Capítulo III: Desarrollo

1 – Descripción del entorno de desarrollo

La plataforma que se va a montar tiene unas amplias posibilidades de expansión mediante el desarrollo de nuevos portlets que añadan diversas funcionalidades al portal.

Por ello, aunque existen numerosas posibilidades en cuanto al entorno de desarrollo de portlets de Liferay, se considera necesario mostrar uno de ellos con el objetivo de facilitar el desarrollo de los mismos.

1.1 – Eclipse Lomboz 3.3

Para el desarrollo de los portlets se ha utilizado Eclipse Lomboz en su versión 3.3.

Eclipse Lomboz es una distribución "especial" de Eclipse que contiene numerosos plugins y otras herramientas que facilitan el desarrollo de portlets de Liferay. Es ésta la principal razón de que se haya elegido éste y no otro entorno de desarrollo.

Gracias a un plugin, es posible lanzar un servidor Tomcat con el que, configurado correctamente, es posible depurar Liferay o cualquiera de los módulos desarrollados para el mismo.

1.2 – Liferay PluginSDK 5.1.2 y Ant

Liferay ofrece varias posibilidades a la hora de desarrollar nuevos portlets. Una de ellas PluginSDK.

PluginSDK contiene varios scripts que, haciendo uso de Ant, permite crear el esqueleto básico de un portlet, un tema o una plantilla de página.

La idea es crear la estructura básica de éstos elementos con esta herramienta y a partir de la misma seguir desarrollando mediante Eclipse.

Incorporar Ant facilita mucho la tarea de desarrollo, ya que consigue integrar en una acción tareas como la compilación, encapsulación del portlet y creación del archivo .war y el despliegue del mismo en el servidor en un solo paso.

1.3 – EasyPHP

EasyPHP es un paquete de software que consiste principalmente en una base de datos MySQL, un servidor web Apache e intérpretes para lenguajes de script.

Lo más importante es que dispone de la utilidad phpMyAdmin, que proporciona un entorno amigable para la creación y gestión de bases de datos existentes en un servidor MySQL. La instalación es inmediata y no necesita ninguna configuración especial.

2 – Desarrollo del portlet de Dapper

2.1 – Descripción

Como se ha comentado con anterioridad Dapper es una herramienta que nos brinda la posibilidad de obtener contenido externo a partir de otras webs, de forma transparente, dinámica y en tiempo real para poder mostrarlo en nuestro portal de la manera que se considere oportuna, pudiendo modificar la estructura o la forma en la que esta información es mostrada.

Para poder aprovechar esta funcionalidad en el portal Liferay se ha desarrollado un portlet que permite hacer uso de Dapps que hayan sido creados previamente (Ver Anexos).

La funcionalidad de este portlet va más allá de la ofrecida por Dapper en ciertos aspectos. En el portlet se ofrece la posibilidad de obtener el contenido en formato HTML o XML. Para cada uno de estos formatos se realizará un procesamiento adicional con el objetivo de obtener un mayor control sobre el resultado final.

Por defecto, si el formato de salida elegido es HTML, la estructura del mismo viene totalmente definida, tanto la distribución de las secciones como el título de la cabecera, las tablas, etc. De cara a mejorar la visualización, se ha hecho opcional el mostrar o no un título y se han eliminado fragmentos de código HTML que introducían ciertas secciones y márgenes. Éstos, en una visualización desde PC resultaban aceptables, pero que en un dispositivo móvil eran excesivos.

Imagen 34: Dapp con el procesado en el portlet y sin él (PC)

En cuanto al contenido con formato XML devuelto por Dapper puede parecer que no tiene mucho sentido, puesto que no tiene ningún interés para el usuario final. Sin embargo es una opción interesante debido a que se ha añadido la posibilidad de añadir plantillas XSL. De esta forma se conoce exactamente el XML recibido, es posible diseñar un XSL a medida para cada Dapp. Con ello conseguimos obtener un mayor control del contenido que se quiere mostrar; su estructura, estilos, permitiendo además la inclusión de código HTML propio en la página que se devuelva como resultado final.

No hay que olvidar que uno de los objetivos de este portlet es adaptar el contenido final, para ser visualizado correctamente en un dispositivo móvil, por lo que tras el proceso de extracción de la información y el tratamiento comentado anteriormente se procederá a la adaptación de las imágenes que pudieran aparecer en dicho contenido.

2.2 – Configuración del portlet

Antes de poder visualizar el contenido de un Dapp dentro del portlet se ha de llevar a cabo un proceso de configuración del mismo. De lo contrario el portlet mostrará el siguiente mensaje:

Imagen 35: Portlet no configurado

Configuración del portlet

Para proceder a la configuración debemos hacer click el botón y seleccionar “Preferencias”:

Imagen 36: Seleccionar preferencias

A continuación se accederá a la página de configuración del portlet donde podremos Crear o Eliminar un Dapp o una plantilla. Nótese que con "Crear Dapp" no nos referimos realmente a crearlo (ver Anexos), sino a crear una entrada en la base de datos para indicarle de alguna manera a Liferay que ese Dapp existe y podemos usarlo.

Para asociar un Dapp al portlet debemos seleccionarlo de la lista de Dapps disponibles y seleccionar "Configurar". En caso de no existir ningún Dapp disponible aparecerá un mensaje indicativo.

Imagen 37: Preferencias

En la página de configuración podemos seleccionar el formato en el que queremos que sea devuelto el contenido. Podemos elegir entre HTML o XML.

Imagen 38: Contenido del Dapp devuelto en HTML

En caso de que el formato elegido sea HTML, podemos elegir un título para mostrar encima del contenido devuelto. Si el formato es XML, tenemos la opción de mostrar únicamente el contenido devuelto directamente en XML o de aplicarle una plantilla XSL, con el objetivo de efectuar transformaciones al XML y generar una página HTML tal y como se indique en la plantilla.

En la siguiente captura se muestra la elección del tipo devuelto como XML y la elección de la plantilla que se le aplicará al mismo.

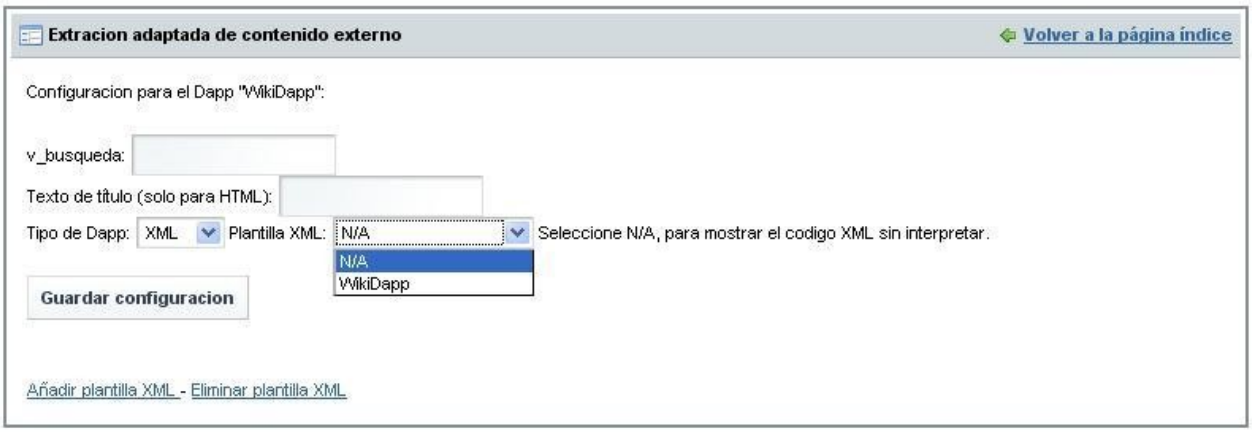

Imagen 39: Contenido del Dapp devuelto como XML

También podemos observar que, en caso de que el Dapp haga uso de parámetros, es en este punto en el que se le asignará el valor correspondiente.

Crear Dapp

Para poder configurar un portlet con un determinado Dapp, este Dapp deber de haber sido creado y dado de alta en la página de Dapper (http://www.dapper.net) y posteriormente en el portal.

Al seleccionar "Crear Dapp" indicamos al portal que el Dapp pasa a ser accesible, es decir, permitimos que pueda ser usado en un portlet de este tipo.

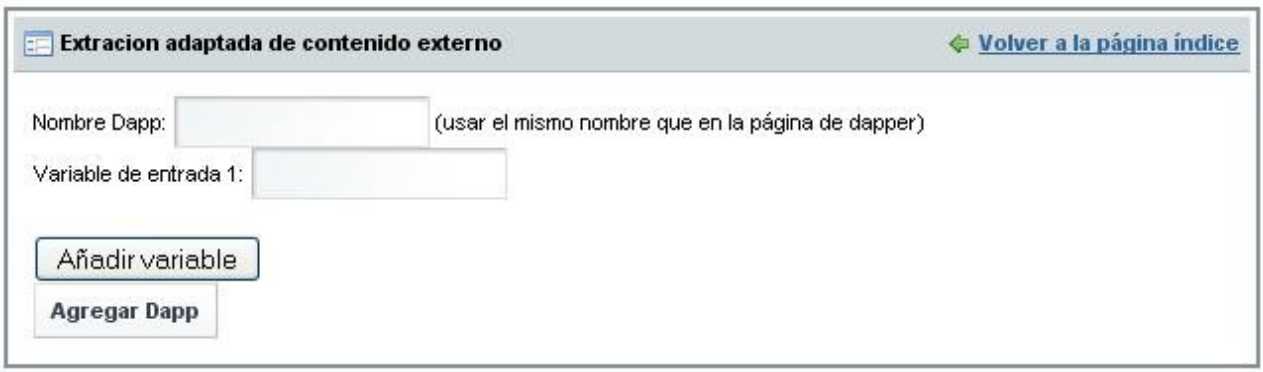

Imagen 40: Crear Dapp

Tan solo tenemos que indicar el nombre del Dapp y de las variables, en caso de que haga uso de ellas. Todos estos nombres (del Dapp y las variables) deben corresponder **exactamente** con los nombres que se les asignaron durante la creación del Dapp, en la página de Dapper.

Eliminar Dapp

Si deseamos eliminar un Dapp del sistema, tan solo debemos seleccionar el enlace correspondiente en la página de configuración de cualquier portlet de Dapper.

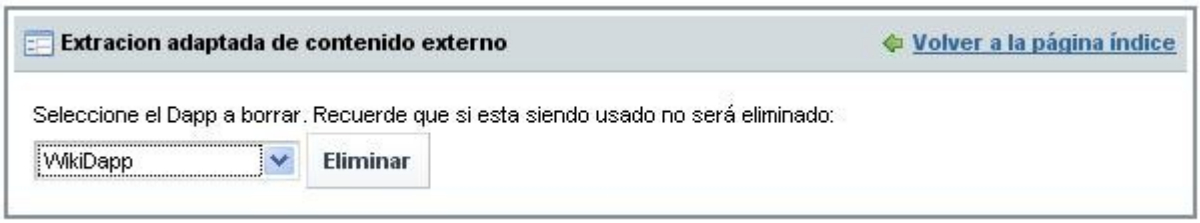

Imagen 41: Eliminar Dapp

Nos parecerá un desplegable en el que se seleccionará el Dapp que se desea eliminar. Es necesario indicar que no se podrá eliminar un Dapp que esté siendo usado en éste o en cualquier otro portlet para evitar inconsistencias en el sistema.

Crear Plantilla

Para hacer uso de las transformaciones mediante XSL, es necesario crear previamente un plantilla. Para ello se debe seleccionar el enlace "Añadir plantilla XML". Esto llevará a un formulario donde debemos asignarle un nombre a la plantilla y escribir o pegar el código de la misma en el espacio correspondiente.

Imagen 42: Añadir plantilla XML

Eliminar plantilla

En caso de que lo que se desee sea eliminar una plantilla se debe seleccionar el enlace correspondiente a "Eliminar plantilla XML". En caso de que la plantilla seleccionada esté siendo usada en algún portlet, no podrá ser eliminada.

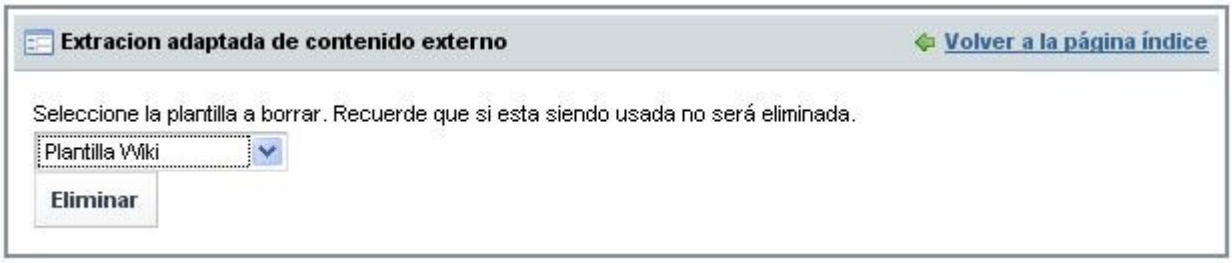

Imagen 43: Eliminar plantilla XML

2.3 – Tablas usadas

Como se observa en la configuración del portlet, es necesario introducir y extraer información para poder hacer uso del mismo. Dicha información se almacena en una base de datos MySQL independiente de la propia de Liferay, llamada "pfcjbb".

Las tablas creadas y el tipo de contenido que almacenan se muestran a continuación:

Nombre de la tabla	**Nombre de los campos**	**Descripción**
dappsinstanciables	id_dapp, nombre_dapp	Almacena los pares nombre-identificador de los Dapps que se han añadido al sistema
dappsinstanciados	id_portlet, id_dapp, nom_var, valor_var	Cada portlet de Dapper posee una configuración propia. Esta tabla relaciona cada portlet mediante el identificador de su instancia con el Dapp que se haya configurado para el mismo, así como el nombre y valor de las variables para ese Dapp
formatodapp	id_portlet, formato_dapp, id_xml	Contiene la información del tipo de contenido configurado (HTML o XML) y el identificador de la plantilla a aplicar (solo XML) asociados a la instancia del portlet.
variablesdapp	id_dapp, nom_var	Indica los nombres de las variables que posee

		un determinado Dapp
xmldapp	id_xml, nombre_plantilla, plantilla_xml	Cada plantilla XML posee un identificador, un nombre y un contenido (la plantilla en sí). Esta tabla relaciona todos estos campos

Tabla 2: Tablas usadas en el portlet de Dapper

2.4 – Clases usadas

A continuación se va a realizar una descripción breve de las clases que componen el portlet de Dapper y aquellas más importantes de las que hace uso.

Clase principal

es.jbb.pfc.dapper.DapperPortlet

El portlet de Dapper posee una única clase principal, que es la encargada de toda la lógica del mismo.

Sus principales métodos son:

doView: Se encarga de cargar la configuración del Dapp y hacer las llamadas correspondientes para recuperar el contenido externo. Posteriormente aplica las transformaciones pertinentes al contenido, según la configuración del portlet. Seguidamente aplica las transformaciones relacionadas con la adaptación a dispositivos móviles y devuelve el resultado al usuario que accede a la página.

doEdit: Contiene toda la lógica de la parte de configuración del portlet. Se encarga de cargar las distintas páginas JSP de configuración a medida que sean necesarias y del acceso a la base de datos para cargar/guardar información de los Dapps y las plantillas XML

Clases auxiliares

En este apartado tenemos 3 clases:

es.jbb.pfc.utils.wurfl.DatosMovil:

La clase DatosMovil es la encargada de obtener todas las características del dispositivo móvil necesarias para la adaptación de las imágenes.

El constructor de DatosMovil necesita que se le pase como parámetro el User-Agent del dispositivo que accede al portlet. En el proceso de instanciación es cuando se accede a WURFL mediante su API de Java y recupera los parámetros del dispositivo, los cuales están accesibles mediante métodos del tipo "*get<parametro>()*"

es.jbb.pfc.utils.adaptation.Adaptacion:

Una vez que el portlet ha recuperado el contenido externo del Dapp que tenga configurado es necesario aplicar un proceso de transformación para adaptar el contenido al dispositivo que haya realizado la petición. Esta clase es la encargada de realizar dicha tarea.

Esta clase contiene 2 métodos:

- mobileAdaptation: Se encarga de analizar el contenido recuperado y modificar las URLs de las imágenes para redirigirlas a Alembik. Es en el servidor Alembik donde se realiza todo el proceso de adaptación de las imágenes.
- deleteTrash: Este método elimina de un documento HTML generado por Dapper, el título principal, algunas imágenes no usadas y los títulos de sección. Estos elementos no son necesarios y afectan negativamente a la visualización desde dispositivos móviles.

es.jbb.pfc.dapper.EliminaEntradaBDServlet

Esta clase es un servlet que debe instalarse en Liferay de forma independiente del proceso de despliegue del portlet de Dapper.

Cuando un portlet de Dapper se configura, se crean una serie de entradas en ciertas tablas de la base de datos. En el momento que eliminamos el portlet es necesario que dicha información sea borrada. El problema es que, si lo eliminamos directamente, perdemos toda referencia al portlet para llevar a cabo el borrado de la información de su configuración almacenada en las tablas. Lo único que podemos hacer es escuchar, en la página JSP, el evento de eliminación del portlet y lanzar un servlet cuando éste se detecte. De esta manera el servlet sí que podrá realizar el borrado pertinente en la base de datos ya que le pasaremos la referencia al portlet justo antes de que éste sea eliminado del sistema. (Ver anexo para la instalación del servlet)

2.5 – Funcionamiento del portlet

A continuación se va a describir detalladamente el funcionamiento del portlet y la interacción con todos los elementos del sistema con el fin de mostrar el proceso completo que tiene lugar desde que un dispositivo accede al portlet de Dapper hasta que se se muestra el contenido adaptado en la pantalla del terminal.

Vamos a suponer que todos los elementos del sistema están en funcionamiento y correctamente configurados. El portal Liferay está en funcionamiento, los Dapps están dados de alta en www.dapper.net, los portlets están correctamente configurados, y el servidor Alembik está activo.

Petición a Liferay

En esta situación, accedemos a una página del portal que contiene un portlet de Dapper. Tras realizar la petición y justo antes de mostrar ningún resultado se ejecuta el método doView() de la clase *es.jbb.pfc.dapper.DapperPortlet.*

Obtención de características

En primer lugar se recupera el User-Agent del dispositivo que accede a la página, por ejemplo un Nokia5800.

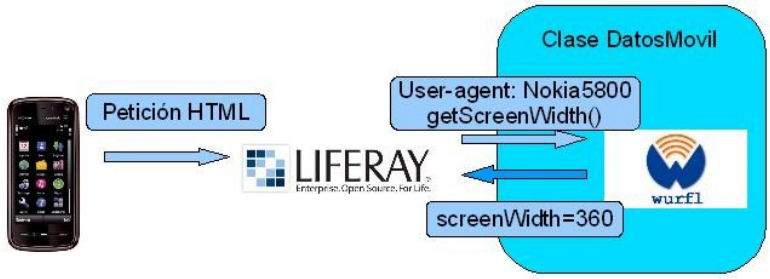

Diagrama 15: Obtención de parámetros del terminal

Seguidamente se procede a la obtención de los parámetros característicos de ese modelo de dispositivo móvil haciendo uso de la clase *es.jbb.pfc.utils.wurfl.DatosMovil*, que a su vez hace uso de WURFL. Asímismo recuperamos toda la información de configuración del portlet de Dapper usando una API para acceder a la base de datos MySQL.

Petición a Dapper y recuperación del contenido

Una vez que tenemos toda esta información debemos acceder al portal de Dapper, que es donde se almacena toda la información del Dapp necesaria para obtener el contenido externo de forma dinámica. Para ello, en código Java, construimos una URL con los parámetros necesarios para la petición y realizamos la misma.

El portal de Dapper, cuando reciba nuestra petición, la analizará y procederá a acceder al contenido. Una vez que tenga todos los contenidos de las páginas que se indicaron al crear el Dapp procederá a analizarlos mismos y, dependiendo de los parámetros que se le hayan indicado en la petición, devolverá al portal de Liferay un documento HTML o XML con toda la información estructurada del contenido recuperado. A partir de este momento no se vuelve a acceder a la página de Dapper hasta que vuelva a haber otra petición en nuestro portal.

Diagrama 16: Extracción del contenido y adaptación de las URLs

Es necesario indicar que toda la información que el portal ha recuperado es texto, por lo que las imágenes y otros recursos no están físicamente en el portal. En su lugar tenemos las referencias, es decir las URL donde se encuentran las imágenes y todo el contenido multimedia.

Transformaciones

A continuación, dependiendo de la configuración del portlet, se procederá a realizar una serie de transformaciones sobre el documento recibido del portal de Dapper.

Si se trataba de un documento HTML se eliminarán partes innecesarias del mismo que Dapper incluye por defecto, como los títulos principales y de sección. En su lugar se mostrará el título que se le haya asignado por configuración.

Si en cambio era un XML sin plantilla asignada, se modifica para que el navegador no intente interpretar el archivo, sino que simplemente se muestre por pantalla.

En caso de que el portlet de Dapper esté configurado para aplicar una plantilla XSL, es en este punto en el que realiza este proceso. Una vez aplicada tenemos como resultado una página HTML que es enviada al dispositivo que realizó la petición.

Adaptación de URL

El portal y el servidor Alembik para la adaptación del contenido no están conectados directamente de ninguna forma. Realmente, lo que el portal realiza es una adaptación de las URL del contenido. Tanto si el documento es un archivo HTML o uno XML, se analizan las URL aparecidas en etiquetas del tipo <img>, que indican la presencia de una imagen en ese lugar. Cada URL de este tipo se sustituye por otra similar, que apunta directamente al servidor Alembik, pasando como parámetros la URL con la ubicación original de la imagen y una serie de parámetros necesarios para la adaptación de la imagen.

De:

```
<img src=”http://www.una_pagina_cualquiera.com/menu/imagen.jpg‟ />
```

A:

```
<img src=”direccion_servidor_alembik/alembik/transcode?
direccion_imagen_original&parametros_de_transformacion”/>
```

El usuario final, lo que recibe realmente cuando accede al portlet es un documento HTML (excepto cuando se trate de un XML sin plantilla). El navegador tiene toda la información en modo texto del contenido, únicamente le queda resolver las peticiones de las URL para obtener las imágenes.

Llamada a Alembik

Para cada una de las URL se le hace una petición al servidor Alembik. Como en la petición aparece la URL original de la imagen, Alembik la resuelve y obtiene físicamente la imagen. A continuación procede a realizar la conversión según el resto de parámetros de la petición y devuelve la imagen al usuario final.

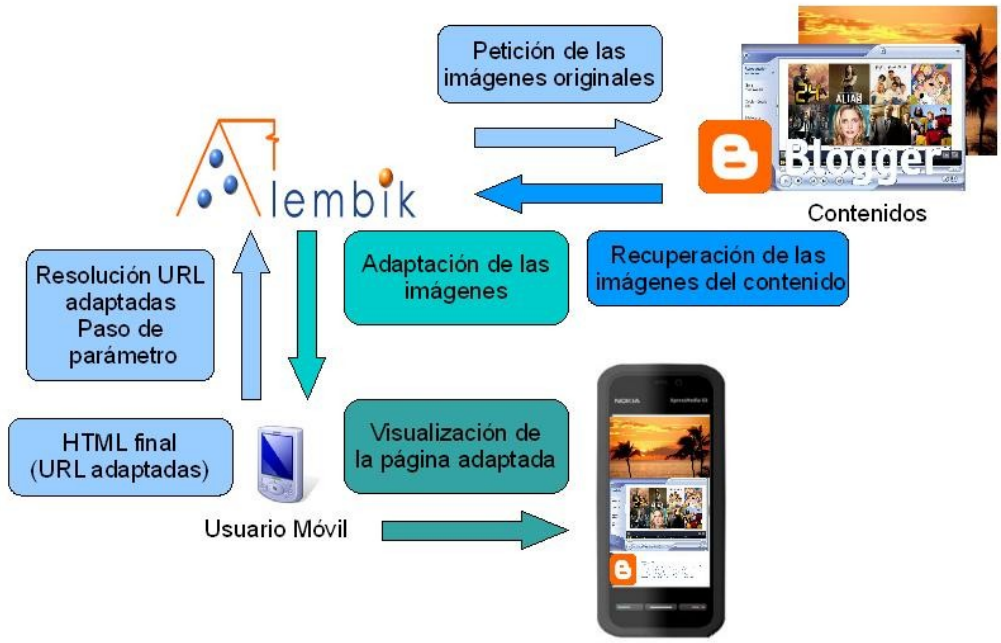

Diagrama 17: Adaptación de las imágenes

Todo este proceso es transparente para el usuario final, ya que virtualmente la imagen en cuestión parece estar almacenada en el portal Liferay.

En la siguiente imagen podemos comparar la visualización de las noticias desde un terminal móvil Nokia 5800. En la primera imagen se muestra el resultado de usar el Dapp de noticias del portal "www.muyinteresante.es" en el portlet del portal Liferay. Junto a esta imagen se muestra el resultado de acceder directamente a la portada de "www.muyinteresante.es". Cabe destacar que para la página de Liferay ocupa unos cientos de KB mientras que la original de la que se extraen las noticias mas de 2 MB.

Imágenes 44 y 45: Capturas de pantalla. Extracción de noticias usando el portlet de Dapper y accediendo directamente al portal original

2.6 – Estructura de archivos

El portlet está formado por la siguiente estructura de archivos. Los archivos que no se han modificado respecto al esqueleto básico original del mismo se han obviado.

En la estructura de archivos del portlet podemos distinguir varios grupos:

- Paquete *es.jbb.pfc.dapper*: Contiene la clase principal del portlet, explicada en el apartado "Clases usadas".
- Archivos XML y .properties de la carpeta WEB-INF: Contienen toda la información de configuración propia del portlet referente al portal.
- Archivos JSP: Constituyen el total de las páginas mostradas en el portlet.
    - view.jsp: Es la única página a la que tiene acceso el usuario final. Muestra el contenido del Dapp en el formato configurado con las imágenes adaptadas.
    - Resto de archivos *.jsp: Son las distintas páginas de configuración del portlet de cara al usuario; elección del Dapp a mostrar, configuración del mismo y añadir/eliminar Dapps y plantillas.
- Archivo build.xml: Útil para el desarrollo del portlet, permite la compilación, generación del archivo .WAR e incluso el despliegue del portlet dentro del portal.

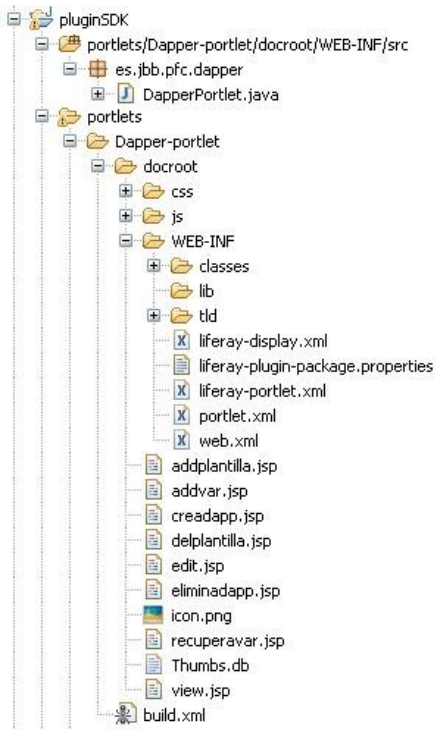

Imagen 46: Estructura de archivos del portlet de Dapper

Además de los propios del portlet, tenemos los siguientes archivos auxiliares:

- Paquete auxiliar *es.jbb.pfc.utils.adaptacion* (ver clases auxiliares)
- Paquete auxiliar *es.jbb.pfc.utils.wurfl* (ver clases auxiliares)
- Servlet contenido en el paquete *es.jbb.pfc.dapper* (ver clases auxiliares)

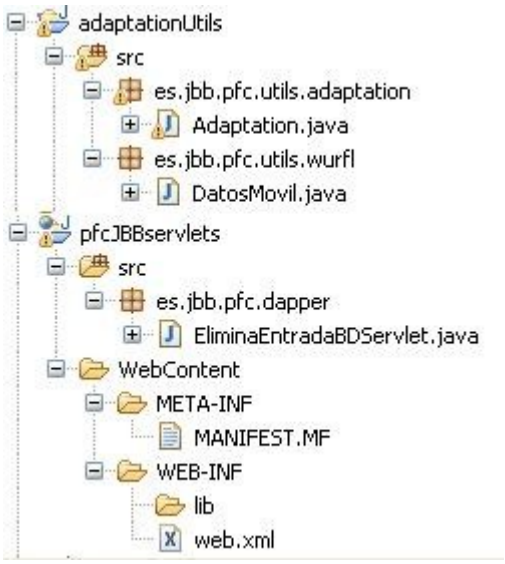

Imagen 47: Archivos/paquetes auxiliares

2.7 – Archivos .properties necesarios

El portlet de Dapper necesita una serie de archivos de configuración que deben de estar ubicados en <*carpeta del servidor*>/common/classes. Estos archivos, con extensión .properties contienen información que no debe estar incluida dentro del código debido a que es posible que deba ser modificada en algún momento. Se optado por usar 4 archivos distintos en vez de uno solo para mantener separados los "bloques" de configuración.

- **configuracion.properties**: Contiene la información de configuración para el acceso a base de datos.
- **dapper.properties**: Contiene todos parámetros para construir la URL de petición de contenido externo.
- **wurfl.properties**: Contiene la ubicación de los archivos wurfl.xml y wurfl_patch.xml.
- **alembik.properties**: Contiene la información de acceso al servidor Alembik y los nombres de los parámetros necesarios para realizar las llamadas al servidor.

3 – Desarrollo del portlet de seguimiento de enlaces o navegación adaptada

3.1 – Descripción

Una característica muy interesante que ofrece Alembik es la posibilidad de realizar una navegación adaptada. Es posible realizar un petición a Alembik para que nos devuelva una determinada página web completa de forma adaptada. A partir de la misma todo el transcurso de la navegación pasará a través del servidor, adaptando todo el contenido posible.

Para aprovechar esta posibilidad se ha desarrollado un portlet que haga uso de esta funcionalidad.

En el portlet se ofrecen 2 posibilidades:

- Por un lado se puede mostrar una lista de enlaces que corresponden a páginas que previamente han sido seleccionadas por el administrador del portal. Esto permite, por ejemplo, la agrupación de enlaces a páginas con una temática determinada.
- La otra posibilidad es que el usuario elija la página web desde la que desea iniciar la navegación.

Todos los enlaces que aparezcan en la versión adaptada de dichas páginas llevarán a su vez a páginas adaptadas. Es por ello por lo que se denomina "navegación adaptada".

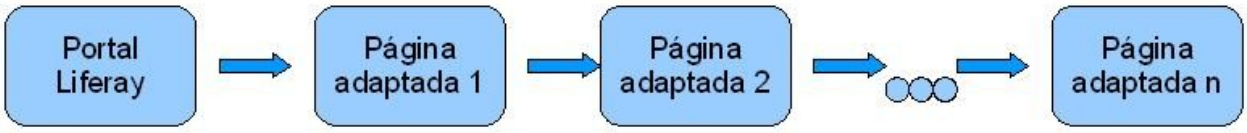

Diagrama 18: Navegación adaptada

Es necesario indicar que durante el proceso de la navegación adaptada se hace un "intento" de adaptación, es decir, no hay garantías de que la visualización en el terminal móvil se realice de forma óptima, puesto que las páginas por las que se desea navegar pueden tener una estructura difícilmente adaptable. En caso de encontrarse con un página de estructura compleja, el resultado de la adaptación puede ser inesperado, pudiendo resultar incluso impracticable.

3.2 – Configuración del portlet

Por defecto, el portlet muestra un cuadro de texto en el que introducir la URL de la página que se desea adaptar.

Imagen 48: Portlet con la configuración por defecto

Si se desea mostrar algunos enlaces por los que empezar la navegación, en necesario hacer click en y seleccionar "Preferencias".

Imagen 49: Selección de "Preferencias"

A continuación aparecerá una lista en la que aparecerán todos los enlaces previamente configurados (ninguno por defecto).

Si se desea añadir un nuevo enlace deberemos seleccionar "Añadir enlace" y aparecerán 2 nuevos cuadros de texto. En el primero se deberá introducir el texto que aparecerá en el enlace, mientras que en "Href" se deberá introducir la URL a donde apuntará el enlace.

En caso de querer eliminar uno de estos enlaces basta con hacer click en "Borrar".

Imagen 50: Configuración del portlet de navegación adaptada

Una vez que se hayan realizado todos las modificaciones deseadas, basta con seleccionar "Enviar" para aplicar los cambios. En la página principal del portlet aparecerán los enlaces creados.

Imagen 51: Portlet de navegación adaptada con enlaces

3.3 – Tablas usadas

No se hace uso de tablas en este portlet. Toda la información de configuración está almacenada en las propiedades del portlet.

3.4 – Clases usadas

A continuación se va a realizar una descripción breve de la clase principal que contiene el portlet de navegación adaptada y la clase auxiliar mas importante de la que hace uso.

Clase Principal

es.jbb.pfc.navegacionadaptada.NavegacionAdaptada

El portlet de navegación adaptada posee una única clase principal, que es la encargada de toda la lógica del mismo.

Sus principales métodos son:

doView: Se encarga de recuperar el User-Agent del dispositivo que accede al portal y de extraer las características propias del mismo. Tras recuperar los enlaces configurados los adapta y los muestra al usuario.

doEdit: Al seleccionar "Propiedades" para configurar el portlet, este método se encarga de recuperar toda información de configuración para ser mostrada.

processAction: Recibe toda la información de la configuración tras realizar cualquier cambio en

la misma y la almacena en las propiedades del portlet.

Clase Auxiliar

En este portlet solo tenemos que destacar una única clase auxiliar.

es.jbb.pfc.utils.wurfl.DatosMovil

La clase DatosMovil es la encargada de obtener todas las características del dispositivo móvil necesarias para la adaptación de las imágenes.

El constructor de DatosMovil necesita que se le pase como parámetro el User-Agent del dispositivo que accede al portlet. En el proceso de instanciación es cuando se accede a WURFL mediante su API de Java y recupera los parámetros del dispositivo, los cuales están accesibles mediante métodos del tipo "*get<parametro>()*".

3.5 – Funcionamiento del portlet

A continuación se va a describir detalladamente el funcionamiento del portlet y la interacción con todos los elementos del sistema con el fin de mostrar el proceso completo que tiene lugar. Esto incluye el acceso al portal Liferay, la carga de la primera página adaptada y la navegación a partir de este punto por los diversos enlaces de las páginas adaptadas.

Vamos a suponer que todos los elementos del sistema están en funcionamiento y correctamente configurados. El portal Liferay esta en funcionamiento, el portlet está configurado con varias direcciones web, y el servidor Alembik funciona correctamente.

Petición a Liferay

En el momento del acceso a la página que contiene el portlet se ejecuta el método doView() de la clase *es.jbb.pfc.navegacionadaptada.NavegacionAdaptada.*

Obtención de características

En primer lugar se recupera el User-Agent del dispositivo que accede a la página.

Seguidamente se procede a la obtención de los parámetros característicos de ese modelo de dispositivo móvil haciendo uso de la clase *es.jbb.pfc.utils.wurfl.DatosMovil*, que a su vez hace uso de WURFL.

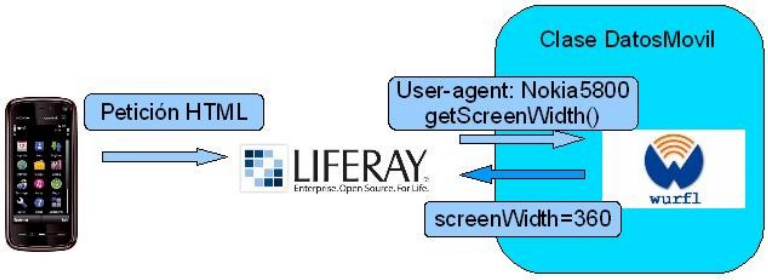

Diagrama 19: Obtención de parámetros del terminal

Adaptación de los enlaces configurados

Posteriormente recuperamos toda la información de configuración del portlet, es decir, los enlaces previamente almacenados y procedemos a su adaptación. En este caso la adaptación de los enlaces consiste en una redirección de la página a la que se pretende acceder. Los enlaces adaptados apuntan directamente al servidor Alembik de forma que se acede al mismo pasando por URL ciertos parámetros, entre los cuales está la URL de la página a la que se pretende acceder.

Como se indicaba en apartados anteriores, es posible introducir manualmente la dirección web de la página a la que se desea acceder. Si se opta por este método el proceso es análogo.

Imagen 52: Acceso al portlet de navegación adaptada

Petición a Alembik

Tras seleccionar cualquiera de los enlaces o introducir manualmente la dirección web, se realizará la petición a Alembik. Mientras se espera la respuesta aparecerá el siguiente mensaje:

Imagen 53: Adaptando contenido

A partir de este punto perderemos toda referencia del portal de Liferay, ya que ha sido el dispositivo, y no el portal quien ha realizado la petición de una página web a Alembik.

El servidor Alembik, tras recibir la petición, procederá a acceder a la dirección web y obtendrá todos los elementos de la misma. Seguidamente se realizará el proceso de adaptación de la página completa.

Durante este proceso de adaptación no sólo se intentan adaptar las imágenes, sino que se reestructura toda la página intentando, en la medida de lo posible, que todo el contenido esté en una única columna, para evitar que sea necesario usar un scroll horizontal en el dispositivo.

Una vez que todo el contenido de la página solicitada esté adaptado, será enviado al dispositivo que realizó la petición.

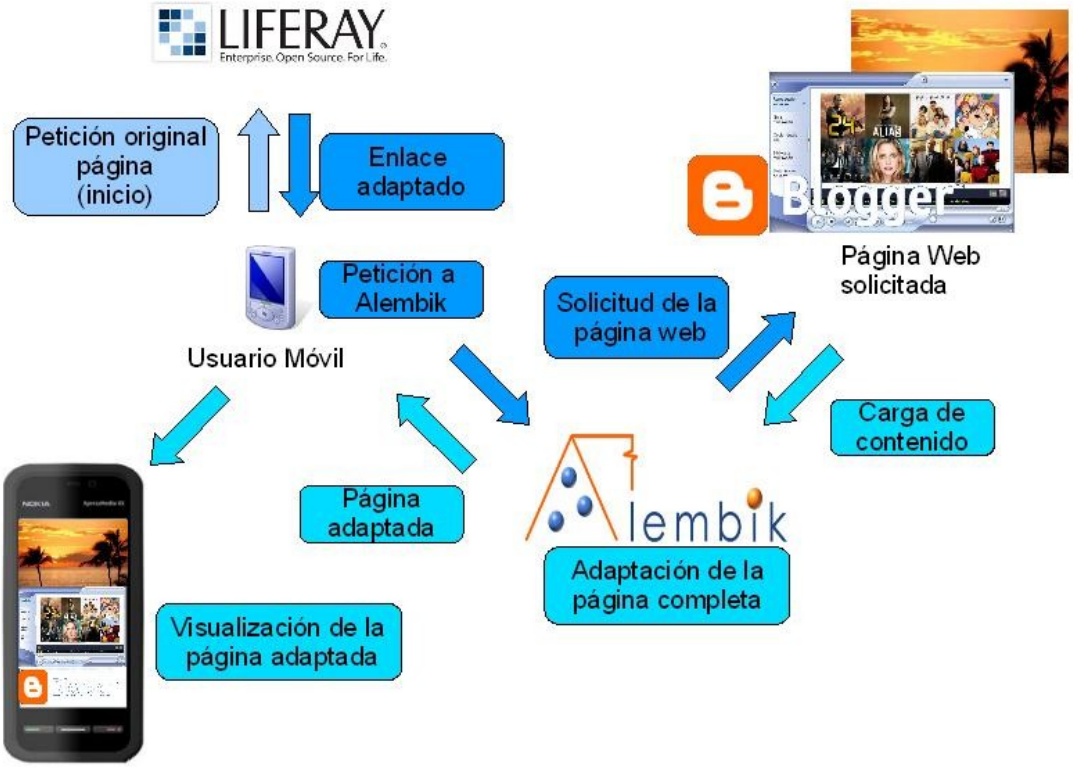

Diagrama 20: Navegación adaptada (primera página solicitada)

Diagrama 21: Navegación adaptada (páginas siguientes)

Una de las adaptaciones más importantes que realiza Alembik es la de los enlaces. Cuando se seleccione un enlace de la página adaptada que implique una redirección a otra página, la petición se realizará de nuevo al servidor Alembik, pasándole los parámetros de adaptación y se repetirá el proceso.

Todos los enlaces de las páginas adaptadas apuntan al servidor Alembik, pero dará la impresión de que el usuario es quien visitará dichas páginas, ya que el proceso de adaptación intermedio se realiza de forma transparente.

A continuación se muestra una navegación de un terminal Nokia 5800 la página principal de "www.20minutos.es" así como del seguimiento del enlace indicado. En el primer caso se muestra el ejemplo de navegación adaptada, mientras que en el segundo se trata de una navegación

"normal". En este último caso se observa claramente que parte del contenido aparece "cortado", por lo que es necesario hacer scroll lateral para visualizar la página correctamente, mientras que en el caso adaptado no es necesario.

Imágenes 54 y 55: Navegación adaptada por el contenido

Imágenes 56, 57 y58: Navegación no adaptada por el contenido

3.6 – Estructura de archivos

El portlet está formado por la siguiente estructura de archivos. Los archivos que no se han modificado respecto al esqueleto básico original del mismo se han obviado.

Podemos distinguir varios grupos de archivos:

- Paquete *es.jbb.pfc.navegacionadaptada*: Contiene la clase principal del portlet, explicada en el apartado "Clases usadas".
- Archivos XML y .properties de la carpeta WEB-INF: Contienen todas la información de configuración propia del portlet referente al portal.
- Archivos JSP: Constituyen el total de las páginas mostradas en el portlet.
    - view.jsp: Es la única página a la que tiene acceso el usuario final, muestra los enlaces configurados previamente y una caja de texto donde introducir manualmente la página web que se desea adaptar.
    - edit.jsp: Es la página de configuración del portlet. En ella se crean los enlaces que se mostraran a los usuarios que accedan al portlet.
- Archivo build.xml: Útil para el desarrollo del portlet. Permite la compilación, generación del archivo .WAR e incluso el despliegue del portlet dentro del portal.

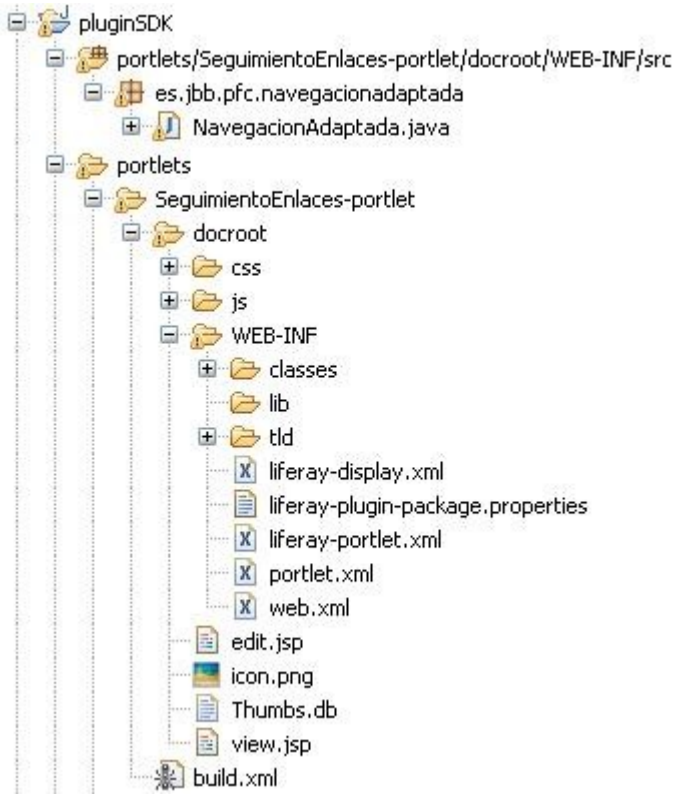

Imagen 59: Estructura de archivos del portlet de navegación adaptada

Además de los propios del portlet tenemos los siguientes archivos auxiliares:

- Paquete auxiliar *es.jbb.pfc.utils.wurfl* (ver clase auxiliar)

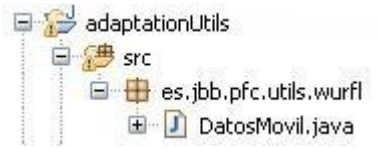

Imagen 60: Archivo/paquete auxiliar

3.7 – Archivos .properties necesarios

El portlet de navegación adaptada necesita tan solo de un archivo de configuración que debe de estar ubicado en <*carpeta del servidor*>/common/classes/. Este archivo, con extensión .properties, contiene información que no debe estar incluida dentro del código debido a que es posible que deba ser modificada en algún momento.

- **alembik.properties**: Contiene la información de acceso al servidor Alembik y los nombres de los parámetros necesarios para realizar las llamadas al servidor