Capítulo 6. Implementación

Resumen

Para demostrar la viabilidad de la arquitectura descrita en el capítulo anterior, se desarrolló una implementación prototípica en el contexto de U-DL-A, específicamente en los ambientes de espacios personales analizados previamente en el capítulo 4. A este prototipo se le denominó PoPS (P*o*rtable *P*ersonal *S*paces) [Castellanos y Sánchez 2003].

En este capítulo se detalla la construcción de este prototipo.

6.1 Selección de tecnologías

Al inicio del desarrollo de PoPS, fue evidente que la selección del lenguaje en el cual se definirían las interfaces genéricas sería un factor muy importante tanto para la definición de las interfaces como para la implementación del convertidor. Dicho lenguaje debería permitir que las interfaces genéricas contaran con las características mencionadas en la Sección 5.2.

- ser interpretado por la mayoría de los micronavegadores.
- permitir un diseño de interfaces con un alto grado de escalabilidad, es decir, en el caso de que una nueva herramienta o servicio se agregara al espacio personal, o se contemplara un nuevo dispositivo, el lenguaje seleccionado debía proporcionar las facilidades para que sólo se agregaran las nuevas especificaciones al diseño original.

Al principio se consideraron tres lenguajes analizados en el capítulo 2:

- UIML
- PersonalJava
- XML

De los tres, el primer lenguaje descartado fue PersonalJava. Si bien Java asegura portabilidad a través de diferentes plataformas de *hardware,* no es un lenguaje declarativo ni de marcado. Además, desafortunadamente en la actualidad son muy pocos los dispositivos móviles que lo soportan: puede instalarse en la mayoría de las PDAs pero sólo algunos de los modelos más modernos (y caros) de celulares lo soportan. Además, los resultados de una encuesta presentados en el Apéndice D reflejaron que sólo el 4.17% de la

comunidad de la UDLA que posee un teléfono celular cuenta con un dispositivo que soporte Java.

Para elegir entre UIML y XML se tomaron en cuenta sus ventajas y desventajas. La principal ventaja de ambos es que, por ser lenguajes de marcado, favorecen la generación de interfaces portátiles, tal y como se explicó previamente en la Sección 2.2.2. Estos dos lenguajes en particular permiten crear interfaces genéricas que pueden convertirse a diferentes lenguajes. En el caso de UIML, la conversión es automática; en el caso de XML, la transformación la realiza un procesador XSLT. Para crear una interfaz en UIML, el desarrollador debe conocer los lenguajes de conversión (WML, VoiceXML, Java o HTML) pues la sección *peers* demanda asociar cada componente de la interfaz genérica con su correspondiente componente en el lenguaje destino; con XML esto no es necesario, pues el mapeo de cada componente lo hace el procesador XSLT tomando como referencia la hoja de estilos XSL. En UIML este mapeo debe especificarse en cada interfaz genérica; en XML sólo una vez, cuando se crea la hoja de estilo. Si se contempla un nuevo lenguaje destino, la sección *peers* de cada interfaz genérica escrita en UIML debe actualizarse; con XML las interfaces genéricas no deben modificarse: únicamente debe crearse una nueva hoja de estilo XSL para el nuevo lenguaje. Hasta el momento UIML sólo contempla cuatro lenguajes de conversión; en cambio, utilizando las tecnologías XSLT un archivo XML puede transformarse en otro documento XML como WML, XHTML o VoiceXML, y también a cualquier otro formato como JSP, PDF o RDF.

Las últimas cuatro ventajas de XML sobre UIML determinaron su elección. Por otro lado, los resultados de una encuesta realizada entre programadores de la UDLA reflejaron que muy pocos conocen UIML en comparación con lo que conocen XML (aunque sea en un nivel bajo). Los resultados de esta encuesta pueden consultarse en el Apéndice E.

En cuanto a los lenguajes específicos a los dispositivos, se determinó que serían WML para celulares y XHTML para PDAs. WML porque es el lenguaje de WAP y todos los navegadores de celulares lo soportan. XHTML por su parte es soportado por todos los *browsers* de PDAs*;* además, lo interpretan algunos micronavegadores de celulares.

Por lo tanto, XML y sus tecnologías asociadas fueron la base para la definición de las

interfaces genéricas y el convertidor de PoPS.

6.2 Componentes de PoPS

Los tres componentes descritos en el capítulo anterior se instanciaron de la siguiente manera en PoPS:

- **Interfaces genéricas:** como PoPS se implementó en los ambientes de espacios personales, a este componente se le denominó **Espacio Personal Genérico (EPG)**. Esta especificación única del espacio personal contiene las interfaces genéricas escritas en XML.
- **Convertidor:** transforma las interfaces genéricas del EPG a XHTML o WML utilizando las tecnologías XSLT y realiza la conversión *por modificación.* Las transformaciones las efectúa un procesador XSLT en conjunto con hojas de estilo XSL. Para cada lenguaje específico se requiere una hoja de estilo diferente, lo que equivale a cada uno de los convertidores descritos en la Sección 5.3.
- **Generador de interfaces:**
    - **Módulo de construcción de interfaces:** se implementó con JSPs que construyen las interfaces que se presentan al usuario.
    - **Módulos de acceso a los servicios específicos del contexto:** se implementaron con JavaBeans que permiten el acceso a diversos servicios de U-DL-A.
    - **Perfiles de dispositivos**: PoPS únicamente contempla dos dispositivos: PDAs y celulares.

En el Apéndice I se encuentra una relación de las herramientas, tecnologías y lenguajes utilizados en cada componente. Las siguientes secciones describen la implementación de cada componente**.**

6.2.1 Espacio personal genérico (EPG)

El EPG es la especificación única del espacio personal que contiene las interfaces genéricas. Estas interfaces están escritas en XML y no dependen de las características de los diferentes dispositivos. Cada interfaz se almacena en un archivo XML diferente (ver

Figura 6.3). Este enfoque permite crear descripciones extensibles: si una nueva herramienta o servicio se agrega al espacio personal, sólo deben agregarse las nuevas especificaciones al diseño original.

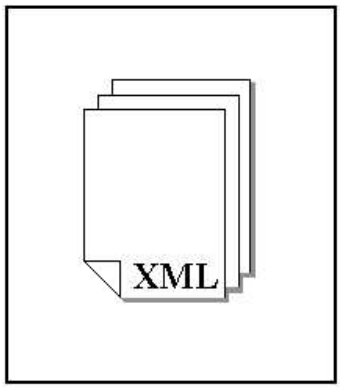

Figura 6.2 Espacio Personal Genérico

En el Apéndice F se encuentra el DTD completo de las interfaces genéricas. La sintaxis de las interfaces es semejante a la de HTML, pero, obviamente, con las restricciones de XML. Por lo tanto, el DTD de estos archivos es un subconjunto del DTD de HTML con dos diferencias significativas: el *tag* inicial es <document> (ver Figura 6.3) y el *tag* para texto es <etiqueta>.

```
<?xml version="1.0" encoding="iso-8859-1"?>
 <document>
   .
     .
       .
     .
   .
</document>
```

Figura 6.3 Ejemplo de un archivo XML del EPG

Se decidió utilizar la sintaxis de HTML porque una encuesta demostró que HTML es el lenguaje de marcado más popular entre los programadores de la UDLA (ver apéndice E). De esta manera los desarrolladores no deben aprender otro lenguaje para crear las interfaces genéricas de PoPS: sólo necesitan el DTD y un editor de XML.

6.2.2 Convertidor

El convertidor realiza las transformaciones utilizando las tecnologías XSLT descritas anteriormente en la Sección 2.3.4. Por lo tanto, está formado por dos componentes: hojas de

estilo XSL y un procesador XSLT (*XSLT engine)*; estos componentes se muestran en la Figura 6.4.

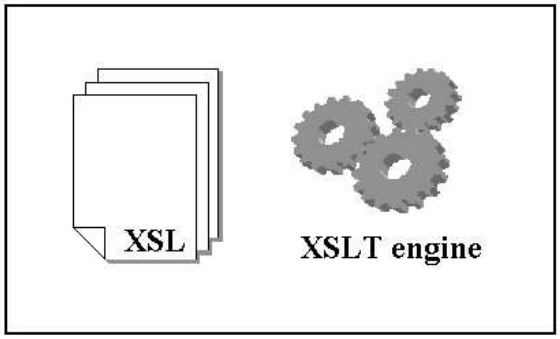

Figura 6.4 Componentes del convertidor

Transforma las interfaces genéricas del EPG a XHTML o WML y realiza la conversión *por modificación* descrita en la Sección 5.5, es decir, las interfaces producidas se almacenan y sólo se regeneran cuando el EPG cambia (ver Figura 6.5).

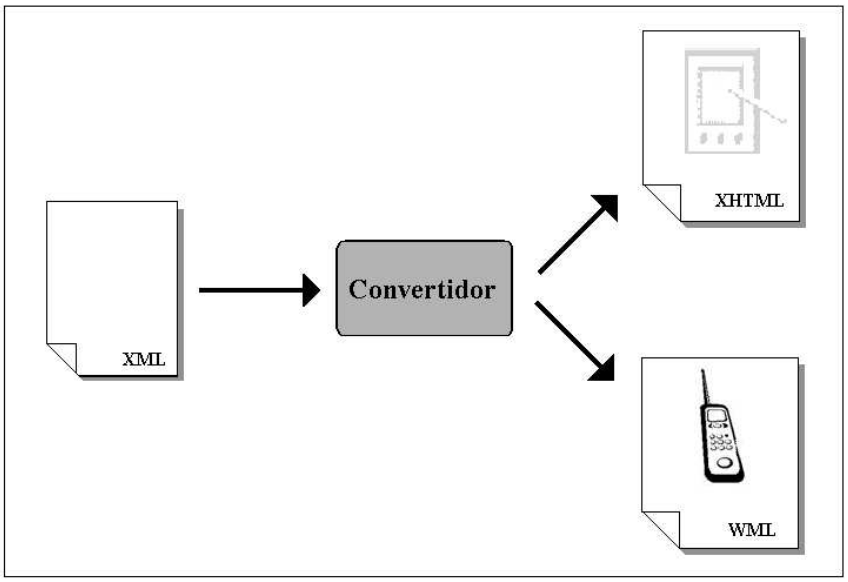

Figura 6.5 Conversión de un archivo XML a XHTML y WML

Cada lenguaje requiere su propia hoja de estilo. Por lo tanto, el convertidor tiene dos archivos XSL: uno para XHTML y otro para WML. La extensibilidad del convertidor se basa en esta característica: para convertir el EPG a un nuevo lenguaje, únicamente se debe diseñar la hoja de estilo correspondiente a ese lenguaje. No es necesario cambiar ningún archivo del EPG. La Figura 6.6 presenta un fragmento de la hoja de estilo XSL para

XHTML.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns="http://www.w3.org/TR/xhtml1/strict">          <-- instrucción XSLT
  <xsl:output method="html" />                            <-- instrucción XSLT
      <xsl:template match="/">                            <-- instrucción XSLT
            <html>                                        <-- tag de HTML
                  <xsl:apply-templates/>                  <-- instrucción XSLT
            </html>                                       <-- tag de HTML
      </xsl:template>                                     <-- instrucción XSLT
                .
                .
                .
</xsl:stylesheet>                                         <-- instrucción XSLT
```

Figura 6.6 Fragmento de la hoja de estilo XSL para XHTML

6.2.3 Generador de interfaces (GI)

Este componente contiene archivos JSPs y JavaBeans (ver Figura 6.7). Los JSPs construyen las interfaces que se presentan al usuario; los JSPs utilizan JavaBeans que permiten el acceso a las bases de datos y servicios de U-DL-A. El servicio de U-DL-A que contempla PoPS es el servicio de administración de agentes, específicamente MAIDL. MAIDL es una aplicación de agentes móviles que visitan las colecciones digitales de bibliotecas de diferentes instituciones y recuperan información de forma transparente bajo la Iniciativa de Archivos Abiertos [Sánchez et al. 2002].

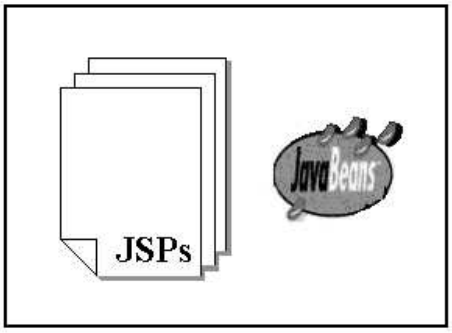

Figura 6.7 Componentes del generador de interfaces

El GI construye las interfaces de los espacios personales adaptando el código producido por el convertidor a las preferencias de los usuarios; estas preferencias constituyen la configuración del espacio personal de cada usuario y se almacenan en una base de datos.
La Figura 6.8 ilustra el diagrama de operación del GI. Realiza tres funciones:

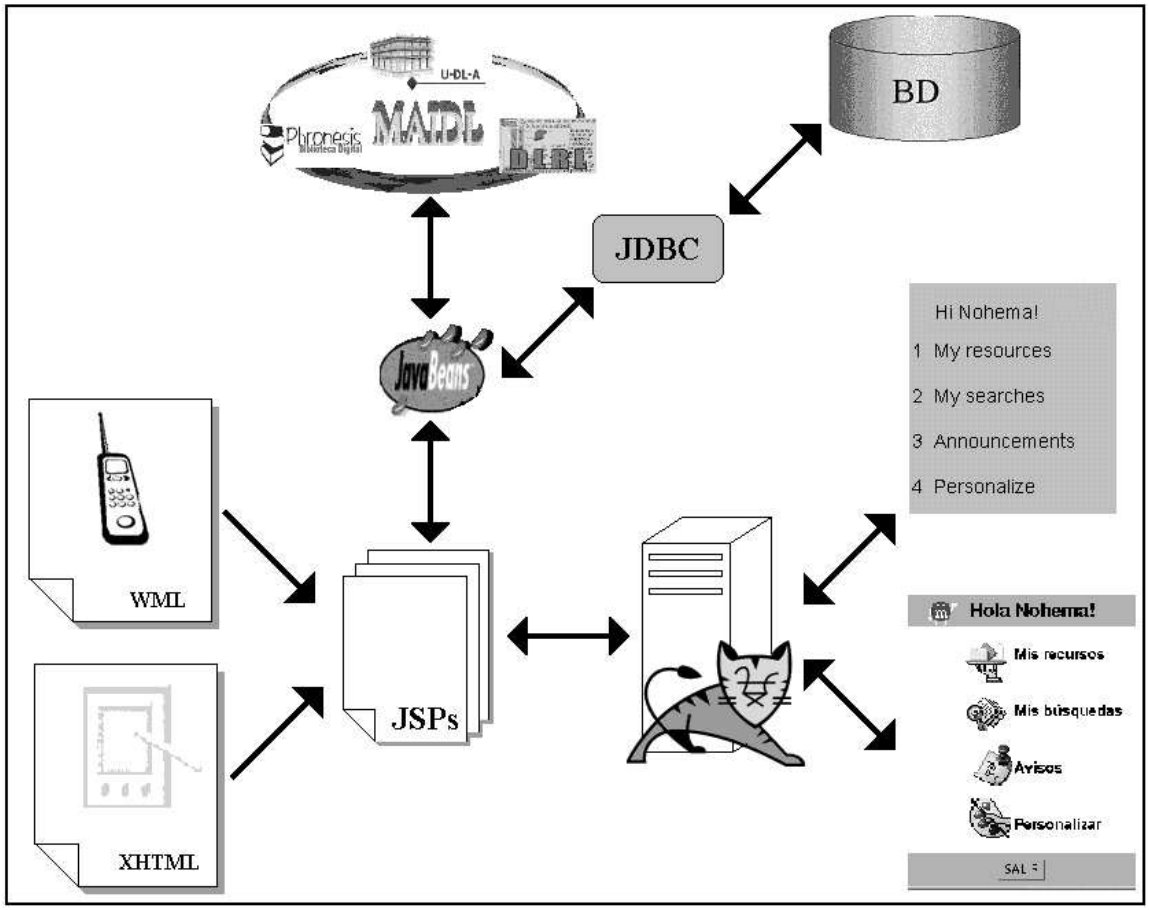

Figura 6.8 Diagrama de operación del generador de interfaces

Construcción de interfaces. Recibe los parámetros del usuario, detecta el tipo de dispositivo y realiza consultas a la base de datos para recuperar la configuración del usuario, las últimas noticias de la biblioteca, los libros en préstamo y las últimas adquisiciones. Tomando en cuenta la configuración del usuario y el tipo de dispositivo, extrae de la interfaz escrita en el lenguaje del dispositivo sólo los componentes requeridos, construye la interfaz correspondiente y la envía al usuario.

Presentación de los resultados de búsquedas. Recibe los parámetros de búsqueda y los envía al sistema MAIDL. Cuando el servicio regresa los resultados, construye la interfaz apropiada para presentarla al usuario. Esta construcción se hace *incrustando* los resultados de la búsqueda en el código del lenguaje del dispositivo.

Actualización de la configuración del espacio personal del usuario. Si los usuarios modifican la configuración de sus espacios personales, actualiza estos cambios en la base de datos.

6.3 Funcionamiento de PoPS

Como se muestra en la Figura 6.9, PoPS funciona de la siguiente manera:

Cuando un usuario invoca a su espacio personal desde un teléfono WAP o una PDA al teclear un URL en el navegador del dispositivo, debe proporcionar su *login* y su *NIP*. El generador de interfaces recibe estos parámetros de entrada así como el tipo de dispositivo y consulta a la base de datos para recuperar la configuración del usuario y los avisos. Después, extrae de la interfaz escrita en el lenguaje del dispositivo sólo los componentes definidos en la configuración y la envía al usuario. El espacio personal se presenta entonces al usuario en el navegador del dispositivo. Cuando el usuario realiza una búsqueda, el generador de interfaces recupera y envía los parámetros de búsqueda al servicio respectivo. Cuando el servicio regresa los resultados de la búsqueda, el generador de interfaces construye la interfaz y la envía al usuario. Si el usuario modifica la configuración de su espacio personal, el generador de interfaces actualiza estos cambios en la base de datos.

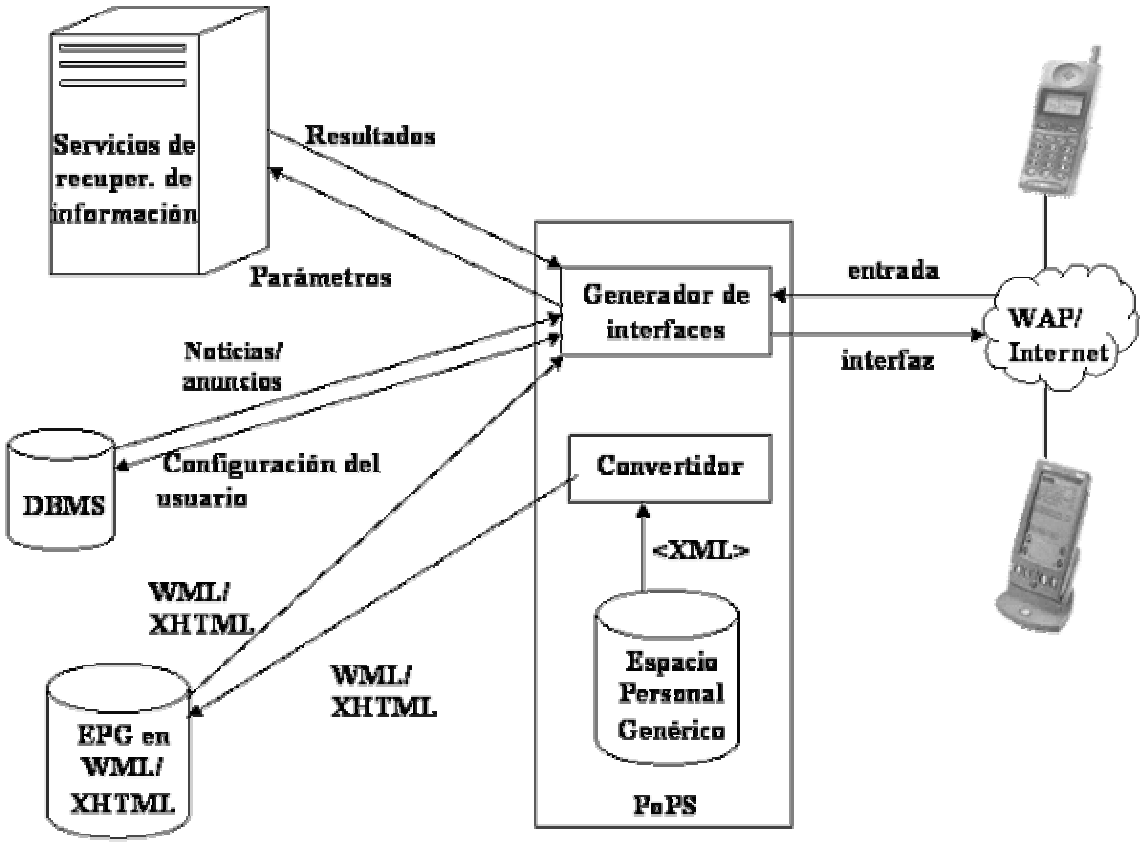

Figura 6.9 Funcionamiento de PoPS

6.4 Acceso a PoPS

Si la universidad contara con un *gateway* de WAP, el acceso a PoPS con el usuario se haría como se ilustra en la Figura 6.10. Los usuarios móviles podrían conectarse a la biblioteca digital desde un teléfono WAP o una PDA. Las PDAs pueden conectarse a PoPS vía Internet; los teléfonos sólo pueden conectarse mediante un *gateway* de WAP.

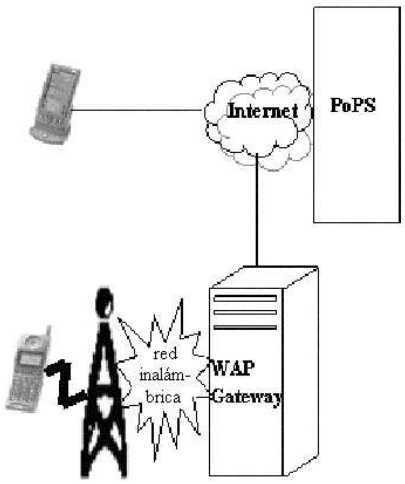

Figura 6.10 Conexión de PoPS con el usuario

El acceso actual a PoPS se muestra en la figura 6.11. Los usuarios móviles se pueden conectar a PoPS desde una PDA o un emulador de WAP. Las PDAs se conectan vía Internet a través de una red inalámbrica; el emulador de WAP se conecta por medio de un *gateway* de WAP que a su vez se conecta a PoPS a través de Internet.

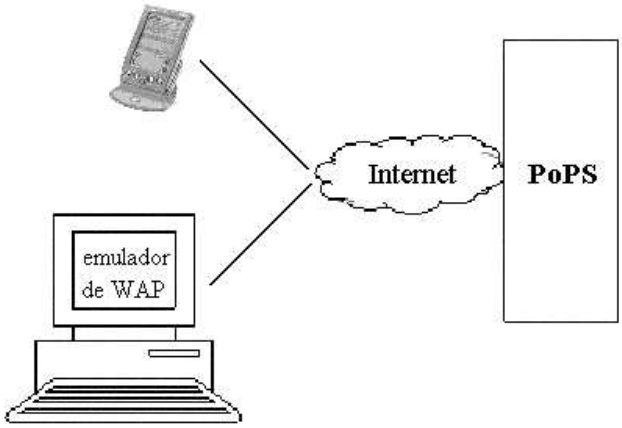

Figura 6.11 Interacción de PoPS con el usuario

6.5 PoPS en el contexto U-DL-A

Como se muestra en la Figura 6.12, PoPS se implementó en el nivel de servicios de la arquitectura U-DL-A, pero tiene implicaciones importantes en el nivel de interfaces de usuario, ya que los espacios personales pueden ahora desplegarse en varios dispositivos.

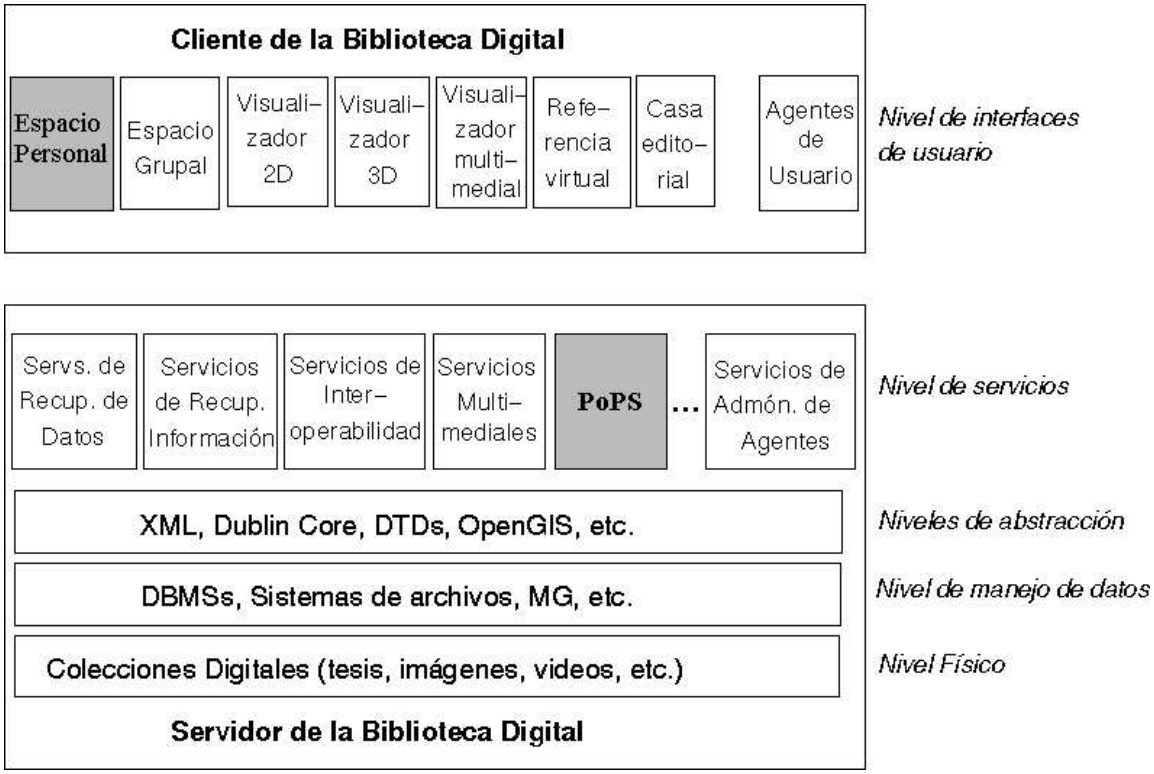

Figura 6.12 PoPS en la arquitectura de U-DL-A

6.6 Diseño de interfaces

Para diseñar las interfaces de PoPS se tomaron en cuenta todos los lineamientos mencionados en la Sección 2.6.2. A continuación se comentan los más importantes:

- *Personalizar*. PoPS almacena en una base de datos la configuración del espacio personal de cada usuario.
- *Identificar las acciones más importantes*. En PoPS no se incluyeron todas las funcionalidades de MiBiBlio 2.0: los únicos servicios disponibles por el momento

son búsquedas, recursos, noticias, libros prestados, y adquisiciones recientes.

- *Desplegar sólo mensajes concisos*. El texto de las interfaces de PoPS es directo y corto.
- *Incluir la opción "Hacia atrás*". Esta funcionalidad se incluye en cada interfaz del espacio personal, al igual que la opción para regresar al inicio y otra para salir de la aplicación.
- *Minimizar la entrada de los datos*. El PoPS el usuario sólo debe proporcionar los siguientes datos alfabéticos: *login*, *NIP* y los nombres de los nuevos recursos que desee agregar. Las demás opciones las puede seleccionar por medio de las teclas de selección, *radio-buttons* y *link-lists.*
- *Evitar redundancia*. En PoPS sólo existe una forma de realizar una misma actividad.
- *Probar inmediatamente el prototipo*. El primer paso en el diseño de las interfaces de PoPS fue el desarrollo de un prototipo el cual fue sometido a pruebas de usabilidad. Los resultados de las pruebas así como las interfaces de este prototipo pueden consultarse en el Apéndice C.
- *Evitar el scroll horizontal.* Las interfaces de PoPS no contemplan este tipo de *scroll*, pero sí el vertical.
- *Considerar el mínimo común denominador capacidades de pantalla.* Las interfaces de PoPS se diseñaron cuatro renglones de 15 a 18 caracteres.

En el Apéndice H se presenta un recorrido por el sistema.