MonitorMashup: Integrador de sistemas de monitorización

Antonio Sánchez Capellán - asanchezcap
Grado de Ingeniería Informática

Albert Grau Perisé

13 de Junio de 2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	**MonitorMashup: Integrador de sistemas de monitorización**
Nombre del autor:	**Antonio Sánchez Capellán**
Nombre del consultor:	**Albert Grau Perisé**
Fecha de entrega (mm/aaaa):	07/2017
Área del Trabajo Final:	**Java EE**
Titulación:	*Grado de Ingeniería Informática*

Resumen del Trabajo (máximo 250 palabras):

"Dado el crecimiento de la externalización de servicios informáticos, no es extraña la presencia de empresas que realizan la gestión de redes y sistemas de varios clientes. Esto obliga estas empresas a utilizar los sistemas de motorización preexistentes de todos ellos. Esto a su vez produce en los operadores incomodidad y falta de eficiencia al tener que alternar entre unos y otros. Por ejemplo, si se presentan varias incidencias en un determinado intervalo de tiempo resulta muy complejo realizar el triaje para decidir el orden de intervención en función de la gravedad del incidente y del tiempo transcurrido desde que sucedió éste. Todos estos factores dificultan el adecuando cumplimiento de los SLA contratados por los clientes. La creación de esta aplicación, que fusionará todos esos datos mostrándolos de forma unificada y coherente, pretende solucionar la problemática expuesta."

Abstract (in English, 250 words or less):

"Given the growth of outsourcing of IT services, it is not surprising the presence of companies that manage the networks and systems of several customers. This forces these companies to use the pre-existing motorization systems of all of them. This in turn produces in the operators discomfort and lack of efficiency having to alternate between them. For example, if there are several incidents in a given time interval, it is very complex to decide on the order of intervention depending on the severity of the incident and the time elapsed since the incident occurred. All these factors make it difficult to adequately comply with the SLAs contracted by customers. The creation of this application, which will merge all these data showing them in a unified and coherent way, tries to solve the problems exposed."

Palabras clave (entre 4 y 8):
java, nagios, monitorización, patrones de diseño, mashup, firma digital, tomcat, jmeter

Índice

Lista de figuras

1. Introducción

1.1. Contexto y justificación del Trabajo

Dado el crecimiento de la externalización de servicios informáticos, no es extraña la presencia de empresas que realizan la gestión de redes y sistemas de varios clientes. Esto obliga estas empresas a utilizar los sistemas de motorización preexistentes de todos ellos. Esto a su vez produce en los operadores incomodidad y falta de eficiencia al tener que alternar entre unos y otros. Por ejemplo, si se presentan varias incidencias en un determinado intervalo de tiempo resulta muy complejo realizar el *triaje* para decidir el orden de intervención en función de la gravedad del incidente y del tiempo transcurrido desde que sucedió éste. Todos estos factores dificultan el adecuando cumplimiento de los SLA contratados por los clientes. La creación de esta aplicación, que fusionará todos esos datos mostrándolos de forma unificada y coherente, pretende solucionar la problemática expuesta.

La aplicación, bautizada ***MonitorMashup***, se conectará a las interfaces de los diferentes sistemas de monitorización y fusionará y priorizará los datos de los incidentes. También podrá conectarse a otras instalaciones de ella misma actuando de esta forma como un sistema distribuido. Para hacer esto, los datos fusionados serán ofrecidos a través de varias API[i]: JSON[ii], XML, HTML formateado, HTML sin formato ni cabeceras... De esta forma los resultados se podrán mostrar directamente desde la aplicación o bien integrados en otras aplicaciones, actuando como un middleware.

1.2. Objetivos del Trabajo

A través de la realización del proyecto, se pretende conseguir un mejor conocimiento de las tecnologías del ecosistema Java: lenguaje, servidores de aplicaciones, frameworks, etc; así como un acercamiento al desarrollo de software en entornos profesionales.

La ejecución del proyecto se desarrollara siguiendo varias de las lineas indicadas para trabajos del Área JEE dentro del ámbito del Itinerario en Ingeniería del software:

- Desarrollo de servlets, uso de Tomcat (contenedor de Servlets)
- Estudio y uso de patrones de diseño, aplicándolos en la arquitectura Java EE
- Comunicación con bases de datos relacionales (por ejemplo MySQL).

Es importante indicar que hará y (sobre todo) qué no hará el software resultante de este proyecto.

Funcionalidades previstas: la aplicación soportará autentificación accesible tanto para usuarios como de tipo automatizado para maquinas. Obtendrá información de los sistemas de motorización de tipo Nagios[iii] y de otras instalaciones de ***MonitorMashup***, los fusionará, y los mostrara en diversos formatos. Ademas, podrá instalarse con o sin necesidad de una base de datos para almacenar la configuración, de forma que pueda instalarse en sistemas con pocos recursos.

Funcionalidades no previstas: la aplicación no mostrará una interfaz compleja ni dispondrá de una gestión avanzada de usuarios y permisos. No se contempla la obtención de información de ningún otro sistema de monitorización mas allá de los mencionados, pero el código deberá estar pensado de forma que extenderlo para que pueda soportarlos sea lo mas sencillo posible.

Cuando se documente la instalación y configuración, dicha documentación se limitará a aquellos aspectos directamente relacionados con el software, quedando por tanto excluido:

- Instalación y/o configuración del Sistema Operativo
- Instalación y/o configuración del Software Base
- Instalación y/o configuración de otros sistemas

1.3. Enfoque y método seguido

Una vez expuesto el problema y detallados los objetivos, se ha optado por el desarrollo de una aplicación a medida, fácilmente ampliable e integrable. Dicha aplicación será desarrollada siguiendo el esquema clásico de desarrollo en cascada[iv] al ser el que mejor se adapta al sistema de evaluaciones que sigue este Trabajo.

1.4. Planificación del Trabajo

1.4.1. Identificación y gestión de riesgos

Riesgos identificados

Código	Descripción	Impacto	Probabilidad
R01	La instalación de un sistema de motorización Nagios y de sistemas a monitorizar para disponer de incidencias de demostración puede escapar del horizonte temporal disponible.	Alto	Media

Plan de contingencia

Código	Acción	Tipo	Riesgo Residual
A1R01	Emplear la instalación de demostración que el equipo de desarrollo de Nagios tiene disponible online.	Corrector	Muy bajo

1.4.2. Descripción de tareas

Según el PMBOK[1] (utilizando como inspiración la versión 3), un proyecto se divide en cinco etapas o grupos de procesos: Iniciación, Planificación, Ejecución, Seguimiento y Control y Cierre. La etapa de Seguimiento y Control sera realizada por el equipo docente, por lo cual se articulará la planificación alrededor de las etapas restantes:

Iniciación

- **Definición inicial**: Se Realizará un descripción somera del producto resultante de la ejecución del proyecto.
- **Definición del alcance**: En esta subfase se decidirá que características deberán estar en la versión final y cuales no.
- **Definición inicial de requisitos**: Se definirán requisitos que no se hayan especificado en las dos fases anteriores, que serán principalmente requisitos no funcionales: entorno de ejecución, herramientas de desarrollo, frameworks y librerías a utilizar...

Planificación

- **Identificación y gestión de riesgos**: Se identificarán los riesgos que se puedan presentar e se indicará como se afrontarán en caso de presentarse.
- **Definición de tareas**: Se definirán las tares en las cuales se dividirá el proyecto.

- **Planificación temporal**: Asignación de tiempos a las tareas definidas y relaciones entre ellas. La planificación deberá tener en cuenta los riesgos detectados.
- **Entrega de la PEC1 (Evento)**: Entrega final de la fase de planificación.

Ejecución

- **Lanzamiento del proyecto (Evento)**: Inicio de la ejecución del proyecto.
- **Análisis**: Se analizará el problema que pretende resolver el proyecto y se definirá la solución a través de modelos y fichas de casos de uso.
- **Diseño**: Partiendo de la especificación de la fase anterior, se realizará el diseño de la solución, por medio de diseño diagramas de clase, de base de datos, de arquitectura...
- **Entrega de la PEC2 (Evento)**: Entrega parcial de la fase de ejecución.
- **Instalación del entorno de desarrollo**: Instalación el entorno de desarrollo, que incluye el entorno de ejecución y el IDE. El riesgo identificado como R01 afecta a esta fase. En caso de no cumplirse la programación temporal, deberá ejecutarse el plan de contingencia A1R01.
- **Implementación del escenario base**: Implementación del núcleo de la aplicación y el escenario mas básico:
  - Entrada: la entrada de la aplicación se simulará con un fichero estático (la salida en formato json de dos instalaciones de ***MonitorMashup***).
  - Salida: La aplicación deberá devolver en formato HTML sin formato las alertas fusionadas proporcionadas en la entrada.
- **Implementación de los escenarios restantes**: En esta fase de implementará la capacidad de actuar frente al resto de posibles entradas y de mostrar todas las posibles salidas.
- **Pruebas**: Realización de pruebas para las posibles entradas y salidas.
- **Elaboración del manual de instalación**: Suponiendo la existencia de un producto entregable definitivo (no estará disponible hasta la fase de cierre), se realizara la documentación necesaria para la instalación de ***MonitorMashup*** en un entorno de producción.
- **Entrega de la PEC3 (Evento)**: Entrega final de la fase de ejecución.

Cierre

- **Elaboración de la presentación**: En esta fase se realizará una presentación que muestre la evolución general del proyecto y el producto resultante.
- **Generación del software ejecutable**: Se generará un ejecutable completo de ***MonitorMashup*** como demostración del éxito de la ejecución del proyecto.

- **Elaboración de la memoria**: Se generará la documentación pormenorizada del proyecto y del producto resultante.
- **Entrega final y cierre del proyecto (Evento)**: Entrega final de la fase de cierre.

1.4.3. Diagrama de Gantt

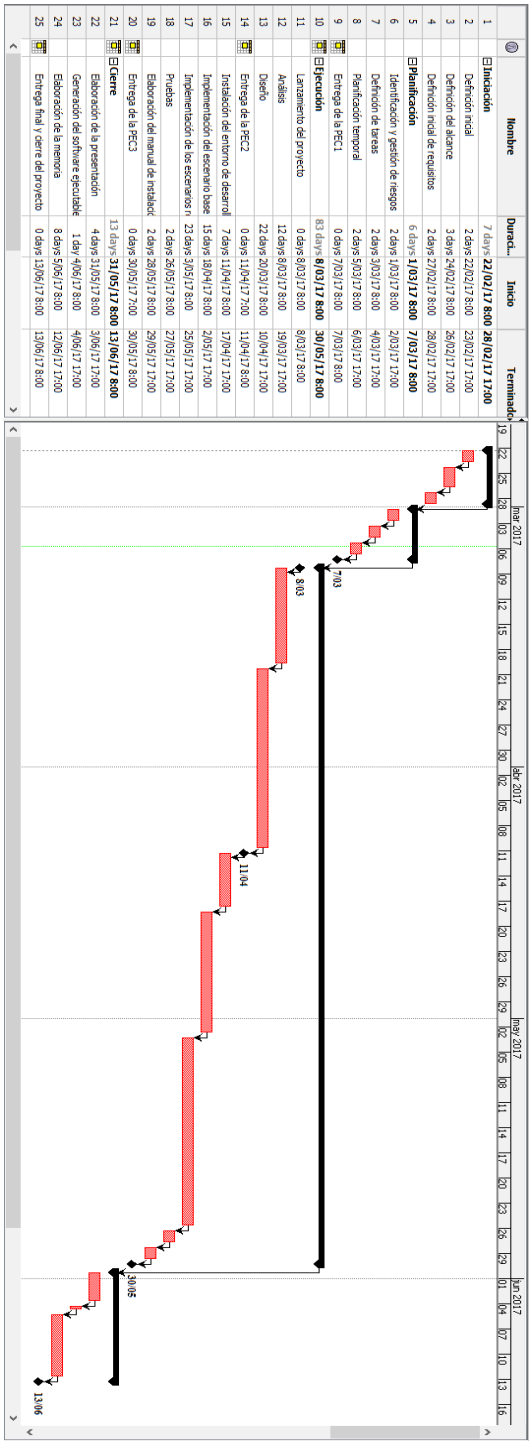


Ilustración 1: Diagrama de Gantt

1.5. Sumario de productos obtenidos

El software completo (código fuente y compilado) se distribuye en un fichero comprimido zip con el siguiente contenido:

- build/: Carpeta con los ficheros java .class[v] compilados.
- distr/: Carpeta que contendrá el fichero MonitorMashup.war[vi]. Este fichero en formato war contiene el software ejecutable.
- doc/: Documentación Javadoc[vii] del código fuente en formato HTML.
- jmeter/: Carpeta con los ficheros necesarios para realizar las pruebas de carga y de aceptación sobre la aplicación con la aplicación Apache JMeter[viii].
- manual/: Esta carpeta contiene el manual completo de instalación.
- src/: Código fuente de la aplicación.
- utils/: En esta carpeta puede encontrarse el fichero *monitormashup.sql* que contiene el script[ix] para la creación de la base de datos. También contiene la utilidad *generateKeys.bat* empleada para la creación de los certificados autofirmados[x] y almacenes (keystores[xi]) empleados en la firma de los informes en pdf.
- WebContent/: Carpeta con el resto de ficheros necesarios para la generación del war de la aplicación: html, jsp, properties, librerías...

Así mismo, junto al software arriba indicado, se incluyen entre los entregables del proyecto esta memoria, un video explicativo y una plantilla de autoevaluación.

1.6. Descripción de los otros capítulos de la memoria

Como se indicó en el apartado , la metodología empleada para la ejecución del proyecto sera el desarrollo en cascada. Siguiendo las fases principales de esta metodología, se irán desganando los principales apartados de esta memoria:

- *2 Tecnologías*: Se describirá el entorno de ejecución, las herramientas de desarrollo y los frameworks y librerías utilizadas.
- *3 Toma de requisitos*: En este capitulo se detallaran todos los requisitos de la aplicación, tanto los funcionales como los no funcionales.
- *4 Análisis*: Utilizando diagramas y fichas de casos de uso se especificará el comportamiento de la aplicación resultante.
- *5 Diseño*: A través de UML[xii] se definirá la arquitectura de clases que se empleará para realizar la aplicación, así como las interacciones entre las principales clases.

- *6 Implementación*: Con el fin de facilitar la interpretación del código fuente entregado, se añaden en este apartado explicaciones a cada fichero principal de la solución.
- *7 Instalación y configuración*: Se ha optado por integrar la instalación como capitulo y no como un anexo. El motivo es que la descripción de la instalación ofrece una visión mas amplia del producto desarrollado, no limitándose a una simple lista numerada de instrucciones.
- *8 Pruebas*: Detalle de las pruebas de aceptación[xiii] realizadas para verificar el cumplimiento de los requisitos.
- *9 Conclusiones*: Donde we realiza un estudio de los resultados obtenidos.

Por último, los dos apartados finales consistirán en la Bibliografía empleada y el glosario de términos. Con esto se completa esta introducción. En el siguiente apartado pueden verse las tecnologías utilizadas.

2. Tecnologías

2.1. Entorno de ejecución

Una vez completada la ejecución del proyecto, la aplicación resultante podrá ser ejecutada en un entorno que cumpla las siguientes especificaciones:

- Apache Tomcat 8.0[xiv]: Se trata de un contenedor de servlets desarrollado por la Apache Software Foundation. Implementa las especificaciones Servlet 3.1 y de JavaServer Pages (JSP) 2.3[xv].
- Maquina Virtual Oracle Java 7[xvi]: Se trata de un requisito de Tomcat 8 y por tanto de la aplicación. Es el software capaz de interpretar y ejecutar instrucciones expresadas en el bytecode de Java, el cual es generado por el compilador del lenguaje Java[xvii].
- Oracle Mysql 5 (Opcional)[xviii]: Es el motor de base de datos relacional de Oracle que se utilizará para almacenar la configuración de la aplicación. Se trata de un software con licencia dual comercial/libre[xix]. Existe otro software derivado (fork[xx]) denominado MariaDB[xxi] que podría ser compatible, aunque verificar este extremo escapa al ámbito de este proyecto. Este requisito del entorno es opcional, ya que la configuración también podrá ser almacenada en ficheros de texto.

2.2. Herramientas de desarrollo

Ademas de los requisitos de funcionamiento indicados en el punto anterior, para el desarrollo de la aplicación serán necesarios los siguientes componentes:

- Eclipse Java EE IDE for Web Developers (Versión: Mars.2 Release 4.5.2)[xxii]: Se trata de un entorno integrado de desarrollo desarrollado por la Fundación Eclipse y distribuido como software libre[xxiii].
- ProjectLibre 1.7.0[xxiv]: Software de administración de proyectos licenciado como software libre. Se empleará para la gestión de las tareas.

2.3. Frameworks y librerías

Durante la ejecución del proyecto, se usarán las siguientes librerías y frameworks:

- JSTL: Se trata de la Standard Tag Library for JavaServer Pages, es decir la librería de tags standart para su uso en ficheros JSP.

- mysql-connector-java: Driver JDBC[xxv] oficial para MySQL en su versión 5.1.40.
- JSON-java[xxvi]: Librería para el tratamiento de JSON en Java, en su versión 20160810.
- itextpdf[xxvii]: Toolkit utilizado para la generación de ficheros PDF en Java. Se utilizará la versión 5.5.10.
- Bouncy Castle Crypto APIs[xxviii]: librerías Java para el tratamiento de certificados y que se usarán para la firma digital de ficheros PDF, utilizándose la versión 155.

Una vez vistas las tecnologías utilizadas se procede a mostrar la toma de requisitos realizada, donde se detallarán todos los de la aplicación, tanto funcionales como no funcionales.

3. Toma de requisitos

Los requisitos son características observables del sistema que satisfacen una necesidad o expresan una restricción que afecta al producto (o al proceso) que estamos desarrollando. Una vez completado su estudio, se procederá al enunciado de estos, siguiendo la clasificación propuesta por la plantilla Volere[2].

3.1. Requisitos funcionales

3.1.1. Requisitos de funcionalidad

1. Sistemas compatibles. La aplicación deberá ser capaz de conectarse con las APIS de los siguientes sistemas de motorización:
   a) Nagios, con versión 411 o con API json compatible.
   b) MonitorMashup, con version 1.0.
2. Formatos de salida. Una vez captada la información de los sistemas de monitorización y fusionados los datos en función del tiempo transcurrido, el producto mostrará dichos datos en los siguientes formatos:
   a) json
   b) XML
   c) HTML formateado
   d) HTML sin formato ni cabeceras
   e) PDF
   f) PDF firmado digitalmente
3. La aplicación deberá soportar autentificación.
4. Validación del XML. La aplicación deberá proporcionar tanto el DTD[xxix] como el XSD[xxx] correspondiente al XML generado como salida.
5. Funcionamiento en modo demo: aún sin configurar, la aplicación dará la opción de acceder a datos estáticos de muestra para todos los formatos de salida disponibles.
6. Funcionamiento como middleware: la aplicación deberá ser capaz de obtener datos de otras instalaciones de ***MonitorMashup***. Cuando obtenga datos de esta forma, deberá mostrar tanto la información correspondiente a dicho sistema, como del sistema de motorización del cual este último obtuvo los datos.

En el punto 2, formatos de salida, se contempla la generación de ficheros PDF y PDF firmado digitalmente[xxxi]. Se ha detectado la conveniencia

de esta funcionalidad (con posterioridad a la toma inicial de requisitos) para dar la posibilidad de generar un informe que permita demostrar el estado de todas las incidencias en un momento dado sin el menor genero de duda ni posibilidad de modificación fraudulenta posterior[3].

3.1.2. Requisitos de datos

1. Datos a almacenar. El sistema almacenara los siguientes datos, por ser necesarios para la conexión a los sistemas compatibles y permitir la interpretación de los datos de salida:
   a) Nombre del cliente
   b) URL del sistema compatible
   c) Usuario y contraseña de acceso al sistema compatible
   d) Identificador del tipo de sistema compatible
   e) URL al sistema de documentación del cliente
2. Datos a obtener. Cuando el sistema recolecte información de un sistema compatible, para cada evento registrado deberá obtener los siguientes datos:
   a) Host afectado
   b) Servicio del host afectado
   c) Estado del evento (Warning, Critical)
   d) Tiempo transcurrido desde el inicio del evento

En el punto 1, datos a almacenar, se ha identificado la necesidad de almacenar una URL que permita acceder al sistema que contenga la documentación relativa al cliente, con el fin de tenerla accesible a la hora de resolver la incidencia.

3.2. Requisitos no funcionales

3.2.1. Requisitos de presentación

1. Cuando la salida del sistema sea HTML formateado, el formado deberá ser proporcionado por medio de CSS[xxxii].
2. El formado mostrado por la salida de tipo HTML formateado y PDF (en sus diferentes variantes) deberá ser similar.

3.2.2. Requisitos de usabilidad y humanidad

1. La salida HTML (en sus diferentes variantes) deberá poder verse correctamente en cualquier tipo de terminal, evitándose en la medida de los posible el uso de Javascript[xxxiii].

3.2.3. Requisitos de cumplimiento

Utilizando el hardware recomendado (ver apartado *3.2.4 Requisitos operacionales y de entorno*), el sistema deberá cumplir:

1. El sistema deberá ser capaz de mostrar al menos 6 clientes[1] con una media de 10 eventos cada uno.
2. El sistema deberá ser capaz de mostrar al menos 200 eventos.
3. Para un sistema con C clientes y E eventos, el tiempo de respuesta medio deberá ser inferior a C+(E*0,01) segundos.

3.2.4. Requisitos operacionales y de entorno

1. El sistema deberá poder ejecutarse en el siguiente entorno: Ver apartado *2.1 Entorno de ejecución*.
2. Hardware recomendado
   a) CPU de 4 nucleos
   b) 16 Gigas de RAM
   c) 20 Gigas de disco
   d) Conexión de 2 mbps
3. Hardware mínimo
   a) CPU de 1 nucleo
   b) 2 Gigas de RAM
   c) 8 Gigas de disco
   d) Conexión de 1 mbps

3.2.5. Requisitos de mantenimiento y soporte

1. El sistema deberá estar diseñado de forma que sea fácilmente extensible[2] a la hora de añadir nuevos formatos de salida.
2. El sistema deberá estar diseñado de forma que sea fácilmente extensible a la hora de añadir interfaces hacia nuevos sistemas de monitorización a los que poder conectarse.

3.2.6. Requisitos de seguridad

1. El sistema empleado para la autentificación en la aplicación será Basic access authentication[xxxiv], detallado en el RFC 2617[xxxv], con

1 Para abreviar, a lo largo del documento se utilizará indistintamente los términos “cliente” y “sistema de monitorización”. Por ejemplo, indicar que “la aplicación se conectará a un cliente” expresará lo mismo que “la aplicación se conectará al sistema de monitorización de un cliente”. Por ejemplo, en el caso de esta nota, lo que se indica es que el sistema será capaz de mostrar la información de los sistemas de monitorización de 6 clientes distintos.

2 Se emplea el término extensible en el marco del Principio Abierto/Cerrado.

el fin de facilitar la integración de los datos generados por la aplicación en otras plataformas.

3.2.7. Requisitos culturales y políticos

1. Tanto la interfaz de administración como todos los formatos de salida deberán emplear el ingles como idioma.

3.2.8. Requisitos legales

No se han identificado requisitos de tipo legal.

Hasta aquí los requisitos identificados. En el siguiente apartado se describirá el análisis realizado. Por medio de diagramas y fichas de casos de uso se especificará el comportamiento de la aplicación.

4. Análisis

4.1. Modelo de casos de uso y actores

En el siguiente diagrama puede verse esquemáticamente el funcionamiento de la aplicación:

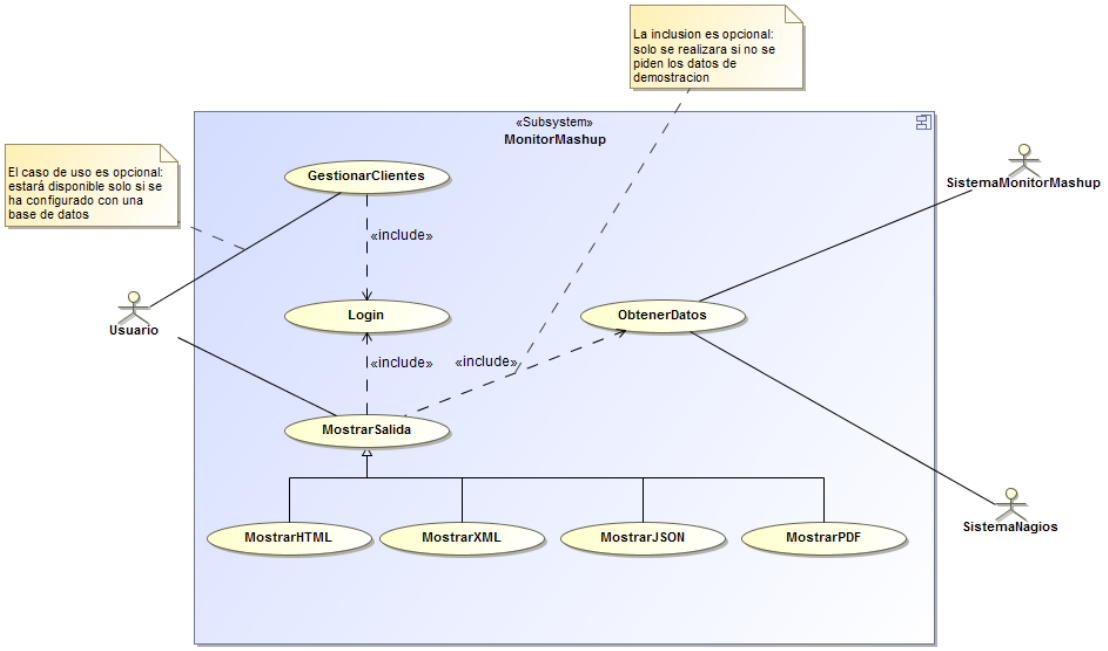


Ilustración 2: Modelo de casos de uso y actores

4.2. Fichas de casos de uso

Caso de uso	**Login**
Actor principal	Usuario
Ámbito	Sistema
Nivel de objetivo	Usuario
Stakeholders e intereses	
• Los usuarios desean poder identificarse en el sistema para poder acceder a la aplicación	
Garantías mínimas	Ninguna
Garantías en caso de éxito	El usuario accede a la aplicación
Precondición	Ninguna
Escenario principal de éxito	

1. El usuario introduce en su navegador la URL de la aplicación 2. El navegador solicita al usuario su nombre de usuario y contraseña 3. El usuario introduce su nombre de usuario y contraseña 4. El sistema comprueba que la contraseña se corresponde a ese nombre de usuario y que el usuario tiene permiso de acceso 5. El navegador muestra la pantalla de inicio de la aplicación
Escenarios alternativos
2a. El usuario pulsa cancelar. El navegador muestra una pagina en blanco (o el comportamiento que tenga definido por defecto) y el caso termina. 4a. El sistema no acepta la contraseña como válida o el usuario no tiene permiso de acceso. Se vuelve al punto 2 del escenario principal.

Caso de uso	**MostrarSalida**
Actor principal	Usuario
Ámbito	Sistema
Nivel de objetivo	Usuario
Stakeholders e intereses	
• Los usuarios quieren que la aplicación les muestre los eventos registrados en todos los sistemas de motorización configurados en tiempo real.	
Garantías mínimas	En caso de no poder obtener datos de uno o varios de los sistemas de motorización configurados, la aplicación mostrara el error por pantalla.
Garantías en caso de éxito	La pantalla de la aplicación muestra los eventos registrados.
Precondición	Usuario autentificado en el sistema (caso **Login**)
Escenario principal de éxito	
1. El usuario solicita acceder a los datos 2. El sistema obtiene los datos (caso **ObtenerDatos**) 3. El sistema proporciona los datos	
Escenarios alternativos	
1a. El usuario solicita acceder a los datos de demostración 1.1. El sistema redirecciona a los datos estáticos 1.2. El caso continua en al paso 3 del escenario principal	

Caso de uso	**MostrarHTML**
Actor principal	Usuario

Ámbito	Sistema
Nivel de objetivo	Usuario
Stakeholders e intereses	
• Los usuarios quieren obtener los datos en formato HTML.	
Garantías mínimas	Ninguna
Garantías en caso de éxito	Los datos son mostrados en formato HTML
Precondición	Ninguna
Escenario principal de éxito	
1. El usuario solicita acceder a los datos (caso **MostrarSalida**) en formato HTML 2. El navegador muestra los datos en formato HTML	
Escenarios alternativos	
1a. El usuario solicita que el HTML esté formateado 2a. El navegador muestra los datos en formato HTML con formato	

Caso de uso	**MostrarPDF**
Actor principal	Usuario
Ámbito	Sistema
Nivel de objetivo	Usuario
Stakeholders e intereses	
• Los usuarios quieren obtener los datos en formato PDF.	
Garantías mínimas	Ninguna
Garantías en caso de éxito	Los datos son descargados en formato PDF
Precondición	Ninguna
Escenario principal de éxito	
1. El usuario solicita acceder a los datos (caso **MostrarSalida**) en formato PDF 2. El navegador envía para su descarga los datos en formato PDF	
Escenarios alternativos	
1a. El usuario solicita que el PDF esté firmado digitalmente 2a. El navegador envía para su descarga los datos en un PDF firmado	

Caso de uso	**MostrarXML**
Actor principal	Usuario
Ámbito	Sistema

Nivel de objetivo	Usuario
Stakeholders e intereses	
• Los usuarios quieren obtener los datos en formato XML.	
Garantías mínimas	Ninguna
Garantías en caso de éxito	Los datos son mostrados en formato XML
Precondición	Ninguna
Escenario principal de éxito	
1. El usuario solicita acceder a los datos (caso **MostrarSalida**) en formato XML 2. El navegador muestra los datos en formato XML	
Escenarios alternativos	

Caso de uso	**MostrarJSON**
Actor principal	Usuario
Ámbito	Sistema
Nivel de objetivo	Usuario
Stakeholders e intereses	
• Los usuarios quieren obtener los datos en formato JSON.	
Garantías mínimas	Ninguna
Garantías en caso de éxito	Los datos son mostrados en formato JSON
Precondición	Ninguna
Escenario principal de éxito	
1. El usuario solicita acceder a los datos (caso **MostrarSalida**) en formato JSON 2. El navegador muestra los datos en formato JSON	
Escenarios alternativos	

Caso de uso	**ObtenerDatos**
Actor principal	SistemaNagios, SistemaMonitorMashup
Ámbito	Sistema
Nivel de objetivo	Usuario
Stakeholders e intereses	

•	
Garantías mínimas	Ninguna
Garantías en caso de éxito	Se suministran los datos de todos los sistemas de motorización configurados correctamente ordenados en función del tiempo transcurrido.
Precondición	Ninguna
Escenario principal de éxito	
1. El sistema se conecta con todos los sistemas de monitorización que tenga configurados en orden secuencial 2. El sistema obtiene los eventos que cada sistema de monitorización tenga registrados en ese momento 3. El sistema fusiona los datos, ordenándolos en función del tiempo transcurrido 4. El sistema devuelve los datos para su posterior procesado	
Escenarios alternativos	
1a. En caso de error, se notifica 2a. En caso de error, se notifica	

Caso de uso	**Gestionar clientes**
Actor principal	Usuario
Ámbito	Sistema
Nivel de objetivo	Usuario
Stakeholders e intereses	
• Los usuarios quieren tener la posibilidad de poder añadir y quitar sistemas de monitorización de los que obtener información en tiempo de ejecución.	
Garantías mínimas	Ninguna
Garantías en caso de éxito	Se añaden usuarios de forma correcta
Precondición	Usuario autentificado en el sistema (caso **Login**)
Escenario principal de éxito	
1. El usuario solicita acceder a la administración de clientes 2. El navegador muestra un listado con los clientes 3. El usuario solicita crear un nuevo clientes 4. El navegador muestra un formulario donde se le solicitan los datos del nuevo cliente 5. El usuario introduce los datos solicitados y acepta 6. El sistema añade el cliente y vuelve al paso 2	
Escenarios alternativos	

3a. El usuario solicita actualizar un cliente
    3.1. El navegador muestra un formulario donde se incluyen los datos actuales del cliente
    3.2. El usuario actualiza los datos y acepta
    3.3. El sistema actualiza el cliente y vuelve al paso 2 del escenario principal
3b. El usuario solicita eliminar un cliente
    3.1. El sistema elimina el cliente (sin pedir confirmación) y vuelve al paso 2 del escenario principal

Con esto se completa el análisis. En el siguiente apartado se detallará el diseño realizado, donde utilizando UML se definirá la arquitectura de clases así como las interacciones entre ellas.

5. Diseño

5.1. Modelo de pantallas

Para visualizar mejor el funcionamiento previsto de la aplicación, y partiendo ya de los datos definitivos de que se disponen, pueden simularse los pasos a seguir después de instalar la aplicación por medio de pantallas estáticas.

Inicialmente se darab de alta a aquellos clientes cuyas alertas se necesite gestionar. Se introducirá el nombre del cliente, la URL de su sistema de motorización, los datos de autentificación en dicho sistema, el tipo de sistema que se trate y la URL que corresponda con la documentación de dicho cliente. Veamos esta pantalla de alta:

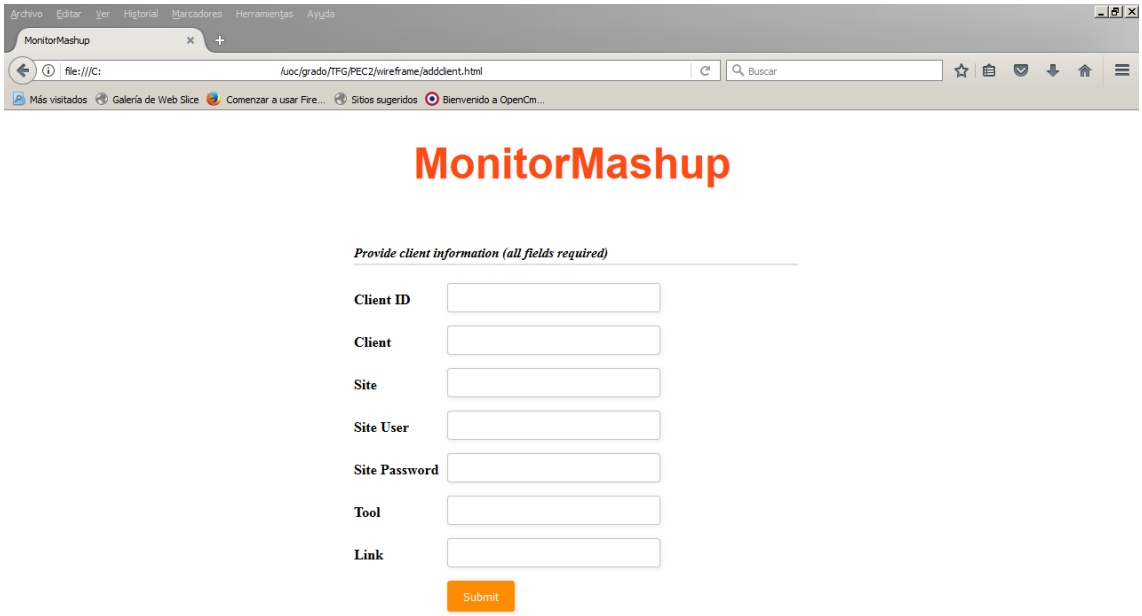
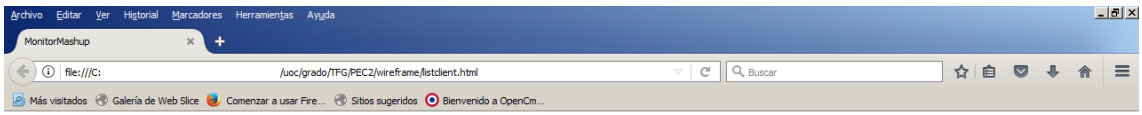


Ilustración 3: Formulario de alta de Cliente

Una vez se hayan dado de alta a todos los clientes, podrán verse en un listado a tal efecto:



MonitorMashup

Id	Client Name	Site	Site User	Site Password	Tool	Link	Action	
1	Client 1	nagios.client1.com	admin	**********	nagios	wiki.client1.com	Update	Delete
2	Client 2	monmashup.client2.com	admin	**********	monitormashup	wiki.client2.com	Update	Delete
3	Client 3	nagios.client3.com	admin	**********	nagios	wiki.client3.com	Update	Delete

Add Client

Ilustración 4: Listado de clientes

En este punto ***MonitorMashup*** ya será capaz de acceder a las incidencias detectadas por los diferentes sistemas de motorización y priorizarlos en función del tiempo de disparo de dichas incidencias:

MonitorMashup

Client	Host	Service	Status	Duration	Link
Client 1	mail.client1.com	SMTP service	Critical	35	wiki.client1.com
Client 3	www.client3.com	Web Page Content	Warning	21	wiki.client3.com
Client 2	secure.client2.com	Authentication Service	Critical	9	wiki.client2.com

Ilustración 5: Listado de tipo HTML con formato

5.2. Diagrama de clases

Para implementar la aplicación se ha realizado el diseño de clases que puede verse en los siguientes apartados. En general se ha prescindido de presentar aquellas clases que al ser instrumentales no añadan información (por ejemplo, las clases DAO[xxxvi] o los servlets).

5.2.1. Paquete edu.uoc.asanchezcap.monitormashup

La clase *GrandCentral* centralizará el funcionamiento de la aplicación. En el siguiente apartado podrá verse esto con mas claridad.

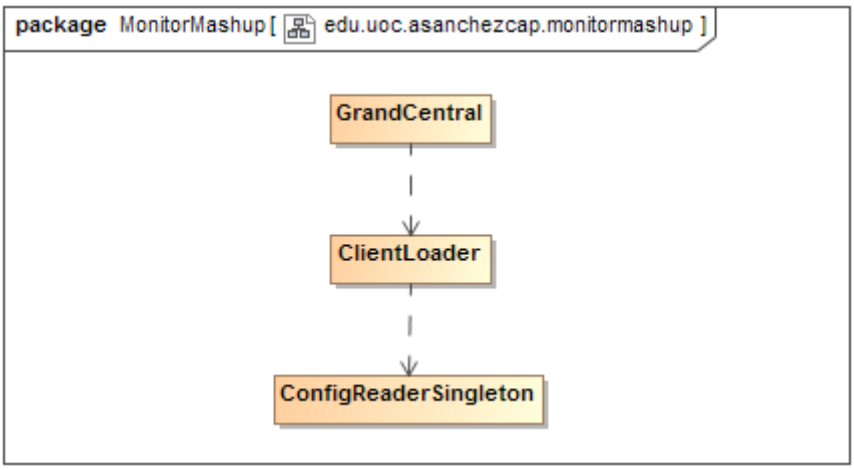


Ilustración 6: Diagrama de clases del paquete monitormashup

5.2.2. Paquete edu.uoc.asanchezcap.monitormashup.model

La clase *Client* almacenará la información de los clientes cuyos sistemas de motorización estén registrados. Será una clase persistente (si esta activo el modo con base de aatos). *MonitorStatus* guardará los datos dinámicos del sistema de motorización y *ServiceStatus* los datos dinámicos de cada servicio afectado por una incidencia.

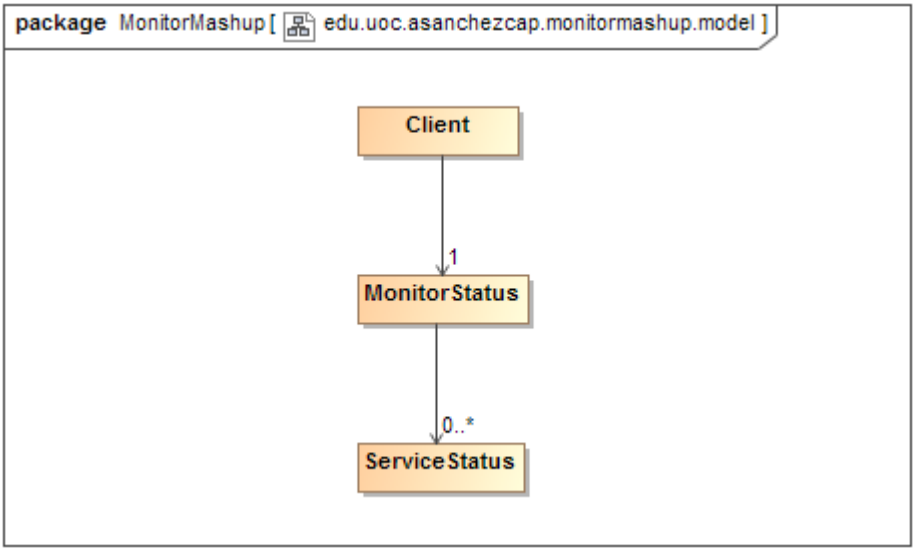


Ilustración 7: Diagrama de clases del paquete model

5.2.3. Paquete edu.uoc.asanchezcap.monitormashup.output

En este paquete estarán ubicadas las clases encargadas de formatear los datos recogidos para adecuarlas a los formatos de salida deseados. Se ha empleado el patrón de diseño Factory Method[xxxvii][4][5], con el fin de cumplir los requisitos (ver *3.2.5 Requisitos de mantenimiento y soporte)* que establecen

que el sistema deberá estar diseñado de forma que sea fácilmente extensible[xxxviii] a la hora de añadir nuevos formatos de salida.

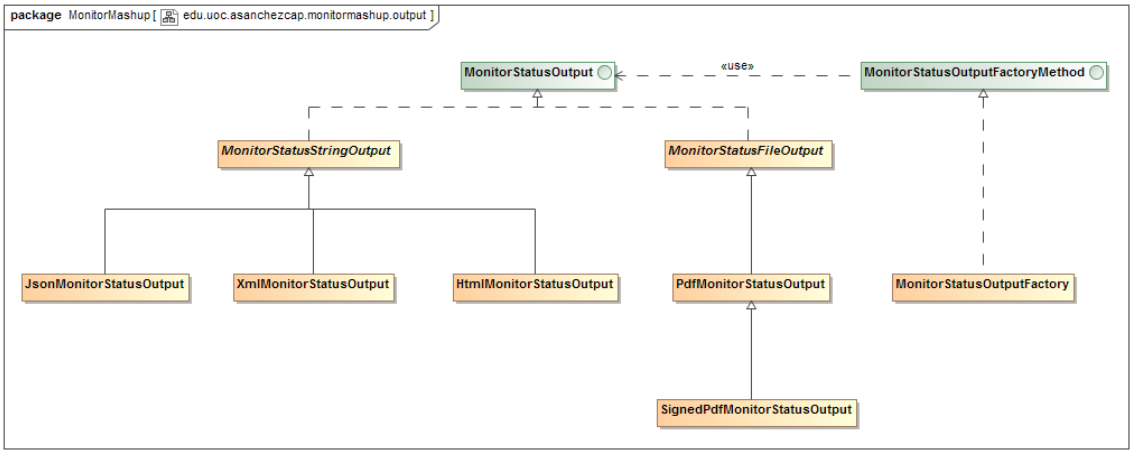


Ilustración 8: Diagrama de clases del paquete output

En el caso que en el futuro fuera necesario desarrollar nuevos formatos de entrada basados en pdf, se recomienda refactorizar dichas clases siguiendo el patrón Decorador[xxxix][GoF02].

5.2.4. Paquete edu.uoc.asanchezcap.monitormashup.input

En este paquete estarán ubicadas las clases encargadas conectarse a los distintos tipos de sistemas de monitorización remotos. También se ha empleado el patrón de diseño Factory Method citado en el epígrafe anterior para facilitar la extensión y añadir fácilmente nuevos sistemas compatibles.

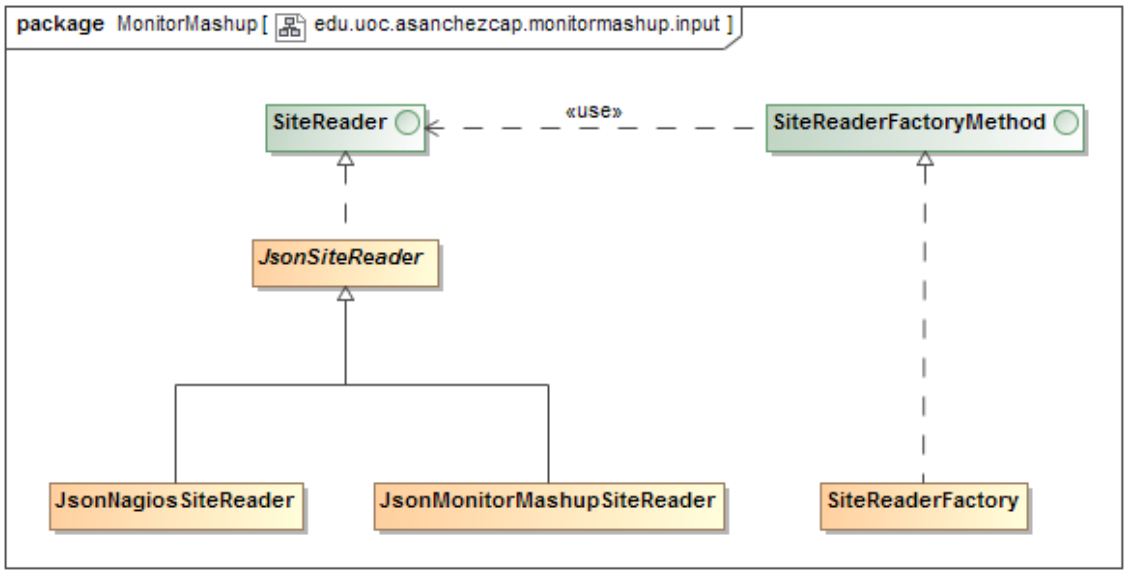


Ilustración 9: Diagrama de clases del paquete intput

5.3. Diagrama de secuencia

En el siguiente diagrama puede verse el funcionamiento normal de la aplicación ante una petición. La clase *GrandCentral* recibirá la solicitud de un *servlet* determinado y en función de los parámetros recibidos procesará la solicitud y devolverá los datos correctamente formateados al *servlet*, que a su vez devolverá los datos al usuario.

Como puede observarse, los clientes (o para ser mas exactos, los sistemas de monitorización de los clientes) son consultados de forma secuencial. Podría ocurrir que ante un numero elevado de clientes a consultar, cada uno de ellos con un numero elevado de incidencias, ***MonitorMashup*** no fuera capaz de procesar todos los datos y el servidor de aplicaciones mostrara un error de timeout[xl]. Esto podría solucionarse abordando el problema por medio de una solución multihilo[xli] (consultando los clientes en paralelo), a costa de un aumento en el tiempo de desarrollo, dada su mayor complejidad.

Sin embargo, se estima que la solución secuencial es capaz de soportar los requisitos (ver *3.2.3 Requisitos de cumplimiento)*, por lo cual se ha optado por esta aproximación al resultar menos costosa.

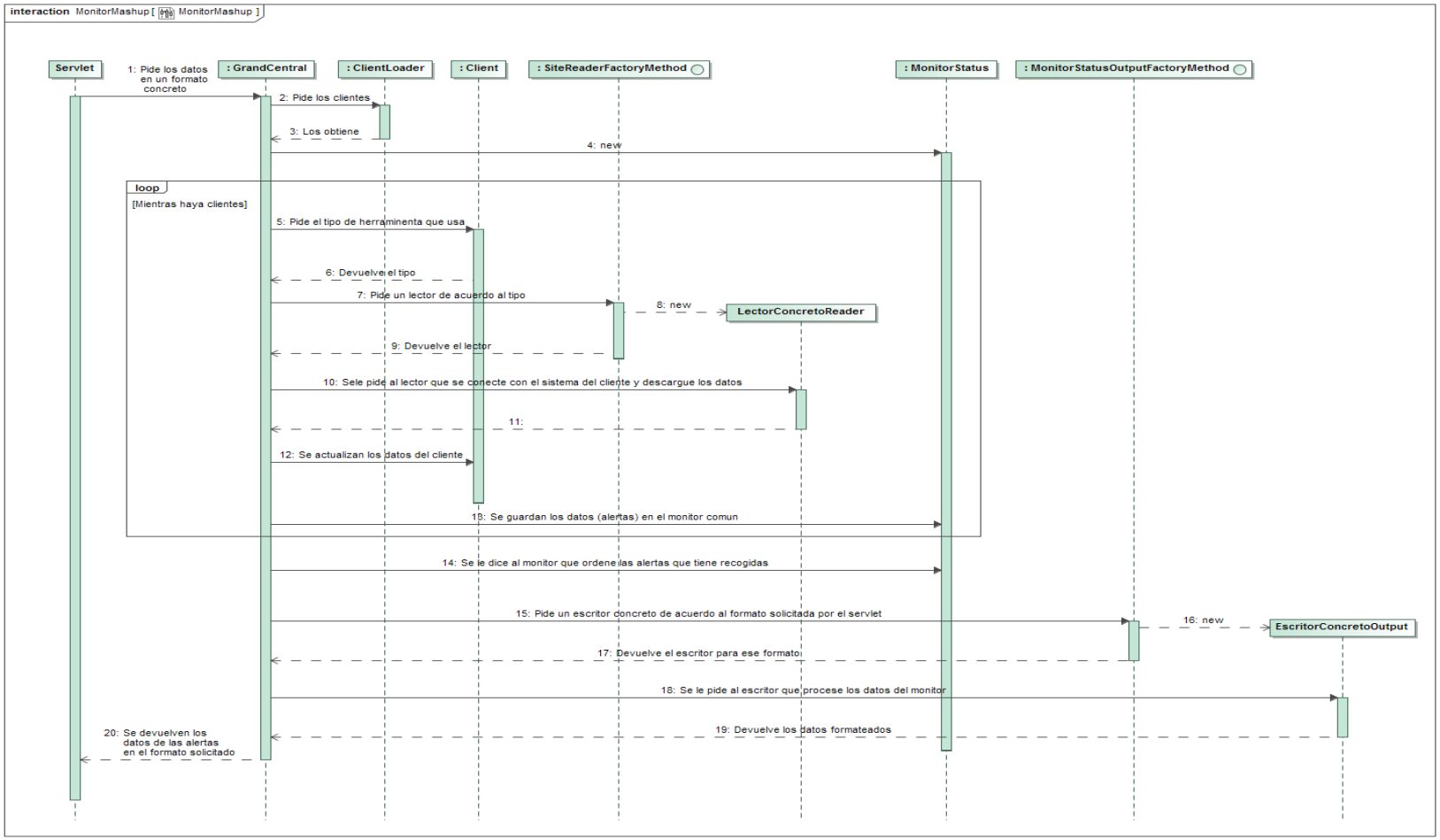


Ilustración 10: Diagrama de secuencia principal de la aplicación

5.4. Diseño relacional de la base de datos

El diagrama de base de datos será muy simple (una única tabla) ya que el resto de datos son dinámicos por naturaleza. Si se "cacheáran"[xlii] los datos de las alertas, se estarían tomando decisiones basadas en datos desactualizados por definición.

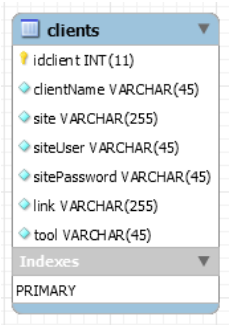


Ilustración 11: Diagrama de base de datos

5.5. Diagrama de arquitectura

5.5.1. Arquitectura Software

El software seguirá como patrón de arquitectura el Modelo-Vista-Controlador o MVC[xliii]:

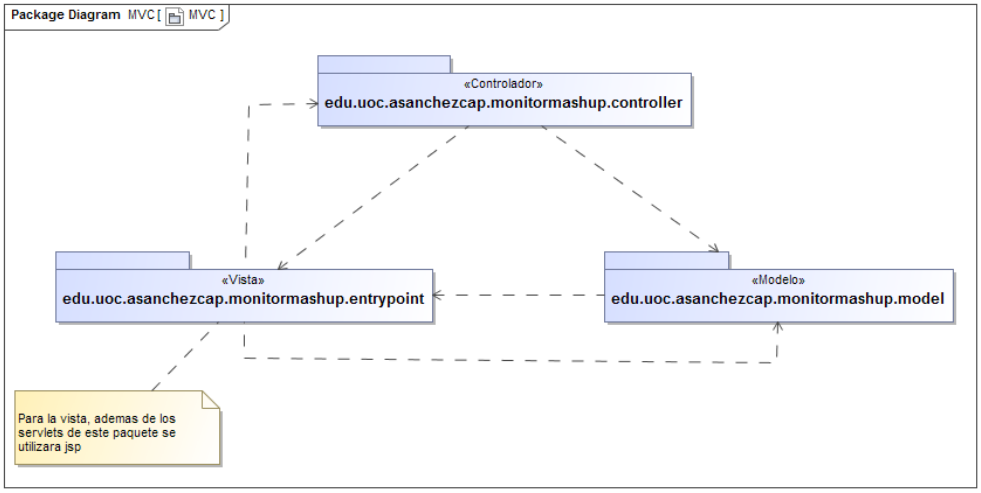


Ilustración 12: Arquitectura Software

5.5.2. Arquitectura Hardware

Una vez completada, la aplicación resultante podrá instalarse tanto en sistemas básicos (un único servidor con Tomcat con o sin base de datos) como en entornos complejos que garanticen alta disponibilidad. En el siguiente ejemplo puede verse una arquitectura de este tipo sirviendo los datos agregados directamente a Internet, indicando cada capa y los protocolos de comunicación entre ellas:

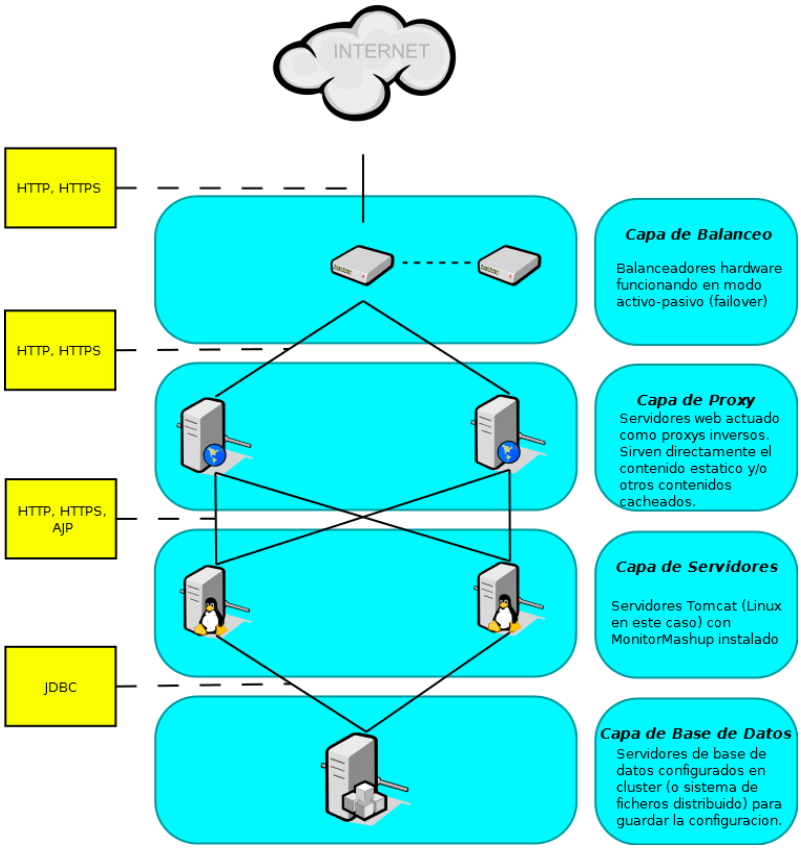


Ilustración 13: Arquitectura Hardware: ejemplo en alta disponibilidad

Al margen de las conexiones indicadas en el gráfico, que refleja la forma en que llegarán las peticiones al sistema, los servidores donde este instalado ***MonitorMashup*** deberán tener permisos en el firewall[xliv] para poder acceder a los sistemas de monitorización de los cuales tenga que obtener información.

Una vez visto el diseño realizado se procede a mostrar su implementación, donde se indicarán los archivos principales de la solución y una breve descripción de ellos.

6. Implementación

6.1. Implementación de clases

6.1.1. Paquete edu.uoc.asanchezcap.monitormashup

- **ClientLoader**: Carga, bien de base de datos, bien del fichero de propiedades, los datos de los sistemas de motorización de los clientes.
- **ConfigReaderSingleton**: Esta clase se encarga de leer el fichero de *properties* y guardar los datos para que estén disponibles en el resto de la aplicación. Utiliza el patrón de diseño Singleton[xlv][GoF02]. La conveniencia del uso de este patrón se detecto durante la fase de implementación.
- **DatabaseConnectorSingleton**: Establece la conexión con la base de datos utilizando los parámetros de configuración almacenados en el fichero de *properties*. Utiliza el patrón de diseño Singleton. La conveniencia del uso de este patrón se detecto durante la fase de implementación.
- **GrandCentral**: Este es el punto de entrada a la aplicación: los *servlets* solicitan los datos a esta clase y ella se encarga de buscar la configuración de los clientes, se conecta a ellos para traer los datos, formatea la salida adecuadamente y devuelve los datos al *servlet* para que los envíe al usuario.
- **MonitorMashupRuntimeException**: Excepción personalizada que se usará para controlar los problemas específicos de la funcionalidad (p.e. que los datos proporcionados en el fichero de *properties* no sean correctos, etc)

6.1.2. Paquete edu.uoc.asanchezcap.monitormashup.controller

- **ClientController**: Esta clase realiza la función de controlador (MVC) de los clientes.

6.1.3. Paquete edu.uoc.asanchezcap.monitormashup.dao

- **ClientDao**: Contiene la lógica de acceso a la base de datos referida a clientes.

6.1.4. Paquete edu.uoc.asanchezcap.monitormashup.entrypoint

- **HtmlEntryPoint**: Servlet encargado de mostrar los servicios recopilados en formato HTML (con o sin formato)
- **JsonEntryPoint**: Servlet encargado de mostrar los servicios recopilados en formato JSON
- **PdfEntryPoint**: Servlet encargado de mostrar los servicios recopilados en formato PDF (firmado o no)
- **XmlEntryPoint**: Servlet encargado de mostrar los servicios recopilados en formato XML

6.1.5. Paquete edu.uoc.asanchezcap.monitormashup.input

- **SiteReaderFactoryMethod**: En esta interfaz se utiliza el patrón método factoría. Define el método que se empleará para crear el tipo concreto de *reader*.
- **JsonMonitorMashupSiteReader**: Obtiene los servicios con *warnings* o *criticals* de otra instancia de MonitorMashup a traves de su api json.
- **JsonNagiosSiteReader**: Obtiene los servicios con *warnings* o *criticals* de un Nagios con versión 411 o con API json compatible.
- **JsonSiteReader**: Esta clase se obtiene después de refactorizar los métodos comunes en las *readers* que emplean json para conectarse a los clientes.
- **SiteReader**: En esta clase se define que condiciones tienen que tener los lectores de sitios (*readers*).
- **SiteReaderFactory**: Con esta clase se implementa el patrón método factoría.

6.1.6. Paquete edu.uoc.asanchezcap.monitormashup.model

- **Client**: Clase que representa los clientes registrados (almacenados en la base de datos o en el fichero de configuración).
- **MonitorStatus**: Esta clase es un mero almacén de servicios. Se utiliza para encapsular la implementación.
- **ServiceStatus**: En esta clase se almacenan los datos de un servicio marcado como warning o critical por alguno de los sistemas de motorización registrados. Almacenara el cliente, el host, el servicio, el estado, la duración en minutos y un enlace. Es ordenable en base a la duración. Se ha optado por duplicar el nombre del cliente y enlace a la documentación de la clase Cliente por si fuera necesario realizar operaciones con ellos (por ejemplo cuando accedemos a un cliente no directamente sino a través de otra instancia de MonitorMashup).

6.1.7. Paquete edu.uoc.asanchezcap.monitormashup.output

- **MonitorStatusOutput**: Interfaz que define el método que tendrán que implementar las clases que ofrezcan los diferentes tipos de salida de un monitor.
- **MonitorStatusOutputFactoryMethod**: En esta interfaz se utiliza el patrón método factoría. Define el método que se empleara para crear el tipo concreto de salida.
- **HtmlMonitorStatusOutput**: Clase concreta que partiendo de un monitor, devuelve un String que contiene un HTML con los datos de los servicios que contiene.
- **JsonMonitorStatusOutput**: Clase concreta que partiendo de un monitor, devuelve un String que contiene un json con los datos de los servicios que contiene.
- **MonitorStatusFileOutput**: Esta clase guarda todos los métodos comunes a aquellos *MonitorStatusOutput* que den como salida un File.
- **MonitorStatusOutputFactory**: Con esta clase se implementa el patrón método factoría. Se utiliza para encapsular aquí el tipo concreto de *MonitorStatusOutput* que es necesario crear para dar la salida, en función de su tipo.
- **MonitorStatusStringOutput**: Esta clase guarda todos los métodos comunes a aquellos *MonitorStatusOutput* que den como salida un String.
- **PdfMonitorStatusOutput**: Clase concreta que partiendo de un monitor, devuelve un File con formato pdf con los datos de los servicios que contiene[6].
- **SignedPdfMonitorStatusOutput**: Partiendo de los servicios suministrados, genera un informe en formato pdf firmado. Realiza la operación partiendo de un fichero generado por la clase padre. Una vez creado, en esta clase se firma y se devuelve.
- **XmlMonitorStatusOutput**: Clase concreta que partiendo de un monitor, devuelve un String que contiene un xml con los datos de los servicios que contiene.

6.2. Otros ficheros implementados

- **utils/generateKeys.bat**[7][1]: Script para MS Windows para la automatización del proceso de generación de certificados susceptibles de ser usados para la firma digital de ficheros pdf desde una aplicación Java.
- **WebContent/index.jsp**: Página principal de la aplicación, desde la cual se accede al resto de opciones de que dispone.

1 La aplicación Java que utiliza el script generateKeys.bat para la creación del keystore ImportKey.java pertenece sus autores Joachim Karrer y Jens Carlberg y aparece publicada por Jochen Seifarth a través de http://www.agentbob.info/ sin una licencia específica. Ha sido modificada ligeramente para adaptarse a las necesidades de MonitorMashup. Se distribuye con su código fuente correspondiente.

- **WebContent/admin/listclients.jsp**: Página jsp que muestra un listado con los clientes almacenados en base de datos[8].
- **WebContent/admin/clients.jsp**: Página jsp que muestra el detalle de un cliente almacenado en base de datos. Tambien se utiliza para dar de alta clientes.
- **WebContent/css/style.css**: Hoja de estilos de la aplicación.
- **WebContent/xml/monitormashupdata.dtd**: Definición de tipo de documento correspondiente a la salida XML de la aplicación.
- **WebContent/xml/monitormashupdata.xsd**: Esquema de documento correspondiente a la salida XML de la aplicación. Junto con el anterior, se suministra para facilitar la integración de ***MonitorMashup*** con aplicaciones de terceros.

6.3. Ficheros de datos

- **utils/monitormashup.sql**: Script de MySQL, utilizado para la creación de la base de datos y la tabla utilizadas por la aplicación.
- **WebContent/cert/pubkey.pem**: Clave publica correspondiente al certificado utilizado para la firma digital de ficheros pdf.
- **WebContent/WEB-INF/classes/config.properties**: Fichero de configuración de la aplicación.
- **WebContent/WEB-INF/classes/keystore.ImportKey**: Almacén de certificados que guarda el certificado utilizado para la firma digital de ficheros pdf.

6.4. Ficheros de demostración

- **WebContent/demo/gethtml.html**: Fichero estático que simula la salida de la aplicación en formato HTML sin maquetación CSS ni cabeceras.
- **WebContent/demo/gethtmlfull.html**: Fichero estático que simula la salida de la aplicación en formato HTML con maquitación CSS.
- **WebContent/demo/getjson.json**: Fichero estático que simula la salida de la aplicación en formato json.
- **WebContent/demo/getxml.xml**: Fichero estático que simula la salida de la aplicación en formato XML.
- **WebContent/demo/monitormashup_report_aaaammdd_hhmm.pdf**: Fichero estático que simula la salida de la aplicación en formato PDF.
- **WebContent/demo/monitormashup_signed_report_aaaammdd_hhmm.pdf**: Fichero estático que simula la salida de la aplicación en formato PDF firmado digitalmente. Junto con todos los anteriores, se suministra para facilitar la integración de ***MonitorMashup*** con aplicaciones de terceros.

Hasta aquí la explicación de la implementación realizada. En el siguiente apartado se describirá la instalación y configuración de la aplicación. Como se indicó al comienzo del documento, se ha optado por integrar la instalación como capitulo y no como un anexo para ofrecer una visión mas amplia del producto desarrollado.

7. Instalación y configuración[1]

7.1. Instalación

7.1.1. Descripción del entregable

Puede verse una breve descripcion del contenido del software entregado en el apartado 1.5 Sumario de productos obtenidos.

7.1.2. Proceso de instalación

La instalación de ***MonitorMashup*** es un proceso simple que se puede realizar en pocos minutos.

Para que el despliegue de la aplicación se produzca sin errores, es necesario adaptar primero la configuración de Tomcat. Para ello, se detendrá el servidor de aplicaciones. Una vez detenido, se editará el fichero de Tomcat *conf/tomcat-users.xml*. En este fichero es necesario buscar el tag <*tomcat-users*>. Dentro de ese tag, se añadirá lo siguiente:

```
<role rolename="app"/>
<user username="asanchezcap" password="asanchezcap" roles="app"/>
```

Es necesario sustituir los atributos *username* y *password* con aquellos a los cuales se quiera dar permisos de acceso y modificación a la aplicación. Se pueden añadir tantos usuarios como se quiera, copiando la linea correspondiente al tag *user* y sustituyendo los atributos de acceso, respetando siempre el atributo *roles*, que debe contener el rol *app*.

La aplicación dispone de una interfaz gráfica para añadir y eliminar los sistemas de monitorización (o clientes) a los cuales se conectará la aplicación. Esta interfaz solo es funcional si se utiliza una base de datos para guardar la configuración de los clientes. Por ello, si no se desea que el usuario que accede a la aplicación tenga la posibilidad de modificar esta configuración, se recomienda definir los clientes a través del fichero del fichero de configuración. Esto podrá verse en detalle en el apartado *7.2 Configuración*.

Una vez guardados los cambios en el fichero *tomcat-users.xml*, se procederá a desplegar la aplicación. Para ello, basta con copiar el fichero

1 En este apartado se muestra un subconjuto del manual completo de instalación, que puede encontrarse dentro del entregable software.

MonitorMashup.war desde la carpeta *dist/* del entregable a la carpeta *webapps* de Tomcat.

Realizadas estas operaciones, se arrancará el servidor de aplicaciones. Una vez completado el arranque, podrá verse una carpeta *MonitorMashup* dentro de la carpeta *webapps* de Tomcat. Esto dará por completada la instalación.

Desde es punto puede optarse por proceder a la configuración de la aplicación siguiendo con el siguiente apartado o por probar su funcionamiento con los datos de prueba incorporados, siguiendo lo indicado en el apartado *7.3 Verificación del proceso de instalación*.

7.2. Configuración

7.2.1. Parametrización del servidor de aplicaciones

Durante la realización de las pruebas de carga no ha sido necesario modificar ningún parámetro de los encontrados por defecto en el servidor de aplicaciones para alcanzar los requisitos de cumplimiento.

7.2.2. Parametrización de la aplicación

La parametrización de la aplicación se realiza a través del fichero *config.properties*. Este fichero puede encontrarse en la carpeta *webapps/MonitorMashup/WEB-INF/classes*. Es importante reseñar que cada vez que se realicen cambios sobre este fichero será necesario recargar el contexto (o reiniciar el servidor de aplicaciones) para que la aplicación utilice los nuevos valores.

En los siguientes apartados se muestran los parámetros configurables.

7.2.2.1. Configuración de los informes

La aplicación permite personalizar los textos que aparecerán en los informes:

Propiedad	Descripción
table.title	Titulo del informe Valor por defecto: MonitorMashup
table.subtitle	Subtitulo del informe Valor por defecto: Services

Propiedad	Descripción
table.client	Encabezado de la columna de clientes del informe Valor por defecto: Client
table.host	Encabezado de la columna de hosts del informe Valor por defecto: Host
table.service	Encabezado de la columna de servicios del informe Valor por defecto: Service
table.status	Encabezado de la columna de estado del informe Valor por defecto: Status
table.duration	Encabezado de la columna de duración del informe Valor por defecto: Duration
table.link	Encabezado de la columna de procedimiento del informe Valor por defecto: Event Procedure

7.2.2.2. Configuración de las propiedades de firma

El fichero *config.properties* permite modificar las propiedades relacionadas con la firma de los ficheros en formato pdf:

Propiedad	Descripción
keystore.file	Nombre del fichero que contiene el certificado utilizado para la firma. Este fichero debera estar en la ruta *MonitorMashup/WebContent/WEB-INF/classes*. Valor por defecto: keystore.ImportKey **Se recomienda NO modificar esta propiedad.**
keystore.password	Contraseña del almacen de certificados identificado en la propiedad *keystore.file*. Valor por defecto: monitormashup
sign.reason	Texto que aparecerá en el campo *Reason* del sello de firma. Valor por defecto: monitormashup report
sign.location	Texto que aparecerá en el campo *Location* del sello de firma. Valor por defecto: monitormashup

7.2.2.3. Configuración del modo de funcionamiento

Propiedad	Descripción
mode	Esta propiedad permite fijar el modo de funcionamiento de la aplicación. Tiene dos valores posibles:

Propiedad	Descripción
	• file: Si indicamos este valor, los datos de los clientes serán gestionados a través del fichero de configuración. • db: Si indicamos este valor, los datos de los clientes serán gestionados a través una base de datos. **La introducción de cualquier otro valor distinto de los anteriores provocará la imposibilidad de aplicación para funcionar.** Valor por defecto: file

7.2.2.4. Configuración de la base de datos

La configuración de la base de datos es opcional. Los datos indicados en las siguientes propiedades solo serán utilizados en el caso de que en la propiedad *mode* se haya indicado el valor “db” (ver apartado *7.2.2.3 Configuración del modo de funcionamiento*).

Propiedad	Descripción
db.driver	Identificador del conector a la base de datos Valor por defecto: com.mysql.jdbc.Driver
db.url	Url de conexión a la base de datos Valor por defecto: jdbc:mysql://localhost:3306/monitormashup
db.user	Usuario de la base de datos Valor por defecto: monitormashup
db.password	Contraseña del usuario de la base de datos Valor por defecto: monitormashup

Para el correcto funcionamiento del modo db, el usuario indicado en la configuración deberá tener acceso de lectura/escritura sobre la base de datos. Previamente, se habrá creado la estructura de dicha base de datos por medio del script suministrado *monitormashup.sql* que se encuentra en la carpeta *utils/* del entregable.

7.2.3. Generación del certificado de firma

7.2.3.1. Generación del certificado

La aplicación es distribuida con un juego de certificados autofirmados. Esto permite comenzar a utilizar la aplicación una vez configurada sin mayor complicación. Sin embargo, se recomienda generar un certificado especifico

para cada instalación. Para facilitar esta operación el entregable dispone de un script para MS Windows que automatiza todo el proceso.

Este script tiene los siguientes requerimientos:

- OpenSSL[xlvi] versión 1: para la generación de los certificados.
- Java 8: para la creación del keystore y la inclusión de los certificados en él.

Para utilizar el script, se seguirán los siguientes pasos:

1. Entrar en la carpeta *utils/* del entregable.
2. Ejecutar el script *generateKeys.bat*.
3. Cuando la aplicación muestre el mensaje "*Enter PEM pass phrase:*", se introducirá la contraseña deseada para la clave privada (mínimo 5 caracteres). Una vez introducida, se pulsará enter.
4. Cuando la aplicación muestre el mensaje "*Verifying - Enter PEM pass phrase:*", se volverá a introducir la contraseña indicada en el paso anterior.
5. En este punto se solicitarán los datos informativos del certificado:
   5.1. *Country Name (2 letter code) [AU]*: Código de país (p.e. ES).
   5.2. *State or Province Name (full name) [Some-State]*: Nombre de provicia o similar (p.e. Sevilla).
   5.3. *Locality Name (eg, city) []*: Localidad (p.e. Camas).
   5.4. *Organization Name (eg, company) [Internet Widgits Pty Ltd]*: Nombre de la empresa u organismo (p.e. Universitat Oberta de Catalunya).
   5.5. *Organizational Unit Name (eg, section) []*: Departamento o sección dentro de la empresa (p.e. Trabajo de Fin de Grado).
   5.6. *Common Name (eg, YOUR name) []*: Nombre del propietario del certificado. Puede utilizarse el nombre del administrador de la aplicación, un identificador de la instalación, etc. Cuando el certificado se utilice para firma, este será el nombre que aparecerá como firmante y se verá en el sello del fichero pdf. (p.e. MonitorMashup Nodo 1).
   5.7. *Email Address []*: Dirección de correo del firmante. Se puede utilizar para indicar la dirección del administrador de la aplicación. (p.e. asanchezcap@uoc.edu).
6. Cuando la aplicación muestre el mensaje "*Enter pass phrase for keys\key.pem:*" se introducirá la contraseña elegida en el paso 3 y confirmada en el paso 4.
7. Por último, el script pedirá que un password para el keystore con el siguiente texto: "*Enter the keystore password (default monitormashup):*". Esta contraseña podrá ser distinta a la definida en el paso 3 y será la que se utilice a efectos de configuración de ***MonitorMashup***.

Una vez finalizada la ejecución del script, se dispondrá de todos los ficheros generados en carpeta *utils/keys*. El certificado en formato DER[xlvii] y PEM[xlviii], la clave privada en formato DER y PEM, la clave pública en formato PEM y el fichero keystore.

7.2.3.2. Instalación del certificado

Con el nuevo certificado generado en el apartado anterior, podrá procederse a configurar ***MonitorMashup*** para que utilice el nuevo certificado. El proceso se detalla en los los siguientes pasos:

1. Copiar el fichero que contiene la clave pública *utils/keys/pubkey.pem y* pegarlo en la carpeta del servidor de aplicaciones */webapps/MonitorMashup/cert/*, sobrescribiendo el existente.
2. Copiar el fichero keystore *utils/keys/keystore.ImportKey y* pegarlo en la carpeta del servidor de aplicaciones */webapps/MonitorMashup/Web-INF/classes/*, sobrescribiendo el existente.
3. Modificar el fichero de configuración y en la propiedad *keystore.password* indicar la nueva contraseña, definida en el punto 7 del apartado anterior. Para mas información sobre la modificación del fichero de configuración, consultar el apartado *7.2.2.2 Configuración de las propiedades de firma*.
4. Recargar el contexto de la aplicación o reiniciar el servidor de aplicaciones.

7.2.4. Gestión de clientes

Una vez elegido el modo de funcionamiento tal como se detalla en el apartado *7.2.2.3 Configuración del modo de funcionamiento*, puede procederse a la gestión de los clientes.

Independientemente del modo seleccionado, los datos que se necesitan conocer de los clientes son idénticos:

1. Nombre del cliente: Nombre con el que se identificará al cliente. Pueden ser sus siglas, su nombre comercial, su razón social…
2. URL del sistema de monitorización: Dirección donde se encuentre la API de acceso al sistema de motorización del cliente. Serán las siguientes:
   2.1. Nagios: http://DIRECCION_IP_O_DOMINIO/nagios/cgi-bin/statusjson.cgi?query=servicelist&details=true&servicestatus=warning+critical
   2.2. MonitorMashup: http://DIRECCION_IP_O_DOMINIO:8080/MonitorMashup/get/json

3. Usuario en el sistema de monitorización: Usuario con permisos de lectura en el sistema de monitorización.
4. Contraseña en el sistema de monitorización: Contraseña del usuario anterior.
5. Enlace a la documentación: Dirección del sistema donde se almacene la documentación relativa al cliente. En esta documentación se espera que existan procedimientos para la resolución de cuanta incidencia pueda aparecer.
6. Tipo de herramienta de conexión: Identificador, propio de ***MonitorMashup***, del tipo de sistema de motorización configurado. Son los siguientes:
   6.1. Nagios: *jsonnagios*
   6.2. MonitorMashup: *jsonmonitormashup*

Disponiendo de estos datos para cada uno de los clientes a configurar, se procederá a darlos de alta en el sistema según el modo elegido.

7.2.4.1. Gestión de clientes en modo file

La configuración a través de un fichero de es el modo por defecto. Este modo, esta pensado para sistemas que requieran un bajo consumo de recursos o bien donde no se quiera permitir que los usuarios puedan modificar los datos de los clientes, quedando esto reservado para un administrador que tenga permisos para modificar el fichero en el servidor donde este instalado el Tomcat que albergue la aplicación y para recargar el contexto (o reiniciar Tomcat) para que la aplicación utilice los nuevos valores.

El fichero de configuración será el mismo visto en el apartado *7.2.2 Parametrización de la aplicación*, el fichero *config.properties*. Este fichero puede encontrarse en la carpeta *webapps/MonitorMashup/WEB-INF/classes*. Las propiedades a configurar serán las siguientes:

Propiedad	Descripción
numeroServidores	En esta propiedad es necesario indicar el numero (entero) se servidores de monitorización a los que se quiera conectar.
server.n.client	Nombre del cliente numero n (n entre 1 y N)
server.n.site	URL del sistema de monitorización numero n (n entre 1 y N)
server.n.siteUser	Usuario en el sistema de monitorización numero n (n entre 1 y N)
server.n.sitePassword	Contraseña en el sistema de monitorización numero n (n entre 1 y N)

Propiedad	Descripción
server.n.link	Enlace a la documentación numero n (n entre 1 y N)
server.n.tool	Tipo de herramienta de conexión numero n (n entre 1 y N)

En el siguiente ejemplo puede verse una configuración donde se utilizan las propiedades de forma correcta:

```
numeroServidores=2

server.1.client=NagiosTest1
server.1.site=http://ip1/nagios/cgi-bin/statusjson.cgi?
query=servicelist&details=true&servicestatus=warning+critical
server.1.siteUser=nagiosadmin
server.1.sitePassword=nagiosadmin
server.1.link=https://doc.nagiostest1.com/wiki
server.1.tool=jsonnagios

server.2.client=NagiosTest2
server.2.site=http://ip2/nagios/cgi-bin/statusjson.cgi?
query=servicelist&details=true&servicestatus=warning+critical
server.2.siteUser=nagiosadmin
server.2.sitePassword=nagiosadmin
server.2.link=https://doc.nagiostest2.com/wiki
server.2.tool=jsonnagios
```

La descripción y posibles valores correspondientes a los servidores están detallados al comienzo de este apartado.

7.2.4.2. Gestión de clientes en modo db

La gestión de clientes se simplifica notablemente al realizarse a través de una base de datos. En este modo de funcionamiento ***MonitorMashup*** proporciona un CRUD[xlix] para la realización de estas operaciones, accesible desde el inicio de de la aplicación.

Estando en la pantalla inicial de la aplicación, y pulsando en el enlace "*Admin zone*", se accede al listado de clientes datos de alta. Desde dicho listado pueden añadirse clientes, pulsando "*Add Client*", o volver al inicio de la aplicación, pulsando "*Exit Admin zone*". Eligiendo añadir un cliente, se accede al formulario de creación. La descripción y posibles valores correspondientes a los clientes están detallados al comienzo de este apartado.

7.3. Verificación del proceso de instalación

Para la verificación de la instalación y de la principal funcionalidad, así como para las pruebas de carga (ver apartado siguiente *8 Pruebas*) se empleará Apache Jmeter. Como ya se expuso en el apartado *1.2 Objetivos del*

Trabajo, queda fuera de las pretensiones de este documento la instalación y configuración de otros sistemas, por lo cual se darán las explicaciones mínimas necesarias para la realización de las pruebas.

7.3.1. Preparación del entorno de verificación

Para iniciar las pruebas es necesario disponer de una instalación de ***MonitorMashup*** con al menos un cliente configurado.

Se descargará desde el sitio oficial **Apache JMeter 3.1**[1], que tiene como requisito Java 7. Una vez descargado, basta con descomprimirlo para empezar a usarse. Se ejecutará el fichero *jmeter.bat* (o *jmeter.sh*) que se encuentra en *apache-jmeter-3.1\bin*. Esto abrirá la interfaz gráfica de jmeter.

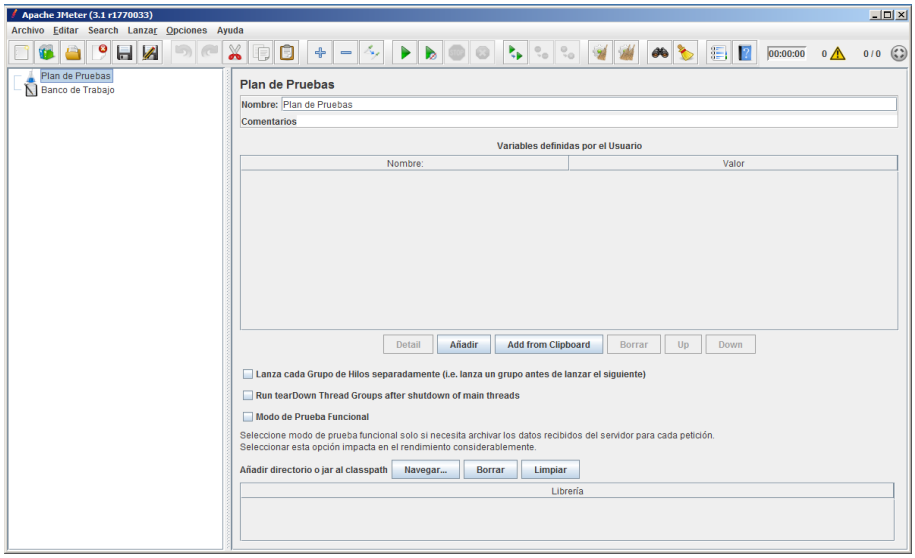


Ilustración 14: Pantalla principal de jmeter

Para abrir el fichero que contiene las pruebas en jmeter, se pulsará sobre *Archivo → Abrir*, y se buscara la carpeta *jmeter/* del entregable. Una vez en la carpeta, se seleccionará el archivo *PlandePruebas_MonitorMashup.jmx* y se pulsará *Abrir*.

Con el archivo cargado, se desplegara *Grupo de Hilos* en la zona izquierda de la ventana y después se pulsará sobre *Variables definidas por el Usuario*. Las variables a definir son las siguientes (cambiar solo el valor):

- HOST: Nombre de dominio o direccion ip donde se encuentra instalado ***MonitorMashup***.
- PORT: Puerto de escucha del servidor de aplicaciones.
- PATH: URL de la aplicación (coincide con el nombre del contexto en el servidor de aplicaciones. Por defecto, MonitorMashup).

1 http://jmeter.apache.org/download_jmeter.cgi

- USER: Usuario con acceso a la aplicación.
- PASS: Contraseña del usuario anterior.

Una vez definidas las variables, el resultado puede verse en la siguiente imagen:

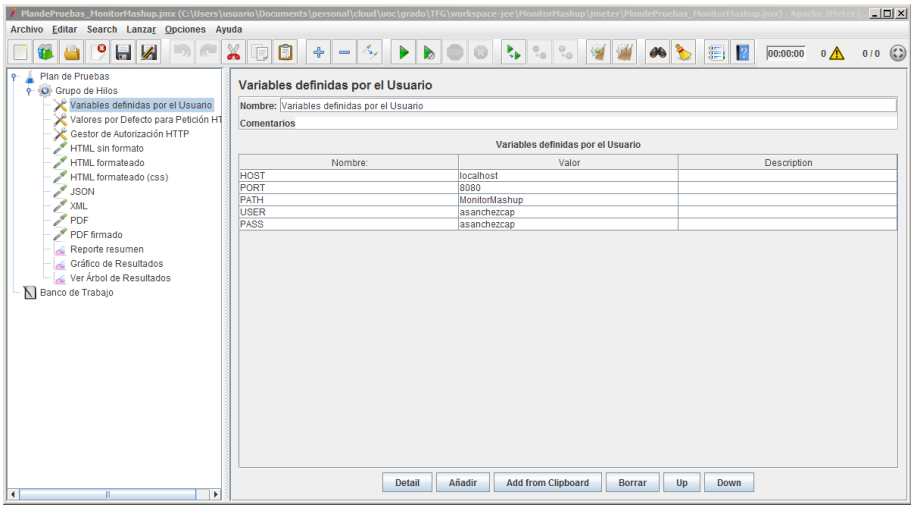


Ilustración 15: Variables definidas en la prueba de verificación en jmeter

7.3.2. Ejecución de la verificación

En este punto ya se esta en disposición de iniciar la prueba. Antes de realizar cada prueba se recomienda limpiar los resultados de los reportes. Esta limpieza se puede hacer de dos maneras: con el atajo de teclado Ctrl+E o bien a través de la opción del menú *Lanzar → Limpiar Todo*.

Para ver el resultado de la prueba se utilizara el Reporte resumen, al que se accede desde el menú de la izquierda. Una vez seleccionado, se arrancara la prueba con el atajo de teclado Ctrl+R o bien a través de la opción del menú *Lanzar → Arrancar*. Terminada la ejecución, se podrán ver los resultados:

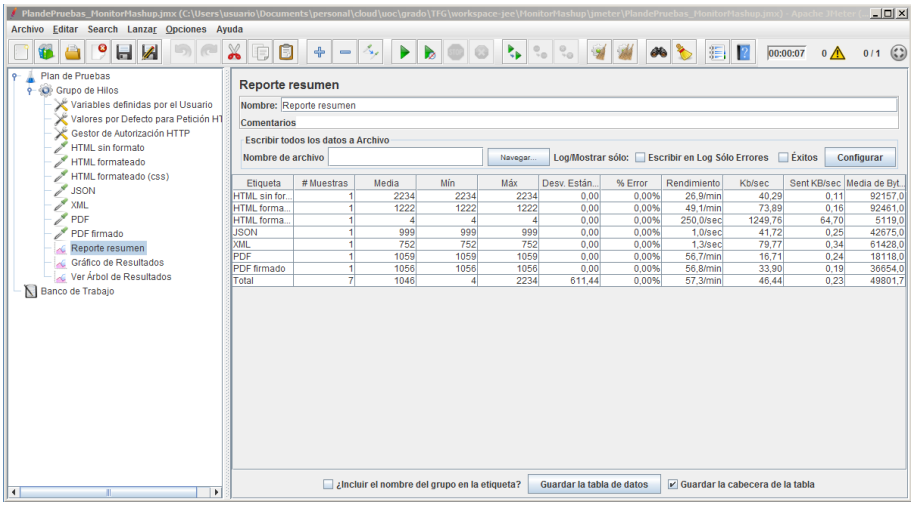


Ilustración 16: Resultados de la prueba de verificación en jmeter

En los resultados se verificará columna “% Error”, comprobando que todas las filas presentan 0,00%. Así mismo, se comprobará que el log del servidor de aplicaciones esta limpio de errores durante la ejecución de la prueba.

Una vez completada la instalación, en el siguiente apartado se continuará utilizando la aplicación jmeter para la realización de las pruebas de aceptación y verificar el adecuado cumplimiento de los requisitos.

8. Pruebas

Para verificar los requisitos de cumplimiento detallados en el apartado *3.2.3 Requisitos de cumplimiento* se ha utilizado la misma prueba de jmeter que en el apartado anterior, por lo que antes de iniciar las pruebas, es necesario realizar las tareas detalladas en el subapartado *7.3.1 Preparación del entorno de verificación*.

Una vez preparado el entorno, es necesario realizar los siguientes cambios:

- En ***MonitorMashup***: se han de configurar 6 clientes con una media de 10 eventos cada uno y un total superior a 200 eventos. Para los resultados que se detallan a continuación se ha usado 6 veces el servidor que el equipo de desarrollo de *nagios* publica en internet como demostración. Dado que habitualmente presenta mas de 200 eventos, estamos hablando de un total de 1200, lo que cubre ampliamente los requisitos de la prueba. Los datos de este servidor son los siguientes:
  - URL: http://nagioscore.demos.nagios.com/nagios/cgi-bin/statusjson.cgi?query=servicelist&details=true&servicestatus=warning+critical
  - Usuario: nagiosadmin
  - Contraseña: nagiosadmin
- En la prueba jmeter: deshabilitar los siguientes elementos de prueba:
  - HTML formateado
  - HTML formateado (css)
  - JSON
  - XML
  - PDF
  - PDF firmado

En el caso de no deshabilitar estas entradas, no se estaría probando lo establecido en el requisito (estaríamos repitiendo la prueba 7 veces, una por formato de salida) y estaríamos provocando un estrés innecesario sobre un servidor publico, causando un perjuicio sobre una web de un proyecto de software libre. En la siguiente imagen puede verse como deshabilitar las entradas:

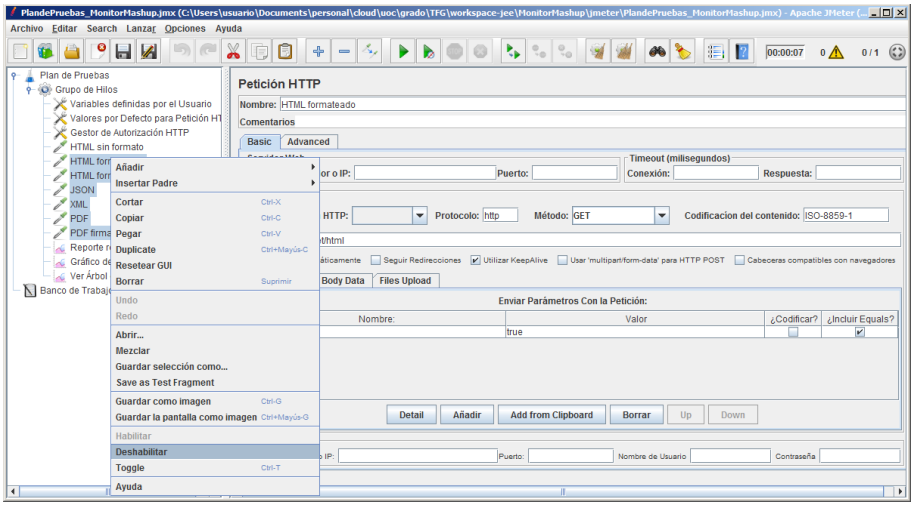


Ilustración 17: Ejemplo de como deshabilitar elementos de prueba en jmeter

En este punto ya se puede iniciar la prueba. Antes de realizar cada prueba se recomienda limpiar los resultados de los reportes con el atajo de teclado Ctrl+E o a través de la opción del menú *Lanzar → Limpiar Todo*. Seleccionamos la entrada *Reporte resumen* en la parte de la izquierda y lanzamos la prueba con el atajo de teclado Ctrl+R o bien a través de la opción del menú *Lanzar → Arrancar*.

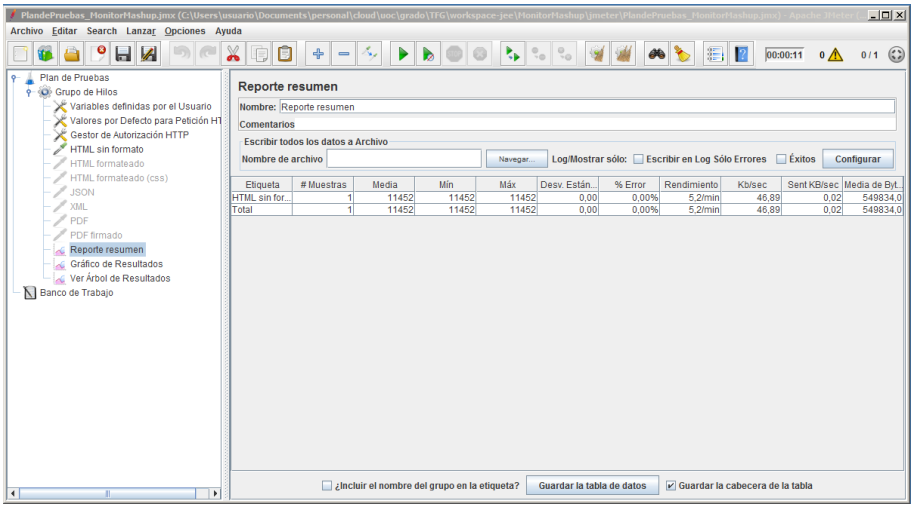


Ilustración 18: Resultados de la prueba de cumplimiento en jmeter

Los resultados de la prueba que pueden verse en la imagen anterior muestran un tiempo de respuesta de 11,452 segundos para 6 clientes con un total de 1404 eventos con un 0% de errores y sin errores en el log del servidor de aplicaciones. Puede concluirse que que se superan los requisitos indicados en el citado apartado*:*

1. El sistema deberá ser capaz de mostrar al menos 6 clientes con una media de 10 eventos cada uno: muestra 6 clientes con una media de 234 eventos.

2. El sistema deberá ser capaz de mostrar al menos 200 eventos: muestra 1404 eventos.
3. Para un sistema con C clientes y E eventos, el tiempo de respuesta medio deberá ser inferior a C+(E*0,01) segundos: 11,452s<6s+(1404*0,01)s=20,04s.

Hasta aquí las pruebas de aceptación. En el siguiente epígrafe se obtendrán las preceptivas conclusiones de lo visto tanto en este apartado como en los anteriores para facilitar una valoración de este Trabajo.

9. Conclusiones

9.1. Cumplimiento de los objetivos

Durante la realización de este trabajo se han cumplido satisfactoriamente tanto el objetivo principal (profundizar en el conocimiento del ecosistema Java EE), como los objetivos concretos del proyecto desarrollado. La aplicación resultante, ***MonitorMashup***, cumple con todos los requisitos indicados en el apartado *3 Toma de requisitos*, tanto funcionales como no funcionales.

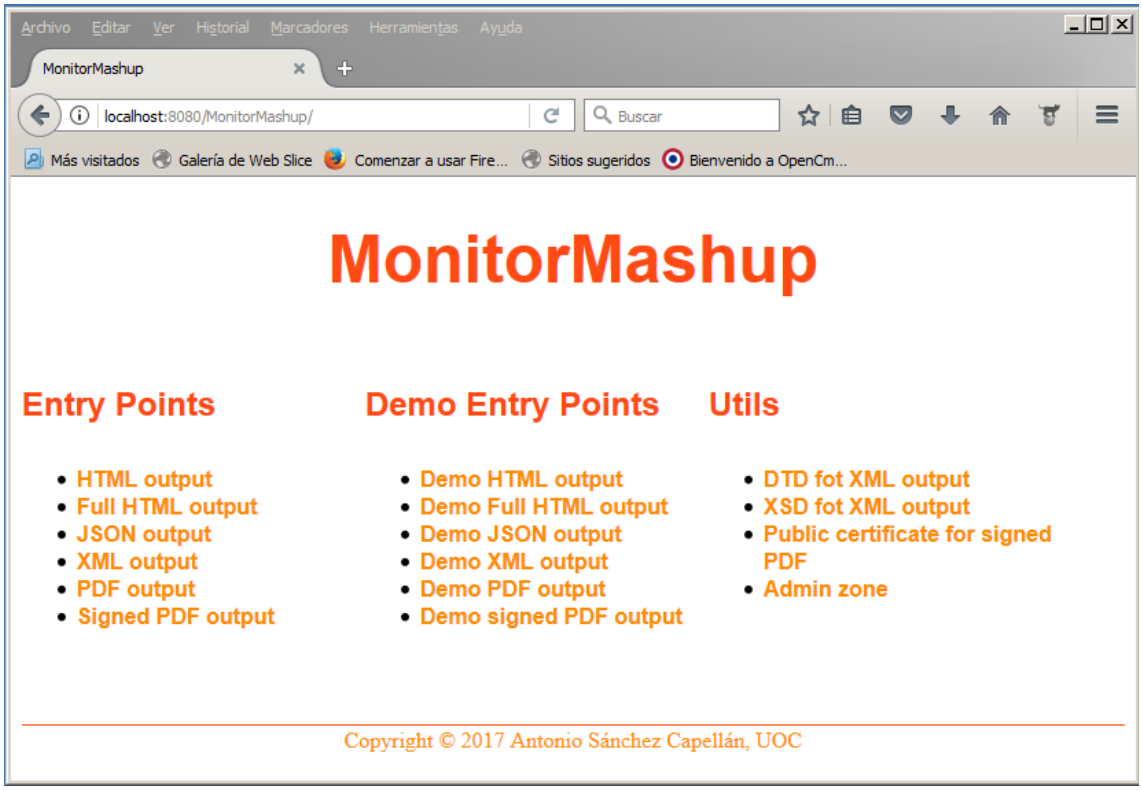


Ilustración 19: Vista de la página principal de ***MonitorMashup***

Cabe reseñar que no todos los requisitos fueron identificados en la toma inicial, sino que algunos fueron detectados durante fases posteriores. Por ejemplo, un requisito identificado de esta forma fue la necesidad de generar informes en formato PDF firmados digitalmente. Estos informes permiten capturar una “foto fija” de la situación de los clientes en un momento dado, impidiendo su modificación posterior, siendo un informe muy útil tanto para la justificación de la facturación de los servicios prestados a los clientes, como para el control del rendimiento de la plantilla.

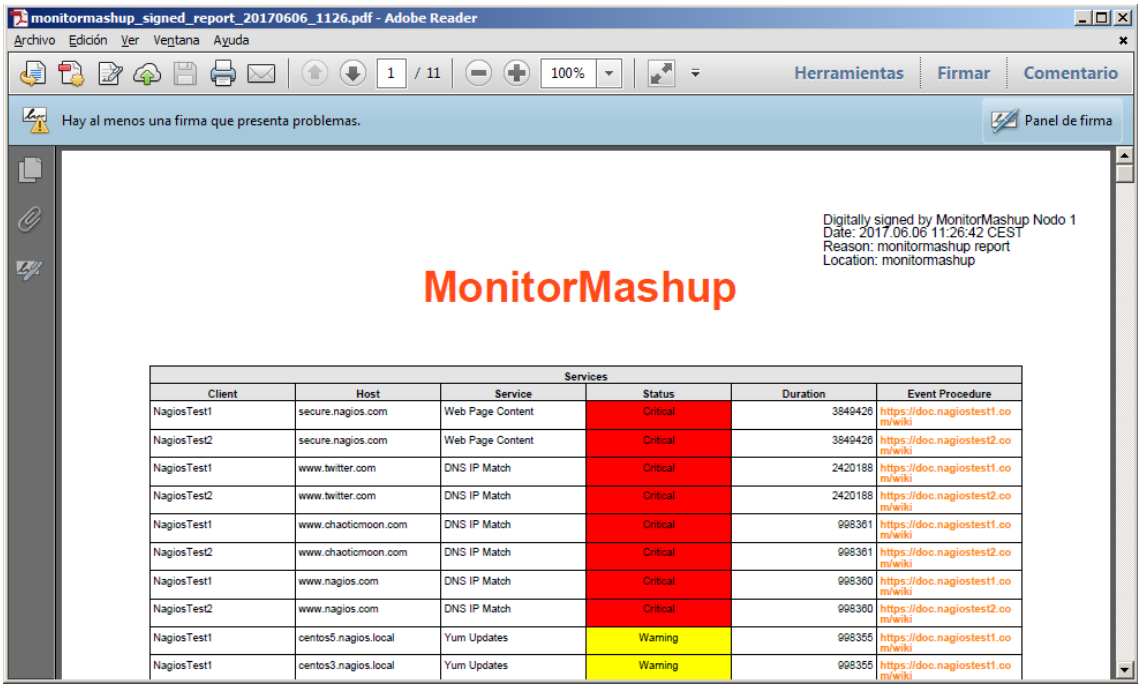


Ilustración 20: Ejemplo de fichero PDF de salida firmado digitalmente

9.2. Seguimiento de la planificación

Como se detalla a lo largo del documento, la consecución de los objetivos se realiza cumpliendo de forma simultanea la planificación prevista en el apartado *1.4 Planificación del Trabajo*. Sin embargo, es importante reseñar dos casos en los que ha sido necesario modificar la realización de las tareas para garantizar su cumplimiento:

Gestión de riesgos

Durante la ejecución del proyecto, se presentó el riesgo identificado como **R01** (ver apartado *1.4.1 Identificación y gestión de riesgos*). Resultando imposible la instalación de un entorno virtual con Nagios y un numero significativo de servidores con diferentes servicios a monitorizar, se optó por ejecutar el plan de contingencia **A1R01**. Es plan consistió en utilizar la instalación de demostración que el equipo de desarrollo de Nagios tiene disponible online. La ejecución del plan permitió mantener el cronograma sin ninguna variación.

Paralelización de tareas

En la fase de Cierre del proyecto, hubiera sido mas realista paralelizar las tareas 22 (Elaboración de la presentación) y 24 (Elaboración de la memoria), modificando en consecuencia la dedicación en cada una al 50%. Las sinergias que surgen durante la realización de ambos documentos justificaría esta realización simultanea: la necesidad de actualizar un documento ante un

cambio en el otro ha sido constante. Planificar estas tareas en paralelo hubiera supuesto un aumento de la duración de ambas y la postergación de la finalización de la presentación. Esta planificación, peor a priori, hubiera sido mas realista.

De ambos casos podemos sacar la siguientes conclusiones en cuanto a planificación:

1. Una identificación temprana de riesgos y la planificación de planes de contingencia son vitales para garantizar la realización en tiempo y forma de los proyectos.
2. La experiencia previa en la realización de un determinado tipo de proyecto permite realizar planificaciones mas ajustadas y realistas.

9.3. Líneas futuras de trabajo

De la ejecución del proyecto realizado y de las pruebas realizadas con el producto resultante, se recomiendan las siguientes mejoras y evoluciones:

1. **Aumento de los sistemas de monitorización soportados**: Actualmente solo están soportados Nagios y ***MonitorMashup***. Nagios ha sido hasta ahora el estándar de la industria en su segmento, pero su falta de evolución esta propiciando la aparición de nuevos sistemas. De estos sistemas, los dos principales candidatos serían Zabbix[l] y Pandora FMS[li].
2. **Cacheo y separación entre la lectura de los clientes y los resultados mostrados**: Con la implementación realizada, la lectura de los clientes se realiza tras cada petición. Una vez realizada la lectura los resultados son mostrados como respuesta a dicha petición. Esto garantiza que los resultados mostrados nunca estarán desactualizados. Si embargo, podría ser positivo separar estas dos acciones. Por un lado se realizaría la lectura de los clientes de forma temporizada. Estos datos serían almacenados en base de datos en bruto o bien procesados. Por otro lado, respondiendo a las peticiones, se mostrarían los resultados. Ajustando la periodización se podría conseguir resultados razonablemente actualizados eliminando la sobrecarga que supone realizar una consulta a cada cliente por cada petición recibida.
3. **Paralelización de las consultas a los clientes**: Tal como se puede ver en la *Ilustración 10: Diagrama de secuencia principal de la aplicación*, las consultas a los clientes se realizan de forma secuencial por medio de un bucle. Aunque esta implementación ha demostrado ser mas que suficiente para mostrar un elevado numero de incidencias en pantalla, la utilización de una aproximación multihilo podría mejorar los tiempos de

respuesta y aumentar la disponibilidad de la aplicación en caso de errores de conexión con los clientes.

4. **Utilización del patrón decorador en la generación de ficheros PDF**: Como ya se comentó en el apartado *5.2.3 Paquete edu.uoc.asanchezcap.monitormashup.output*, en caso de que fuera necesario aumentar el numero de opciones relacionadas con ficheros PDF (en este momento hay dos, con y sin firma digital), sería conveniente refactorizar el código actual utilizando el patrón decorador en lugar de utilizar herencia como hasta ahora.
5. **Mejoras en el algoritmo de clasificación**: El software desarrollado clasifica y ordena las incidencias recopiladas en función de la duración del evento. El poder incorporar y configurar nuevos elementos a la clasificación mejoraría enormemente la funcionalidad. Por ejemplo, incorporar los SLA asociados a cada cliente: los tiempos de respuesta, los horarios de respuesta (24x7, 8x5…), los tiempos de resolución, etc.

Con la exposición de las lineas de trabajo propuestas se dan por finalizadas las conclusiones. En los siguientes apartados podrán consultarse tanto las referencias bibliográficas como un pequeño glosario de términos.

10. Bibliografía

[1] José Ramón Rodríguez. La gestión de proyectos. Conceptos básicos.
UOC, PID 00153562.

[2] Jordi Pradel Miquel y Jose Raya Martos. Introducción a la ingeniería de requisitos.
UOC, PID 00230179.

[3] Digital Signatures in a PDF
https://www.adobe.com/devnet-docs/acrobatetk/tools/DigSig/Acrobat_DigitalSignatures_in_PDF.pdf
Visitado en Abril 2017.

[4] Gamma E. et al. Patrones de diseño (César Fernández Acebal, trad.).
ADDISON WESLEY, España, 2002. (Obra original publicada en 1995).

[5] Vaskaran Sarcar. Java Design Patterns. A Tour with 23 Gang of Four Design Patterns in Java.
Apress, New York, 2016.

[6] iText 5 examples.
http://developers.itextpdf.com/content/itext-5-examples
Visitado en Mayo 2017.

[7] Import private key and certificate into Java Key Store (JKS).
http://www.agentbob.info/agentbob/79-AB.html
Visitado en Mayo 2017.

[8] Simple CRUD Using Jsp, Servlet and MySQL
https://danielniko.wordpress.com/2012/04/17/simple-crud-using-jsp-servlet-and-mysql/
Visitado en Mayo 2017.

11. Glosario

i **API**: https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

ii **JSON**: https://es.wikipedia.org/wiki/JSON

iii **Nagios**: https://es.wikipedia.org/wiki/Nagios

iv **Desarrollo en cascada**: https://es.wikipedia.org/wiki/Desarrollo_en_cascada

v **Fichero class**: https://en.wikipedia.org/wiki/Java_class_file

vi **Fichero war**: https://es.wikipedia.org/wiki/WAR_(archivo)

vii **Documentación Javadoc**: https://es.wikipedia.org/wiki/Javadoc

viii **Apache JMeter**: http://jmeter.apache.org/

ix **Fichero de script**: https://es.wikipedia.org/wiki/Script

x **Certificado autofirmado**: https://en.wikipedia.org/wiki/Self-signed_certificate

xi **Keystore**: https://en.wikipedia.org/wiki/Keystore

xii **UML**: https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado

xiii **Pruebas de aceptación**: https://es.wikipedia.org/wiki/Pruebas_funcionales

xiv **Apache Tomcat**: https://es.wikipedia.org/wiki/Tomcat

xv **Documentación de Tomcat 8**: http://tomcat.apache.org/tomcat-8.0-doc/index.html

xvi **Máquina Virtual Java**: https://es.wikipedia.org/wiki/M%C3%A1quina_virtual_Java

xvii **JVM de Oracle**: https://www.java.com/es/download/

xviii **MySQL**: https://es.wikipedia.org/wiki/MySQL

xix **Licenciamiento de MySQL**: https://www.mysql.com/about/legal/

xx **Fork**: https://en.wikipedia.org/wiki/Fork_(software_development)

xxi **MariaDB**: https://es.wikipedia.org/wiki/MariaDB

xxii **Eclipse IDE**: https://es.wikipedia.org/wiki/Eclipse_(software)

xxiii **Licencia Eclipse**:
https://eclipse.org/org/documents/epl-v10.php

xxiv **ProjectLibre**:
https://es.wikipedia.org/wiki/ProjectLibre
http://www.projectlibre.com/product/projectlibre-open-source

xxv **JDBC**:
https://es.wikipedia.org/wiki/Java_Database_Connectivity

xxvi **JSON-java**:
https://github.com/stleary/JSON-java

xxvii **itextpdf**:
http://itextpdf.com/

xxviii **The Bouncy Castle Crypto APIs**:
https://www.bouncycastle.org/java.html

xxix **DTD**:
https://es.wikipedia.org/wiki/Definici%C3%B3n_de_tipo_de_documento

xxx **XSD**:
https://es.wikipedia.org/wiki/XML_Schema

xxxi **Firma digital**:
https://es.wikipedia.org/wiki/Firma_digital

xxxii **CSS**:
https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada

xxxiii **JavaScript**:
https://es.wikipedia.org/wiki/JavaScript

xxxiv **Basic access authentication (Autenticación de acceso básica)**:
https://es.wikipedia.org/wiki/Autenticaci%C3%B3n_de_acceso_b%C3%A1sica

xxxv **RFC2617**:
https://tools.ietf.org/html/rfc2617

xxxvi **DAO**:
https://es.wikipedia.org/wiki/Data_Access_Object

xxxvii **Patrón de diseño Método Factoría**:
https://es.wikipedia.org/wiki/Factory_Method_(patr%C3%B3n_de_dise%C3%B1o)

xxxviii **Principio Abierto/Cerrado**:
https://es.wikipedia.org/wiki/Principio_de_abierto/cerrado

xxxix **Patrón Decorador**:
https://es.wikipedia.org/wiki/Decorator_(patr%C3%B3n_de_dise%C3%B1o)

xl **Timeout**:
https://en.wikipedia.org/wiki/Timeout_(computing)

xli **Multihilo**:
https://es.wikipedia.org/wiki/Multihilo

xlii **Cache**:
https://en.wikipedia.org/wiki/Cache_(computing)

xliii **Patrón de arquitectura MVC (Modelo - Vista - Controlador)**:
https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador

xliv **Firewall**:
https://es.wikipedia.org/wiki/Cortafuegos_(inform%C3%A1tica)

xlv **Patrón de diseño Singleton**:
https://es.wikipedia.org/wiki/Singleton

xlvi **OpenSSL**:
https://es.wikipedia.org/wiki/OpenSSL
xlvii **Formato DER**:
https://en.wikipedia.org/wiki/X.690#DER_encoding
xlviii **Formato PEM**:
https://en.wikipedia.org/wiki/Privacy-enhanced_Electronic_Mail
xlix **CRUD**:
https://es.wikipedia.org/wiki/CRUD
l **Zabbix**:
http://www.zabbix.com/
https://es.wikipedia.org/wiki/Zabbix
li **Pandora FMS**:
https://pandorafms.org/es/
https://es.wikipedia.org/wiki/Pandora_FMS