Universidad de Valladolid

E. T. S. Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Procesador del Log de HPSA

Autor:

D. José Alberto González Olmedo

Tutor:

D. Pablo de la Fuente Redondo

Agradezco a compañeros, amigos, familia y profesores
por todo el apoyo prestado que me ha permitido
completar la titulación de grado en informática.

INDICE

CAPITULO 1. Introducción

1.1 Propósito del sistema

El propósito del sistema es el procesamiento y visualización de la información presente en los ficheros de Log de la aplicación HPSA. El sistema proporcionará una forma de realizar consultas sencillas sobre la información tratada anteriormente.

1.2 Ámbito del sistema

Una vez que se conoce el propósito del sistema es necesario conocer como surgió la iniciativa para su creación. Para ello a continuación se va a exponer el ámbito para el que se decidió crear este sistema.

Primero, hay que conocer que la aplicación HPSA es una aplicación que se utiliza en compañías telefónicas para mantener el inventario de sus equipos de red y para activar y desactivar líneas y servicios como el buzón de voz, la llamada en espera, además de otros. Esta aplicación es la que genera los ficheros Log que tratará la nueva aplicación.

La idea de crear esta nueva aplicación surgió porque después de varias implantaciones del HPSA en clientes con un gran volumen de operaciones por segundo, los trabajadores de la empresa HP se percataron que la interfaz de visualización de los datos del Log era mejorable. Estos empleados destacaron que se podía mejorar sobre todo la visualización, filtrado y búsqueda de información.

Por los motivos expuestos anteriormente se propuso la creación de una aplicación que lea los archivos de Log, los integre en un servicio de indexación de información, y que posteriormente permita visualizar la información y realizar búsquedas sobre la información procesada. Esta aplicación se integrará en el HPSA EP. El HPSA EP es un pack de extensión de las funcionalidades del programa HPSA.

1.3 Alcance del sistema

El sistema que se pretende desarrollar es una aplicación para el procesamiento de los archivos de Log para la aplicación HPSA. La aplicación realizará la lectura y copia de información de los archivos de Log y los integrará en un servicio de indexación de información; sentando la base para realizar estudios estadísticos y consultas filtrando por los contenidos de los campos.

En esta aplicación no se va a generar los archivos de log, sino simplemente se procesarán. También se desarrollará una interfaz web sencilla para realizar operaciones de búsqueda básicas.

El nuevo sistema no interaccionará directamente con la aplicación HPSA. El único intercambio de información entre las dos aplicaciones será a través de los ficheros de Log de HPSA.

1.4 Objetivos

Para este trabajo, los objetivos son:

- Proporcionar una aplicación que procese y almacene la información del Log en un sistema de indexación de contenidos, desde donde después poder realizar búsquedas y análisis estadísticos. Desde esta aplicación se podrán realizar operaciones de búsqueda básica, buscando concordancias por campos de las entradas del Log.

- Se desea que la nueva aplicación no afecte al rendimiento de HPSA. Por lo que la aplicación no debe interaccionar con HPSA y de esta forma estar aislada teniendo solo contacto con los ficheros de Log. Este punto es un requisito importante, ya que el HPSA es una aplicación que requiere un alto rendimiento y se desea que no se interfiera en la calidad de servicio ofrecido a sus clientes. Por lo que operaremos con archivos ya creados y no trabajaremos directamente con el sistema de Log del HPSA.

1.5 Estructura del documento

Antes de describir la estructura del documento es necesario realizar una breve exposición del propósito de éste. El propósito general de este documento es hacer constatar el trabajo realizado durante el trabajo fin de grado. Para ello se realizará una exposición de las principales tareas llevadas a cabo y la documentación que se han ido generando durante el proyecto.

Esta memoria ha sido estructurada de modo que en cada capítulo se muestra la documentación que se ha ido generando durante el proyecto. El documento se divide en los siguientes 5 capítulos: Introducción; Planificación y Seguimiento; Análisis; Diseño e Implementación; Pruebas y Conclusiones.

En el capítulo de "Introducción" se expone información general sobre el proyecto. Los puntos más importantes tratados son: el contenido de esta memoria, una noción general del sistema desarrollado y el método de colaboración con HP.

En el capítulo de "Planificación y Seguimiento" se expone la planificación realizada al comienzo del proyecto, con las tareas a realizar y la previsión del tiempo dedicado para completarse. Dentro de este capítulo también se expone a modo de conclusión un seguimiento de las actividades de forma que se pueda ver el desfase entre el tiempo planificado y el tiempo que finalmente se ha empleado.

En el capítulo de "Análisis", se van a exponer los requisitos que debe cumplir el sistema a implementar. Para completar la información aportada por los requisitos, se optó por realizar un análisis a través de la creación de los casos de uso del sistema así como de un análisis del dominio sobre el que se implementará la aplicación.

En el capítulo de "Diseño e Implementación", se van a mostrar los diagramas creados previamente a la implementación de la aplicación. En estos diagramas se establece la arquitectura del sistema y la interacción entre los distintos elementos que deben crearse.

En el capítulo de "Pruebas", se exponen los resultados de las tareas que tenían como objetivo comprobar que la aplicación cumple con los requisitos presentes en el capítulo de "Análisis".

Finalmente, en el último capítulo del documento se exponen las conclusiones personales del autor sobre el trabajo realizado y las posibles futuras líneas de trabajo.

En el final del documento se encuentran unos anexos, con información tanto de este documento como de los materiales adjuntados a él en su entrega.

1.6 Colaboración con HP

Para la realización de este proyecto se ha colaborado activamente con la empresa HP. Esta empresa dispone de una plataforma para la colaboración con estudiantes de distintas universidades. La plataforma en la que se colaboró para la realización de este proyecto la denominaron "Observatorio 2012", por el nombre de la plataforma y el año en que se realizó el trabajo.

La empresa HP, dentro de su proyecto de colaboración con universidades, realizó una exposición en el "Salón de Grados" informando del sistema de colaboración y de los proyectos que ofertaban.

Posteriormente los alumnos interesados debían enviar una solicitud indicando dos proyectos en los que deseaban colaborar, a través de un correo electrónico al director de la E. T. S. de Ingeniería Informática. Finalmente, una vez cumplido el plazo que se dio para presentar solicitudes, se me comunico a través de correo electrónico que había sido seleccionado para realizar el proyecto "Procesador del Log de HPSA", que documento mediante esta memoria.

Una vez que me fue asignado el trabajo, comencé a intercambiar correos electrónicos tanto con el tutor de la universidad como con Rubén Rodríguez, el tutor de HP. Durante las primeras semanas, el intercambio de correos iba orientado a conocer los requisitos de la aplicación. Una vez que se acordaron los requisitos del sistema, hubo un periodo de tiempo que dediqué a estudiar las tecnologías que posteriormente usaría, durante este periodo no hubo comunicación con el tutor.

Los siguientes correos electrónicos que se intercambiaron fueron en la fase de desarrollo de la aplicación. En estos correos recibí consejos y comentarios a los artefactos que iba desarrollando. También cabe destacar que el "Workspace", de Eclipse, en el que se desarrolló la aplicación era compartido con Rubén Rodríguez, a través de "Dropbox".

Solo cabe destacar que todas las interacciones con Rubén Rodríguez fueron a través de correos electrónicos. Aunque nos intercambiamos nuestros números de teléfono no fue nunca necesario su uso.

Finalmente quiero destacar que la posibilidad de colaborar con HP, una empresa potente en el área de la informática, fue una de las principales causas por las que me ofrecí para realizar este proyecto.

CAPITULO 2. Planificación y seguimiento

2.1 Planificación.

Al comenzar un nuevo proyecto lo primero que hay que hacer es establecer una planificación, para crear el sistema cumpliendo con los objetivos en unos plazos fijados por el cliente. Por lo tanto, lo primero que se realizó fue una planificación sobre las tareas que se iban a realizar durante el desarrollo del proyecto y una estimación del tiempo que se iba a emplear en cada una.

A la hora de realizar una planificación es necesario determinar la fecha de inicio, la fecha de fin y una estimación de horas para completar el trabajo. La fecha de inicio que se estableció fue el 11 de febrero de 2013. Se consideró que la duración estimada para completar el proyecto sería de 400 horas aproximadamente, estimación realizada por parte de Rubén Rodríguez, tutor del proyecto de HP. La fecha establecida para la finalización del proyecto fue el inicio del mes de junio de 2013.

Dadas las fechas de inicio y de fin del proyecto, para poder realizar el proyecto en las horas estimadas por el tutor, el alumno debería dedicar 5 horas diarias a las tareas que engloban el proyecto. Esta estimación de dedicación diaria se calculó considerando que las tareas se realizarían de lunes a viernes.

Teniendo en cuenta las anteriores estimaciones, se consideró un plan de contingencia en caso de que se produjese un retraso en la planificación de las tareas, suponiendo este retraso un riesgo para cumplir con la fecha de fin. El plan de contingencia consiste en la dedicación de más horas de las previstas, pudiendo ser durante los fines de semana o de lunes a viernes, en función de las prioridades del desarrollador.

La planificación del proyecto se divide en las siguientes fases: “Planificación del proyecto”, “Análisis”, “Diseño”, “Implementación”, “Pruebas” y “Documentación”. Las fases se realizarán de forma secuencial al igual que sus tareas. Aunque en algunos casos, las tareas se podrían haber realizado en otro orden o de manera paralelizada, si se hubiera contado con un mayor número de desarrolladores en el proyecto. Por lo anteriormente comentado, se comenzará una tarea cuando la anterior se haya finalizado, para que el desarrollador este centrado en la actividad que está realizando.

Las fases enumeradas en el párrafo anterior, se encuentran documentadas en los capítulos de esta memoria, exceptuando la fase de “Documentación”. Las fases de “Diseño” e “Implementación” se documentan en el mismo capítulo que lleva por nombre la conjunción de los nombres de las fases. El código de la aplicación que es resultado de la fase de “Implementación” se encuentra en el CD Anexo. Como resultado de la fase de “Documentación” se obtuvo esta memoria, un manual de uso y un manual de instalación de la aplicación.

Una vez establecidas las fechas de inicio y fin del proyecto, los tiempos de desarrollo del proyecto y las fases, se decidió las tareas que se llevarían a cabo en cada fase. Una vez que las tareas se habían enumerado, se estimó el tiempo que se dedicaría a cada una de ellas para

completarlas. Esta estimación se asignó en función de la experiencia del autor, en proyectos realizados anteriormente y los conocimiento obtenidos durante los estudios realizados.

A continuación se muestra un resumen de la planificación mostrando las fases, las tareas y la estimación asignada para ser completadas:

Nombre de tarea	Estimación
Desarrollo del Procesador del Log de HPSA	**400 horas**
Planificación del proyecto	**15 horas**
Determinar el ámbito general del proyecto	5 horas
Determinar planificación	5 horas
Realización del documento de planificación	5 horas
Análisis	**35 horas**
Establecer requisitos funcionales	5 horas
Establecer requisitos no funcionales	5 horas
Realizar análisis del Log	7 horas
Definir casos de uso	3 horas
Realizar casos de uso	10 horas
Completar documento de análisis	5 horas
Diseño	**100 horas**
Formación en las tecnologías	60 horas
Establecer arquitectura	7 horas
Diseño de prototipo de GUI	3 horas
Realizar casos de uso de diseño	10 horas
Diseñar pruebas con datos reales (funcionalidad y GUI)	10 horas
Diseñar fichero de Log artificial y diseño de pruebas	10 horas
Implementación	**165 horas**
Preparar herramientas de desarrollo	15 horas
Implementar GUI	30 horas
Implementar código	100 horas
Limpiar código	20 horas
Pruebas	**30 horas**
Realizar pruebas con datos reales	15 horas
Realizar pruebas con fichero artificial	15 horas
Documentación	**55 horas**
Realizar manual de ayuda	10 horas
Realizar manual de instalación	10 horas
Realizar memoria de TFG	30 horas
Dar formato a la memoria	5 horas
Fin del proyecto	**0 horas**

Tabla 1: Planificación Nombre de las tareas y estimaciones

Una vez enumeradas las tareas que se realizarán en cada fase, el siguiente paso es realizar un resumen con lo que se pretende obtener de cada tarea de forma que quede claro el objetivo de cada una.

El siguiente paso, es establecer una línea temporal en la que se establece el calendario en el cual se muestre cuando se van a realizar las tareas. Para crear el calendario se ha usado Microsoft Project 2010. Usando esta herramienta se obtiene un diagrama de Gantt en el que se muestra el calendario de la planificación realizada.

Sobre los diagramas de Gantt que se mostrarán a continuación, hay que aclarar que la fecha que se muestra en la zona superior del diagrama indica la fecha del lunes de la semana. También hay que aclarar que las barras que representan las tareas, muestran las fechas de inicio y fin de la tarea, siendo considerados los fines de semana como días en los que no se realizarán tareas, aunque algunas barras se superpongan en esos días en el gráfico.

En las siguientes páginas se va a realizar una descripción de las tareas que se realizarán en cada fase, de forma que se muestra un diagrama de Gantt por cada fase. De esta forma la planificación establecida para el proyecto es la siguiente:

- "Planificación del proyecto": se llevarían a cabo las tareas presentes en el siguiente diagrama. La tarea "Determinar el ámbito general del proyecto" es necesaria para considerar la magnitud que podría tener el proyecto. Una vez realizada esa primera tarea, lo siguiente es determinar que tareas se van a realizar para completar el proyecto con éxito.

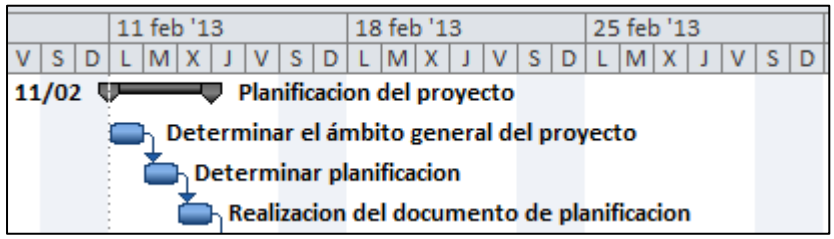


Ilustración 1: Diagrama de Gantt de Planificación

- "Análisis": durante esta fase, se realizarán tareas propias del análisis como la recopilación de requisitos, la enumeración y descripción de los casos de uso. En este proyecto en particular se ha optado por crear una tarea en la que se estudian los ficheros Log que posteriormente van a ser tratados durante la ejecución de la aplicación.

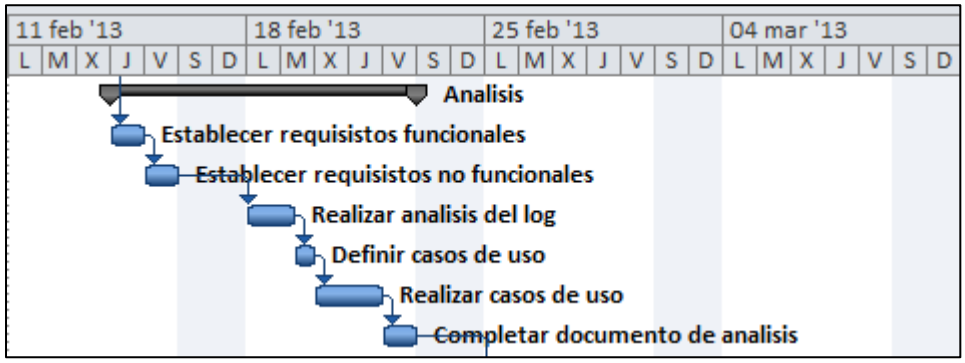


Ilustración 2: Diagrama de Gantt de la planificación de Análisis

- “Diseño”: durante esta fase se van a realizar tareas propias del diseño de aplicaciones, como la elección de la arquitectura, diseño de interfaces y el diseño de las pruebas que se realizarán en la fase de “Pruebas”. Dentro de esta fase cabe destacar la tarea “Formación en las tecnologías”, ya que no es una tarea propia del diseño de aplicaciones, pero es necesaria ya que antes de comenzar el diseño es necesario conocer las ventajas y las limitaciones de las tecnologías utilizadas.

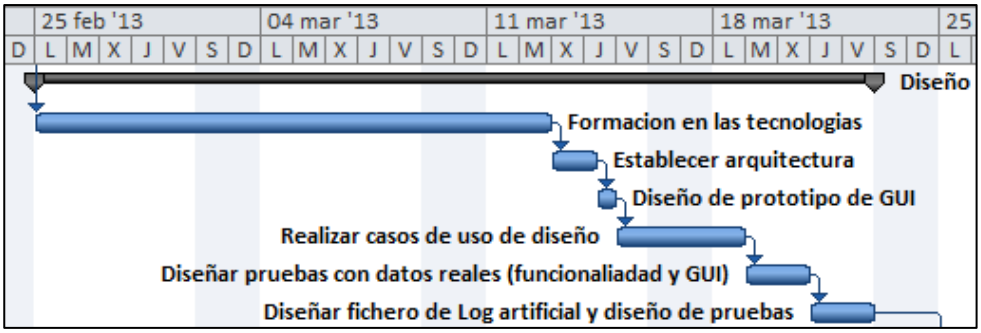


Ilustración 3: Diagrama de Gantt de la planificación de Diseño

- “Implementación”: durante esta fase se van a realizar tareas que tendrán como objetivo tener una versión ejecutable de la aplicación. De las tareas vistas en el diagrama de Gantt, solo cabe aclarar que durante la tarea “Limpiar código” se realizará una revisión del código borrando comentarios no necesarios o dando formato al código, para que en caso de ser consultado pueda ser entendido más fácilmente. Durante la tarea de “Implementar código”, cabe la posibilidad de adquirir nuevos conocimientos sobre las tecnologías según se solucionen los problemas que surjan en la implementación.

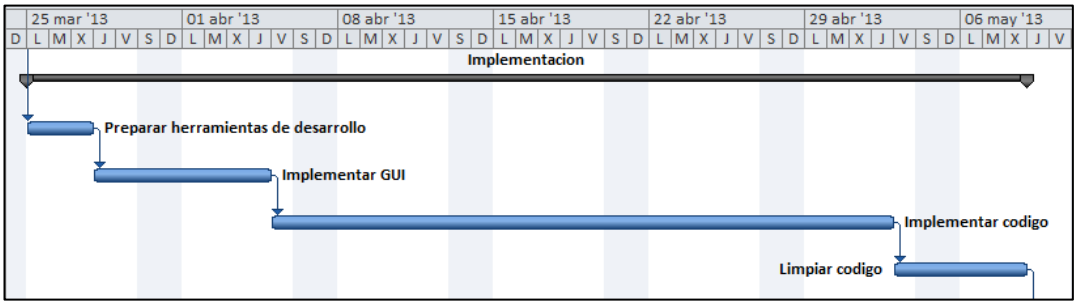


Ilustración 4: Diagrama de Gantt de la planificación de Implementación

- “Pruebas”: durante esta fase se van a realizar tareas para comprobar que la aplicación funciona acorde con los requisitos establecidos en la fase de Análisis. Las pruebas que se realizarán habrían de haber sido diseñadas en la fase de Diseño. En caso de ser detectado algún error será subsanado en esta misma fase, por lo que el tiempo adjudicado a las tareas de prueba es más amplio que el realmente necesario para ejecutar los casos de prueba diseñados.

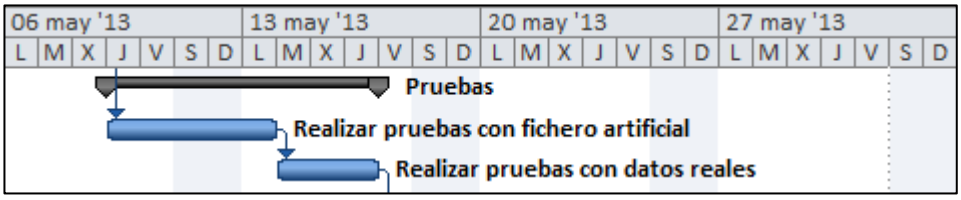


Ilustración 5: Diagrama de Gantt de la planificación de Pruebas

- “Documentación”: durante esta fase se crearán los documentos acordados en la fase de Análisis. En esta fase se crearán los manuales propios de una aplicación, manual de ayuda y manual de instalación, y esta memoria, que documenta el trabajo realizado para cumplir con los objetivos de este proyecto. La realización de esta memoria es una tarea meramente académica por lo que no se considera necesario entregarla junto con el resto de artefactos del proyecto a los responsables de HP. El manual de ayuda y el manual de instalación estarán presentes en el CD anexo y se mostrará su contenido en los anexos de esta memoria.

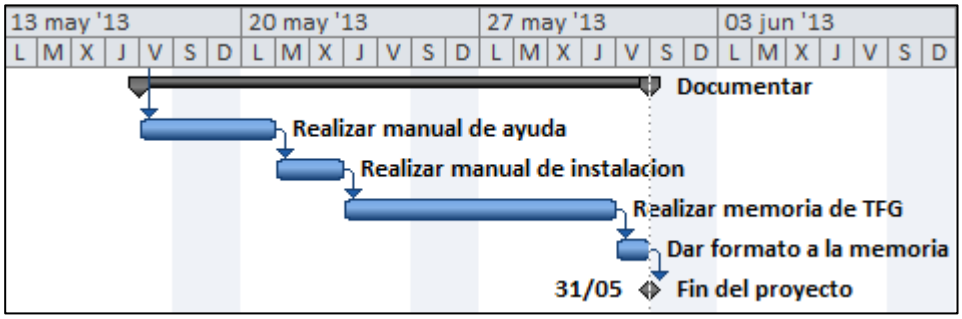


Ilustración 6: Diagrama de Gantt de la planificación de Documentar

Una vez establecida la planificación, el siguiente paso es establecer los mecanismos necesarios para comprobar la evolución del proyecto. Como la dedicación del desarrollador no va a ser siempre uniforme, aunque estaba planificado 5 horas diarias, no se va a realizar un seguimiento sobre las fechas de inicio y fin de las tareas, sino que el seguimiento se va a realizar sobre los tiempos dedicados a cada tarea.

El seguimiento del proyecto se va a realizar anotando el tiempo dedicado a cada tarea, de forma que cuando finalice el proyecto se pueda ver si se ha cumplido con la planificación establecida. De esta forma se podrá obtener un “feedback” para futuras planificaciones.

2.2 Seguimiento.

En este apartado de seguimiento se pretende mostrar el grado de cumplimiento con la planificación fijada al inicio del proyecto. Para ello vamos a comprobar los siguientes factores: orden de realización de las tareas, se cumple con las fechas establecidas, como de acertadas han sido las estimaciones de duración de las tareas.

Durante la realización del proyecto cabe destacar que finalmente no se siguió el orden establecido anteriormente, ya que para realizar la tarea “Formación en las tecnologías” de la fase de Diseño fue necesario realizar la tarea “Preparar herramientas de desarrollo” de la fase de Implementación. Este cambio fue motivado porque durante la tarea de formación se realizaron algunos ejercicios prácticos sobre la tecnología desarrollando ejemplos, presentes en la bibliografía consultada. El resto de tareas se realizó en el orden planificado.

En cuanto a las fechas establecidas para la realización de las tareas, no se fueron cumpliendo ya que la dedicación no fue la previamente fijada, 5 horas diarias de lunes a viernes, siendo usados los fines de semana para que aproximadamente se cumplieran las horas dedicadas por semana. El cambio en el orden antes mencionado y la dedicación a cada tarea también influyo en la variación de las fechas de inicio de cada tarea. Por ultimo en lo referente a las fechas de inicio de las tareas,

cabe destacar que la memoria fue realizada en fechas posteriores, cuando el desarrollador del proyecto quedo libre de otras cargas de trabajo ajenas a este proyecto.

A lo largo del proyecto, se ha ido anotando el tiempo que se ha dedicado a cada tarea. Mediante este seguimiento se trata de comprobar como de acertadas fueron las estimaciones realizadas sobre la duración de las tareas. A continuación se muestra un resumen, en el que se puede ver una comparativa entre las predicciones y el tiempo que finalmente se ha dedicado a cada tarea:

Nombre de tarea	Estimación	Dedicación
Desarrollo del Procesador del Log de HPSA	**400 horas**	**349 horas**
Planificación del proyecto	**15 horas**	**15 horas**
Determinar el ámbito general del proyecto	5 horas	5 horas
Determinar planificación	5 horas	4 horas
Realización del documento de planificación	5 horas	6 horas
Análisis	**35 horas**	**20 horas**
Establecer requisitos funcionales	5 horas	3 horas
Establecer requisitos no funcionales	5 horas	2 horas
Realizar análisis del Log	7 horas	2 horas
Definir casos de uso	3 horas	2 horas
Realizar casos de uso	10 horas	3 horas
Completar documento de análisis	5 horas	8 horas
Diseño	**100 horas**	**105 horas**
Formación en las tecnologías	60 horas	60 horas
Establecer arquitectura	7 horas	5 horas
Diseño de prototipo de GUI	3 horas	2 horas
Realizar casos de uso de diseño	10 horas	8 horas
Diseñar pruebas con datos reales (funcionalidad y GUI)	10 horas	10 horas
Diseñar fichero de Log artificial y diseño de pruebas	10 horas	20 horas
Implementación	**165 horas**	**115 horas**
Preparar herramientas de desarrollo	15 horas	20 horas
Implementar GUI	30 horas	10 horas
Implementar código	100 horas	70 horas
Limpiar código	20 horas	15 horas
Pruebas	**30 horas**	**35 horas**
Realizar pruebas con datos reales	15 horas	15 horas
Realizar pruebas con fichero artificial	15 horas	20 horas
Documentación	**55 horas**	**59 horas**
Realizar manual de ayuda	10 horas	4 horas
Realizar manual de instalación	10 horas	15 horas
Realizar memoria de TFG	30 horas	35 horas
Dar formato a la memoria	5 horas	5 horas
Fin del proyecto	**0 horas**	**0 horas**

Tabla 2: Comparación entre estimación y dedicación de las tareas

En general, se puede ver que la estimación ha sido bastante generosa ya que en muchas de las tareas el tiempo estimado ha sido superior al que finalmente se dedicó. Resultando finalmente el tiempo dedicado al proyecto 51 horas menos a las estimadas inicialmente. Este desfase entre previsión y dedicación fue debido a que durante la tarea "Determinar el ámbito general del proyecto" no se esclareció suficientemente la carga de trabajo que supondría la implementación de la aplicación.

Durante la fase de análisis, una vez enumerados y descritos los casos de uso, se pudo ver que la previsión realizada anteriormente iba a ser excesiva, como al final sucedió, ya que el tiempo previsto para la implementación era excesivo para lo que posteriormente se necesitaría.

Finalmente como conclusión de este apartado, cabe destacar que se cumplieron con los plazos de entrega de la aplicación al personal de HP, siendo acorde con los requisitos establecidos.

CAPITULO 3. Análisis.

3.1 Visión general

La aplicación que se pretende crear va a realizar un procesamiento de los datos de los ficheros Log que genera la aplicación HPSA. Pudiendo después visualizar la información procesada realizando consultas sencillas.

El procesamiento de los ficheros Log, se llevará a cabo realizando la lectura y copia de la información de los ficheros Log, integrando los datos en un servicio de indexación de información. Sentando la base para realizar estudios estadísticos o consultas más avanzadas en posteriores versiones.

La información sobre los ficheros Log que van a ser procesados se obtendrá de un fichero de configuración. En este fichero de configuración se establecerá la carpeta que contiene los ficheros Log que van a ser procesados, así como el tipo de Log a procesar. En esta aplicación se realizará solo el procesamiento de los ficheros Log de tipo "mwfm" (En el punto 3.3 se detalla el contenido de estos ficheros).

El nuevo sistema se integrará en el HPSA EP, de forma que no debe influir en el rendimiento del HPSA.

3.2 Requisitos

En este puntos se van a enumerar los requisitos prefijados por el personal colaborador de HP para la aplicación a desarrollar.

3.2.1 Requisitos funcionales

Req 1. El sistema procesará los ficheros Log introduciendo los datos en un sistema de indexación de contenidos. Los ficheros Log que el sistema procesará, serán los que correspondan con la carpeta y el tipo de Log, estos datos vienen determinados en un fichero de configuración.

Req 1.1. Los nombres de los ficheros de Log siguen el siguiente patrón: "xyx_número.log.xml". Donde "número" es un número asignado al fichero y "xyz" es un código alfabético que indica el tipo del Log. Los siguientes ejemplos muestran posibles nombres de ficheros de Log: mwfm_0.log.xml, mwfm_1.log.xml...

Req 1.2. Dentro de cada carpeta, puede haber N ficheros de Log de distintos tipos.

Req 1.3. La aplicación solo procesará los ficheros de tipo "mwfm". Los tipos de Logs pueden ser: "mwfm", "dw"…

Req 2. Los ficheros Log de tipo “mwfm” tienen los siguientes campos: level, HostName, Time, Module, Part, Component, Topic, Thread y Message. En el punto 3.3 de esta memoria se realiza un análisis más en profundidad de este tipo de ficheros.

Req 2.1. Los campos del fichero serán tratados como texto plano a excepción del campo “Time” que será tratado como fecha.

Req 3. El sistema permitirá realizar búsquedas y consultas sobre los datos de los ficheros procesados por la aplicación.

3.2.2 Requisitos no funcionales

3.2.2.1 Facilidad de uso

Req 1. Para poder utilizar este sistema bastará con que el usuario disponga de un conocimiento básico en navegación Web y el uso de un explorador de Internet.

3.2.2.2 Prestaciones

Req 1. Se mostrará la información del tiempo que tarda el sistema en cargar la información de cada fichero de Log, este tiempo se mostrará en el Log del servidor.

3.2.2.3 Restricciones de diseño

Req 1. El servidor Web sobre el que se ejecutará la aplicación será JBoss 7.

Req 2. La aplicación será implementada en lenguaje Java.

Req 3. Los ficheros Log a procesar (Logs generados por HPSA) están en formato XML.

Req 4. El servicio usado para la indexación de información será Solr.

Req 5. El interfaz de la aplicación se desarrollará en Struts.

Req 6. Se utilizará SAX para el parseo de los ficheros Log.

3.2.2.4 Documentación adjunta

Req 1. La aplicación irá acompañada de un manual de uso, como soporte de ayuda para los usuarios del sistema.

Req 2. La aplicación irá acompañada de un manual para su instalación.

3.3 Análisis del Log

Los ficheros de Log generados por la aplicación HPSA son generados en formato XML. El contenido entre el elemento raíz (“< >”) y el elemento de cierre (“</>”) es el valor asignado para ese campo. Aunque entre estos dos elementos puede haber también otras etiquetas.

Los ficheros de Log que van a ser procesados por la aplicación son los de tipo “mwfm”, por lo que el análisis se centrará en estos ficheros. El elemento “<LogEntries>” marca el inicio del fichero de Log, de forma que entre esta etiqueta y “</LogEntries>” se encuentran todas las entradas de Log presentes en el documento.

Una entrada de Log se encuentra delimitado entre <LogEntry> y </LogEntry>. En algunas entradas de Log existe el atributo “level” dentro del elemento raíz <LogEntry>, el resto de atributos de las entradas de Log se encuentran documentados entre etiquetas con el nombre de cada atributo de los designados en los requisitos.

El contenido de los atributos de las entradas de Log, vienen dados como textos planos, exceptuando el atributo “Time” que viene como texto con formato de fecha.

A continuación se muestra un ejemplo de un fichero de Log de tipo “mwfm”, con una entrada de Log:

```
<LogEntries>
    <LogEntry level="DEBUG" >
        <HostName>Ruben-laptop</HostName>
        <Time>18-dic-2012  12:29:57</Time>
        <Module>mwfm</Module>
        <Part>COMPONENT</Part>
        <Component>sosa_async_responser</Component>
        <Topic></Topic>
        <Thread>Thread-23</Thread>
        <Message>SosaAsyncResponser._checkErrors</Message>
    </LogEntry>
    ...
</LogEntries>
```

Ilustración 7: Ejemplo de una entrada de un Log

En este ejemplo se pueden ver todas las características comentadas anteriormente. Incluso también se puede ver que algunos campos de la entrada de Log pueden encontrarse vacías.

3.4 Modelo de Casos de Uso.

3.4.1 Descripciones generales de Actores.

Usuario: los usuarios serán los empleados de HP y el personal de las empresas cliente que cuente con el HPSA EP. Estos usuarios deben tener permisos de sus propias empresas.

3.4.2 Diagramas del Modelo de Casos de Uso.

El siguiente diagrama muestra los casos de uso del nuevo sistema, en esta aplicación el único caso de uso es "Cargar y Consultar":

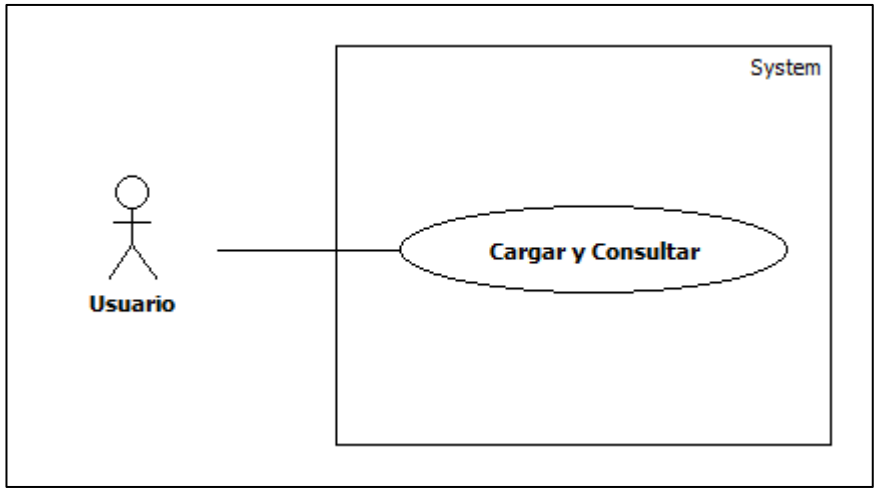


Ilustración 8: Diagrama de Casos de Uso

El caso de uso "Cargar y Consultar" se lanza cuando se accede a la aplicación, se cargan los datos de los Log que corresponden con los datos del fichero de configuración y muestra un formulario en el que realizar consultas.

3.4.3 Descripción de Casos de uso

UC-0001	**Cargar y consultar**	
Versión	1.0 (20/02/13)	
Descripción	Carga los ficheros de Log que marque el fichero de configuración y se realiza una búsqueda de entradas del Log en función de unos criterios.	
Actores	Usuario	
Precondición	El servidos Web está iniciado	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor accede a la aplicación.
	2	El sistema procesa los datos de los ficheros Log, según los criterios del fichero de configuración.
	3	El sistema pregunta los criterios de búsqueda.
	4	El usuario introduce los criterios de búsqueda.
	5	El sistema busca las entradas del Log que corresponden con los criterios de búsqueda introducidos por el usuario y muestra los resultados.
	6	El caso de uso finaliza.
Postcondición	Se han mostrado las entradas del Log correspondientes con los criterios de búsqueda introducidos.	
Flujos Alternativos	**Alt**	**Acción**
	1	En el paso 5 si alguno de los criterios introducidos anteriormente no es correcto se vuelve al paso 3.
	2	En cualquier punto el usuario puede salir de la aplicación cancelando cualquier operación y el caso de uso finaliza.
Importancia	Alta	

Tabla 3: Descripción del caso de uso "Cargar y consultar"

3.5 Modelo de dominio

El dominio de la aplicación es relativamente sencillo al constar solo de 4 clases de objetos. En el siguiente diagrama se ve una descripción de los ficheros que tratará la aplicación.

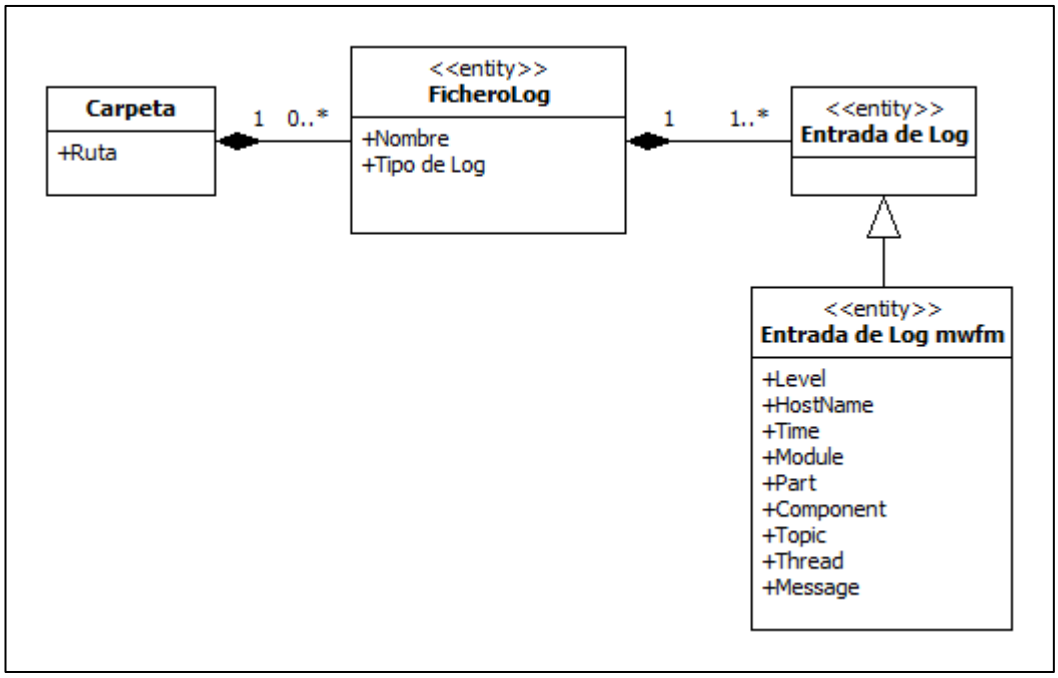


Ilustración 9: Modelo de dominio

En el diagrama se puede ver que en cada carpeta puede haber varios ficheros de Log, que contienen entradas de Log y una de estas posibles entradas de Log son los ficheros de “mwfm”.

3.6 Realización de Casos de uso

En este apartado se va a realizar un análisis del caso de uso “Cargar y Consultar”. En las siguientes páginas se van a mostrar diagramas especificando el funcionamiento que deberá implementar la aplicación acorde con la descripción del caso de uso realizada anteriormente.

Diagrama en el que se muestra la interacción del usuario con el sistema:

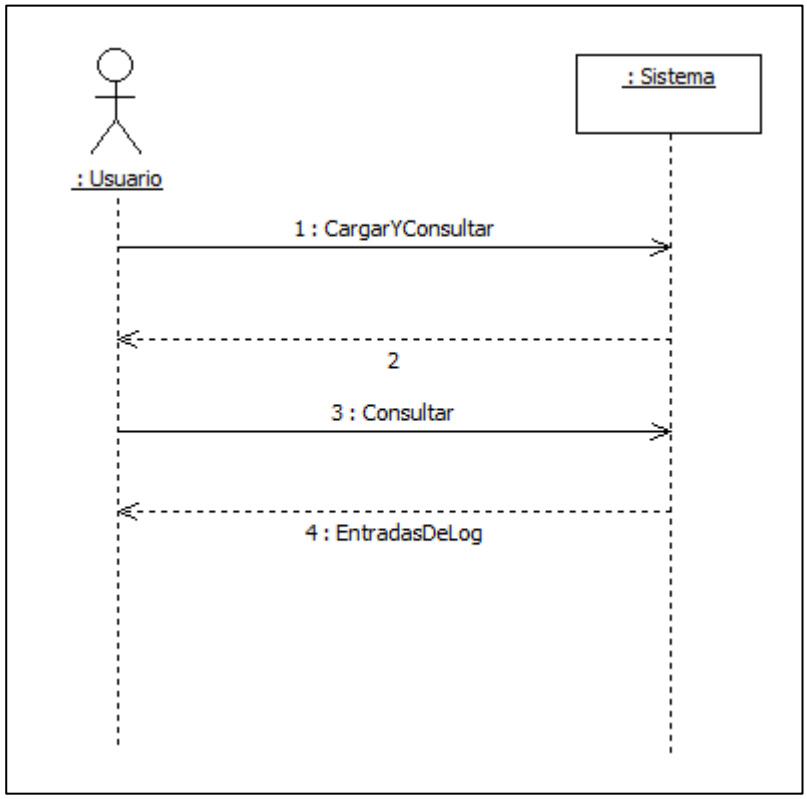


Ilustración 10: Diagrama de secuencia 1 de análisis de "Cargar y consultar"

El siguiente diagrama de secuencia se muestra más en detalle el caso de uso, con la interacción tanto del usuario con el sistema como las distintas comunicaciones entre los objetos del dominio:

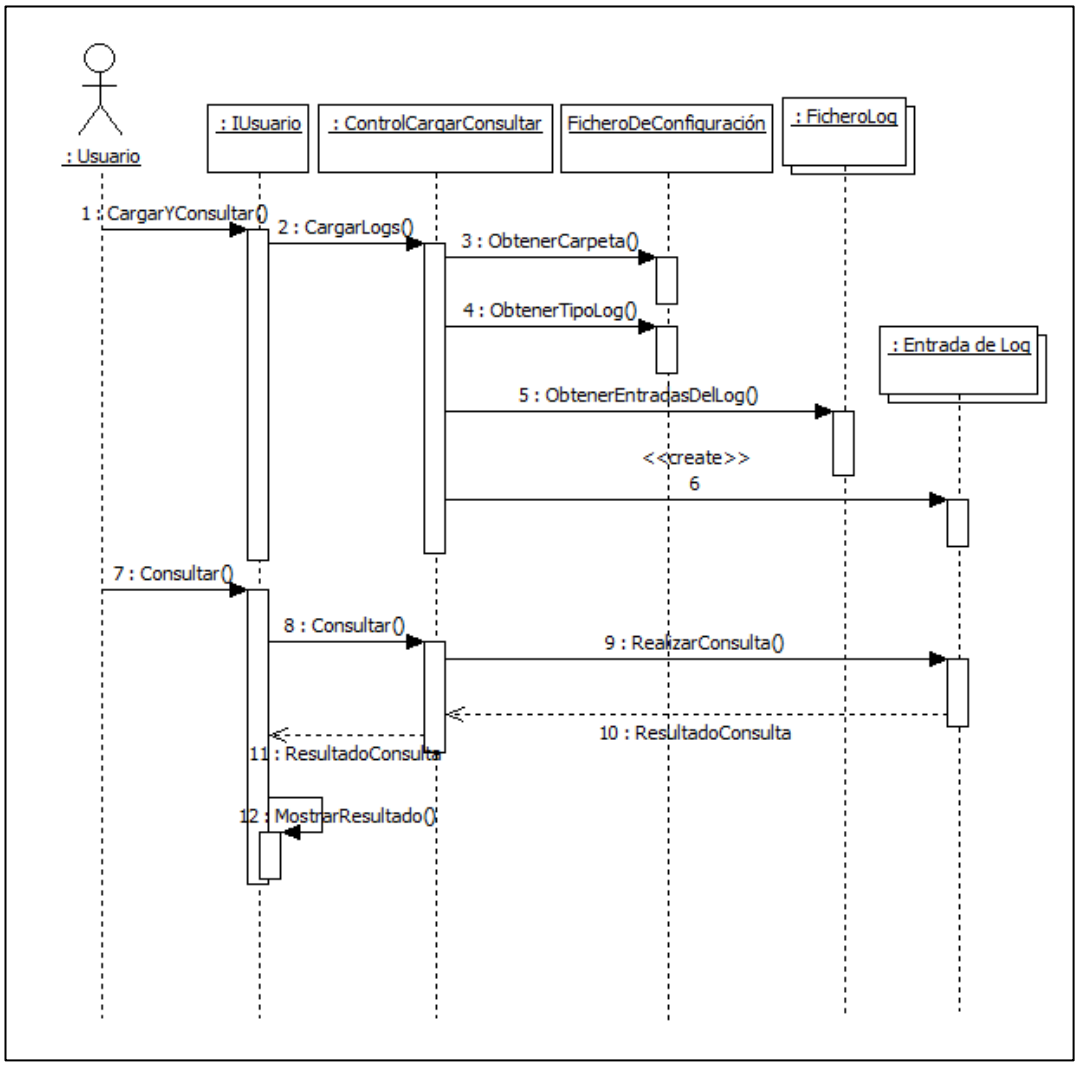


Ilustración 11: Diagrama de secuencia 2 de análisis de "Cargar y consultar"

3.7 Diagrama de Clases de Análisis (BCE)

En este diagrama se muestra un diagrama de clases usando la terminología BCE. En este diagrama se pueden ver las operaciones y los atributos que corresponde con cada clase definida.

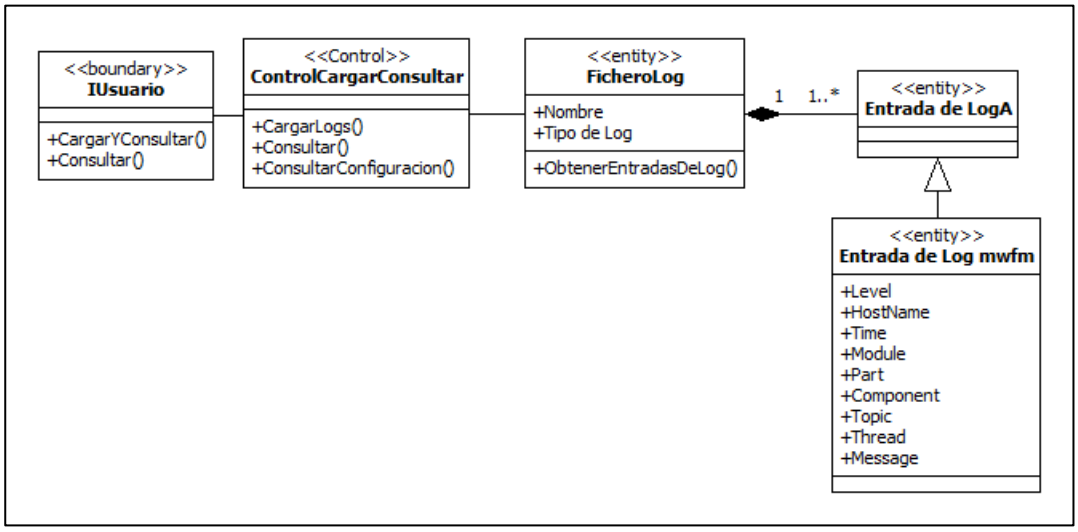


Ilustración 12: Diagrama de Clases de Análisis (BCE)

En este diagrama de clases de análisis, se puede ver que las clases "entity" son similares a las apreciadas en el diagrama del "Modelo de Dominio". La clase "boundary" corresponde con la idea de la interfaz de la aplicación. Y la clase "Control" representa la clase que determina el funcionamiento de la aplicación.

CAPITULO 4. Diseño e Implementación.

4.1 Descripción de las tecnologías

4.1.1 Struts

Struts es un framework, de código libre (open-source), para el desarrollo de aplicaciones Web bajo el patrón Modelo-Vista-Controlador (MVC). Las aplicaciones creadas con Struts deben ser ejecutadas sobre una plataforma Java EE (Java Enterprise Edition). Este framework fue creado por Craig McClanahan que lo dono a "Apache Software Foundation". Inicialmente "Apache Software Foundation" introdujo Struts en el proyecto Jakarta, aunque actualmente es un proyecto independiente denominado "Apache Struts".

Este framework fue implementado en Java buscando independencia del sistema operativo que posea la computadora sobre la que se ejecuta, teniendo como único requisito que la computadora soporte una plataforma Java EE.

Como se ha comentado anteriormente, Struts se basa en el patrón de arquitectura de software MVC el cual se utiliza ampliamente en el desarrollo de aplicaciones Web. De acuerdo con este patrón, las aplicaciones desarrolladas sobre este framework se divide en tres partes: el modelo, la vista y el controlador.

El objetivo de usar un framework como Struts es realizar esta división para que el mantenimiento de las aplicaciones creadas sea más fácil. Las tres partes de la arquitectura están orientadas a unas responsabilidades específicas. El modelo es responsable de la lógica de negocio, la vista de las paginas HTML que se muestran al cliente y el controlador de gestionar la información entre la vista y el modelo.

Dentro de esta arquitectura, Struts proporciona el controlador y librerías con las diferentes etiquetas para diseñar las vistas. Las librerías más usadas son proporcionadas para definir las interfaces en JSP, aunque también se pueden definir en XML/XSLT.

Los desarrolladores que usan Struts son responsables de implementar el modelo, las diferentes vistas que tendrá la aplicación y el fichero de configuración. Este fichero normalmente es denominado "struts-config.xml" y con él se definen las relaciones entre el modelo, la vista y el controlador.

Las peticiones del usuario son enviadas al controlador, de forma que en función de lo definido en el fichero de configuración se definen las acciones que se realizarán. Estas acciones se encargarán de relacionarse con el modelo de forma que se realice la petición realizada. Una vez realizada una acción se devuelve un texto que identifica la siguiente vista a mostrar por el controlador. Las transiciones y las acciones a realizar en cada petición se definen en el fichero de configuración.

Por lo tanto, a la hora de desarrollar una aplicación usando este framework es necesario tener conocimientos en XML, JSP y Java. En XML se definirá el fichero de configuración de Struts. En JSP se definirán las vistas de la aplicación. Y en Java se desarrolla el modelo, en el cual se implementa la lógica de negocio de la aplicación, que contiene las acciones que se realizan en respuesta a las interacciones de los usuarios con la aplicación.

Una característica que hace atractivo el uso de Struts es que permite internacionalizar la aplicación. Ya que simplemente con el cambio de una característica del fichero de configuración se pueden cambiar los mensajes mostrados en la interfaz de la aplicación.

Otra característica atractiva de Struts es que proporciona un sistema de validación de los campos de los formularios. En Struts te permite que esta validación se pueda hacer de varias formas, se puede hacer a través de ficheros XML y de clases Java. Struts proporciona un conjunto de validaciones que abarcan un gran número de posibilidades.

Para realizar estas validaciones es necesario definir una característica del fichero de configuración y un nuevo fichero XML en el que se definen las validaciones a realizar. En este fichero se debe definir el campo del formulario, la clase en la cual está definida la validación, el método que realiza la validación, los parámetros del método y el mensaje que se mostrará en caso de que no se cumpla con la validación. Estas validaciones se realizarían en el servidor pero en caso de querer que estas validaciones se realicen en el cliente simplemente se tendría que añadir a estas definiciones un código en JavaScript.

Finamente decir a modo de resumen, que Struts es un framework para el desarrollo de aplicaciones Web siguiendo el patrón MVC, en el cual el desarrollador se debe encargar de definir el modelo, las vistas y el fichero de configuración de Struts.

4.1.2 Solr

Solr es un motor de búsqueda e indexación de contenidos. Entre las características destacan las búsquedas de texto completo, resaltado de resultados, clustering dinámico, y manejo de documentos con texto enriquecido (Word, PDF…).

Solr es un sistema de código abierto y está basado en Apache Lucene. Este sistema ha sido implementado en Java, teniendo como requisito ser ejecutado en un contenedor de Servlets. Solr es escalable, permitiendo realizar búsquedas distribuidas y la replicación de índices.

Como breve notación histórica, Solr fue creado en 2004 por Yonik Seeley en CNET Networks con la intención de crear un buscador en la Web de la empresa. Posteriormente fue donado a "Apache Software Foundation" en 2006. En 2010 se fusionaron los proyectos Solr y Lucene de forma que actualmente se distribuyen por separado pero pasaron a ser desarrollados por el mismo equipo. Actualmente, el desarrollo y las últimas versiones están orientados a aprovechar la actual tendencia de poseer la información en la nube.

Solr proporciona un API estilo REST, aunque no completo, de forma que en vez de usar drivers o APIs programáticas se usa el protocolo HTTP. Solr permite realizar peticiones HTTP para indexar o consultar documentos. Y permite la recuperación de documentos en formato XML y JSON.

Por el protocolo de comunicación, Solr es independiente al lenguaje de la nueva aplicación y de los tipos de datos ya que en el protocolo HTTP solo se intercambia información en formato de texto.

Otras características destacables se comentan en los siguientes párrafos. Solr contiene caches internas para responder con mayor velocidad a las peticiones recibidas. También incluye una interfaz web que proporciona: estadísticas de rendimiento y del uso de cache, realizar búsquedas mediante un formulario, navegar por los términos más populares del índice, visualizar un desglose detallado de las matemáticas de puntuación y las fases de análisis de texto.

Finalmente cabe destacar que existen librerías para poder usar este sistema desde distintos lenguajes, destacando que el paquete para su uso en Java es SolrJ, aunque no es necesario su uso, ya que se pueden definir librerías propias.

Desde el paquete SolrJ se pueden realizar búsquedas sobre los datos indexados en el servidor Solr de distintas formas: por coincidencias en cada campo, un conjunto de valores con coincidencia en cualquier campo, se puede limitar el número de resultados devueltos, se pueden ordenar los resultados devueltos y algunas posibilidades más.

4.1.3 SAX

SAX es un API para realizar lecturas de ficheros XML. SAX son las siglas de "Simple API for XML". Originalmente fue creado únicamente para programación en Java, pero actualmente existen APIs para más lenguajes de programación.

SAX es un procesador de ficheros XML de acceso secuencial y basado en eventos. Es decir, va recorriendo el xml en orden (de principio a fin) y va generando eventos conforme va encontrando determinadas partes del XML, como el principio de etiqueta, el fin de etiqueta, texto…

A la hora de usar SAX es necesario crear una nueva clase que herede del manejador por defecto de SAX "DefaultHandler" y sobrescribir los métodos correspondientes a los eventos deseados. Los métodos más comúnmente sobrescritos de "DefaultHandler" son: startDocument, endDocument, startElement, endElement y characters.

Esos métodos son ejecutados en función de los eventos lanzados en función de lo leído en el fichero XML. El método startDocument se ejecuta al comenzar el procesado del documento XML. El método endDocument se ejecuta al finalizar el procesado del documento XML. El método startElement se ejecuta al comenzar el procesado de una etiqueta XML. Es en este método donde se leen los atributos de las etiquetas. El método endElement se ejecuta al finalizar el procesado de una etiqueta XML. El método characters se ejecuta al encontrar una cadena de texto.

La principal ventaja de usar SAX es que para realizar la lectura de datos se realiza sin una gran carga de memoria debido al procesamiento secuencial.

Entre sus desventajas destaca que no permite el acceso aleatorio y no permite la creación de documentos XML, ya que solo sirve para la realización de lecturas.

4.2 Arquitectura del sistema

4.2.1 Visión global

Como viene predeterminado por el uso de Struts, se ha optado por una arquitectura basada en el patrón Modelo Vista Controlador. Por ello la aplicación se distribuirá en 3 paquetes lógicos, que no se llegarán a implementar de forma que solo se mostrarán en los diagramas de los diseños.

El paquete Vista debe contener los paquetes orientados a mostrar las interfaces a los usuarios. Este paquete englobará los ficheros Java que definen los formularios y los ficheros JSP que son los que definen las interfaces que finalmente van a ser visionados en el navegador web utilizado.

El paquete Controlador debe contener el paquete que se encarga de controlar las acciones que se reciben a través de las interfaces.

El paquete Modelo incluye los paquetes que se encargarán del tratado de los datos de la aplicación. Estos paquetes se encargarán de la selección y lectura de los ficheros Log, de la introducción y consulta de los datos en el sistema de indexado y del acceso al fichero de configuración.

El diagrama de la arquitectura descrita anteriormente es el siguiente:

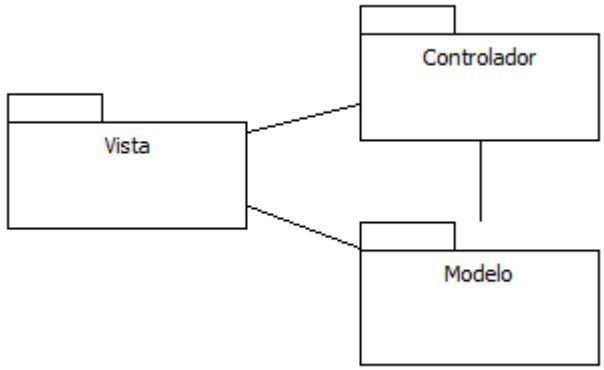


Ilustración 13: Arquitectura del sistema

Una vez que se han puesto las bases de la arquitectura se van a ir diseñando más en detalle los paquetes comentados anteriormente.

4.2.2 Diseño de la arquitectura

Una vez elegida la arquitectura se van a ir diseñando más en detalle los paquetes comentados anteriormente. Antes de realizar ese diseño más detallado, se debe comprender el ámbito en el que se producirá el despliegue de la aplicación. Para ello se adjunta un diagrama en el que se muestran los nodos y los otros sistemas involucrados

4.2.2.1 Topología del sistema (Diagrama de despliegue)

A continuación se muestra el diagrama que representa la situación en donde se desplegará el nuevo sistema:

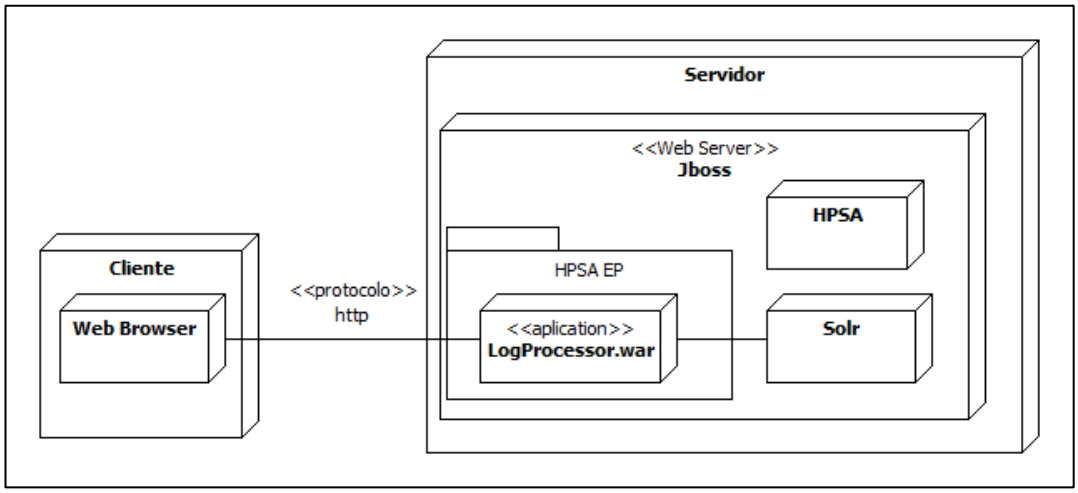


Ilustración 14: Diagrama de despliegue

En el diagrama se pueden ver los distintos sistemas que intervienen con la aplicación y los protocolos de comunicación. La aplicación se usará a través de un navegador Web mediante el protocolo "http".

La aplicación a desarrollar está representada por el nodo "LogProcessor.war". En este diagrama también se puede ver que el sistema estará englobado dentro del paquete HPSA EP y se comunicará con Solr, que debe estar instalado en el servidor Web, JBoss.

4.2.3 Diseño lógico de la arquitectura del sistema

4.2.3.1 Primera Versión de la arquitectura lógica

Para empezar con el diseño de la arquitectura, se comienza diseñando la estructura de paquetes en la que después se incluirán las clases a crear. En el diagrama se pueden ver los distintos paquetes en los que se estructurará la aplicación, así como la arquitectura establecida en los puntos anteriores.

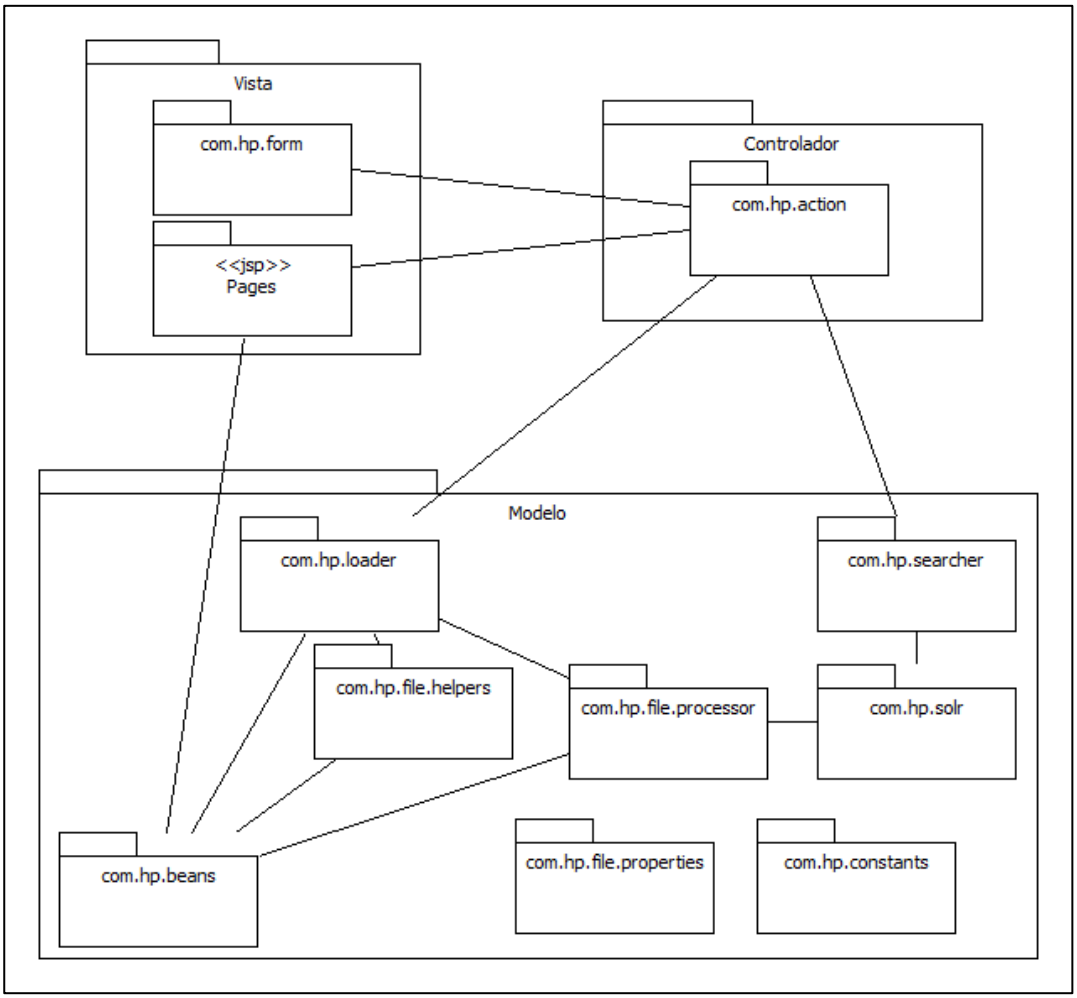


Ilustración 15: Arquitectura del sistema - Estructura de paquetes

Una vez visto el diagrama, en el que se puede ver la arquitectura y los paquetes en los que se dividirá la aplicación se van a describir el contenido que contendrá cada paquete.

Dentro del paquete “Vista”:

- “com.hp.form”: En este paquete, se encuentran las clases que define los campos de los formularios de la aplicación.

- “Pages”: Esta paquete contiene los ficheros .jsp que definen las interfaces de la aplicación.

Dentro del paquete “Controlador”:

- “com.hp.action”: Dentro de este paquete se encuentran las clases que definen las acciones a realizar como respuesta a los eventos producidos por el usuario.

Dentro del paquete “Modelo”:

- “com.hp.loader”: Este paquete se encarga de iniciar el proceso para la carga de los ficheros de Log que cumplen los requisitos en la carpeta especificada en el fichero de configuración.

- “com.hp.searcher”: Este paquete se encarga de llamar a las clases encargadas de realizar la búsqueda en Solr de entradas de Log.

- “com.hp.file.helpers”: Las clases de este paquete, se encargarán de obtener los ficheros Log que van a ser procesados y cargados en Solr.

- “com.hp.beans”: Dentro de este paquete se encuentran las clases que definen los objetos del modelo de dominio necesarios en la aplicación. En la aplicación será necesario implementar clases para el fichero de Log y para las entradas de Log.

- “com.hp.file.processor”: Este paquete contiene las clases encargadas de realizar el parseo de los ficheros de Log.

- “com.hp.solr”: Este paquete se encargará de la comunicación con Solr.

- “com.hp.file.properties”: Este paquete se encarga de realizar la lectura del contenido del fichero de configuración, que tendrá información sobre la configuración de Solr y sobre los ficheros de Log a procesar.

- “com.hp.constants”: En este paquete se definen las constantes usadas en la aplicación.

Una vez descritos los paquetes que han de ser generados para la construcción de la aplicación es necesario determinar cuáles son los paquetes dependientes de las tecnologías usadas. Los paquetes que son dependientes de Struts son “com.hp.form”, “Pages” y “com.hp.action”. El paquete “com.hp.solr” depende de Solr. El paquete “com.hp.file.processor” depende de SAX.

Sobre esta primera versión de la arquitectura lógica solo cabe destacar que los paquetes “com.hp.loader” y “com.hp.searcher” son los paquetes que soportan la interfaz de acceso al Modelo de la aplicación.

4.2.3.2 Segunda Versión de la arquitectura lógica

En el siguiente diagrama se detallan las clases que se deben implementar en cada uno de los paquetes descritos en el punto anterior.

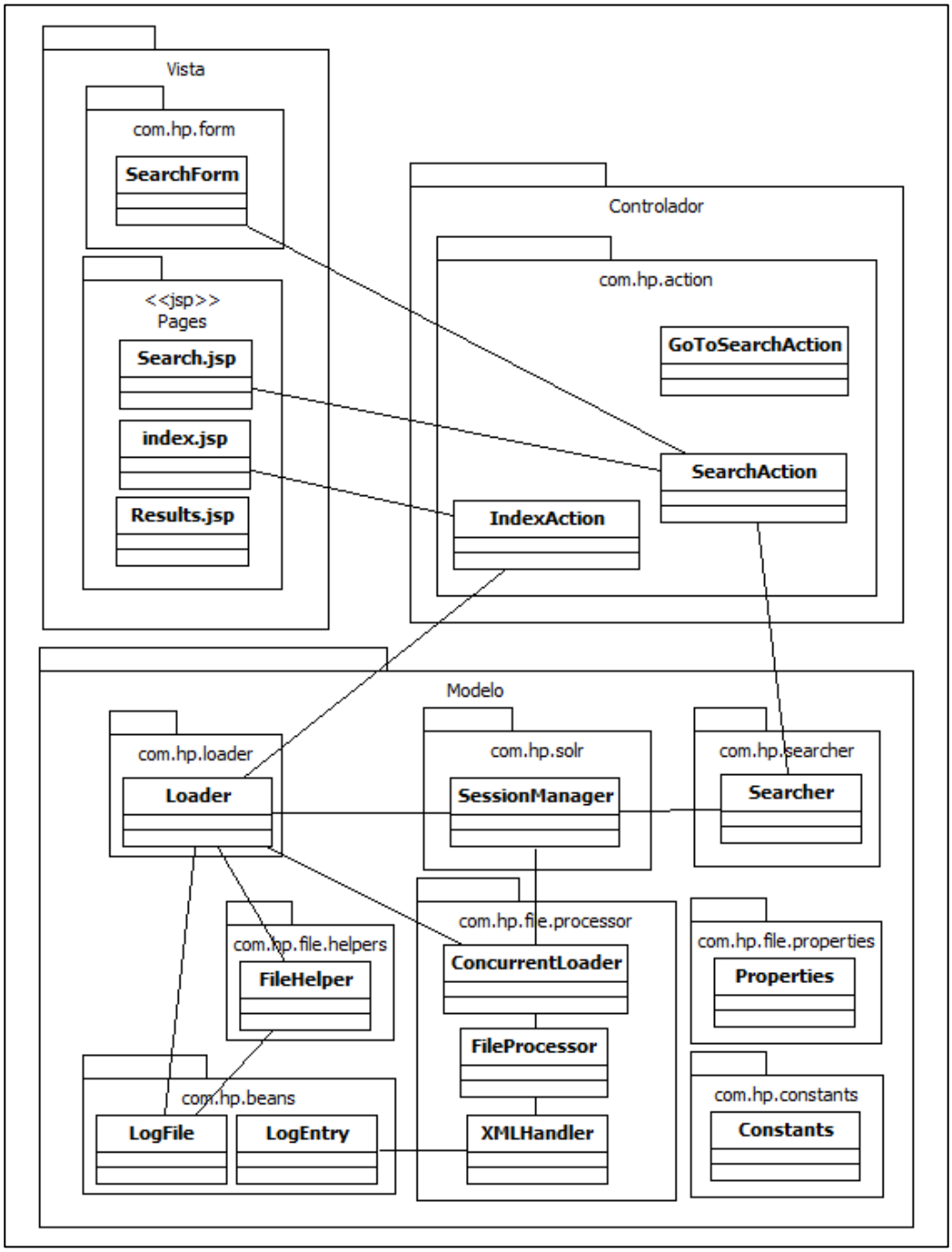


Ilustración 16: Segunda Versión de la arquitectura lógica

Una vez visto las clases que se deben implementar es necesario establecer una descripción del contenido que tendrá cada una de ellas. En las siguientes líneas se muestra dicha descripción:

- GoToSearchAction.java: esta clase se encarga de realizar la transición desde la vista de los resultados a la vista de búsqueda, esta transición se realizará cuando en la página Results.jsp se pulse el botón "New Search".

- IndexAction.java: esta clase realizará la primera fase del caso de uso, iniciará la carga de datos. Para ello, se comunicará con la clase "Loader". Por ultimo realizará la transición al formulario de búsqueda.

- SearchAction.java: esta clase es la encargada de lanzar la búsqueda, acorde con los parámetros introducidos en el formulario. Y posteriormente realizar la transición a la página de resultados.

- SearchForm.java: define la estructura del formulario con los criterios sobre los que se realizará la consulta.

- Loader.java: se encarga de interactuar con FileHelper.java para la obtención de los ficheros de Log a tratar y posteriormente lanzará el procesamiento de parseo para cada fichero obtenido anteriormente. El procesamiento de cada fichero se realizará paralelamente. Inicialmente realizará un borrado del contenido de Solr, para que anteriores cargas no interfieran con los datos que se deseen cargar.

- Searcher.java: se encargará de interactuar con "SessionManager.java" para obtener los resultados de la consulta.

- SessionManager.java: se encargará de la comunicación con el sistema Solr, siendo responsable de todas operaciones que se realicen sobre el sistema.

- FileHelper.java: se encargará de obtener los ficheros Log que van a ser procesados y cargados en Solr, acorde con los parámetros del fichero de configuración.

- LogEntry.java: definirá una entrada del Log.

- LogFile.java: definirá un fichero de Log.

- ConcurrentLoader.java: esta clase será la encargada de interactuar con la clase "SessionManager" para introducir el contenido del fichero analizado, como resultado también introducirá una línea en el Log del servidor mostrando el tiempo en que se han realizado las anteriores operaciones.

- FileProcessor.java: esta clase es la encargada de interactuar con SAX para realizar la transición de datos del fichero XML a los correspondientes objetos Java.

- XMLHandler.java: esta clase definirá la forma en que SAX va a realizar la equivalencia desde el fichero XML a los objetos Java.

- Constants.java: definirá las constantes que serán usadas en las distintas clases de la aplicación.

- Properties.java: se encarga de realizar la lectura del contenido del fichero de configuración.

Una vez descritas las clases, solo queda definir los ficheros JSP contenidos en el paquete "Pages", que describen las interfaces de la aplicación:

- Index.jsp: este fichero no definirá una interfaz de la aplicación, simplemente cumplirá con la función de ser la puerta de entrada a la aplicación y cuando se acceda a ella se comenzará el tratamiento de los ficheros Log.

- Result.jsp: este fichero definirá como se mostrarán los resultados de las búsquedas.

- Search.jsp: este fichero definirá la interfaz en la que se muestra el formulario con los campos en función de los cuales se realizará la búsqueda.

4.3 Casos de Uso de Diseño

4.3.1 Descripción de los casos de uso de diseño

DUC-0001	**Cargar y consultar**	
Versión	1.0 (08/03/13)	
Descripción	Carga los ficheros de Log que designa el fichero de configuración y se realiza una búsqueda de entradas del Log en función de unos criterios.	
Actores	Usuario	
Precondición	El servidos Web está iniciado	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor accede a la aplicación a través de un navegador Web.
	2	El sistema consulta el fichero de configuración
	3	El sistema carga los ficheros de Log que coinciden con los datos del fichero de configuración en Solr.
	3	El sistema muestra un formulario con los siguientes campos: level, hostname, fecha inicial y fecha final, module, Part, Component, Topic y Thread.
	4	El usuario introduce los criterios de búsqueda y pulsa el botón “Search”.
	5	El sistema busca las entradas del Log que corresponden con los criterios de búsqueda introducidos por el usuario
	6	El sistema muestra los resultados de la búsqueda mostrando las entradas de Log correspondientes en formato tabla.
	7	El caso de uso finaliza.
Postcondición	Se han mostrado las entradas del Log correspondientes con los criterios de búsqueda introducidos.	
Flujos Alternativos	**Alt**	**Acción**
	1	En el paso 5 si alguno de los criterios introducidos anteriormente no es correcto se vuelve al paso 3.
	2	En el paso 6, si se pulsa el botón “Nueva Búsqueda” se vuelve al paso 3.
	3	En cualquier punto el usuario puede salir de la aplicación cancelando cualquier operación y el caso de uso finaliza.
Importancia	Alta	

Tabla 4: Descripción del diseño del caso de uso "Cargar y consultar"

4.3.2 Realización de casos de uso de diseño

En este apartado se va a mostrar el diseño del caso de uso a través de diagramas de secuencia. Se ha optado por dividir el caso de uso en dos diagramas, para que la representación sea más clara. De este modo diferenciamos el caso de uso en dos partes: el primer diagrama muestra como se cargan los ficheros en el sistema de indexado y el segundo muestra como se realizan las búsquedas en el sistema.

El siguiente diagrama de secuencia esquematiza la carga de los datos:

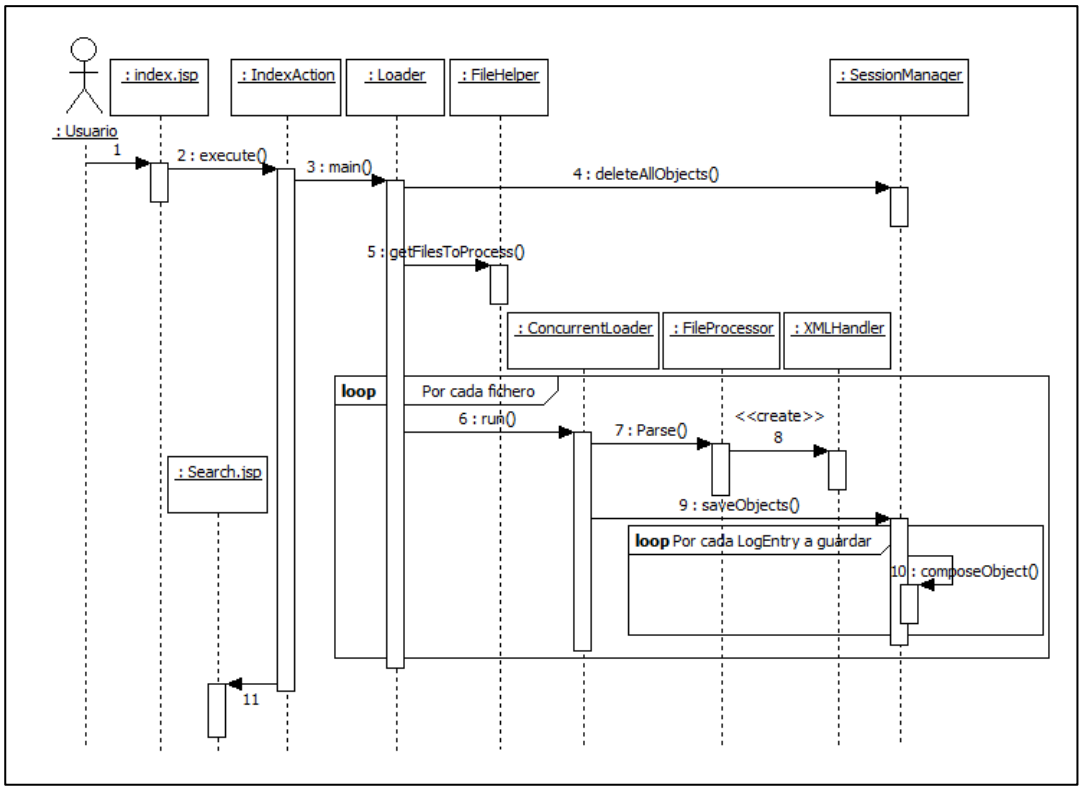


Ilustración 17: Diagrama de secuencia de la carga de datos

En el anterior diagrama se pueden ver la colaboración de las clases para llevar a cabo la carga de los datos de los ficheros de Log. Se puede ver que para que los datos presentes en Solr, sean los acordes con el fichero de configuración primero se realiza un borrado de los datos anteriores, ya que los parámetros del fichero de configuración pueden haber sido modificados. De esta forma se evita que datos de cargas anteriores interfieran en las posibles nuevas búsquedas.

La consulta de los datos del fichero de configuración ha sido omitida para que no se vea afectada la claridad del diagrama. Esta consulta la realizaría la clase "FileHelper" en el momento en que recibe la orden "getFilesToProcess()", para calcular que ficheros deben ser procesados de acuerdo al contenido del fichero de configuración. Para consultar los parámetros del fichero de configuración se deberán realizar las llamadas pertinentes a la clase "Properties".

También aclarar que la lectura de cada fichero se realizará en hilos distinto, paralelizando este proceso para que pueda ser más rápida la lectura de todos los ficheros.

El siguiente diagrama de secuencia, corresponde a la parte de la búsqueda:

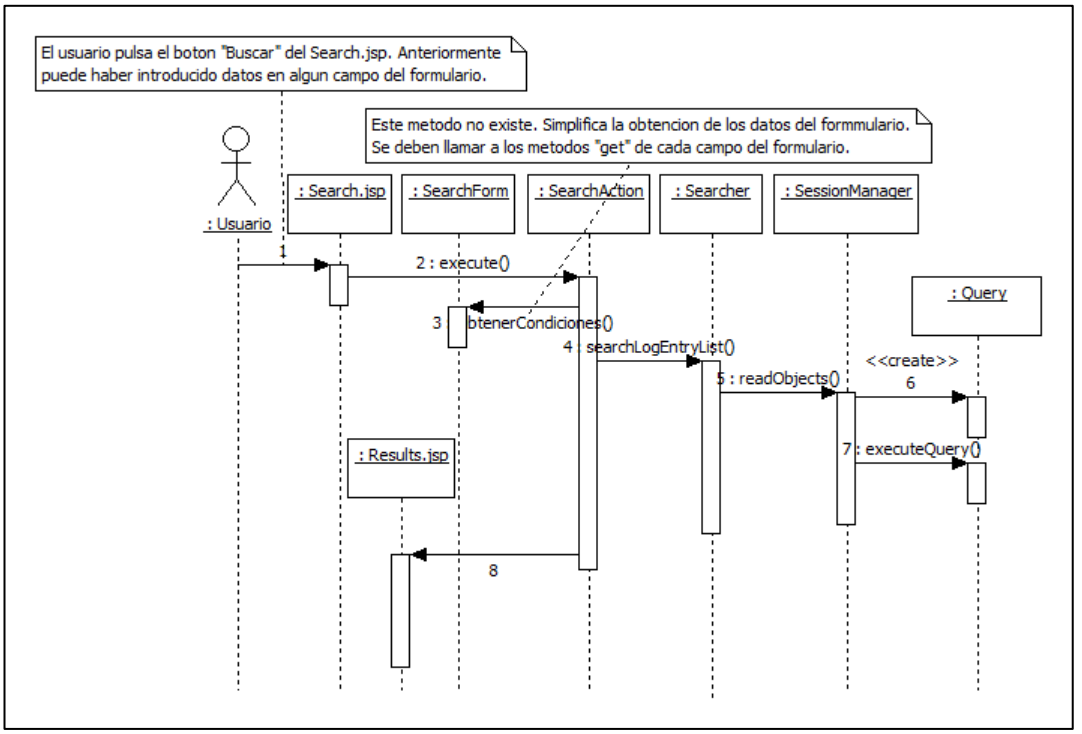


Ilustración 18: Diagrama de secuencia de la búsqueda

En el anterior diagrama se pueden ver cómo interactúan las clases para llevar a cabo la consulta sobre los datos cargados anteriormente. Se puede ver que como se realiza la obtención de los campos del formulario, a través de los métodos “get” de la clase “SearchForm” simplificado en el método “ObtenerCondiciones()” presente en el diagrama.

Una vez obtenidos los valores introducidos por el usuario en el formulario y pasados a la clase “SessionManager”, se crea una consulta con las condiciones recibidas. A continuación cuando el “SearchAction” tenga el resultado de la consulta, se mostrará en “Results.jsp”

Los retornos y parámetros de las operaciones presentes en los diagramas de secuencia se detallarán en las siguientes páginas en las que se puede ver un diseño más detallado de las clases.

4.4 Diagrama de Clases de Diseño

En la siguiente imagen se muestra el diagrama de clases con atributos y operaciones:

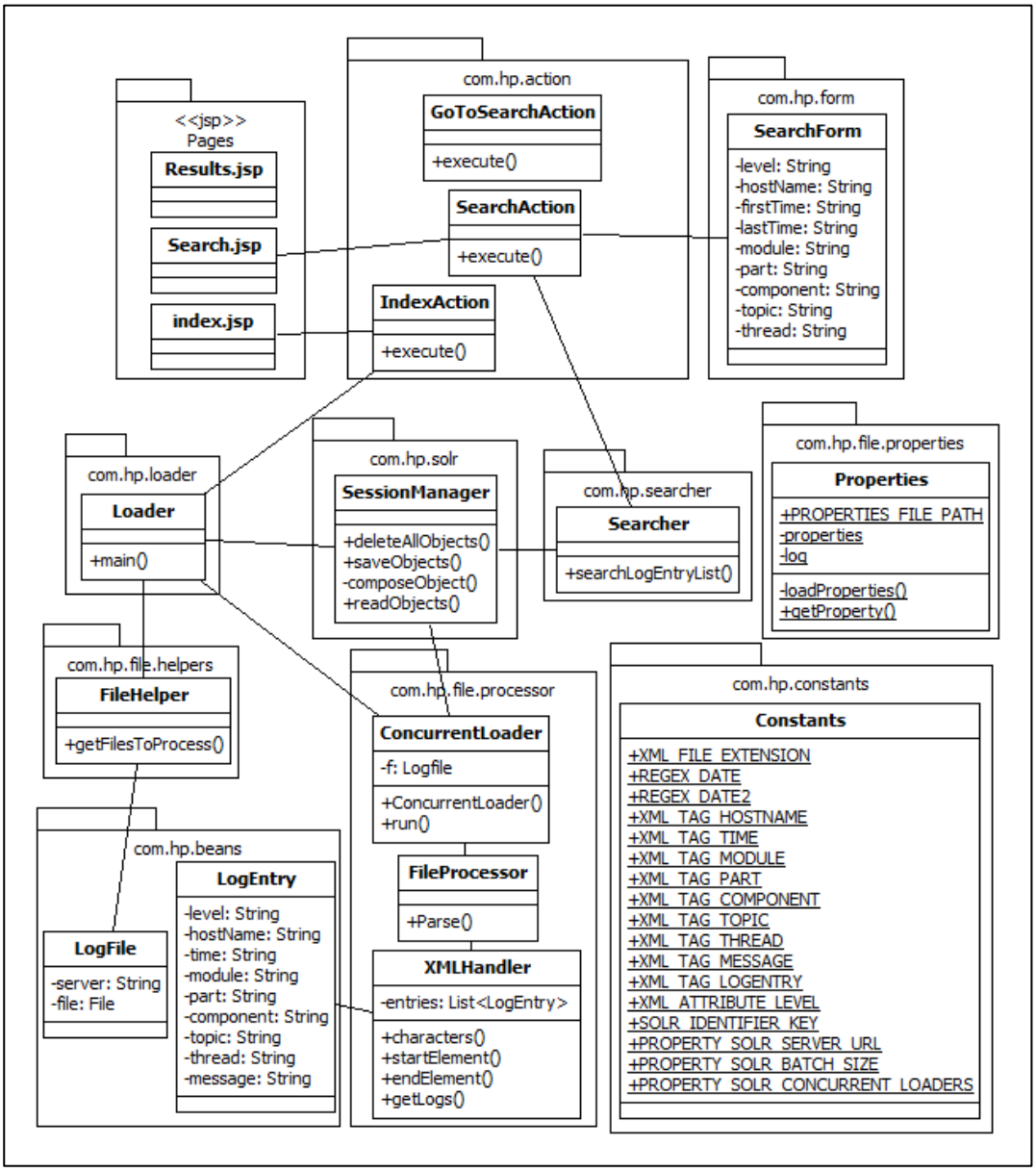


Ilustración 19: Diagrama de Clases de Diseño

Según lo visto en el diagrama de clases de diseño va a ser necesario implementar 14 clases y 3 paginas JSP. En el diagrama anterior no se muestran todas las características de las clases, para que

el diagrama no estuviera demasiado recargado. Estas características se detallarán en el siguiente punto.

4.5 Diseño detallado de los paquetes

En las siguientes páginas se muestran las clases con un nivel superior de detalle definiendo los parámetros y los tipos de valores devueltos en las operaciones, así como el tipo de los atributos y los valores de las constantes. Para ello se va a ir exponiendo un diagrama de clases por cada paquete realizando un breve comentario sobre el contenido del paquete

En lo que se refiere a las páginas JSP no creo necesario realizar una descripción, ya que dicha descripción se realizó en el punto 4.2.3.

4.5.1 "com.hp.action"

Este paquete contiene las clases encargadas de gestionar las acciones realizadas por los usuarios. En Struts deben extender la clase Action implementando la función "execute()". Estas clases quedarían definidas como se muestra en el diagrama siguiente:

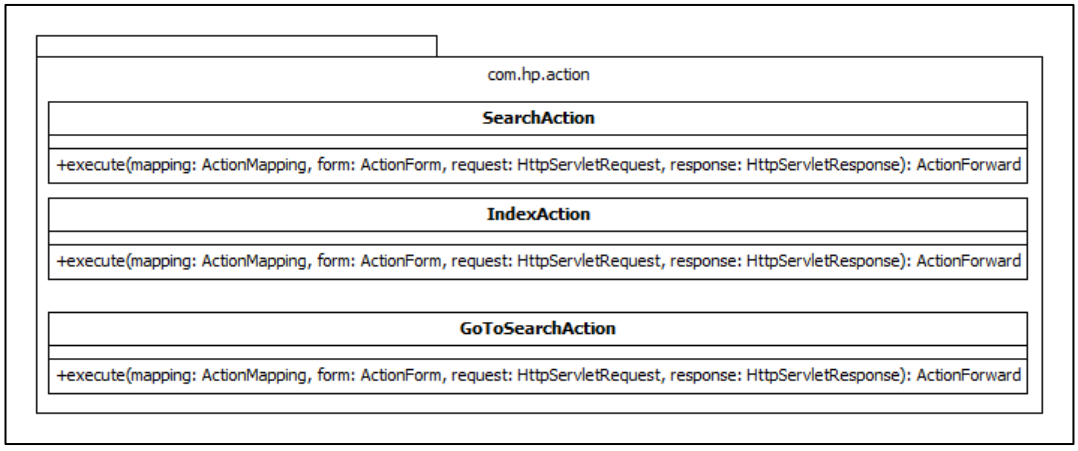


Ilustración 20: Diseño detallado del paquete "com.hp.action"

4.5.2 "com.hp.loader"

Este paquete contiene una única clase que es la encargada de iniciar la carga de los datos de los ficheros de Log. Para ello contará con una única operación que interactúa con otras clases del modelo para borrar los datos anteriores, obtener los ficheros a cargar y realizar la carga de los datos correspondientes.

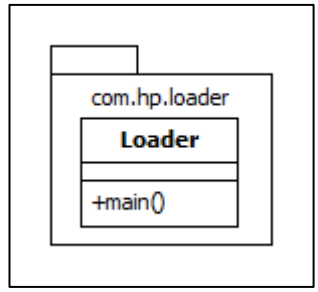


Ilustración 21: Diseño detallado del paquete "com.hp.loader"

4.5.3 “com.hp.form”

Este paquete tiene una sola clase que define los campos del formulario de búsqueda a través de sus atributos, teniendo que definir un “get” y un “set” por cada atributo. En Struts debe extender la clase “ValidatorForm” para poder realizar validaciones sobre los valores que se dan a sus atributos. La clase quedaría definida como se muestra en el diagrama siguiente:

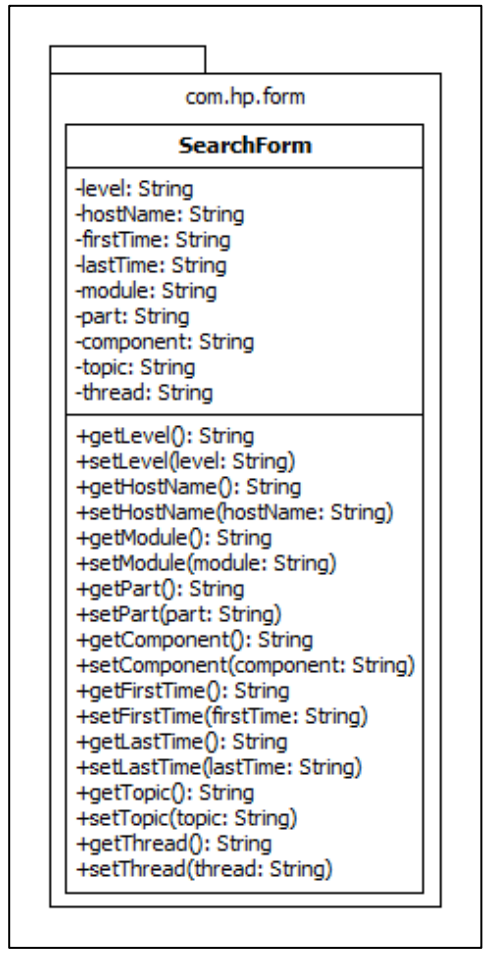


Ilustración 22: Diseño detallado del paquete "com.hp.form"

4.5.4 “com.hp.searcher”

Este paquete contiene una sola clase que se encarga de iniciar la búsqueda en el modelo. La única operación que contiene la clase tiene como parámetros los campos del formulario de búsqueda y devolviendo una lista con los LogEntry resultante de la búsqueda. La clase quedaría definida como se muestra en el diagrama siguiente:

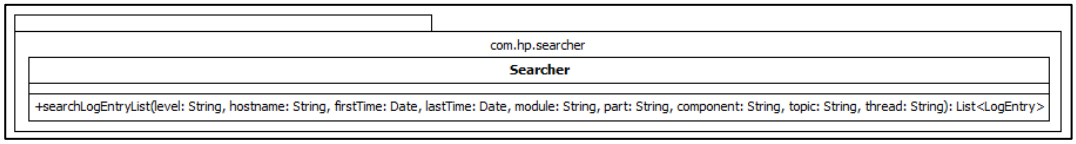


Ilustración 23: Diseño detallado del paquete "com.hp.searcher"

4.5.5 “com.hp.solr”

Este paquete tiene una sola clase que se encarga de interactuar con Solr. La clase tiene una operación para borrar los datos de cargas anteriores, otra para guardar los datos nuevos y otra a través de la que se realiza la consulta. Cabe destacar que en Solr los objetos se deben guardar como un objeto especificado de Solr por lo que fue necesario crear la función “composeObject()” para realizar la conversión de LogEntry al formato especifico de Solr. La clase quedaría definida como se muestra en el diagrama siguiente:

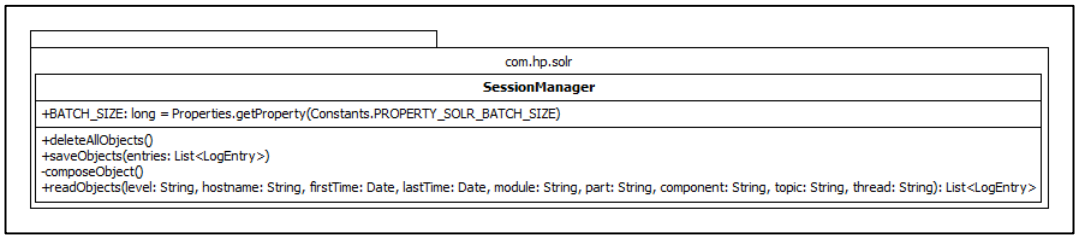


Ilustración 24: Diseño detallado del paquete "com.hp.solr"

4.5.6 “com.hp.file.helpers”

Este paquete tiene una sola clase que se encarga de la obtención de los ficheros de Log que van a ser procesados en la aplicación, en función de los datos presentes en el fichero de configuración. Esta clase posee una única operación en la que se devuelve una lista de los ficheros a procesar. La clase quedaría definida como se muestra en el diagrama siguiente:

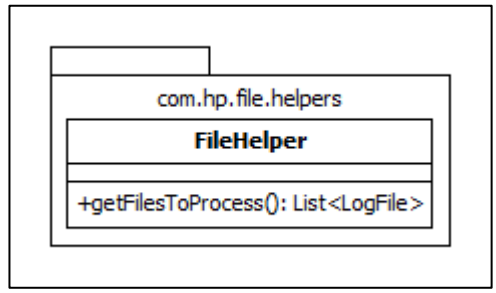


Ilustración 25: Diseño detallado del paquete "com.hp.file.helpers"

4.5.7 “com.hp.file.processor”

Este paquete contiene las clases que definen la forma de realizar la equivalencia entre el fichero XML y los objetos LogEntry correspondientes. La clase “ConcurrentLoader” es la encargada de definir el fichero que se procesa y se comunica con el paquete “com.hp.solr” para introducir los datos. La clase “FileProcessor” es la encargada de la creación de los objetos de SAX necesarios para realizar la equivalencia. La clase “XMLHandler” define la lógica de como leer los datos del fichero XML y convertirlo en objetos LogEntry. Las clases quedarían definidas como se muestra en el diagrama siguiente:

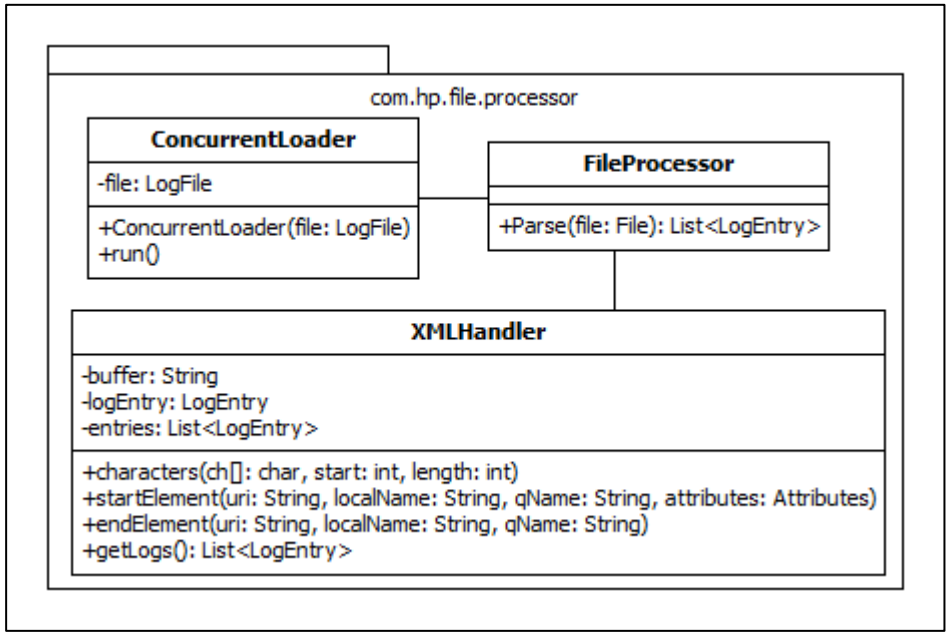


Ilustración 26: Diseño detallado del paquete "com.hp.file.processor"

4.5.8 “com.hp.file.properties”

Este paquete tiene una sola clase en la que se realiza la lectura del fichero de propiedades. La clase quedaría definida como se muestra en el diagrama siguiente:

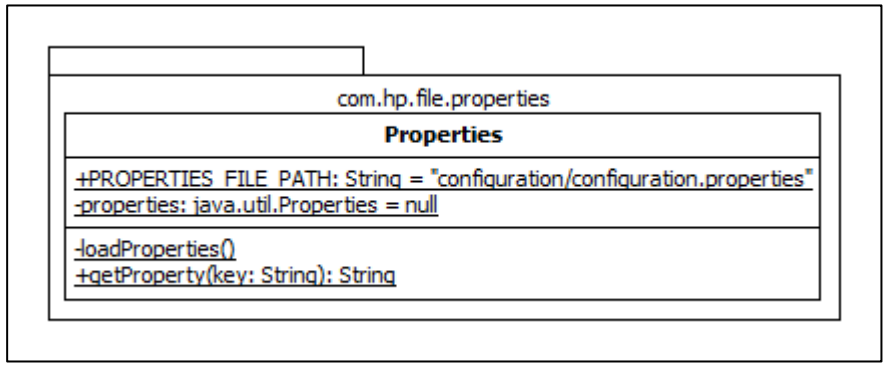


Ilustración 27: Diseño detallado del paquete "com.hp.file.properties"

4.5.9 “com.hp.beans”

Este paquete contiene las clases que definen las entidades necesarias de las mostradas en el modelo de dominio de análisis. Las dos clases existentes definen los ficheros de Log y las entradas de Log, definiendo los atributos necesarios para la aplicación. En cada clase también se definen métodos “get” y “set” para cada atributo. Entre las operaciones cabe aclarar que las operaciones “getTimeObject()” y “setTimeObject()” obtienen y asignan el valor del atributo “time” en función de un objeto de la clase “Date”, estando el atributo “time” siempre con el formato definido por la constante “REGEX_DATE” definida en la clase “Constants”. Las clases quedarían definidas como se muestra en el diagrama siguiente:

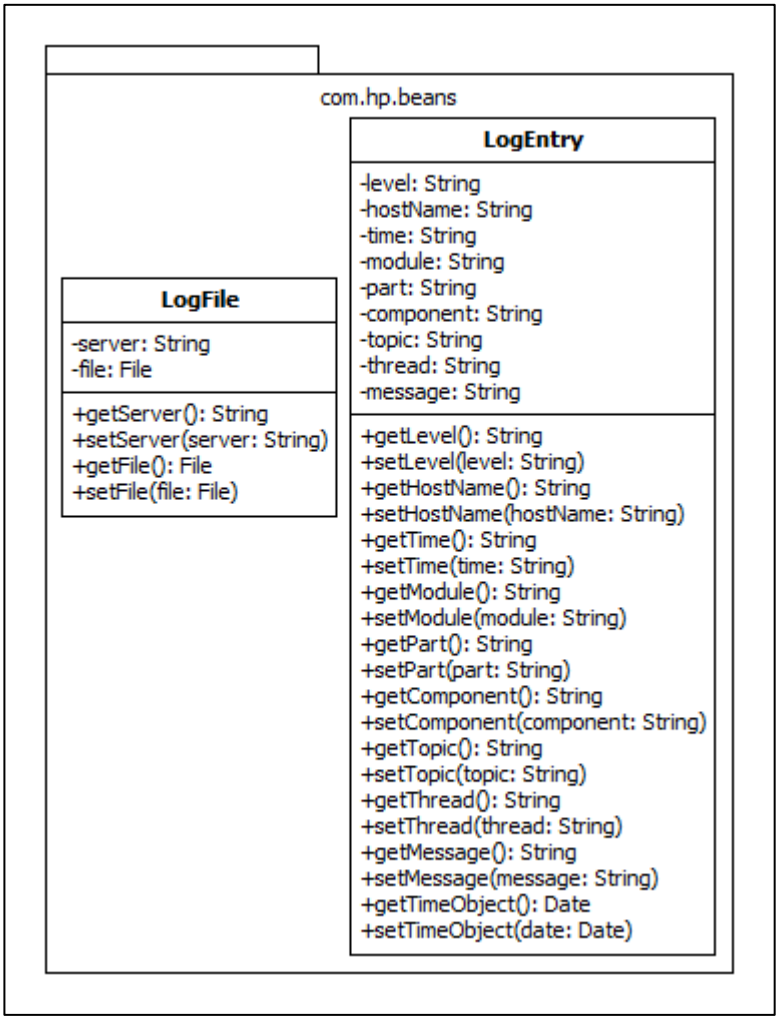


Ilustración 28: Diseño detallado del paquete "com.hp.beans"

4.5.10 “com.hp.constants”

Este paquete contiene una sola clase que define las constantes usadas dentro de la aplicación. Las constantes son variadas ya que se definen: etiquetas de los ficheros XML, el formato de las fechas usadas y las claves del fichero de propiedades. Las constantes de los formatos de fechas son dos en función desde donde provienen las fechas: la constante “REGEX_DATE” representa el formato de las fechas en los ficheros de Log y “REGEX_DATE2” es el formato de entrada de fechas a través de la interfaz de la aplicación. La clase quedaría definida como se muestra en el diagrama siguiente:

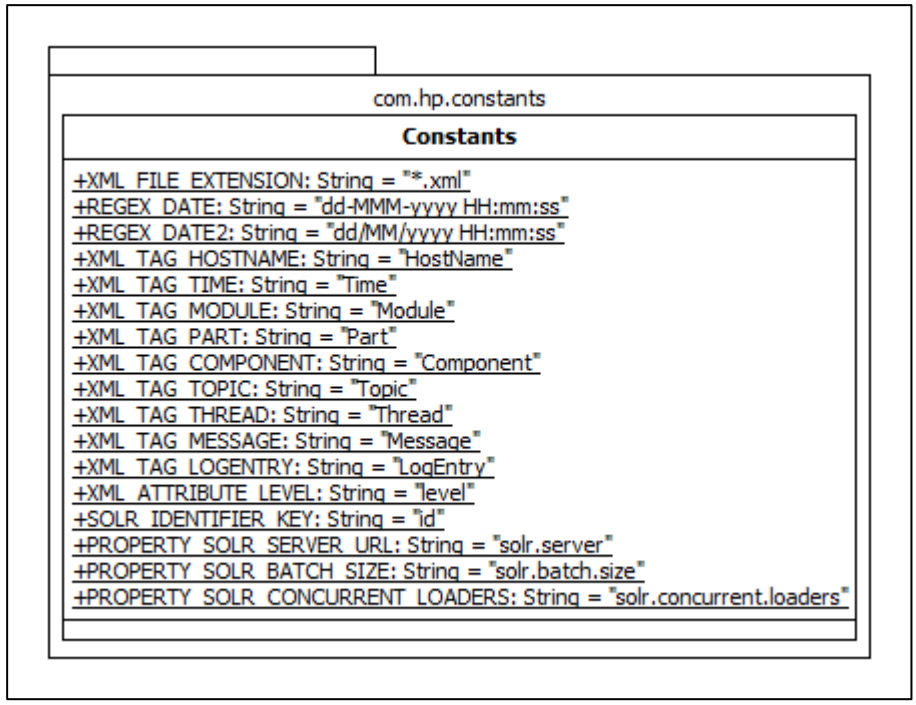


Ilustración 29: Diseño detallado del paquete "com.hp.file.constants"

4.6 Diseño de las interfaces

En este apartado se van a mostrar el diseño de las interfaces necesarias para crear la aplicación. Primero se ha determinado que es necesario crear dos interfaces para la aplicación. La primera debe ser la interfaz de consulta, en ella se debe determinar qué características deben de tener las entradas de Log que a buscar. La segunda debe ser la interfaz a través de la cual se muestran los resultados de la búsqueda.

4.6.1 Interfaz de consulta

La interfaz de consulta debe de constar de un campo de entrada de texto por cada campo de la entrada de Log por los que se quiera buscar. Los campos por los que se desea realizar la búsqueda son: level, hostname, fecha inicial, fecha final, module, Part, Component, Topic y Thread. Por lo tanto la interfaz dispondrá de nueve campos de introducción de texto y un botón, que será pulsado cuando se deseen ver los resultados para los parámetros introducidos en cada campo.

De los campos mencionados anteriormente solo se deben realizar comprobaciones sobre los campos de fecha de modo que se compruebe si el formato del texto introducido es acorde con el formato de fecha establecido. El resto de campos son simples cadenas de caracteres sin ningún formato establecido.

La interfaz presentará un aspecto similar a la siguiente imagen:

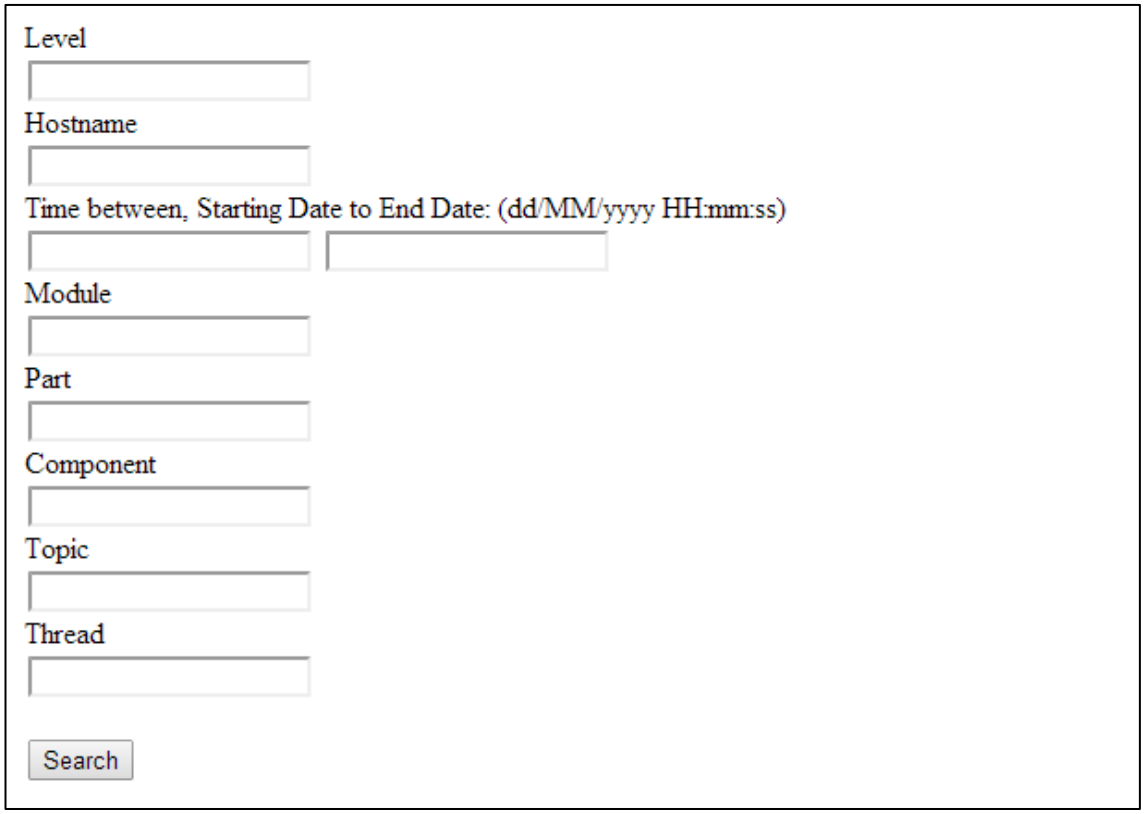


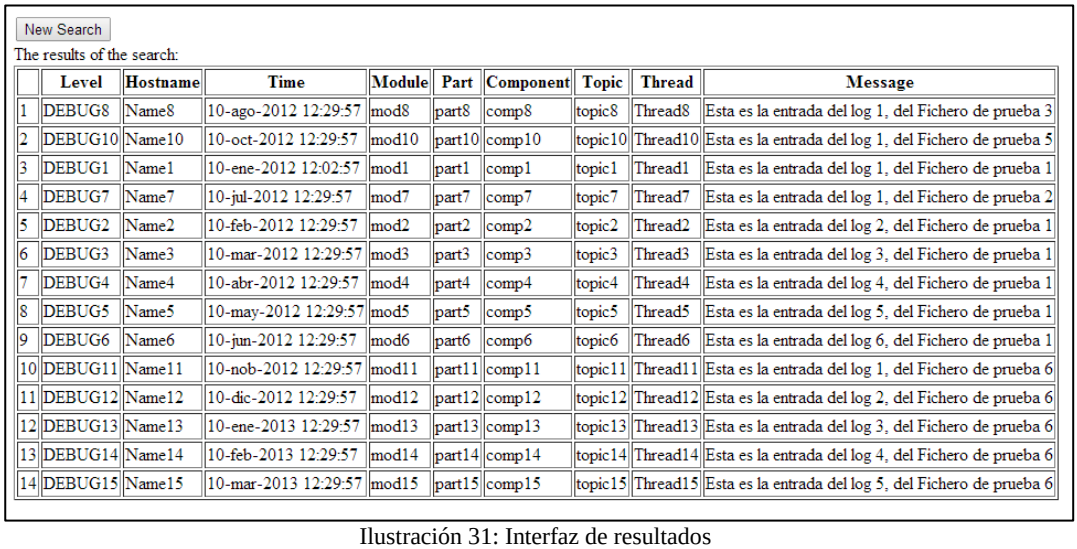
Ilustración 30: Interfaz de consulta

4.6.2 Interfaz de resultados

La interfaz de resultados mostrará las entradas de Log acordes con los parámetros introducidos en la interfaz de búsqueda. Para visualizar mejor las entradas de Log se introducirán en una tabla de forma que en cada columna se introducirá un atributo de las entradas de Log, estableciendo en la primera columna un índice numérico para establecer la cantidad de entradas de Log acordes con la búsqueda realizada. En la primera fila se mostrará una cabecera que indicará el atributo representado en cada columna.

La interfaz también debe disponer de un botón para volver a la interfaz de búsqueda en caso de que el usuario desee realizar una nueva búsqueda.

La interfaz presentará un aspecto similar a la siguiente imagen:



	Level	Hostname	Time	Module	Part	Component	Topic	Thread	Message
1	DEBUG8	Name8	10-ago-2012 12:29:57	mod8	part8	comp8	topic8	Thread8	Esta es la entrada del log 1, del Fichero de prueba 3
2	DEBUG10	Name10	10-oct-2012 12:29:57	mod10	part10	comp10	topic10	Thread10	Esta es la entrada del log 1, del Fichero de prueba 5
3	DEBUG1	Name1	10-ene-2012 12:02:57	mod1	part1	comp1	topic1	Thread1	Esta es la entrada del log 1, del Fichero de prueba 1
4	DEBUG7	Name7	10-jul-2012 12:29:57	mod7	part7	comp7	topic7	Thread7	Esta es la entrada del log 1, del Fichero de prueba 2
5	DEBUG2	Name2	10-feb-2012 12:29:57	mod2	part2	comp2	topic2	Thread2	Esta es la entrada del log 2, del Fichero de prueba 1
6	DEBUG3	Name3	10-mar-2012 12:29:57	mod3	part3	comp3	topic3	Thread3	Esta es la entrada del log 3, del Fichero de prueba 1
7	DEBUG4	Name4	10-abr-2012 12:29:57	mod4	part4	comp4	topic4	Thread4	Esta es la entrada del log 4, del Fichero de prueba 1
8	DEBUG5	Name5	10-may-2012 12:29:57	mod5	part5	comp5	topic5	Thread5	Esta es la entrada del log 5, del Fichero de prueba 1
9	DEBUG6	Name6	10-jun-2012 12:29:57	mod6	part6	comp6	topic6	Thread6	Esta es la entrada del log 6, del Fichero de prueba 1
10	DEBUG11	Name11	10-nob-2012 12:29:57	mod11	part11	comp11	topic11	Thread11	Esta es la entrada del log 1, del Fichero de prueba 6
11	DEBUG12	Name12	10-dic-2012 12:29:57	mod12	part12	comp12	topic12	Thread12	Esta es la entrada del log 2, del Fichero de prueba 6
12	DEBUG13	Name13	10-ene-2013 12:29:57	mod13	part13	comp13	topic13	Thread13	Esta es la entrada del log 3, del Fichero de prueba 6
13	DEBUG14	Name14	10-feb-2013 12:29:57	mod14	part14	comp14	topic14	Thread14	Esta es la entrada del log 4, del Fichero de prueba 6
14	DEBUG15	Name15	10-mar-2013 12:29:57	mod15	part15	comp15	topic15	Thread15	Esta es la entrada del log 5, del Fichero de prueba 6

Ilustración 31: Interfaz de resultados

CAPITULO 5. Pruebas.

Para comprobar que la aplicación cumple con los requisitos establecidos en la fase de análisis, se va realizar un conjunto de pruebas. Estas pruebas las vamos a diferenciar en dos clases, unas pruebas realizadas con ficheros Log diseñados específicamente para probar la aplicación y otras pruebas con un fichero de Log real cedido por el tutor de HP. También se han realizado pruebas de la interfaz, para comprobar la navegabilidad en la aplicación y las entradas de datos sobre los campos en que se realiza una validación. En las siguientes páginas se documentan todas las pruebas realizadas.

Antes de iniciar las pruebas es necesario que el servidor Web este iniciado. Todas las pruebas van a ser realizadas de forma local, estando el servidor Web y el navegador en el mismo equipo. En el equipo en que se han realizado las pruebas la URL de acceso a la aplicación es "http://127.0.0.1:8080/LogProcessor/", siendo "127.0.0.1" la IP que identifica al equipo local, "8080" el puerto del servidor Web y "/LogProcessor/ el identificador de la aplicación. Pudiendo tener que variar estas variables en función de la configuración del equipo en que se realicen las pruebas.

Para comprobar que los ficheros Log han sido cargados en su totalidad se comprobarán los datos de los ficheros que han sido cargados en Solr. Para comprobar que la carga de datos se realiza correctamente se deben seguir los siguientes pasos:

1. Iniciar la aplicación, introduciendo la dirección web correspondiente con nuestra aplicación.

2. Acceder a la interfaz web del Solr de nuestro servidor Web o realizar una búsqueda a través de la interfaz de nuestra aplicación sin rellenar ningún campo pulsando el botón de buscar una vez accedido al formulario de búsqueda.

Todas las pruebas mostradas en la presente memoria se van a realizar sobre la interfaz de búsqueda. Para documentar dichas pruebas se va a rellenar por cada prueba una tabla similar a la mostrada a continuación:

Identificador:	
Objetivo:	
Entrada:	
Salida esperada:	
Salida obtenida:	
Solución:	
Modificaciones:	
Salida Final:	

Tabla 5: Tabla a completar por cada prueba

Los cinco primeros campos de las tablas serán rellenados en todas las pruebas, mientras los últimos tres se rellenarán solo en los casos en que se detecten diferencias entre la salida esperada y la obtenida. En los siguientes párrafos se realizará una breve descripción del contenido de cada campo.

Los campos que serán rellenados en todas las pruebas son: "Identificador", "Objetivo", "Entrada", "Salida esperada" y "Salida obtenida". El campo "Identificador" es usado para identificar cada prueba, a este campo se le asignará un número que se asignará en el orden en que se realizan las pruebas. En el campo "Objetivo" se introducirá una breve descripción de lo que se pretende con la realización de la prueba. En el campo "Entrada" se especificará el valor de los campos a rellenar y las acciones que se realizarán. En el campo "Salida esperada" se especificará la salida que se espera en respuesta a la entrada especificada anteriormente. Y en el campo "Salida obtenida" se especificará la salida obtenida por el uso de la aplicación.

Los campos que serán rellenados en caso de que la salida obtenida en una prueba sea diferente de la salida esperada son los tres últimos campos: "Descripción del fallo", "Modificaciones" y "Salida Final". En el campo "Descripción del fallo" se realizará una breve exposición del motivo por el que se ha producido el fallo. En el campo "Solución:" se especificarán los cambios que se han realizado en la aplicación para solventar el fallo detectado. Y en el campo "Salida Final" se especificará el resultado de realizar la prueba nuevamente con las modificaciones realizadas.

Las pruebas se harán de forma secuencial en el orden en que aparecen en este documento.

5.1 Pruebas de la interfaz

El objetivo general de este conjunto de pruebas es comprobar que la navegabilidad en la aplicación es la esperada y que las validaciones se realizan correctamente. Los únicos campos sobre los que se realizan validaciones son los campos de las fechas, en los que se comprueba el formato del texto introducido. Para realizar estas pruebas los ficheros de Log que se procesarán serán los específicamente creados para probar la aplicación.

Identificador:	1
Objetivo:	Comprobar que el acceso inicial es el esperado.
Entrada:	Se introducirá la URL en el navegador
Salida esperada:	Se muestra el formulario de búsqueda.
Salida obtenida:	OK. Se ha mostrado el formulario de búsqueda.

Identificador:	2
Objetivo:	Comprobar la navegación de los botones.
Entrada:	En el formulario de búsqueda se pulsa el botón “Search”.
Salida esperada:	Se muestra el formulario de los resultados.
Salida obtenida:	OK. Se ha mostrado el formulario con los resultados.

Identificador:	3
Objetivo:	Comprobar la navegación de los botones.
Entrada:	En el formulario de resultados se pulsa el botón “New Search”.
Salida esperada:	Se muestra el formulario de búsqueda.
Salida obtenida:	OK. Se ha mostrado el formulario de búsqueda.

Identificador:	4
Objetivo:	Comprobar la validación de los datos de fecha.
Entrada:	En el primer campo de fecha se introduce “01-ene-2012 00:00:00” y se pulsa el botón “Search”
Salida esperada:	Se muestra un mensaje en el que se informa de que el formato de la fecha es incorrecto y no se cambia de formulario.
Salida obtenida:	OK. En el formulario de búsqueda se ha mostrado el mensaje “The field Starting Date isn't formated correctly.”.

Identificador:	5
Objetivo:	Comprobar la validación de los datos de fecha.
Entrada:	En el segundo campo de fecha se introduce “11-ene-2012 00:00:00” y se pulsa el botón “Search”.
Salida esperada:	Se muestra un mensaje en el que se informa de que el formato de la fecha es incorrecto y no se cambia de formulario.
Salida obtenida:	OK. En el formulario de búsqueda se ha mostrado el mensaje “The field End Date isn't formated correctly”.

Identificador:	6
Objetivo:	Comprobar la validación de los datos de fecha.
Entrada:	En el primer campo de fecha se introduce “01/01/2012 00:00:00” y en el segundo “11/01/2012 00:00:00”, finalmente se pulsa el botón “Search”.
Salida esperada:	Se muestran los resultados de la búsqueda.
Salida obtenida:	OK. Se muestra el formulario de resultados.

5.2 Pruebas con los ficheros Log diseñados.

En este apartado se van a detallar las pruebas realizadas con los ficheros Log diseñados para probar la aplicación. Todas las pruebas se realizarán sobre la interfaz de búsqueda. En este apartado se van a realizar búsquedas por cada campo de forma que haya coincidencias y otras en que no en función de cada campo del formulario de consulta.

Identificador:	7
Objetivo:	Probar búsqueda sin condiciones y comprobación de datos cargados.
Entrada:	Se pulsa el botón "Search".
Salida esperada:	Se muestran todas las entradas de Log de los ficheros de prueba.
Salida obtenida:	OK. Se muestran las 15 entradas de Log presentes en los 6 ficheros de prueba.

Identificador:	8
Objetivo:	Probar búsqueda en función del campo Level.
Entrada:	En el campo Level se introduce "DEBUG1" y se pulsa el botón "Search".
Salida esperada:	Se muestra la entrada de Log que coincide con el parámetro introducido.
Salida obtenida:	OK. Se muestra la entrada de Log con Level="DEBUG1".

Identificador:	9
Objetivo:	Probar búsqueda en función del campo Level.
Entrada:	En el campo Level se introduce "DEBUG" y se pulsa el botón "Search".
Salida esperada:	Se muestra un mensaje en la vista de resultados, diciendo que no existen entradas de Log con esas condiciones.
Salida obtenida:	OK. En la vista de resultados se muestra el mensaje "The search, not found a log entry with the conditions.".

Identificador:	10
Objetivo:	Probar búsqueda en función del campo Hostname.
Entrada:	En el campo Hostname se introduce "Name2" y se pulsa el botón "Search".
Salida esperada:	Se muestra la entrada de Log que coincide con el parámetro introducido.
Salida obtenida:	OK. Se muestra la entrada de Log con Hostname ="Name2".

Identificador:	11
Objetivo:	Probar búsqueda en función del campo Hostname.
Entrada:	En el campo Hostname se introduce "name" y se pulsa el botón "Search".
Salida esperada:	Se muestra un mensaje en la vista de resultados, diciendo que no existen entradas de Log con esas condiciones.
Salida obtenida:	OK. En la vista de resultados se muestra el mensaje "The search, not found a log entry with the conditions.".

Identificador:	12
Objetivo:	Probar búsqueda en función del campo Module.
Entrada:	En el campo Module se introduce "mod3" y se pulsa el botón "Search".
Salida esperada:	Se muestra la entrada de Log que coincide con el parámetro introducido.
Salida obtenida:	OK. Se muestra la entrada de Log con Module ="mod3".

Identificador:	13
Objetivo:	Probar búsqueda en función del campo Module.
Entrada:	En el campo Module se introduce “mod” y se pulsa el botón “Search”.
Salida esperada:	Se muestra un mensaje en la vista de resultados, diciendo que no existen entradas de Log con esas condiciones.
Salida obtenida:	OK. En la vista de resultados se muestra el mensaje “The search, not found a log entry with the conditions.”.

Identificador:	14
Objetivo:	Probar búsqueda en función del campo Part.
Entrada:	En el campo Part se introduce “part4” y se pulsa el botón “Search”.
Salida esperada:	Se muestra la entrada de Log que coincide con el parámetro introducido.
Salida obtenida:	OK. Se muestra la entrada de Log con Part =“part4”.

Identificador:	15
Objetivo:	Probar búsqueda en función del campo Part.
Entrada:	En el campo Part se introduce “part” y se pulsa el botón “Search”.
Salida esperada:	Se muestra un mensaje en la vista de resultados, diciendo que no existen entradas de Log con esas condiciones.
Salida obtenida:	OK. En la vista de resultados se muestra el mensaje “The search, not found a log entry with the conditions.”.

Identificador:	16
Objetivo:	Probar búsqueda en función del campo Component.
Entrada:	En el campo Component se introduce “comp5” y se pulsa el botón “Search”.
Salida esperada:	Se muestra la entrada de Log que coincide con el parámetro introducido.
Salida obtenida:	OK. Se muestra la entrada de Log con Component =“comp5”.

Identificador:	17
Objetivo:	Probar búsqueda en función del campo Component.
Entrada:	En el campo Component se introduce “comp” y se pulsa el botón “Search”.
Salida esperada:	Se muestra un mensaje en la vista de resultados, diciendo que no existen entradas de Log con esas condiciones.
Salida obtenida:	OK. En la vista de resultados se muestra el mensaje “The search, not found a log entry with the conditions.”.

Identificador:	18
Objetivo:	Probar búsqueda en función del campo Topic.
Entrada:	En el campo Topic se introduce “topic6” y se pulsa el botón “Search”.
Salida esperada:	Se muestra la entrada de Log que coincide con el parámetro introducido.
Salida obtenida:	OK. Se muestra la entrada de Log con Topic =“topic6”.

Identificador:	19
Objetivo:	Probar búsqueda en función del campo Topic.
Entrada:	En el campo Topic se introduce “topic” y se pulsa el botón “Search”.
Salida esperada:	Se muestra un mensaje en la vista de resultados, diciendo que no existen entradas de Log con esas condiciones.
Salida obtenida:	OK. En la vista de resultados se muestra el mensaje “The search, not found a log entry with the conditions.”.

Identificador:	20
Objetivo:	Probar búsqueda en función del campo Thread.
Entrada:	En el campo Thread se introduce “Thread7” y se pulsa el botón “Search”.
Salida esperada:	Se muestra la entrada de Log que coincide con el parámetro introducido.
Salida obtenida:	OK. Se muestra la entrada de Log con Thread =“Thread7”.

Identificador:	21
Objetivo:	Probar búsqueda en función del campo Thread.
Entrada:	En el campo Thread se introduce “Thread7” y se pulsa el botón “Search”.
Salida esperada:	Se muestra un mensaje en la vista de resultados, diciendo que no existen entradas de Log con esas condiciones.
Salida obtenida:	OK. En la vista de resultados se muestra el mensaje “The search, not found a log entry with the conditions.”.

Identificador:	22
Objetivo:	Probar búsqueda en función de los campos de fechas.
Entrada:	Se introducen en los campos: el primero=“01/01/2012 00:00:00” y el segundo=“11/01/2012 00:00:00”; y se pulsa el botón “Search”.
Salida esperada:	Se muestra la entrada de Log que coincide con el parámetro introducido.
Salida obtenida:	OK. Se muestra la entrada de Log con Time=”10-ene-2012 12:02:57”

Identificador:	23
Objetivo:	Probar búsqueda en función de los campos de fechas.
Entrada:	Se introducen en los campos: el primero=“01/01/2012 00:00:00” y el segundo=“09/01/2012 00:00:00”; y se pulsa el botón “Search”.
Salida esperada:	Se muestra un mensaje en la vista de resultados, diciendo que no existen entradas de Log con esas condiciones.
Salida obtenida:	OK. En la vista de resultados se muestra el mensaje “The search, not found a log entry with the conditions.”.

En estas pruebas realizadas con los ficheros artificiales, no se ha detectado ningún error. En estas pruebas se ha comprobado la posibilidad de cargar varios ficheros presentes en la carpeta especificada en el fichero de configuración. Y se ha probado la búsqueda en función de cada campo del formulario.

5.3 Pruebas con el fichero Log real

En este apartado se van a detallar las pruebas realizadas con el fichero Log con datos reales cedido para probar la aplicación. Para ello se realizará una prueba para comprobar que se ha cargado todo el fichero y el resto serán pruebas para comprobar que los datos cargados son los presentes en el fichero Log y que las búsquedas en función de un campo devuelven los valores esperados. En las últimas pruebas se probará el funcionamiento para búsquedas combinando los valores en varios campos.

Identificador:	24
Objetivo:	Probar búsqueda sin condiciones y comprobación de datos cargados.
Entrada:	Se pulsa el botón "Search".
Salida esperada:	Se muestran las 205 entradas del fichero de Log.
Salida obtenida:	OK. Se muestran las 205 entradas del Log

Identificador:	25
Objetivo:	Probar búsqueda en función de un campo.
Entrada:	En el campo Level se introduce "INFORMATIVE" y se pulsa el botón "Search".
Salida esperada:	Se muestran 3 entradas de Log que coinciden con el parámetro introducido.
Salida obtenida:	OK. Se muestran las 3 entrada de Log con Level=" INFORMATIVE"

Identificador:	26
Objetivo:	Probar búsqueda en función de un campo.
Entrada:	En el campo Hostname se introduce "Ruben-laptop" y se pulsa el botón "Search".
Salida esperada:	Se muestran 205 entradas de Log que coinciden con el parámetro introducido.
Salida obtenida:	OK. Se muestran las 205 entrada de Log con Hostname=" Ruben-laptop"

Identificador:	27
Objetivo:	Probar búsqueda en función de un campo.
Entrada:	En el campo Module se introduce "mwfm" y se pulsa el botón "Search".
Salida esperada:	Se muestran 205 entradas de Log que coinciden con el parámetro introducido.
Salida obtenida:	OK. Se muestran las 205 entrada de Log con Module=" mwfm"

Identificador:	28
Objetivo:	Probar búsqueda en función de un campo.
Entrada:	En el campo Part se introduce "COMPONENT" y se pulsa el botón "Search".
Salida esperada:	Se muestran 204 entradas de Log que coinciden con el parámetro introducido.
Salida obtenida:	OK. Se muestran las 204 entrada de Log con Part=" COMPONENT"

Identificador:	29
Objetivo:	Probar búsqueda en función de un campo.
Entrada:	En el campo Component se introduce “keep_alive” y se pulsa el botón “Search”.
Salida esperada:	Se muestran 147 entradas de Log que coinciden con el parámetro introducido.
Salida obtenida:	No OK. Se muestran 145 entrada de Log con Component=” keep_alive”.
Descripción del fallo:	Se comprueba que hay datos erróneos en la aplicación: 2 campos Component se han cargado erróneamente faltando partes del texto del campo.
Solución:	Se ha comprobado que el error se produce en el parseo de datos del fichero de Log. Se produce debido a que un conjunto de caracteres entre etiquetas no se lee como un único conjunto de caracteres. Para solucionarlo se modificó la clase “XMLHandler”. Se modificaron los métodos: characters(), startElement() y endElement(), haciendo que el atributo buffer sea igual a el mismo más la nueva cadena en el método characters() y se inicie en una cadena vacía en el método startElement(). Este fallo no se detectó en las pruebas con ficheros diseñados debido a que el fallo se da cuando se produce la lectura de ficheros grandes (en este caso, aproximadamente se daba un error cada 20.000 caracteres del fichero XML).
Salida Final:	OK. Se muestran las 147 entradas

Identificador:	30
Objetivo:	Probar búsqueda en función de un campo.
Entrada:	En el campo Topic se introduce “STARTUP” y se pulsa el botón “Search”.
Salida esperada:	Se muestran 1 entrada de Log que coincide con el parámetro introducido.
Salida obtenida:	OK. Se muestra la entrada de Log con Topic=” STARTUP”

Identificador:	31
Objetivo:	Probar búsqueda en función de un campo.
Entrada:	En el campo Thread se introduce “Thread-23” y se pulsa el botón “Search”.
Salida esperada:	Se muestran 49 entradas de Log que coinciden con el parámetro introducido.
Salida obtenida:	OK. Se muestran las 49 entradas de Log con Thread=” Thread-23”

Identificador:	32
Objetivo:	Probar búsqueda en función de los campos de fechas.
Entrada:	Se introducen en los campos: el primero =“18/12/2012 12:30:06” y el segundo=“18/12/2012 12:33:13”y se pulsa el botón “Search”.
Salida esperada:	Se muestran 146 entradas de Log que coinciden con los parámetros introducidos.
Salida obtenida:	OK. Se muestran las 146 entradas de Log con Time entre “18/12/2012 12:30:06” y “18/12/2012 12:33:13”

Identificador:	33
Objetivo:	Probar búsqueda en función de varios campos.
Entrada:	En el campo Level se introduce “DEBUG”. En Component se introduce “scheduler_module”. En Thread se introduce “Sched1”. Y se pulsa el botón “Search”.
Salida esperada:	Se muestran 2 entradas de Log que coinciden con los parámetros introducidos.
Salida obtenida:	OK. Se muestran las 2 entradas de Log que coinciden con los parámetros comentados en la entrada.

Identificador:	34
Objetivo:	Probar búsqueda en función de varios campos.
Entrada:	En el campo Level se introduce “DEBUG2”. Y en los campos de fechas se introducen “18/12/2012 12:30:14” y “18/12/2012 12:30:34”.
Salida esperada:	Se muestran 22 entradas de Log que coinciden con los parámetros introducidos.
Salida obtenida:	OK. Se muestran las 22 entradas de Log que coinciden con los parámetros comentadas en la entrada.

Identificador:	35
Objetivo:	Probar búsqueda en función de varios campos.
Entrada:	En el campo Level se introduce “DEBUG2”. Y en el campo Thread se introduce “MSC service thread 1-4”.
Salida esperada:	Se muestra una entrada de Log que coincide con los parámetros introducidos.
Salida obtenida:	No OK. Se muestra el mensaje de que no existen coincidencias con los parámetros introducidos
Descripción del fallo:	Se comprueba que por cada campo de la entrada se realiza bien la búsqueda. Pero para Thread = “MSC service thread 1-4” no hay resultados coincidentes
Solución:	Se ha comprobado que el error se produce en la función readObjects() de la clase SessionManager. El error se produce debido a que cuando se introducen los filtros del objeto Query a través del método addFilterQuery() en caso de que existan espacios en el criterio es necesario que vaya entre comillas.
Salida Final:	OK. Se muestran la entrada de Log que coincide con los parámetros comentados en la entrada.

Una vez corregidos los errores detectados en las dos pruebas de este bloque, se volvió a realizar este bloque de pruebas para comprobar que los cambios introducidos en el código no hubieran provocado otros errores.

CAPITULO 6. Conclusiones.

La principal conclusión obtenida es que con la realización de este proyecto se dan por cumplidos los objetivos iniciales del proyecto. Ya que se proporciona un método que procesa los ficheros Log correspondientes con lo determinado por el fichero de configuración.

El primer objetivo, proporcionar una aplicación que procese y almacene la información del Log, en un sistema de indexación de contenidos, le considero cumplido ya que la aplicación creada realiza dichas acciones. La nueva aplicación también proporciona un método para buscar entradas en los ficheros de Log que también era parte del primer objetivo.

El segundo objetivo, que la nueva aplicación no afecte al rendimiento de HPSA, le consideramos cumplido ya que la nueva aplicación no interfiere en el funcionamiento de la aplicación ya que su único intercambio de información se produce a través de los ficheros de Log , que son ficheros resultados de la ejecución del HPSA.

La segunda conclusión obtenida es en referencia a las tecnologías usadas, Struts, Solr y SAX. Sobre Struts tengo la conclusión de que es una tecnología relativamente complicada comparada con otras tecnologías en las que solo es necesario definir las interfaces en lenguaje Java, sin necesidad de usar ficheros .jsp. Sobre Solr destacar que proporciona una forma de indexar datos de forma rápida y sencilla a través del paquete de interfaz que te proporcionan en Java. En cuanto a SAX cabe destacar la velocidad de procesamiento de ficheros XML, aunque no esté permitido el acceso aleatorio.

En este apartado me parece también necesario exponer las posibles líneas futuras de trabajo en la aplicación. A mi parecer, la aplicación se podría mejorar con las siguientes modificaciones:

- Mostrar estadísticas en función de los datos resultantes de las búsquedas. Estas estadísticas podrían ir acompañados de gráficos, de forma que la aplicación genere un resultado rápidamente asimilable por parte de los usuarios.

- La posibilidad de modificar los datos del fichero de configuración desde la aplicación. De forma que no se tenga que tener conocimiento sobre su posición en el sistema de ficheros ni se pueda modificar cualquier valor de los presentes en el fichero.

BIBLIOGRAFIA

En este apartado se va a realizar una enumeración de las referencias bibliográficas consultadas para la realización de las prácticas. Clasificando las referencias en función del soporte de la referencia.

A continuación se enumeran los libros consultados:

[1] Craig Larman, UML Y PATRONES: Una introducción al análisis y diseño orientado a objetos y al proceso unificado, Segunda edición 2003. PRENTICE HALL. ISBN: 84-205-3438-2.

[2] David Roldán Martínez, Pedro J. Valderas Aranda y Óscar Pastor López, Aplicaciones Web: Un enfoque práctico, Edición de 2010. Ra-Ma Editorial. ISBN: 978-84-7897-957-8.

A continuación se enumeran las referencias bibliográficas de las Webs consultadas, que han sido organizadas a través del programa Zotero:

[3] "JBoss: Curso JBoss." [Online]. Available: http://itdba.wikispaces.com/file/view/Curso+de+Jboss+Application+Server+Dia2.pdf. [Accessed: 16-Jan-2014].

[4] "JBoss: Documentation JBoss 7.1." [Online]. Available: https://docs.jboss.org/author/display/AS71/Documentation. [Accessed: 16-Jan-2014].

[5] "JBoss: Tutorial de montaje de aplicaciones en servidor JBoss," 29-Nov-2010. [Online]. Available: http://www.youtube.com/watch?v=lpCkNPqvC5s&feature=youtube_gdata_player. [Accessed: 16-Jan-2014].

[6] "JSP: Java Server Pages Tutorial." [Online]. Available: http://www.programacionfacil.com/java_jsp/start. [Accessed: 16-Jan-2014].

[7] "JSP: Programando con JSPs." [Online]. Available: http://geneura.ugr.es/~jmerelo/JSP/. [Accessed: 16-Jan-2014].

[8] "JSP: Tutorial." [Online]. Available: http://www.jsptut.com/. [Accessed: 16-Jan-2014].

[9] "REST: Representational State Transfer," Wikipedia, la enciclopedia libre. [Online]. Available: http://es.wikipedia.org/w/index.php?title=Representational_State_Transfer&oldid=71767627. [Accessed: 30-Jan-2014].

[10] “SAX: How to read XML file in Java.” [Online]. Available: http://www.mkyong.com/java/how-to-read-xml-file-in-java-sax-parser/. [Accessed: 16-Jan-2014].

[11] “SAX: Parseando XML en Java mediante SAX,” Le Funes. [Online]. Available: http://lefunes.wordpress.com/2008/03/01/parseando-xml-en-java-mediante-sax/. [Accessed: 16-Jan-2014].

[12] F. Méndez, “Solr: Querying using a pure AJAX application,” Developer Blog, 20-Jun-2011. [Online]. Available: http://gbif.blogspot.com.es/2011/06/querying-solr-using-pure-ajax.html. [Accessed: 16-Jan-2014].

[13] “Solr: The Base Solr Install.” [Online]. Available: http://synapticloop.com/tomes/solr/solr-tutorial/the-base-solr-install/. [Accessed: 16-Jan-2014].

[14] “Solr: una introducción.” [Online]. Available: http://www.dosideas.com/noticias/java/913-apache-solr-una-introduccion.html. [Accessed: 30-Jan-2014].

[15] “Solr: wikipedia,” Wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Apache_Solr&oldid=592837194. [Accessed: 30-Jan-2014].

[16] “Solr: with JBoss.” [Online]. Available: http://wiki.apache.org/solr/SolrJBoss. [Accessed: 16-Jan-2014].

[17] “SolrJ/Solr: cross-version compatibility.” [Online]. Available: http://wiki.apache.org/solr/Solrj. [Accessed: 16-Jan-2014].

[18] “SolrJ: example.” [Online]. Available: http://wiki.constellio.com/index.php/Solrj_example. [Accessed: 16-Jan-2014].

[19] “Struts: Aplicación paso a paso con Struts - Monografias.com.” [Online]. Available: http://www.monografias.com/trabajos28/aplicacion-paso-paso-struts/aplicacion-paso-paso-struts.shtml. [Accessed: 16-Jan-2014].

[20] “Struts: Desarrollo de aplicaciones Web.” [Online]. Available: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=struts_appdemo. [Accessed: 16-Jan-2014].

[21] “Struts: Hello World Example in Eclipse.” [Online]. Available: http://www.dzone.com/tutorials/java/struts/struts-tutorial/struts-tutorial-using-eclipse-1.html. [Accessed: 16-Jan-2014].

[22] “Struts: Manual Básico de Struts.” [Online]. Available: http://www.programacion.com/articulo/manual_basico_de_struts_156. [Accessed: 30-Jan-2014].

[23] “Struts: simple example | Manikandan’s Weblog.” [Online]. Available: http://manikandanmv.wordpress.com/tag/struts-simple-example/. [Accessed: 16-Jan-2014].

[24] “Struts: Tutorial Creating Struts application in Eclipse. Struts Tutorial with Eclipse” [Online]. Available: http://viralpatel.net/blogs/tutorial-creating-struts-application-in-eclipse/. [Accessed: 16-Jan-2014].

[25] “Struts: Validaciones” [Online]. Available: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=validacionStruts. [Accessed: 16-Jan-2014].

[26] “Struts: wikipedia” Wikipedia, la enciclopedia libre. [Online]. Available: http://es.wikipedia.org/w/index.php?title=Apache_Struts&oldid=72039182. [Accessed: 30-Jan-2014].

[27] “Struts: wikipedia Apache Struts,” Wikipedia, the free encyclopedia. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Apache_Struts&oldid=592006834. [Accessed: 30-Jan-2014].

ANEXO I. Contenido del CD.

Contenido del CD:

1. Memoria en formato PDF
2. Código del sistema
3. War para JBoss 7.1.1
4. Manual de uso
5. Manual de instalación
6. Otros ficheros adjuntos (Ficheros de Log usados en Pruebas, servidor JBoss)

ANEXO II. Definiciones, acrónimos y abreviaturas

En este anexo se van a realizar aclaraciones sobre algunas de las palabras presentes en esta memoria que son relevantes para la comprensión de este documento:

- API (Application Programming Interface):

  Es la interfaz de una librería que contiene un conjunto de funciones y métodos para poder ser usar la librería desde otro software.

- HPSA (HP Service Activator):

  Es una aplicación que se utiliza en compañías telefónicas para mantener el inventario de sus equipos de red, para activar y desactivar líneas y servicios como el buzón de voz, la llamada en espera y otros servicios. Esta aplicación es la que genera los ficheros Log que lee la nueva aplicación.

- HPSA EP (HPSA Extension Pack):

  Es un conjunto de paquetes para ampliar la funcionalidad del HPSA. En esta extensión es donde se añadirá la nueva aplicación creada en este proyecto añadiendo una nueva funcionalidad.

- Log:

  Es un fichero que contiene un registro oficial de los eventos dados en una aplicación en un rango de tiempo determinado.

- MVC(Modelo-Vista-Controlador):

  Patrón arquitectónico en el que se define una división de la aplicación en tres secciones con distintas responsabilidades. En el Modelo se implementa la lógica de negocio. En la Vista se describen las interfaces que se mostrarán a los usuarios. Y el controlador gestiona el intercambio de información entre la vista y el modelo.

- REST (Representational State Transfer):

  Es una técnica de arquitectura software para sistemas distribuidos. Este término se usa en el para describir cualquier interfaz web simple que utiliza XML y HTTP, sin necesidad de abstracciones para el intercambio de mensajes. El conjunto de principios de la arquitectura son: Un protocolo cliente/servidor sin estado, un conjunto de operaciones bien definidas que se aplican a todos los recursos de información (POST, GET, PUT y DELETE), Una sintaxis universal para identificar los recursos y el uso de hipermedios, para la información de la aplicación y para las transiciones de estado de la aplicación.

- SAX (Simple API for XML):

  Sistema usado para procesar ficheros XML, se detallan sus características en el punto 4.1.3.

- Solr:

  Sistema usado para la indexación de la información presentes en los ficheros de Log procesados, se detallan sus características en el punto 4.1.2.

- Struts:

  Framework usado para mostrar las interfaces de la aplicación. Se detallan sus características en el punto 4.1.1.

- XML (eXtended Markup Language):

  Es un lenguaje de marcas. Mediante este lenguaje se puede estructurar la información de forma lógica a través de etiquetas. Un fichero XML es un documento de texto plano con las marcas del lenguaje XML.

ANEXO III. Ficheros Log diseñados para pruebas

Para realizar las pruebas anteriormente descritas se usó un fichero de Log con datos reales y otros creados artificialmente. Los ficheros creados específicamente para probar la aplicación son 6. Se usó este número de ficheros con el objetivo de probar el procesamiento de varios ficheros y para probar la búsqueda en función de todos los campos del formulario. A continuación se muestran los ficheros Log que han sido diseñados específicamente para probar la aplicación:

- **Fichero "mwfm_1.log.xml":**

```
<LogEntries>
<LogEntry level="DEBUG1" >
  <HostName>Name1</HostName>
  <Time>10-ene-2012 12:02:57</Time>
  <Module>mod1</Module>
  <Part>part1</Part>
  <Component>comp1</Component>
  <Topic>topic1</Topic>
  <Thread>Thread1</Thread>
 <Message>Esta es la entrada del log 1, del Fichero de prueba 1</Message>
</LogEntry>
<LogEntry level="DEBUG2" >
  <HostName>Name2</HostName>
  <Time>10-feb-2012 12:29:57</Time>
  <Module>mod2</Module>
  <Part>part2</Part>
  <Component>comp2</Component>
  <Topic>topic2</Topic>
  <Thread>Thread2</Thread>
 <Message>Esta es la entrada del log 2, del Fichero de prueba 1</Message>
</LogEntry>
<LogEntry level="DEBUG3" >
  <HostName>Name3</HostName>
  <Time>10-mar-2012 12:29:57</Time>
  <Module>mod3</Module>
  <Part>part3</Part>
  <Component>comp3</Component>
  <Topic>topic3</Topic>
  <Thread>Thread3</Thread>
 <Message>Esta es la entrada del log 3, del Fichero de prueba 1</Message>
</LogEntry>
<LogEntry level="DEBUG4" >
  <HostName>Name4</HostName>
  <Time>10-abr-2012 12:29:57</Time>
```

```
  <Module>mod4</Module>
  <Part>part4</Part>
  <Component>comp4</Component>
  <Topic>topic4</Topic>
  <Thread>Thread4</Thread>
 <Message>Esta es la entrada del log 4, del Fichero de prueba 1</Message>
</LogEntry>
<LogEntry level="DEBUG5" >
  <HostName>Name5</HostName>
  <Time>10-may-2012 12:29:57</Time>
  <Module>mod5</Module>
  <Part>part5</Part>
  <Component>comp5</Component>
  <Topic>topic5</Topic>
  <Thread>Thread5</Thread>
 <Message>Esta es la entrada del log 5, del Fichero de prueba 1</Message>
</LogEntry>
<LogEntry level="DEBUG6" >
  <HostName>Name6</HostName>
  <Time>10-jun-2012 12:29:57</Time>
  <Module>mod6</Module>
  <Part>part6</Part>
  <Component>comp6</Component>
  <Topic>topic6</Topic>
  <Thread>Thread6</Thread>
 <Message>Esta es la entrada del log 6, del Fichero de prueba 1</Message>
</LogEntry>
</LogEntries>
```

- **Fichero “mwfm_2.log.xml”:**

```
<LogEntries>
<LogEntry level="DEBUG7" >
  <HostName>Name7</HostName>
  <Time>10-jul-2012 12:29:57</Time>
  <Module>mod7</Module>
  <Part>part7</Part>
  <Component>comp7</Component>
  <Topic>topic7</Topic>
  <Thread>Thread7</Thread>
 <Message>Esta es la entrada del log 1, del Fichero de prueba 2</Message>
</LogEntry>
</LogEntries>
```

- **Fichero “mwfm_3.log.xml”:**

```
<LogEntries>
<LogEntry level="DEBUG8" >
  <HostName>Name8</HostName>
  <Time>10-ago-2012 12:29:57</Time>
  <Module>mod8</Module>
  <Part>part8</Part>
  <Component>comp8</Component>
  <Topic>topic8</Topic>
  <Thread>Thread8</Thread>
 <Message>Esta es la entrada del log 1, del Fichero de prueba 3</Message>
</LogEntry>
</LogEntries>
```

- **Fichero “mwfm_4.log.xml”:**

```
<LogEntries>
<LogEntry level="DEBUG9" >
  <HostName>Name9</HostName>
  <Time>10-sep-2012 12:29:57</Time>
  <Module>mod9</Module>
  <Part>part9</Part>
  <Component>comp9</Component>
  <Topic>topic9</Topic>
  <Thread>Thread9</Thread>
 <Message>Esta es la entrada del log 1, del Fichero de prueba 4</Message>
</LogEntry>
</LogEntries>
```

- **Fichero “mwfm_5.log.xml”:**

```
<LogEntries>
<LogEntry level="DEBUG10" >
  <HostName>Name10</HostName>
  <Time>10-oct-2012 12:29:57</Time>
  <Module>mod10</Module>
  <Part>part10</Part>
  <Component>comp10</Component>
  <Topic>topic10</Topic>
  <Thread>Thread10</Thread>
 <Message>Esta es la entrada del log 1, del Fichero de prueba 5</Message>
</LogEntry>
</LogEntries>
```

- **Fichero “mwfm_6.log.xml”:**

```
<LogEntries>
<LogEntry level="DEBUG11" >
  <HostName>Name11</HostName>
  <Time>10-nov-2012 12:29:57</Time>
  <Module>mod11</Module>
  <Part>part11</Part>
  <Component>comp11</Component>
  <Topic>topic11</Topic>
  <Thread>Thread11</Thread>
 <Message>Esta es la entrada del log 1, del Fichero de prueba 6</Message>
</LogEntry>
<LogEntry level="DEBUG12" >
  <HostName>Name12</HostName>
  <Time>10-dic-2012 12:29:57</Time>
  <Module>mod12</Module>
  <Part>part12</Part>
  <Component>comp12</Component>
  <Topic>topic12</Topic>
  <Thread>Thread12</Thread>
 <Message>Esta es la entrada del log 2, del Fichero de prueba 6</Message>
</LogEntry>
<LogEntry level="DEBUG13" >
  <HostName>Name13</HostName>
  <Time>10-ene-2013 12:29:57</Time>
  <Module>mod13</Module>
  <Part>part13</Part>
  <Component>comp13</Component>
  <Topic>topic13</Topic>
  <Thread>Thread13</Thread>
 <Message>Esta es la entrada del log 3, del Fichero de prueba 6</Message>
</LogEntry>
<LogEntry level="DEBUG14" >
  <HostName>Name14</HostName>
  <Time>10-feb-2013 12:29:57</Time>
  <Module>mod14</Module>
  <Part>part14</Part>
  <Component>comp14</Component>
  <Topic>topic14</Topic>
  <Thread>Thread14</Thread>
 <Message>Esta es la entrada del log 4, del Fichero de prueba 6</Message>
</LogEntry>
<LogEntry level="DEBUG15" >
  <HostName>Name15</HostName>
  <Time>10-mar-2013 12:29:57</Time>
  <Module>mod15</Module>
  <Part>part15</Part>
  <Component>comp15</Component>
  <Topic>topic15</Topic>
  <Thread>Thread15</Thread>
 <Message>Esta es la entrada del log 5, del Fichero de prueba 6</Message>
```

</LogEntry>
</LogEntries>

ANEXO IV. Manual de instalación.

En este anexo se muestra el contenido del manual de instalación existente en el CD anexo a esta memoria. En este anexo se ha omitido el índice y la bibliografía para simplificar su visualización.

CAPITULO 1. Introducción

1.1 Propósito

El propósito de este documento es tener un control de los programas necesarios para realizar un despliegue de la aplicación. También se detallarán las instalaciones que se deben realizar en los equipos para instalar la aplicación.

1.2 Ámbito

Se documenta el proceso para la instalación de las herramientas que han sido necesarias para desplegar la aplicación construida.

1.3 Alcance

Este documento solo pretende ser un manual de como desplegar la aplicación. No se pretende ser una guía para la instalación o configuración de los software necesarios para el despliegue.

1.4 Definiciones, Acrónimos y Abreviaturas

<Log>
Es un fichero que contiene un registro oficial de eventos durante un rango de tiempo en particular.

<Java EE 6> (Java Enterprise Edition 6 SDK)
Se designa a la versión Enterprise Edition de Java. Es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java.

CAPITULO 2. Software necesario

2.1 Lista de software

A continuación se tiene una lista con todos los programas necesarios para realizar un despliegue de la aplicación. En la siguiente lista se especifica el software necesario:

- Java EE 6: Es la plataforma Java.
- JBoss 7.1.1: Es el servidor Web en el que se desplegará la aplicación.
- Solr: Sistema de indexado usado en la aplicación. Debe estar instalado sobre el servidor Web.

2.2 Instrucciones de instalación en Windows

Para conseguir una correcta instalación de todos los componentes necesarios para la ejecución de la aplicación se deben seguir los siguientes pasos. A continuación se detalla como instalar las distintas herramientas necesarias.

2.2.1 Instalación de Java EE 6.

Si no se tiene instalado anteriormente el Java EE 6, lo primero a hacer es instalarlo. Para ello debes descargarte el instalador que corresponda con tu sistema operativo y ejecutarlo. La dirección web desde la que descargar esta versión de la plataforma esta introducida en la bibliografía.

2.2.2 Instalación de JBoss y Solr.

Estas dos herramientas pueden ser instaladas por separado, pero en el CD en que se encuentra este manual en la carpeta de ficheros adjuntos existe una versión en un fichero RAR. Para instalarlo simplemente se debe descomprimir el contenido en una carpeta del sistema, por ejemplo "C:\obs2012". En los puntos posteriores nos referiremos a esta capeta como carpeta raíz ("Raiz\")

2.2.3 Puesta en marcha de JBoss.

El modo de iniciarlo es ejecutar el fichero "standalone" presente en "Raiz\jboss-as-7.1.1.Final\bin\". Una vez iniciada la ejecución de este fichero se abrirá una ventana en la que se muestra la ejecución del servidor mostrando su Log en tiempo real.

2.2.4 Parar JBoss.

Si lo que se desea es parar JBoss es necesario posicionar el foco de ejecución sobre la ventana en que se muestra su Log en tiempo real y pulsar <Ctrl> + "C". De este modo el servidor a través de la ventana pregunta: ¿Desea terminar el trabajo por lotes (S/N)?. A esta pregunta se responde pulsando S y <Intro> y finalmente se cierra la ventana.

CAPITULO 3. Instalación de la aplicación

Para realizar un despliegue de la aplicación es necesario copiar el fichero War presente en el CD adjunto y pegarlo en "Raiz\jboss-as-7.1.1.Final\standalone\deployments\".

Una vez realizado el despliegue, antes de iniciar la aplicación, es necesario crear el fichero de configuración. Para ello se tiene que copiar el fichero de configuración "configuration.properties", presente en el War (el contenido del War se puede visualizar con un sistema de descompresión, como Winrar) en "Raiz\jboss-as-7.1.1.Final\bin\configuration\", si la carpeta "configuration" no existe se debe crear y después copiar el fichero.

Una vez creado el fichero de configuración se deben establecer los valores de la configuración. En función de la instalación de Solr realizada se debe especificar la dirección de acceso al servidor, el tamaño del "batch" y los hilos que pueden actuar de forma concurrente sobre Solr. Los otros tres valores del fichero de configuración son los valores que delimitarán los ficheros que van a ser procesados, como el host y la carpeta en que se encuentran y el patrón de inicio del nombre de fichero que indica el tipo de Log que va a ser procesado.

A continuación se muestra el contenido del fichero de configuración existente en el War:

```
solr.server=http://localhost:8080/solr
solr.batch.size=1000
solr.concurrent.loaders=5

server.0.name=LOCALHOST
server.0.directory=C:\\Log\\LogHPSA
server.0.filename.filter.pattern=mwfm
```

CAPITULO 4. Iniciar la aplicación

Para iniciar la ejecución de la aplicación es necesario iniciar un navegador Web y en el introducir la dirección web para el acceso a la aplicación. La dirección variará dependiendo de como se haya realizado la instalación del servidor Web, pudiendo ser distinto el puerto a través del que se accede. El puerto por defecto en el que se instala el servidor es el 8080, y este valor no cambia a no ser que cuando se inicie la ejecución del servidor exista otra aplicación usando este puerto.

El acceso a la aplicación se realizará a través de una dirección URL similar a la siguiente: "http://127.0.0.1:8080/LogProcessor/". Donde "127.0.0.1" indica el localhost, "8080" es el puerto donde se instaló el servidor web y /LogProcessor/ es el nombre de la aplicación a la que se desea acceder. Si la configuración del servidor fuera distinta sería necesario modificar la URL anterior.

ANEXO V. Manual de uso.

En este anexo se muestra el contenido del manual de uso existente en el CD anexo a esta memoria. En este anexo se han omitido el índice de contenidos y el índice de imágenes para simplificar la visualización del manual.

CAPITULO 1. Introducción

1.1 Propósito

El propósito de este documento es mostrar la forma de usar la aplicación. Explicando la funcionalidad de las interfaces y el modo de ejecutar los casos de uso de la aplicación.

1.2 Ámbito

Se documenta el proceso a seguir para el uso correcto de la aplicación.

1.3 Alcance

Este documento solo pretende ser un manual de como usar la aplicación. Si se desea acceder a un manual para la instalación o el acceso a la aplicación se debe consultar el "Documento de instalación".

CAPITULO 2. Ejecución de la aplicación.

En este apartado se va a exponer el orden normal de uso de la aplicación. Lo primero es necesario aclarar que esta aplicación solo tiene un caso de uso: "Cargar y Consultar".

Para iniciar "Cargar y Consultar" se debe introducir en un explorador web la URL de acceso a la aplicación, este paso está documentado en el "Documento de instalación". Una vez iniciado "Cargar y Consultar" se deben seguir los siguientes pasos:

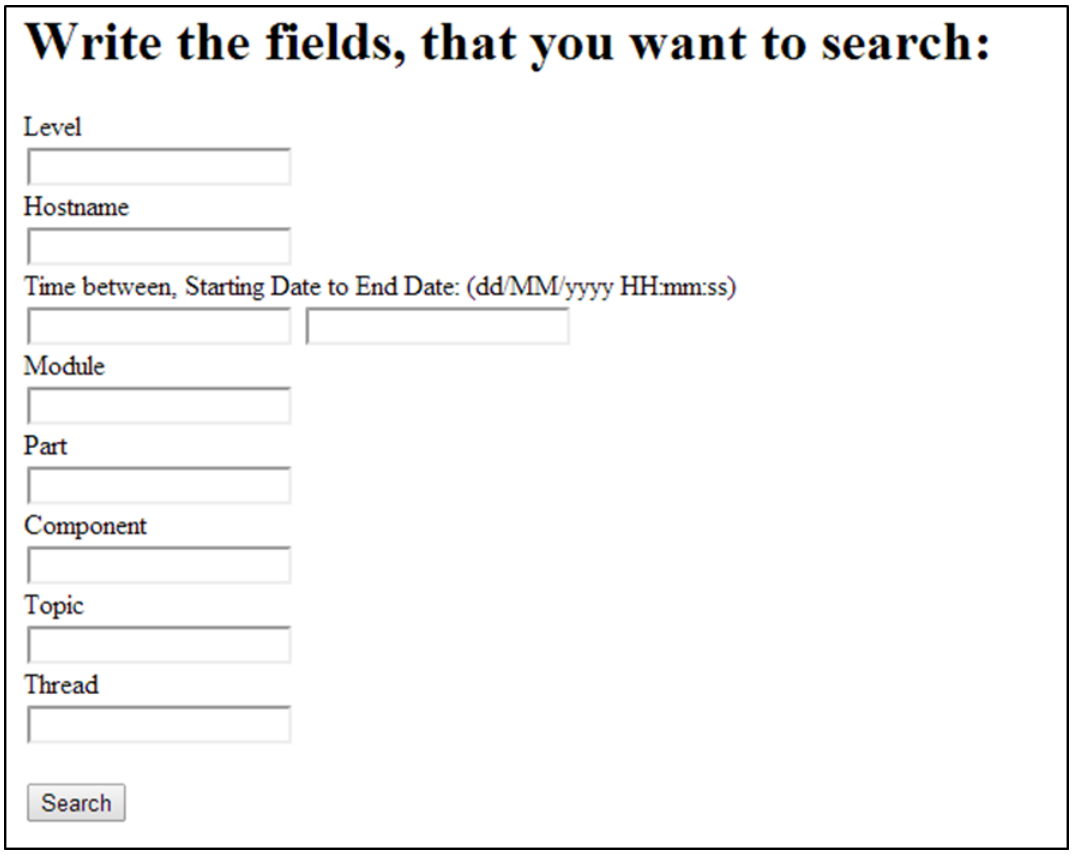
1. Cuando se accede a la aplicación se muestra el "Formulario de Búsqueda".
2. En este formulario se deben rellenar los campos que se deseen para realizar la búsqueda y se debe pulsar el botón "Search", en la parte inferior.
3. Se muestran las entradas de Log que coinciden con los parámetros introducidos en el anterior formulario.
4. Si se pulsa el botón "New Search" se vuelve al "Formulario de Búsqueda", volviendo al punto 2 de esta lista de acciones.

En el próximo punto se puede ver una descripción más exhaustiva de las interfaces de la aplicación.

CAPITULO 3. Interfaces y funcionalidades

3.1 Formulario de búsqueda

A continuación se muestra una imagen en la que se puede ver el formulario de búsqueda:



La búsqueda se realizará de forma que el contenido de los campos del formulario sea igual al de las entradas de Log, exceptuando las fechas que serán acordes al rango establecido en los campos de fechas.

El contenido que se debe introducir en los campos del formulario de búsqueda puede ser cualquier texto con todos los caracteres permitidos. La única particularidad, son los campos de las fechas que deben de cumplir con un formato establecido.

El formato para las fechas es “dd/MM/yyyy HH:mm:ss”, donde: “dd” es el día de 00 a 31, “MM” es el mes en formato de numero de 01 a 12, “yyyy” indica el año, “HH” es la hora, “mm” es el minuto y “ss” es el segundo. Con este formato se indica una fecha que puede ser la fecha inicial o la final en función del campo primero o segundo que se rellene.

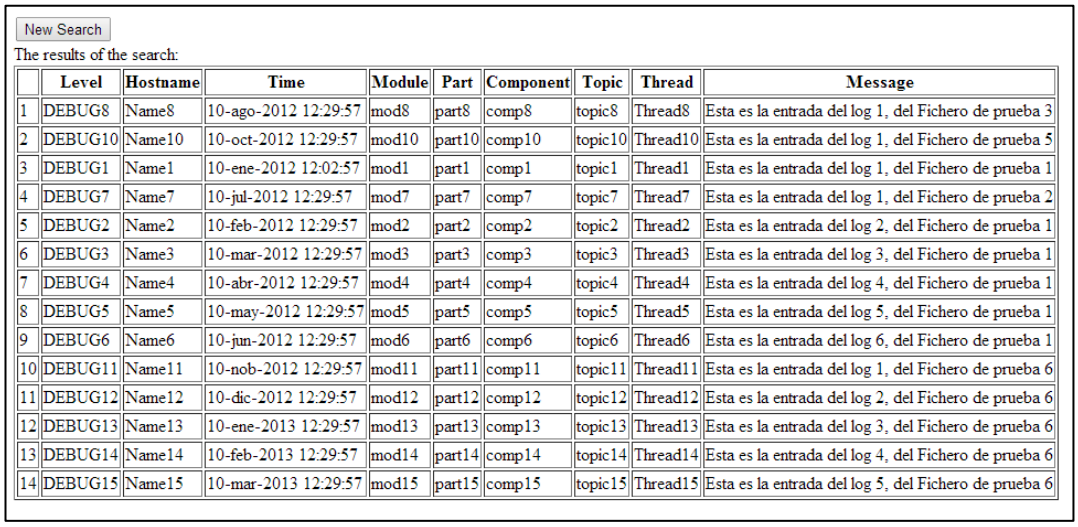
En el caso de querer obtener las entradas de Log correspondientes a un rango de fechas es necesario rellenar ambos campos, en el caso de que se rellene un solo campo se omitirá el dato introducido en el otro.

Ninguno de los campos es obligatorio, por lo que no es necesario rellenar ningún campo para realizar una búsqueda. En una búsqueda sin ningún campo rellenado se mostrarán todas las entradas de Log que hayan sido procesadas.

En caso de que alguno de los campos de fechas no cumplan con el formato establecido se mostrará el formulario de búsqueda con los mensajes de error en rojo. A continuación se muestra una imagen en la que se puede ver el formulario de búsqueda, una vez pulsado el botón "Search" y no cumpliéndose con las validaciones de los campos de fecha:

3.2 Formulario de resultados

A continuación se realiza una descripción de la interfaz en la que se muestran los resultados de la búsqueda:


New Search

The results of the search:

	Level	Hostname	Time	Module	Part	Component	Topic	Thread	Message
1	DEBUG8	Name8	10-ago-2012 12:29:57	mod8	part8	comp8	topic8	Thread8	Esta es la entrada del log 1, del Fichero de prueba 3
2	DEBUG10	Name10	10-oct-2012 12:29:57	mod10	part10	comp10	topic10	Thread10	Esta es la entrada del log 1, del Fichero de prueba 5
3	DEBUG1	Name1	10-ene-2012 12:02:57	mod1	part1	comp1	topic1	Thread1	Esta es la entrada del log 1, del Fichero de prueba 1
4	DEBUG7	Name7	10-jul-2012 12:29:57	mod7	part7	comp7	topic7	Thread7	Esta es la entrada del log 1, del Fichero de prueba 2
5	DEBUG2	Name2	10-feb-2012 12:29:57	mod2	part2	comp2	topic2	Thread2	Esta es la entrada del log 2, del Fichero de prueba 1
6	DEBUG3	Name3	10-mar-2012 12:29:57	mod3	part3	comp3	topic3	Thread3	Esta es la entrada del log 3, del Fichero de prueba 1
7	DEBUG4	Name4	10-abr-2012 12:29:57	mod4	part4	comp4	topic4	Thread4	Esta es la entrada del log 4, del Fichero de prueba 1
8	DEBUG5	Name5	10-may-2012 12:29:57	mod5	part5	comp5	topic5	Thread5	Esta es la entrada del log 5, del Fichero de prueba 1
9	DEBUG6	Name6	10-jun-2012 12:29:57	mod6	part6	comp6	topic6	Thread6	Esta es la entrada del log 6, del Fichero de prueba 1
10	DEBUG11	Name11	10-nob-2012 12:29:57	mod11	part11	comp11	topic11	Thread11	Esta es la entrada del log 1, del Fichero de prueba 6
11	DEBUG12	Name12	10-dic-2012 12:29:57	mod12	part12	comp12	topic12	Thread12	Esta es la entrada del log 2, del Fichero de prueba 6
12	DEBUG13	Name13	10-ene-2013 12:29:57	mod13	part13	comp13	topic13	Thread13	Esta es la entrada del log 3, del Fichero de prueba 6
13	DEBUG14	Name14	10-feb-2013 12:29:57	mod14	part14	comp14	topic14	Thread14	Esta es la entrada del log 4, del Fichero de prueba 6
14	DEBUG15	Name15	10-mar-2013 12:29:57	mod15	part15	comp15	topic15	Thread15	Esta es la entrada del log 5, del Fichero de prueba 6

En esta interfaz se pueden ver los resultados de la búsqueda mostrados en formato tabla. La única forma de interacción con la aplicación es a través del botón “New Search” que hace que se muestre nuevamente la interfaz de búsqueda.

Los resultados que se pueden observar son los datos presentes en los ficheros de prueba de la aplicación de forma que se muestran todos los datos de las entradas de Log que han sido procesados. Las entradas de Log se muestran en formato tabla de modo que cada columna corresponde con un atributo de las entradas de Log. Y cada fila corresponde con una entrada de Log.

ANEXO VI. Índice de Imágenes

ANEXO VII. Índice de Tablas

A continuación se muestra un índice de las tablas presentes en la actual memoria, no se han añadido las tablas de las pueblas realizadas por ser un conjunto amplio cuya información se encuentra indexada por el índice de contenidos: