Análisis de tecnologías sw para laboratorios remotos

Javier García Zubía *, Pablo Orduña *, José María Sáenz Ruiz de Velasco **, Inés Jacob Taquet **, Jesús Luis Díaz Labrador ** y Javier Oliver Bernal **
* Dpto. de Arquitectura de Computadores, Automática y Electrónica y Telecomunicaciones
** Dpto. Ingeniería del software
Universidad de Deusto
Avda. Universidades 24, 48007, Bilbao, España
zubia@eside.deusto.es

Resumen

Los Laboratorios Remotos o WebLab son ya un recurso didáctico de primer orden en las facultades de ingeniería, sin embargo en muchos casos adolecen de un pobre diseño sw, tanto en el cliente como en el servidor, lo que degrada su calidad y su utilidad académica. El presente trabajo analiza y selecciona las mejores tecnologías para implementar el cliente y el servidor de un WebLab.

1. Introducción

Actualmente los WebLabs han demostrado sobradamente su utilidad académica no tanto para sustituir a los laboratrios presenciales, como para complementarlos y potenciarlos. En una primera etapa, los WebLabs estaban organizados y promovidos por un laboratorio o departamento, pero su éxito ha conllevado que deba ser la propia universidad la encargada de ofrecer este servicio. Este cambio supone un reconocimiento para los WebLabs, pero también nuevos requisitos (seguridad, accesibilidad, universalidad, etc.) que generalmente no son tomados como esenciales al inicio del diseño, pero que son los que conforman un servicio profesional. Un planteamiento incorrecto es diseñar primero un prototipo que funcione –it runs- y luego añadirle otras funcionalidades, desgraciadamente este planteamiento no es válido y muchas veces acaba en que hay que rehacer la totalidad de la aplicación. Esto es fácilmente asumible por un profesional de la informática, pero no es tan evidente en otros casos.

Por ejemplo, en el mes de noviembre del 2006 la Universidad de Deusto organizó un workshop invitando a una decena de investigadores en el área de los laboratorios remotos [1]. Buena parte de las dicusiones se centraron en los aspectos hw y académicos, pero sin embargo quedó patente que no era lo mismo desarrollar en Java, que en Adobe o en AJAX, y que una mala elección inicial de la tecnología lastraba la calidad del laboratorio remoto, sobre todo en aspectos esenciales como la universalidad, la seguridad y la accesibilidad. Las razones son que mayoritariamente los investigadores tienen un perfil electrónico o de regulación y que las tecnologías web 2.0 son de reciente aparición.

Otro ejemplo clarificador se obtiene al leer el imponente trabajo [2]. En él se analizan más de un centenar de artículos centrados en laboratorios, pues bien, solo uno de ellos [3] relaciona sw y laboratorios remotos, centrándose el resto en el hw y en los aspectos académicos.

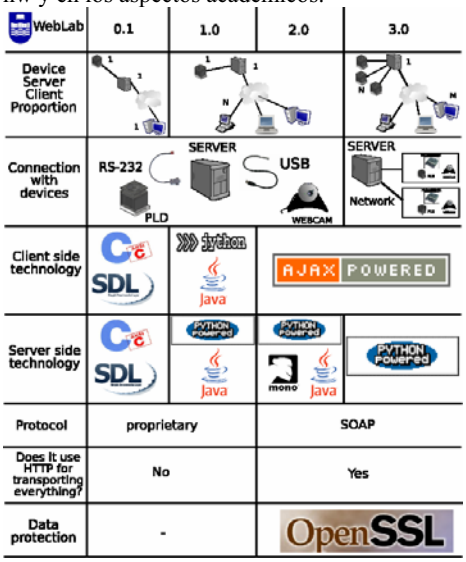

Figura 1. Evolución tecnológica del WebLab-Deusto.

El presente trabajo aprovecha la experiencia del equipo investigador desde el 2001 en el desarrollo de WebLabs (ver Fig. 1 y [4][5][6]) para analizar y seleccionar entre las tecnologías propias del cliente y del servidor la más adecuada para diseñar un laboratorio remoto. El trabajo describe las necesidades de un WebLab, las posibilidades de las diferentes tecnologías para el cliente –aplicaciones de escritorio, ActiveX, Java, Adobe Flash, AJAX y HTML– y las correspondientes al servidor –Python, .NET y Java.

Los apartados 2 y 3 describen y analizan las distintas tecnologías para implementar el cliente y el servidor de un WebLab, seleccionando justificadamente una para cada entorno, AJAX y Python, respectivamente. El apartado 4 refleja las conclusiones del trabajo.

2. Aplicación del cliente

El cliente en un laboratorio remoto es el sw que el usuario va a utilizar para acceder al servicio. Dependiendo del experimento, el cliente puede necesitar enviar un fichero, o recibirlo; puede necesitar ver por WebCam qué está pasando en el laboratorio; puede necesitar interactuar con el experimento; o puede necesitar otros servicios.

La parte cliente debería evitar cualquier restricción innecesaria en el usuario más allá de las funcionalidades, es decir, cualquier alumno con cualquier PC, SO y navegador debe poder acceder al WebLab, ya que lo contrario es difícilmente asumible por la universidad (aunque sí podría serlo por el laboratorio). Así pues, un cliente es mejor cuanto más "fino" es, cuanto más SO soporta, cuanto más accesible es, cuanto menos dependiente es de plug-in, etc. Los WebLabs deben ser herramientas didácticas universales.

2.1. Tecnologías del cliente

Actualmente existe un abanico muy amplio de tecnologías que pueden ser utilizadas para implementar el cliente de un laboratorio remoto, desde la más ligera aplicación web hasta la más pesada aplicación de escritorio. Todas ellas pueden ser clasificadas en dos grupos:

- Aplicaciones de escritorio. Aquellas que se ejecutan en el escritorio del ordenador del usuario.
- Aplicaciones web. Aquellas que son ejecutadas en el navegador del escritorio del usuario.

Una aplicación de escritorio puede ser desarrollada en muchas plataformas (C, C++, Delhi, Java, .NET, Python, etc) y tener muy pocas restricciones, pero es poco portable, más intrusiva que las aplicaciones web, ya que usualmente tienen acceso a todo el sistema del usuario, y necesita de un proceso de instalación. Estas desventajas se compensan con una mayor potencia y complejidad, ya que al no tener las restricciones propias de un navegador y de los estándares asociados a él, estas aplicaciones pueden explotar por cuenta propia recursos como 3D, protocolos binarios específicos para la aplicación, etc. En cualquier caso la calidad y seguridad de una aplicación de escritorio recae totalmente en su diseñador y programador, y esto desde el punto de vista del administrador de sistemas de la universidad suele ser totalmente inaceptable a no ser que ellos hayan sido los desarrolladores; no se debe olvidar que este departamento es el encargado de asegurar y ser responsable de la integridad del servicio informático de la universidad, y que un mal diseño puede comprometerla.

Atendiendo a las razones anteriores, el trabajo se va a centrar principalmente en las aplicaciones web, que junto a las aplicaciones de escritorio, pueden reclasificarse en:

- Aplicaciones intrusivas. Estas aplicaciones tienen los mismos privilegios que las aplicaciones del usuario, por ejemplo pueden acceder al disco duro, pueden leer y escribir ficheros, pueden ejecutar programas, pueden abrir y cerrar conexiones, etc.
- Aplicaciones no intrusivas. Estas aplicaciones garantizan al usuario que pueden ejecutadas sin poner en riesgo su seguridad, ya que no pueden hacer nada que no pueda hacer el navegador en el que están siendo ejecutadas.

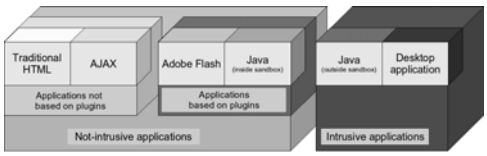

Figura 2. Clasificación de las tecnologías del cliente

Viendo la Fig. 2 se puede decir que cuanto más potente es una tecnología, menos universal y más intrusiva es. Conceptualmente, la mejor tecnología es aquella que cubriendo todos los requisitos del sistema, más universal sea. El resto del trabajo va a analizar las diferentes estrategias para acabar seleccionando AJAX, ya que aunque no es la más potente sí es la más universal, y en el caso de un servicio académico su "universalidad" es un requisito irrenunciable.

Aplicaciones de escritorio

Son las más intrusivas y las que más comprometen la seguridad del usuario. Específicamente las tecnologías Java y .NET pueden ser no intrusivas bajo control del usuario [7] [8]. En cualquier caso su calidad depende claramente del equipo de desarrollo, y por tanto no serán tenidas en cuenta en el análisis final.

ActiveX

Java y ActiveX son probablemente, entre las tecnologías de aplicaciones web, los sistemas más potentes en términos de flexibilidad, pero ActiveX solo está disponible bajo Internet Explorer y sus aplicaciones son por defecto intrusivas, aunque el cliente debe aceptar la ejecución de sw intrusivo. Estas características hacen que las aplicaciones en ActiveX estén más cerca de las aplicaciones de escritorio que de las aplicaciones web.

Java applets

Java es una plataforma potente para desarrollar Rich Internet Applications, RIA. Para poder usar Java, el cliente debe tener instalada la máquina virtual de Java, JRE [9] [10].

Una buena característica de Java es que puede ser instalada en diferentes sistemas operativos, y puede ser embebida en muchos navegadores. La desventaja de JRE es su disponibilidad, que si bien era mucha hace tres o cuatro años, ahora está empezando a decrecer, y no todos los ordenadores la tienen instalada. Otra desventaja es que si el WebLab ha sido desarrollado para JRE 1.5, no funcionará para JRE 1.4, y el usuario tendrá que actualizar la máquina virtual, pero además, y aun siendo extraño, una aplicación hecha bajo JRE 1.3 puede no funcionar bajo JRE 1.5, todo esto exige al usuario tener disponibles varias máquinas virtuales y saber bajo cuál debe ejecutar cada aplicación, y/o exige a los desarrolladores tener diferentes versiones de un mismo WebLab para las diferentes JRE, con el problema de mantenimiento que esto supone.

Es interesante volver sobre la disponibilidad de la JRE, ya que si no está instalada y el usuario está en un cibercafé o en un ordenador universitario sin privilegios de administrador, no podrá instalar la máquina virtual y por tanto no podrá acceder al laboratorio remoto, quedando degradada su universalidad.

Por último cabe destacar que una aplicación Java no es inicialmente intrusiva ya que se ejecuta en la sand box, pero si el cliente debe acceder a algún fichero, entonces la aplicación deberá "abandonar" la sand box, convirtiéndose en una aplicación intrusiva, perdiendo una de sus principales ventajas. La solución a este problema podría implementarse en HTML o JavaScript dentro de la aplicación Java, pero esto complicaría innecesariamente el desarrollo y mantenimiento del sistema.

Adobe Flash

Adobe Flash [11] es actualmente la tecnología líder en RIA. El usuario de una aplicación Adobe Flash debe tener instalado el Adobe Flash Player, el cual interpretará los ficheros en formato swf. Una vez instalado el Adobe Flash Player, las aplicaciones desarrolladas serán no intrusivas, multiplataformas y muy potentes: vídeo, vídeo en tiempo real, sonido, acceso no intrusivo a ficheros, uso de ActionScript, acceso a servicios web, etc. La combinación del potencial de Adobe Flash en servicios web y en animación hacen de esta tecnología una de las más adecuadas para implementar laboratorios remotos.

El uso de Adobe Flash está muy extendido y está disponible en Windows, Linux y Mac OS [12]. En cualquier caso esta disponibilidad es relativa ya que todavía no está disponible para 64 bits, lo que es una clara desventaja. Además, la versión 7 es la única disponible para Linux hasta mediados de enero de 2007, mientras que para Windows ya está desarrollada la versión 9 [13]. Otro problema importante es que solo tiene un gran distribuidor, Adobe, y es por tanto un sw propietario. Por ejemplo, en diciembre de 2006 fue descubierto un error de seguridad en Adobe Reader que comprometía gravemente a Windows [14], de tal forma que si el *site* tenía disponible un pdf, la sesión del cliente podía ser robada. Esto mismo podría pasar con el Adobe Flash Player y

quedar remarcado por la posición aislada de Adobe.

AJAX

En los últimos dos años la tecnología de referencia en RIA es AJAX [15]. AJAX es la combinación de varias tecnologías web ya existentes (XHTML, Javascript, CSS, DOM, etc) con un nuevo componente: XHTMLHttpRequest. Este componente permite llamar a servicios web XML asíncronamente desde Javascript. AJAX es el acrónimo de Asynchronous Javascript And XML.

El aspecto principal de AJAX es que todos los componentes, excepto XMLHttpRequest, son estándares que los navegadores ya soportan. Así, si un navegador soporta el nuevo componente, entonces cualquier aplicación cliente en AJAX se podrá ejecutar en dicho navegador del usuario. Esta característica es muy importante, y hace de AJAX la más portable y universal de las tecnologías explicadas hasta ahora, ya que esta característica no reside tanto en el SO como en el navegador que es el encargado de soportar los estándares. De esta forma, una aplicación implementada en AJAX es directamente ejecutable en un teléfono celular, PDA, etc. siempre que su navegador soporte XMLHttpRequest. Este es el caso del navegador Opera usado por muchos teléfonos celulares [16], el de las últimas versiones del Explorer para Windows CE y el del navegador en código abierto desarrollado por Nokia para sus últimos modelos. Por tanto, un laboratorio remoto implementado en AJAX es accesible desde una multitud de dispositivos, lo que no hace sino otorgarle más universalidad, que es uno de los principales requisitos de este tipo de aplicaciones.

Grandes empresas como Google o Yahoo han desarrollado algunas de sus más famosas aplicaciones en AJAX, por ejemplo Google Maps, Google Mail o Flickr, y por ello AJAX está siendo utilizado en diversidad de aplicaciones, lo que no hace sino aumentar su potencial.

La principal desventaja de AJAX es que no tiene recursos de vídeo y audio. Un laboratorio remoto con bajas exigencias de vídeo y sin audio, bien puede ser desarrollado solo en AJAX, pero si se necesita un buen rendimiento, la aplicación necesita integrar funciones implementadas en Adobe Flash, por ejemplo.

Aplicaciones tradicionales en HTML

Las aplicaciones HTML son aplicaciones web que solo usan estándares como HTML, XHTML, CSS, etc. Estas aplicaciones no tienen por sí mismas capacidad de interacción con el servidor, de vídeo, de audio, etc., lo que supone una clara desventaja en el caso de un WebLab, más allá de la ventaja que supone su perfecta integración en cualquier navegador.

Un aspecto destacable es que es posible desarrollar aplicaciones accesibles en HTML, lo que permite que personas discapacitadas puedan acceder a los servicios web así implementados. Lo anterior no es tan fácil en el resto de tecnologías, excepto para Adobe Flash, cuya versión 6 ayuda al diseñador con funciones para la accesibilidad [17]. Lo anterior puede parecer poco importante, pero los responsables de un laboratorio remoto y la propia universidad no pueden ignorar el alcance de las leyes que favorecen la integración de los discapacitados en la enseñanza [18] [19].

2.2. Análisis y selección de la tecnología del cliente

La pregunta que surge es, ¿cuál es la mejor tecnología para implementar el cliente de un laboratorio remoto? Teniendo claros los requisitos del WebLab a implementar, la pregunta sería ¿pueden cubrirse los requisitos con HTML? Si la respuesta es sí, entonces esta es la tecnología; si la respuesta es no, entonces se repite la pregunta para AJAX; y así sucesivamente para Adobe, etc. Es decir, se debe recorrer la Fig. 2 de izquierda a derecha hasta encontrar la tecnología adecuada para los requisitos.

La Tabla 1 puntúa de 1 a 5 las características de cada tecnología para implementar la aplicación del cliente. Las características utilizadas son:

- Paradigma: ¿Es la tecnología correspondiente el paradigma actual de diseño?
- Multiplataforma: ¿Es ejecutable la aplicación bajo cualquier SO?
- Intrusividad: ¿Hasta qué punto no es intrusiva la aplicación?
- Proveedores: ¿Cuántos proveedores tiene la tecnología de desarrollo?
- Instalación previa: ¿Require la aplicación la instalación previa de un sw, plug-in, máquina virtual, etc?
- Precio: ¿Es gratuita la tecnología?

- Dispositivos móviles: ¿Es ejecutable directamente la aplicación en un teléfono móvil, PDA, etc?
- Flexibilidad: ¿Es utilizable la tecnología en diferentes contextos?
- Accesibilidad: ¿Es adecuada la tecnología para desarrollar aplicaciones accesibles por discapacitados?
- Comunidad de desarrolladores: ¿Existe una comunidad de usuarios y desarrolladores detrás de la tecnología?
- Protocolos: ¿Son diversos y potentes los protocolos que soporta la tecnología?
- Herramientas de desarrollo: ¿Son potentes las herramientas de desarrollo?
- Estandarización: ¿Esta la tecnología basada en estándares?
- Ancho de banda: ¿Cuánto ancho de banda necesita la tecnología?
- Audio y vídeo: ¿Permite la tecnología el uso de audio y vídeo?
- Integración en el navegador: ¿Es la tecnología parte intrínseca del navegador?

Analizando los resultados de la Tabla 1:

- AJAX es la tecnología mejor valorada.
- Mirando solo a los aspectos más destacables, AJAX es también la tecnología mejor valorada (ver Tabla 2).
- Si la aplicación del cliente necesita vídeo y audio de calidad, al menos debe ser usado Adobe Flash.
- Si se necesita interacción con el hw, como es usual en un laboratorio remoto, entonces HTML debe ser descartado.
- Java Applets es similar a Adobe Flash en la mayoría de características, pero está menos disponible en términos de máquina virtual.
- ActiveX no es recomendable para desarrollar WebLabs porque no aporta ninguna ventaja que otras tecnologías no tengan, y sin embargo no es multiplataforma.

Para un WebLab específico, el equipo de desarrollo debe elegir de la Tabla 1 los requisitos más importantes, o añadir nuevos a la tabla. Por ejemplo la Tabla 2 muestra la comparación entre AJAX y Adobe Flash para el desarrollo del WebLab-Deusto. La tecnología más adecuada para el WebLab-Deusto es AJAX.

	Java Applets	Adobe Flash	AJAX	HTML	Active X
Paradigma	***	****	*****	****	*
Multiplataforma	** (1)	**** (2)	***** (3)	***** (3)	* (4)
Intrusividad	*****/* (5)	*****	*****	*****	*
Proveedores	***	*	*****	*****	*
Instalación previa	**	***	*****	*****	*
Precio	*****	*****/** (6)	*****	*****	*** (7)
Dispositivos móviles	** (8)	** (8)	**** (9)	****	** (8)
Flexibilidad	****	****	***	*	*****
Accesibilidad	**	****	**	*****	**
Comunidad de desarrolladores	*****	*****	*****	*****	*****
Protocolos	*****	*****	****	**	*****
Herramientas de desarrollo	*****	***	*****	*****	***
Estandarización	****	***	****	*****	**
Ancho de banda	*****	*****	***	**	*****
Audio y vídeo	***	*****	**	*	*****
Integración en el navegador	*	*	*****	*****	*** (10)
Suma	**56**	**57**	**65**	**64**	**45**

Tabla 1. Análisis de las tecnologías del cliente

1. Mientras la máquina virtual de Java está disponible bajo varios SO, no es posible asumir que esté siempre instalada, sobre todo la última.
2. Es común encontrar Adobe Flash Player instalado (más que la máquina virtual). En cualquier caso, no está disponible para arquitecturas de 64 bits.
3. Es totalmente asumible que todos los usuarios tienen instalados navegadores.
4. Solo se ejecuta bajo un único navegador, el Explorer, y en único SO, Windows.
5. Depende de si se trabaja en la sand box o no.
6. El Player es gratuito, pero el diseñador sí paga por el editor de Adobe, aunque ya hay alternativas gratuitas.
7. ActiveX exige Windows, que no es gratuito.
8. Con restricciones y dependiendo del dispositivo.
9. Es necesario usar navegadores con AJAX, como Opera o Nokia OSS Web Browser.
10 ActiveX es solamente parte integral del Explorer, no del resto de navegadores.

	Adobe Flash	AJAX
Paradigma	****	*****
Multiplataforma	****	*****
Intrusividad	*****	*****
Instalación previa	***	*****
Dispositivos móviles	**	****
Herramientas de desarrollo	***	*****
Audio y vídeo	*****	**
Integración en el navegador	*	*****
Suma	**27**	**36**

Tabla 2. Análisis de las tecnologías del cliente centrado en WebLab-Deusto

Las ventajas que AJAX tiene en términos de disponibilidad, independencia de un único proveedor, buena carga de trabajo e integración en el navegador, hacen de ella la tecnología más adecuada cuando la página web necesita interacción. La principal desventaja de AJAX para WebLabs es que no dispone de capacidad para vídeo y audio de calidad, que sí son aportadas por Adobe Flash o Java. Pero como Adobe y Java son interoperables con AJAX, entonces la integración de módulos desarrollados para audio y vídeo en Adobe Flash o Java en la aplicación es trivial. Por ejemplo, Google Mail es una aplicación AJAX que soporta conversaciones online y usa un módulo en Adobe Flash para generar sonidos cada vez que alguien manda un mensaje [20]. Si el usuario no dispone del Adobe Flah Player, todo el Google Mail funcionará con normalidad, excepto los sonidos de mensaje; esta misma situación se puede dar al diseñar laboratorios remotos.

2.3. Herramientas de implementación del cliente

Cada una de las tecnologías explicadas con anterioridad tiene asociadas al menos dos herramientas de desarrollo. Por ejemplo, para desarrollar aplicaciones de escritorio hay muchas herramientas para cada lenguaje, y en HTML lo mismo; lo mismo ocurre para desarrollar applets de Java; en peor situación está Adobe Flash, pero ya existen algunas herramientas distintas de la propia de Adobe; y finalmente hay docenas de librerías disponibles para integrar AJAX en diferentes plataformas de desarrollo web. Así pues, el equipo de desarrollo puede optar entre multitud de herramientas: Google Web Toolkit, OpenLazslo, AJAX.NET, AJAX para PHP, etc. Las dos primeras van a ser descritas en detalle.

Recientemente la plataforma de código abierto Google Web Toolkit [21][22] ofrece al programador la API con varios widgets y controles escritos en Java, y de hecho el programador desarrolla al completo la web en Java, pero finalmente Google Web Toolkit compilará el cliente en AJAX.

Otra herramienta de desarrollo muy interesante para RIA es OpenLazslo [23], que desarrollada en código abierto ofrece al programador la API en un lenguaje llamado LZX (el lenguaje de programación OpenLazslo consiste en un dialecto de XML que puede crear interfaces de usuario y aceptar métodos y retornos escritos en Javascript). El punto más interesante de OpenLaszlo (ver Fig. 3) es que desde la versión 4 soportará múltiples runtimes, así el diseñador describirá la aplicación en LZX y la compilará en Adobe Flash, en AJAX o en J2ME (Java para dispositivos móviles). Se espera que en el futuro el número de runtimes se incremente [24]. El problema es que de momento esta versión 4 no está disponible excepto como versión beta, el trabajo realizado permite ya compilar en AJAX, y Laszlo Systems ya está trabajando con Sun Microsystems para implementar el compilador en J2ME [25].

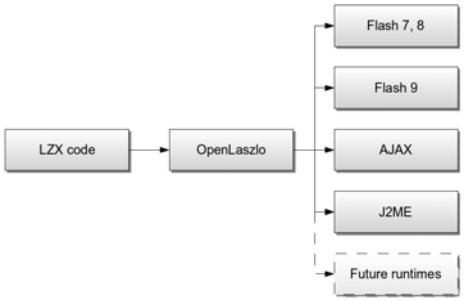

Figura 3. Plataforma OpenLazslo

Cuando la plataforma esté disponible, el equipo de diseño solo trabajará en un lenguaje, LZX, pero podrá obtener tantos clientes como desee, incluso clientes para distintas versiones de por ejemplo Adobe Flash, y así dar servicio a Microsoft, Linux, etc. Así la portabilidad no dependerá del equipo de diseño, sino de la plataforma, y por tanto siempre habrá un cliente para cada usuario, y no un cliente para todos los usuarios.

3. Aplicación del servidor

Aunque una parte muy importante del laboratorio remoto es la parte del cliente y las tecnologías asociadas a él, la mayor parte del esfuerzo y de la potencia residen en el servidor (el cliente es responsable de la universalidad). Si el servidor es bueno y el cliente es pobre, todo el sistema será pobre, pero si el servidor es pobre, no tendrá ninguna importancia la calidad del cliente, el sistema lo será.

El problema reside en que puede que una decisión tomada en el servidor, afecte al cliente, y viceversa. Este situación debe ser tenida en cuenta desde el principio, ya que va a afectar al correcto desarrollo del proyecto. Por ejemplo, si se elige Google Web Toolkit u OpenLaszlo, el equipo de desarrollo necesitará usar Java al menos en una parte del servidor, lo que podría hacer recomendable el usarlo en todo él.

Estas dependencias y la portabilidad no son un grave problema para el servidor, tanto como lo eran para el cliente, simplemente exigen al administrador del servicio utilizar un determinado entorno de explotación.

En cualquier caso un buen diseño en el servidor no depende de la tecnología usada, y hay muchas disponibles; no al menos en el mismo sentido que en el caso del cliente. La tecnología elegida puede hacer más fácil desarrollar un servidor escalable, seguro y mantenible, pero esto solo se logrará si el desarrollador sabe usar la herramienta. La Tabla 3 analiza tres de las tecnologías más usadas para implementar servidores.

	Python	.NET	Java
Multiplataforma	*****	* (1)	****
Precio	*****	** (1)	**** (2)
Comunidad de desarrolladores	*****	*****	*****
Herramientas de desarrolladores	***	*****	*****
Velocidad de desarrollo	*****	***	***
Librerías de servicios web	**	*****	*****
Lenguaje ***	*****	***	**
Robustez	***	****	****
Dinamismo ***	*****	****	****
Suma	**38**	**32**	**36**

Tabla 3. Análisis de las tecnologías del servidor

1. Hay una plataforma popular en código abierto desarrollada por Novell llamada Mono [26], la cual es compatible en muchos sentidos con .NET. Si el WebLab se ejecuta bajo Mono, el costo decrecerá y podrá ser utilizado en diferentes plataformas.
2. Depende de la herramienta y del framework usado.

La tecnología elegida para desarrollar WebLab-Deusto es Python [27] porque es un lenguaje de programación dinámica muy potente, tiene una gran comunidad de código abierto y permite desarrollar prototipos muy rápidamente. Como desventaja de Python cabe destacar que su rendimiento no es muy óptimo. Python es usado internamente por Google, Yahoo, NASA, Industrial & Magic y otras empresas [28], incluso Microsoft y Sun han desarrollado intérpretes de Python para sus entornos .NET y Java, llamados IronPython [29] y Jython [30], respectivamente.

4. Conclusiones

Utilizando la experiencia acumulada desde 2001 como diseñadores del WebLab-Deusto el trabajo ha remarcado en primer lugar la importancia del sw en la calidad final de un laboratorio remoto, sobre todo si este va a ser utilizado como una herramienta didáctica de la universidad, y no solo del laboratorio. En segundo lugar se han analizado diferentes tecnologías para el desarrollo de las aplicaciones cliente y servidor.

Al analizar la aplicación del cliente se proponen como básicas dos características ordenadas por su importancia: la universalidad y la potencia, entendida la primera como la posibilidad de que cualquier usuario o alumno pueda acceder al laboratorio remoto. El análisis concluye con la elección de AJAX como la tecnología más adecuada y potente a la hora de diseñar un laboratorio remoto, seguida de Adobe Flash que destaca por su potencia en el tratamiento de audio y vídeo.

El escenario de diseño del servidor no es tan exigente como el del cliente, ya que aunque su calidad condiciona el del resto del laboratorio remoto, no hay tantas restricciones en cuanto a la universalidad. En el caso de WebLab-Deusto, la elección es Python ya que permite un rápido prototipado, aunque su rendimiento no es muy óptimo.

Referencias

[1] International Meeting on Professional Remote Labs. Bilbao, 12-13 noviembre de 2006. http://weblab.deusto.es

[2] Ma, J. y Nickerson, J.V., 2006, "Hands-on, Simulated, and Remote Laboratories: A Comparative Literature Review", *ACM Computing Surveys*, Vol. 38, Nº 3, Article 3.

[3] Kolberg, S. y Fjeldly, T.A., 2004, "Web Services remote educational laboratories", Proceedings of the International Conference on Engineering Education, Gainesville, FL 1-6.

[4] Garcia-Zubia et al, 2006, "Questions and answers for designing useful WebLabs", *International Journal of Online Engineering,* VOL II, Nº 3, ISSN: 1861-2121, www.ijoe.org, Austria.

[5] Garcia-Zubia et al, 2005, "Evolving towards better architectures for remote laboratories: a practical case", *International Journal of Online Engineering,* VOL I, Nº 2, ISSN: 1861-2121, www.ijoe.org, Austria.

[6] García Zubía, J. y Sáenz Ruiz de Velasco, J.M., 2005, "Diseño de laboratorios remotos virtuales: WebLab", Actas de JENUI 2005, pp: 405-412, ISBN: 84-9732-421-8.

[7] [secJava] http://java.sun.com/javase/6/docs/technotes/guides/security/permissions.html

[8] [secNet] http://msdn2.microsoft.com/en-gb/library/930b76w0(vs.71).aspx

[9] [appletJava] http://java.sun.com/applets/

[10] [downloadJava] http://java.sun.com/javase/downloads/

[11] [adobeMacromedia] http://www.adobe.com/aboutadobe/invrelations/adobeandmacromedia.html

[12] [platformsFlash] http://www.adobe.com/products/flashplayer/productinfo/systemreqs/

[13] [flash9linux] http://blogs.adobe.com/penguin.swf/2007/01/flash_player_9_for_linux_x86.html

[14] [adobeReaderBug] http://www.securityfocus.com/archive/1/455790/30/0/

[15] [ajax] http://www.adaptivepath.com/publications/essays/archives/000385.php

[16] [operaDevices] http://www.opera.com/products/mobile/products/

[17] [flashAccessibility] http://www.adobe.com/resources/accessibility/best_practices/bp_fp.html

[18] [BOCG, 2002] Boletín Oficial de las Cortes Generales, Num. 68-13, 3 de julio de 2002, http://www.congreso.es/public_oficiales/L7/CONG/BOCG/A/A_068-13.PDF

[19] [BOE, 2003] Boletín Oficial del Estado, Num. 289, 3 de diciembre de 2003, http://www.boe.es/boe/dias/2003/12/03/pdfs/A43187-43195.pdf

[20] [flashNeededForGMail] https://mail.google.com/support/bin/answer.py?ctx=%67mail&hl=en&answer=35877

[21] [GWTOpenSource] http://googlewebtoolkit.blogspot.com/2006/12/gwt-13-release-candidate-is-100-open_12.html

[22] [GWT] http://code.google.com/webtoolkit/

[23] [OpenLaszlo] http://www.openlaszlo.org/

[24] [OLLegals] http://www.openlaszlo.org/legals

[25] [Orbit] https://orbit.dev.java.net/

[26] [Mono] http://www.mono-project.com

[27] [Python] http://www.python.org

[28] [PythonSuccess] http://www.python.org/about/success/

[29] [IronPython] www.codeplex.com/Wiki/View.aspx?ProjectName=IronPython

[30] [Jython] http://www.jython.org