizarlo...*/
265             if (nomFileTxt.getText().contains("_encrypt"))
266             {
267
268             /*String nombre_actual_archivo = nomFileTxt.getText();
269             String nombre_archivo_descifrado = nomFileTxt.getText().replace("
                  _encrypt.xml", "_decrypt.xml");*/
270
271             String [] arg = {nombre_actual_archivo, nombre_archivo_descifrado};
272
273             Decrypt.main(arg);
274
275
276
277             }
278             /*-------------------------------------------------------*/
279
280             //Ahora se selecciona el archivo descifrado para poder obtener su
                  informacion y mostrarla
281             File file2 = new File (nomFileTxt.getText().replace("_encrypt.xml", "
                  _decrypt.xml"));
282
283             SAXBuilder builder = new SAXBuilder();
284
285             Document doc = builder.build(file2);
286
287             Element root = doc.getRootElement();
288
289             /*
290             List<Element> allChildren = root.getChildren();
291             JOptionPane.showMessageDialog(null,"Se obtiene la lista ");
292
293             for (Element hijo: allChildren )
294             {
295               String nombre = hijo.getNadatos.xmlme();
296               String texto = hijo.getValue();
297
298               System.out.println("Etiqueta: "+nombre +". Texto: "+texto);
299               JOptionPane.showMessageDialog(null,"Etiqueta: "+nombre +". Texto: "+
                    texto);
300             }
301             */
302
303             String nombre_pt = root.getChildText("nombre_proyecto");
304
305             String alumno = root.getChildText("alumno");
306
307             String fecha = root.getChildText("fecha");
```

```
308
309             String pal_clave= root.getChildText("pal_clave");
310
311             String obj_grales = root.getChildText("obj_grales");
312
313             String obj_part = root.getChildText("obj_part");
314
315             String antec = root.getChildText("antecedentes");
316
317             String justif = root.getChildText("justificacion");
318
319             String desc_tec = root.getChildText("desc_tec");
320
321             String esp_tec = root.getChildText("esp_tec");
322
323             String desarrollo = root.getChildText("desarrollo");
324
325             String resultados = root.getChildText("resultados");
326
327             String biblio = root.getChildText("biblio");
328
329             areaTxt.setText("Datos:"+nombre_pt);
330
331             System.out.println("Datos:"+nombre_pt+","+alumno+","+fecha+", "+
                    pal_clave+","+obj_grales+","+obj_part+","+antec
332                  +","+justif+","+desc_tec+","+esp_tec+","+desarrollo+","+resultados+"
                        , "+biblio);
333
334             System.out.println();
335             System.out.println("Aki solo el nombre del proyecto:"+nombre_pt);
336
337             areaTxt.setText("Datos:"+nombre_pt+","+alumno+","+fecha+", "+pal_clave+"
                    ,"+obj_grales+","+obj_part+","+antec
338                  +","+justif+","+desc_tec+","+esp_tec+","+desarrollo+","+resultados+"
                        ,"+biblio);
339
340             //Para mostrar en la pÃ¡gina html
341             getAppletContext().showDocument(new URL("javascript:carga(\"" +
                    nombre_pt+"\",\""+alumno+"\",\""+fecha
342                  +"\",\""+pal_clave+"\",\""+obj_grales+"\",\""+obj_part+"\",\""+antec
                        +"\",\""+justif
343                  +"\",\""+desc_tec+"\",\""+esp_tec+"\",\""+desarrollo+"\",\""+
                        resultados+"\",\""+biblio+"\")"));
344
345             //Se elimina el archivo descifrado
346             File file3 = new File(nombre_archivo_descifrado);
347             file3.delete();
348           }
349
350           else {
351             JOptionPane.showMessageDialog(null,
352                  "Debe seleccionar un archivo.");
353           }
354
355
356         }
357         catch (Exception ex) {
358           //JOptionPane.showMessageDialog(null, "Debe seleccionar un archivo
                excepcion ");
359           //System.err.println("Error: " + ex);
360           //ex.printStackTrace();
```

```
361           }
362
363         } else if (e.getSource() == guardarBtn){
364
365               try{
366
367                 if (!nomFileTxt.getText().equals("")) {
368                   if(!nomFileTxt.getText().contains(".xml"))
369                   {
370                     JOptionPane.showMessageDialog(null, "El nombre de su archivo debe
                          contener el formato 'ruta/nombre_archivo.xml'");
371                   }
372                   else
373                   {
374                     String nombre_de_archivo = nomFileTxt.getText();
375                     if(nombre_de_archivo.contains(".xml") && !nombre_de_archivo.
                          contains("_encrypt.xml"))
376                     {
377                       nombre_de_archivo = nombre_de_archivo.replace(".xml", "_encrypt.
                            xml");
378                     }
379                     else if(nombre_de_archivo.contains("_encrypt.xml"))
380                     {
381                       ;//nombre_de_archivo = nombre_de_archivo.replace("_encrypt.xml",
                            "_encrypt.xml");
382                     }
383                     JOptionPane.showMessageDialog(null, "Su archivo cifrado se
                          guardarÃ¡ como: "+ nombre_de_archivo);
384                     getAppletContext().showDocument(new URL("javascript:
                          obtenerDatosDeFormulario()"));
385                   }
386                 }
387                 else
388                 {
389                   JOptionPane.showMessageDialog(null, "Debe escribir el nombre con el
                        que desea guardar el archivo. (Puede mencionar la ruta/path)");
390                 }
391
392             }
393               catch(Exception e1){
394                 e1.printStackTrace();
395               }
396
397         }
398       }// fin de actionPerformed
399     }
400 }
```

La información ingresada por el usuario desde alguno de los formularios de los Sistemas Web, es cifrada para una mayor seguridad, así, el procedimiento es el siguiente:

- El usuario ingresa a uno de los Formularios (ya sea Formulario_reporte.jsp o Formulario_paper.jsp)
- Ingresa la información solicitada
- En el applet, ingresa la ruta y nombre del archivo donde se guardará la información (con el siguiente formato: \ruta\nombre_archivo.xml)
- El archivo xml es generado y posteriormente es cifrado, se genera una llave en un archivo que se guarda en el disco (sólo con esta llave el archivo podrá ser descifrado.)
- Se guarda el archivo con el siguiente formato: \ruta\nombre_archivo_encrypt.xml. El programa le concatena la cadena "_encrypt.xml "
- Posteriormente el usuario puede ingresar al otro Formulario, y podrá llenar los campos coincidentes de ambos formularios, con el archivo generado anteriormente.
- Así, el usuario oprime el botón "seleccionar archivo "del applet, busca el archivo nombre_archivo_enc
- Después oprime el botón " Cargar datos " y entonces los campos que contengan información coincidente con el sistema anterior, serán autocompletados, donde anterior a este paso la información es descifrada (leyendo y verificando la clave en el archivo).

A partir de este punto el usuario podrá volver a guardar otro archivo si es que cambia la información o desea añadir más.

El código de Cifrado se muestra a continuación:

```
1  package para_cifrar_p4;
2
3  import java.io.File;
4  import java.io.FileOutputStream;
5  import java.io.IOException;
6
7  import java.security.Key;
8
9  import javax.crypto.SecretKey;
10 import javax.crypto.KeyGenerator;
11
12 import org.apache.xml.security.keys.KeyInfo;
13 import org.apache.xml.security.encryption.XMLCipher;
14 import org.apache.xml.security.encryption.EncryptedData;
15 import org.apache.xml.security.encryption.EncryptedKey;
16
17 import org.w3c.dom.Document;
18 import org.w3c.dom.Element;
19
20 import javax.swing.JOptionPane;
21 import javax.xml.transform.TransformerFactory;
22 import javax.xml.transform.Transformer;
23 import javax.xml.transform.dom.DOMSource;
24 import javax.xml.transform.stream.StreamResult;
25 import javax.xml.transform.OutputKeys;
26
27 /**
28 * La clase Encrypt lee la entrada de un archivo, cifra el contenido
```

```
29    * del archivo y, a continuacion, almacena el archivo cifrado en el disco. Con el
          fin de
30    * lograr esto, la herramienta utiliza el marco de Apache XML para crear dos
31    * claves simÃ©tricas para los siguientes fines:
32    * 1) para cifrar los datos reales del archivo XML
33    * 2) para cifrar la clave utilizada para cifrar los datos del archivo XML
34    *
35    * Los datos cifrados se escriben en el disco y la clave utilizada para cifrar la
36    * clave de encriptacion de datos tambien se almacena en el disco.
37
38  */
39
40
41  public class Encrypt
42  {
43     static
44     {
45        org.apache.xml.security.Init.init();
46     }
47
48     public static Document parseFile(String fileName)
49        throws Exception
50     {
51        javax.xml.parsers.DocumentBuilderFactory dbf =
52           javax.xml.parsers.DocumentBuilderFactory.newInstance();
53        dbf.setNamespaceAware(true);
54        javax.xml.parsers.DocumentBuilder db = dbf.newDocumentBuilder();
55        Document document = db.parse(fileName);
56
57        return document;
58     }
59
60     public static SecretKey GenerateKeyEncryptionKey()
61        throws Exception
62     {
63        String jceAlgorithmName = "DESede";
64        KeyGenerator keyGenerator =
65           KeyGenerator.getInstance(jceAlgorithmName);
66        SecretKey keyEncryptKey = keyGenerator.generateKey();
67
68        return keyEncryptKey;
69     }
70
71     public static void storeKeyFile(Key keyEncryptKey)
72        throws IOException
73     {
74        byte[] keyBytes = keyEncryptKey.getEncoded();
75        File keyEncryptKeyFile = new File("keyEncryptKey");
76        FileOutputStream outStream = new FileOutputStream(keyEncryptKeyFile);
77        outStream.write(keyBytes);
78        outStream.close();
79
80        System.out.println("Key encryption key stored in: "
81                           + keyEncryptKeyFile.toURL().toString());
82     }
83
84     public static SecretKey GenerateSymmetricKey()
85        throws Exception
86     {
87        String jceAlgorithmName = "AES";
88        KeyGenerator keyGenerator =
```

```
89          KeyGenerator.getInstance(jceAlgorithmName);
90       keyGenerator.init(128);
91       return keyGenerator.generateKey();
92    }
93
94    public static void writeEncryptedDocToFile(Document doc,
95                                                String fileName)
96       throws Exception
97    {
98       File encryptionFile = new File(fileName);
99       FileOutputStream outStream =
100         new FileOutputStream(encryptionFile);
101
102      TransformerFactory factory = TransformerFactory.newInstance();
103      Transformer transformer = factory.newTransformer();
104      transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION,
105                                    "no");
106      DOMSource source = new DOMSource(doc);
107      StreamResult result = new StreamResult(outStream);
108      transformer.transform(source, result);
109
110      outStream.close();
111
112      System.out.println("Encrypted XML document written to: " +
113                         encryptionFile.toURL().toString());
114   }
115
116   public static void usage()
117   {
118      System.err.println("        - java EncryptTool "
119                         + "infilename outfilename elementtoencrypt");
120      System.err.println("example - java EncryptTool "
121                         + "test.xml encrypted.xml CreditCardNumber");
122   }
123
124   public static void main(String args[])
125      throws Exception
126   {
127    //JOptionPane.showMessageDialog(null, " Entra al main ");
128
129    if (args.length < 2)
130      {
131         usage();
132         System.exit(1);
133      }
134
135      // parse file into document
136      Document document = parseFile(args[0]);
137
138      // generate symmetric key
139      Key symmetricKey = GenerateSymmetricKey();
140
141      // Get a key to be used for encrypting the symmetric key
142      Key keyEncryptKey = GenerateKeyEncryptionKey();
143
144      // Write the key to a file
145      storeKeyFile(keyEncryptKey);
146
147      // initialize cipher
148      XMLCipher keyCipher =
149         XMLCipher.getInstance(XMLCipher.TRIPLEDES_KeyWrap);
```

```
150         keyCipher.init(XMLCipher.WRAP_MODE, keyEncryptKey);
151
152         // encrypt symmetric key
153         EncryptedKey encryptedKey = keyCipher.encryptKey(document,
154                                                          symmetricKey);
155
156         // specify the element to encrypt
157         Element rootElement = document.getDocumentElement();
158         Element elementToEncrypt = rootElement;
159         if (args.length > 2)
160         {
161            elementToEncrypt =
162               (Element)rootElement.getElementsByTagName(args[2]).item(0);
163            if (elementToEncrypt == null)
164            {
165               System.err.println("Unable to find element: " + args[2]);
166               System.exit(1);
167            }
168         }
169
170         // initialize cipher
171         XMLCipher xmlCipher =
172            XMLCipher.getInstance(XMLCipher.AES_128);
173         xmlCipher.init(XMLCipher.ENCRYPT_MODE, symmetricKey);
174
175         // add key info to encrypted data element
176         EncryptedData encryptedDataElement =
177            xmlCipher.getEncryptedData();
178         KeyInfo keyInfo = new KeyInfo(document);
179         keyInfo.add(encryptedKey);
180         encryptedDataElement.setKeyInfo(keyInfo);
181
182         // do the actual encryption
183         boolean encryptContentsOnly = true;
184         xmlCipher.doFinal(document,
185                           elementToEncrypt,
186                           encryptContentsOnly);
187
188         // write the results to a file
189         writeEncryptedDocToFile(document, args[1]);
190
191      }
192   }
```

El código de Descifrado de información es el siguiente:

```
1  package para_cifrar_p4;
2
3  import java.io.File;
4  import java.io.FileOutputStream;
5
6  import java.security.Key;
7
8  import javax.crypto.SecretKey;
9  import javax.crypto.SecretKeyFactory;
10 import javax.crypto.spec.DESedeKeySpec;
11
12 import org.apache.xml.security.encryption.XMLCipher;
13 import org.apache.xml.security.utils.JavaUtils;
14 import org.apache.xml.security.utils.EncryptionConstants;
15
16 import org.w3c.dom.Document;
17 import org.w3c.dom.Element;
18
19 import javax.xml.transform.TransformerFactory;
20 import javax.xml.transform.Transformer;
21 import javax.xml.transform.dom.DOMSource;
22 import javax.xml.transform.stream.StreamResult;
23 import javax.xml.transform.OutputKeys;
24
25 /**
26  * La clase Decrypt lee un archivo cifrado de disco,
27  * descifra el contenido del archivo con una clave previamente almacenada.
28  */
29
30 public class Decrypt
31 {
32    static
33    {
34       org.apache.xml.security.Init.init();
35    }
36
37    private static Document loadEncryptedFile(String fileName)
38       throws Exception
39    {
40       File encryptedFile = new File(fileName);
41       javax.xml.parsers.DocumentBuilderFactory dbf =
42          javax.xml.parsers.DocumentBuilderFactory.newInstance();
43       dbf.setNamespaceAware(true);
44       javax.xml.parsers.DocumentBuilder builder =
45          dbf.newDocumentBuilder();
46       Document document = builder.parse(encryptedFile);
47
48       System.out.println("Encryption document loaded from: " +
49                          encryptedFile.toURL().toString());
50       return document;
51    }
52
53    private static SecretKey loadKeyEncryptionKey()
54       throws Exception
55    {
56       String fileName = "keyEncryptKey";
57       String jceAlgorithmName = "DESede";
58
59       File kekFile = new File(fileName);
```

```
60
61          DESedeKeySpec keySpec =
62             new DESedeKeySpec(JavaUtils.getBytesFromFile(fileName));
63          SecretKeyFactory skf =
64             SecretKeyFactory.getInstance(jceAlgorithmName);
65          SecretKey key = skf.generateSecret(keySpec);
66
67          System.out.println("Key encryption key loaded from: "
68                             + kekFile.toURL().toString());
69          return key;
70       }
71
72       private static void writeDecryptedDocToFile(Document doc,
73                                                   String fileName)
74          throws Exception
75       {
76          File encryptionFile = new File(fileName);
77          FileOutputStream outStream = new FileOutputStream(encryptionFile);
78
79          TransformerFactory factory = TransformerFactory.newInstance();
80          Transformer transformer = factory.newTransformer();
81          transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION,
82                                        "no");
83          DOMSource source = new DOMSource(doc);
84          StreamResult result = new StreamResult(outStream);
85          transformer.transform(source, result);
86
87          outStream.close();
88
89          System.out.println("Decrypted data written to: " +
90                             encryptionFile.toURL().toString());
91       }
92
93       private static void usage()
94       {
95          System.err.println("usage - java DecryptTool "
96                             + "infile outfile");
97          System.err.println("example - java DecryptTool "
98                             + "encrypted.xml original.xml");
99       }
100
101      public static void main(String args[])
102         throws Exception
103      {
104         if (args.length <2)
105         {
106            usage();
107            System.exit(1);
108         }
109
110         // load the encrypted file into a Document
111         Document document = loadEncryptedFile(args[0]);
112
113         // get the encrypted data element
114         String namespaceURI = EncryptionConstants.EncryptionSpecNS;
115         String localName = EncryptionConstants._TAG_ENCRYPTEDDATA;
116         Element encryptedDataElement =
117            (Element)document.getElementsByTagNameNS(namespaceURI,
118                                                     localName).item(0);
119
120         // Load the key encryption key.
```

```
121         Key keyEncryptKey = loadKeyEncryptionKey();
122
123         // initialize cipher
124         XMLCipher xmlCipher = XMLCipher.getInstance();
125         xmlCipher.init(XMLCipher.DECRYPT_MODE, null);
126
127         xmlCipher.setKEK(keyEncryptKey);
128
129         // do the actual decryption
130         xmlCipher.doFinal(document, encryptedDataElement);
131
132         // write the results to a file
133         writeDecryptedDocToFile(document, args[1]);
134     }
135 }
```

La última y quinta etapa consistió en desarrollar un módulo servlet capaz de generar archivos latex en base a las plantillas predisen adas para los sistemas, para posteriormente convertirlos en archivos PDF y mostrarlos al cliente.

Se ultizó el contenedor de servlets Tomcat 6.0, para arrancar el servicio, es necesario abrir una ventana de comandos, posicionarse en la carpeta bin de Tomcat, en este caso:

cd Apache\ Tomcat/apache-tomcat-6.0.26/bin/

y ejecutar:

./catalina.sh start o **./catalina.sh run**

El primer comando se usa para iniciar Tomcat y entonces se puede acceder a él desde un Navegador, el segundo comando también inicia el Tomcat pero en un modo de Debug, ya que despliega información o errores que pueden llegar a suceder.

Para detener el servicio de Tomcat se ejecuta:

Para: **./catalina.sh start** se utiliza **./catalina.sh**

Y para: **./catalina.sh run** se utiliza la tecla **Ctrl+C**

Una vez que el usuario tiene información en los campos del Formulario (cualquiera) ya puede oprimir el botón ” Generar Reporte PDF ”

El código del módulo servlet encargado de generar un Reporte para Proyecto Terminal se muestra a continuación:

```
1 package com.pt2.pt2Web.controller;
2 import com.pt2.pt2Web.classes.FileManager;
3
4 import java.io.*;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10
11 /**
12 * <p>Title: FormularioTestServlet.java</p>
13 *
14 * <p>Description: Servlet utilizado para generar un Reporte de Proyecto Terminal en
      formato PDF </p>
15 *
16 * @author Maritza HernÃ¡ndez Arias
17 * @version 1.0
18 */
19
20 /**
21  * Servlet implementation class FormularioTestServlet
22  */
23 public class FormularioTestServlet extends HttpServlet {
24   private static final long serialVersionUID = 1L;
25
26   /**
```

```
27     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
           response)
28     */
29    protected void doPost(HttpServletRequest request, HttpServletResponse response)
          throws ServletException, IOException {
30      log ("Probando Servlet");
31
32      /**
33       * Se recuperan los valores ingresados por el usuario desde el Formulario
34       */
35
36      String nombre_proyecto = request.getParameter( "nombre_pt" );
37      String nombre_autor = request.getParameter( "alumno" );
38      String fecha = request.getParameter( "fecha" );
39      String obj_grales = request.getParameter( "obj_grales" );
40      String obj_particulares = request.getParameter( "obj_part" );
41      String antecedentes = request.getParameter( "antec" );
42      String justificacion = request.getParameter( "justif" );
43      String desc_tecnica = request.getParameter( "desc_tec" );
44      String esp_tecnica = request.getParameter( "esp_tec" );
45      String desarrollo = request.getParameter( "desarrollo" );
46      String resultados = request.getParameter( "resultados" );
47      String ref_biblio = request.getParameter( "biblio" );
48
49      /**
50         * Los textos recibidos son procesados para que Latex pueda comprender ciertos
               caracteres,
51         * como lo son las letras con acentos o sÃmbolos como pueden ser : '\', '%',
              '_', etc.
52         */
53
54      FileManager manager = new FileManager();
55
56      String nombre_proyecto2 = manager.poner_acentos( nombre_proyecto );
57      String nombre_autor2 = manager.poner_acentos( nombre_autor );
58      String fecha2 = manager.poner_acentos( fecha );
59      String obj_grales2 = manager.poner_acentos( obj_grales );
60      String obj_particulares2 = manager.poner_acentos( obj_particulares );
61      String antecedentes2 = manager.poner_acentos( antecedentes );
62      String justificacion2 = manager.poner_acentos( justificacion );
63      String desc_tecnica2 = manager.poner_acentos( desc_tecnica );
64      String esp_tecnica2 = manager.poner_acentos( esp_tecnica );
65      String desarrollo2 = manager.poner_acentos( desarrollo );
66      String resultados2 = manager.poner_acentos( resultados );
67      String ref_biblio2 = manager.poner_acentos( ref_biblio );
68
69      /*SE TIENE QUE CAMBIAR LA RUTA CUANDO SE SUBA AL SERVIDOR*/
70
71      /**
72       * La plantilla en Latex es leida.
73       * Posteriormente se buscan los valores a reemplazar (p. ej. @Titulo) y son
             reemplados por los textos
74       * que ingreso el usuario en los apartados correspondientes
75       */
76
77      manager.readFile("/home/itza/Escritorio/templates_pt2Web/plantilla_rep2.tex");
78
79      String arr_buscar [] = {"@Titulo","@Autor","@Fecha", "@ObjetivosGenerales","
           @ObjetivosParticulares","@Antecedentes", "@Justificacion","
           @DescripcionTecnica","@EspecificacionesTecnicas","@Desarrollo","@Resultados",
           "@Bibliografia"};
```

```
80
81      String arr_reemplazar [] = {nombre_proyecto2,nombre_autor2, fecha2, "\\item "+
            obj_grales2,"\\item "+obj_particulares2,antecedentes2,justificacion2,
            desc_tecnica2, esp_tecnica2, desarrollo2, resultados2, ref_biblio2};
82
83      String archivo = manager.replaceValues( "/home/itza/Escritorio/templates_pt2Web/
            plantilla_rep2.tex",arr_buscar,arr_reemplazar);
84
85
86      /**
87       * Se crea el archivo .tex fisicamente
88       */
89
90      FileWriter file = null;
91      try
92        {
93          // Crea la instancia del fichero
94          file = new FileWriter("/home/itza/Escritorio/plantillas_llenas/reporte/
                plantilla_reporte_creada.tex");
95          // El carÃ¡cter \r\n es el que hace que salte una linea en el fichero
                despuÃ©s de escribir
96          file.write(archivo);
97          // Cierra el fichero
98          System.out.println(archivo);
99          file.close();
100       }
101       catch(IOException ioe)
102       {
103         // Aqui pones lo que quieres que haga cuando se da una excepcion de
                escritura
104         System.out.println("Imposible escribir archivo .tex");
105       }
106
107       /**
108        * Se compila el archivo .tex y se genera el archivo PDF
109        */
110
111       try{
112         File dir = new File ("/home/itza/Escritorio/plantillas_llenas/reporte");
113         Process proceso1 = Runtime.getRuntime().exec( "pdflatex /home/itza/
                Escritorio/plantillas_llenas/reporte/plantilla_reporte_creada.tex", null,
                 dir  );
114         System.out.println("Ya compilo el comando latex");
115
116       /**
117        * Se abre el documento PDF para mostrarlo al usuario
118        */
119
120       /* Comando para abrir el documento PDF */
121       String [] cmd = {"gnome-open", "/home/itza/Escritorio/plantillas_llenas/
              reporte/plantilla_reporte_creada.pdf"};
122       Process process4 = Runtime.getRuntime().exec(cmd);
123
124       }
125
126       catch(IOException ioe)
127       {
128         System.out.println("No se pudo ejecutar el comando");
129       }
130
131     /**
```

```
132       * Redireccion a la pÃ¡agina principal
133       */
134      String base = request.getContextPath();
135      response.sendRedirect(base+"/Formulario_reporte.jsp");
136
137    }
138
139 }
```

El código del módulo servlet encargado de generar un Paper para Proyecto Terminal se muestra a continuación:

```
1  package com.pt2.pt2Web.controller;
2
3  import com.pt2.pt2Web.classes.FileManager;
4
5  import java.io.*;
6
7  import javax.servlet.ServletException;
8  import javax.servlet.http.HttpServlet;
9  import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12
13 /**
14 * <p>Title: FormularioTestPaperServlet.java</p>
15 *
16 * <p>Description: Servlet utilizado para generar un paper de un Proyecto Terminal en
       formato PDF </p>
17 *
18 * @author Maritza HernÃ¡ndez Arias
19 * @version 1.0
20 */
21
22 /**
23  * Servlet implementation class FormularioTestPaperServlet
24  */
25 public class FormularioTestPaperServlet extends HttpServlet {
26   private static final long serialVersionUID = 1L;
27
28   /**
29    * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
         response)
30    */
31   protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
32     // TODO Auto-generated method stub
33 log ("Probando Servlet de Paper");
34
35       /**
36        * Se recuperan los valores ingresados por el usuario desde el Formulario
37        */
38
39       String nombre_proyecto = request.getParameter( "nombre_pt" );
40       String nombre_autor = request.getParameter( "alumno" );
41       String fecha = request.getParameter( "fecha" );
42       String pal_clave = request.getParameter( "pal_clave" );
43       String obj_grales = request.getParameter( "obj_grales" );
44       String obj_particulares = request.getParameter( "obj_part" );
45       String antecedentes = request.getParameter( "antec" );
46       String justificacion = request.getParameter( "justif" );
47       String desc_tecnica = request.getParameter( "desc_tec" );
48       String esp_tecnica = request.getParameter( "esp_tec" );
49       String desarrollo = request.getParameter( "desarrollo" );
50       String resultados = request.getParameter( "resultados" );
51       String ref_biblio = request.getParameter( "biblio" );
52
53       /**
```

```
54          * Los textos recibidos son procesados para que Latex pueda comprender ciertos
                caracteres,
55          * como lo son las letras con acentos o sÃmbolos como pueden ser : '\', '%',
                '_', etc.
56          */
57
58         FileManager manager = new FileManager();
59
60         String nombre_proyecto2 = manager.poner_acentos( nombre_proyecto );
61         String nombre_autor2 = manager.poner_acentos( nombre_autor );
62         String fecha2 = manager.poner_acentos( fecha );
63         String pal_clave2 = manager.poner_acentos( pal_clave );
64         String obj_grales2 = manager.poner_acentos( obj_grales );
65         String obj_particulares2 = manager.poner_acentos( obj_particulares );
66         String antecedentes2 = manager.poner_acentos( antecedentes );
67         String justificacion2 = manager.poner_acentos( justificacion );
68         String desc_tecnica2 = manager.poner_acentos( desc_tecnica );
69         String esp_tecnica2 = manager.poner_acentos( esp_tecnica );
70         String desarrollo2 = manager.poner_acentos( desarrollo );
71         String resultados2 = manager.poner_acentos( resultados );
72         String ref_biblio2 = manager.poner_acentos( ref_biblio );
73
74
75         /*SE TIENE QUE CAMBIAR LA RUTA CUANDO SE SUBA AL SERVIDOR*/
76
77         /**
78          * La plantilla en Latex es leida.
79          * Posteriormente se buscan los valores a reemplazar (p. ej. @Titulo) y son
                reemplados por los textos
80          * que ingreso el usuario en los apartados correspondientes
81          */
82
83         manager.readFile("/home/itza/Escritorio/templates_pt2Web/plantilla_paper.tex")
                ;
84
85         String arr_buscar [] = {"@Titulo","@Autor", "@Fecha", "@PalabrasClave","
                @ObjetivosGenerales","@ObjetivosParticulares","@Antecedentes", "
                @Justificacion","@DescripcionTecnica","@EspecificacionesTecnicas","
                @Desarrollo","@Resultados","@Bibliografia"};
86
87         String arr_reemplazar [] = {nombre_proyecto2, nombre_autor2, fecha2,
                pal_clave2, "\\item "+obj_grales2, "\\item "+obj_particulares2,
                antecedentes2, justificacion2, desc_tecnica2, esp_tecnica2, desarrollo2,
                resultados2, ref_biblio2};
88
89         String archivo = manager.replaceValues( "/home/itza/Escritorio/
                templates_pt2Web/plantilla_paper.tex",arr_buscar,arr_reemplazar);
90
91         /**
92          * Se crea el archivo .tex fisicamente
93          */
94
95         FileWriter file = null;
96         try
97         {
98           // Crea la instancia del fichero
99           file = new FileWriter("/home/itza/Escritorio/plantillas_llenas/paper/
                plantilla_paper_creada.tex");
100          // El carÃ¡cter \r\n es el que hace que salte una linea en el fichero
                despuÃ©s de escribir
101          file.write(archivo);
```

```
102           // Cierra el fichero
103           file.close();
104         }
105         catch(IOException ioe)
106         {
107           // Aqui pones lo que quieres que haga cuando se da una excepcion de
                escritura
108           System.out.println("Imposible escribir archivo .tex");
109         }
110
111
112         /**
113          * Se compila el archivo .tex y se genera el archivo PDF
114          */
115
116         try{
117           File dir = new File ("/home/itza/Escritorio/plantillas_llenas/paper");
118           Process proceso1 = Runtime.getRuntime().exec( "pdflatex /home/itza/
                Escritorio/plantillas_llenas/paper/plantilla_paper_creada.tex", null, dir
                  );
119           System.out.println("Ya compilo el comando latex");
120
121           /**
122            * Se abre el documento PDF para mostrarlo al usuario
123            */
124
125           String [] cmd = {"gnome-open", "/home/itza/Escritorio/plantillas_llenas/
                paper/plantilla_paper_creada.pdf"};
126           Process process2 = Runtime.getRuntime().exec(cmd);
127
128           }
129
130           catch(IOException ioe)
131           {
132             System.out.println("No se pudo ejecutar el comando");
133           }
134
135           /**
136            * Redireccion a la pagina principal
137            */
138            String base = request.getContextPath();
139            response.sendRedirect(base+"/Formulario_paper.jsp");
140
141       }
142
143 }
```

Bibliografía

1.http://www.latex-project.org/
Consultada el 3 de febrero de 2009

2.www.geocities.com/jose_luis_cardenas/formato_paper.doc
Consultada el 3 de febrero de 2009

3.http://java.sun.com/applets/
Consultada el 7 de febrero de 2009

4.http://java.sun.com/products/servlet/index.jsp
Consultada el 14 de febrero de 2009

5.http://www.roboform.com/es/
Consultada el 2 de marzo de 2009

6.http://blog.sindominio.net/blog/adolflow/general/2006/06/08/odt_ya_es_un_formato_estandar
Consultado el 14 de febrero de 2009

7.http://download.microsoft.com/download/c/f/4/cf47d3c5-79df-4f7d-8262-cbcc55e03266/Hispasec_D
Consultada el 18 de febrero de 2009

8.http://www.symantec.com/es/mx/norton/security_response/malware.jsp
Consultada el 19 de febrero de 2009

9.http://java.sun.com/developer/technicalArticles/xml/JavaTechandXML/index.html
Consultada el 2 de marzo de 2009

10.http://msdn.microsoft.com/eses/library/system.security.cryptography.des.aspx
Consultada el 6 de marzo de 2009

11.http://www.jdom.org/
Consultada el 6 de marzo de 2009

12.tomcat.apache.org/
Consultada el 6 de marzo de 2009

13.http://www.openformats.org/es62
Consultada el 7 de marzo de 2009

14.Bruce Eckel, ”Piensa En Java”
Editorial Prentice Hall,1998

15.Bishop, David, ”Introduction to cryptography with Java applets”
Editorial Jones and Barlett, 2002

16.Gabriel Valiente Feruglio, Çomposición de textos científicos con LaTeX”
Editorial Alfaomega, 2001.