ISSN: 1692-7257 - Volumen 1 - Número 13 - Año 2009

J2EE AND. NET PLATFORMS IN THE DEVELOPMENT OF WEB SERVICES

PLATAFORMAS J2EE Y .NET EN EL DESARROLLO DE SERVICIOS WEB

MSc. Luz Marina Santos Jaimes, Ing. Jorge Omar Portilla Jaimes
Ing. John Jairo Méndez

Universidad de Pamplona, Facultad de Ingenierías y Arquitectura
Grupo de Investigación Ciencias Computacionales
Ciudadela Universitaria. Pamplona, Norte de Santander, Colombia.
Tel.: 57-7-5685303 Ext. 164
E-mail: santos@unipamplona.edu.co, {oportillajaimes, jjjpropio}@yahoo.es

Abstract: This paper presents a summary of results comparing the J2EE platform in front of the platform. NET aspects of Web Services, showing the main strengths and weaknesses that can be obtained by choosing between the two platforms.

Resumen: Este artículo presenta una síntesis de los resultados obtenidos en la comparación de la plataforma J2EE frente a la plataforma .NET en aspectos concernientes a los Servicios Web, mostrando las principales fortalezas y debilidades que se pueden obtener al escoger entre las dos plataformas.

Keywords: Web Services, J2EE, .NET, Net security, Performance.

1. INTRODUCION

Actualmente las plataformas J2EE y .NET son las principales alternativas para desarrollar soluciones basadas en arquitectura SOA; ambas ofrecen lo necesario para llevar a cabo de forma satisfactoria soluciones distribuidas. Existen dudas a la hora de escoger entre J2EE y .NET, no se conocen las ventajas reales que cada plataforma puede aportar sobre ciertos requerimientos particulares.

El presente artículo presenta las fortalezas y debilidades de las plataformas J2EE y .NET teniendo en cuenta los criterios en el plano de los servicios Web como son: seguridad a nivel de mensaje, portabilidad, interoperabilidad y desempeño, de tal forma que sirva de base en la toma de decisión sobre cual plataforma elegir en un desarrollo distribuido de software.

1.1 Servicios Web

Una aplicación software identificada por un URI, cuyas interfaces se pueden definir, describir y descubrir mediante documentos XML. Un Servicio Web soporta interacciones directas con otros agentes software, utilizando mensajes XML intercambiados mediante protocolos basados en Internet.

Los Servicios Web están basados en los siguientes estándares (Romero, 2004):

- **XML**. Es un estándar para la representación estructurada de datos y para crear etiquetas, definido por el W3C[1].
- **SOAP**. Es un estándar definido por la W3C, que permite realizar peticiones y recibir respuestas a Servicios y Clientes Web en formas de mensajes utilizando el estándar XML.

[1] Consorcio World Wide Web. Guía Breve de Servicios Web. http://www.w3c.es/.

- **WSDL**. Describe la interfaz externa de un servicio Web y la forma de cómo utilizarlo. Se basa en XML.
- **Registro UDDI**. Es un archivo XML, que describe el proveedor y los servicios que ofrece.

En la figura 1 se observan los pasos al requerir un servicio Web (Molinari, 2004):

1. El cliente pregunta al registro UDDI para ubicar un servicio.
2. El registro le indica al cliente un documento WSDL.
3. El cliente accede al documento WSDL
4. WSDL provee lo necesario para que el cliente pueda interactuar con el servicio.
5. El cliente envía un requerimiento usando SOAP
6. El servicio Web retorna una respuesta SOAP.

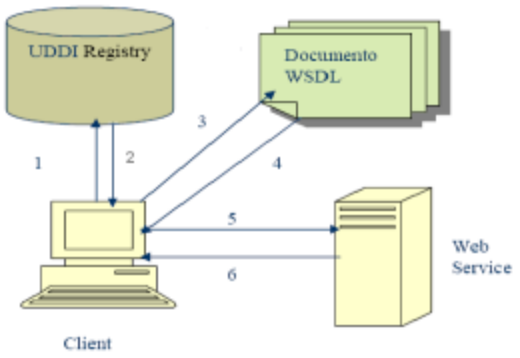

Fig. 1. Dinámica de un Servicio Web.

1.2 Generalidades de la Plataforma .NET

.NET es un proyecto de Microsoft para el desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma de hardware y que permite un rápido desarrollo de aplicaciones. Basado en esto, la empresa desarrolla una estrategia horizontal integrando todos sus productos, desde el sistema operativo hasta las herramientas de mercado (Carpe, 2001).

La arquitectura .NET se divide en tres niveles o capas: presentación, negocios y acceso a datos (Garrido, 2006).

El *Framework* de .NET es un conjunto de servicios de programación diseñados para simplificar el desarrollo de aplicaciones en el entorno altamente distribuido de Internet. .NET *Framework* incluye CLR y bibliotecas de clases (Samper, 2005). Cuando se compila cualquier código fuente soportado por .NET en realidad se compila a MSIL. Para poder ejecutar MSIL se debe convertir mediante un compilador JIT o *jitter* a código de máquina que se ejecuta en la plataforma del cliente (Aquino, 2002).

1.3 Generalidades de la Plataforma J2EE

J2EE es la arquitectura creada por Sun para el desarrollo de todo tipo de aplicaciones para empresas y usuarios en general. Sun lo define como un estándar para el desarrollo de aplicaciones empresariales multicapa. A diferencia de la plataforma .NET, J2EE solamente soporta el lenguaje Java. Las aplicaciones Java están típicamente compiladas en un lenguaje intermedio llamado *bytecode*, que es normalmente interpretado o compilado a código nativo mediante la JVM. La JVM se sitúa en un nivel superior al hardware del sistema, y este actúa como un puente que entiende tanto el *bytecode*, como el sistema sobre el que se pretende ejecutar.**]**.

Las aplicaciones realizadas en J2EE se pueden dividir en 2, 3 o más capas. En la primera capa es donde se encuentran las interfaces como páginas JSP, *Servlet* y *Applet*. En la segunda capa se encuentra los componentes EJB, los servicios Web y toda la lógica de negocio. La última capa es para acceder a Bases de Datos. Cada una de estas capas pueden subdividirse en subcapas (Garrido, 2006).

1.4 Criterios de comparación

*Seguridad a nivel de Mensajes***.** En el momento de tomar decisiones de negocio sobre que tecnologías implementar y de seleccionar la plataforma de desarrollo de las mismas, es fundamental considerar las amenazas en la seguridad de la información. Ya que, garantizar la seguridad conforma un objetivo fundamental en la actualidad para implementar los WSs (OCDE, 2002). El hecho de que se haya tomado la seguridad en el desarrollo de los servicios Web como criterio de comparación, radica en que la seguridad constituye uno de los principales desafíos para los sistemas distribuidos.

Tanto los servicios Web como los sistemas o aplicaciones Web necesitan el mismo tipo de seguridad en el nivel de transporte. No obstante, los servicios Web al utilizar mensajes SOAP basados en el estándar XML generan un nivel adicional de seguridad, ya que, cualquier persona podría ser capaz

de interceptar un mensaje alterando la autenticidad, integridad o confidencialidad de este. Por lo tanto, es necesario implementar algún tipo de seguridad en los mensajes de peticiones y de respuestas. Para solucionar problemas de seguridad en los servicios Web han surgido un conjunto de especificaciones, que definen una serie de extensiones para el protocolo SOAP (Gutiérrez, et al, 2005).

La *portabilidad* de una aplicación se basa en la facilidad para ser ejecutada en distintos entornos lógicos o físicos; sean sistemas operativos o hardware, obteniéndose así una independencia de la máquina o sistema cómputo en la cual se va a ejecutar el programa. Al considerarse que un software o aplicación es portable se presume que el sistema puede migrarse a otra plataforma a un costo razonable.

Se ha tomado la Portabilidad como un criterio de comparación, ya que, es muy probable requerir desplegar WSs en sistemas operativos diferentes, de donde originalmente fueron implementados.

Dentro la arquitectura SOA la *interoperabilidad* es tal vez el principio más importante. Los WSs como tecnología de implementación de esta arquitectura deben ofrecer importantes beneficios de interoperabilidad, y permitir la ejecución en múltiples plataformas de software y arquitecturas de hardware (Cheng, et al, 2007; Skonnard, 2007).

Se tomó este criterio de análisis porque es en la interoperabilidad, donde se puede encontrar el mayor beneficio de los WSs, al interconectar sistemas distribuidos implementados en diversas plataformas.

El *desempeño* es el rendimiento (R) en términos de tiempo de respuesta (t), por cada usuario de un sistema (S) sin una determinada carga de trabajo T. R es una respuesta de S a un estímulo de T, y puede ilustrarse con la representación funcional R(S, T) (Bostad, 2006)..

Actualmente existe la tendencia a implementar gran cantidad de aplicaciones con respuestas en tiempo real. Es decir, se requiere que las aplicaciones respondan a sus clientes en el menor tiempo posible y de la mejor forma. Lo anterior, hace que se haya tomado el desempeño como criterio fundamental, para el análisis comparativo entre J2EE y .NET. Pues, tal análisis determina cuál de estas dos plataformas presenta mayores ventajas al respecto.

2. ANALISIS COMPARATIVO ENTRE J2EE Y .NET

Para realizar la comparación se escogió NetBeans 6.5 IDE (para J2EE) y Visual Studio 2008 (para .NET). Lo anterior debido a que, son herramientas, robustas que incorporan lo necesario para el desarrollo y consumo de los servicios Web. Es importante resaltar que NetBeans 6.5 y VS 2008 son las últimas versiones de cada IDE.

2.1 Seguridad a nivel de mensaje

La plataforma J2EE a través de sus servidores de aplicaciones, soporta varias especificaciones de seguridad, entre estas especificaciones se encuentra *WS-Security*.

WSIT es una especificación de tecnologías abiertas de servicios Web pensada para interoperar en forma transparente con tecnología .NET. WSIT trata aspectos claves de interoperabilidad como: arranque y configuración, mensajería confiable, manejo de transacciones y seguridad a nivel de mensajes.

WSIT en el área de seguridad implementa un conjunto de estándares publicados por el consorcio OASIS.

Metro[2] es un *Stack* para servicios Web propuesto por Sun, el cual contiene a JAX-WS (lo que incluye a JAXB, JAXP, StAX, SAAJ; utilizado para el mapeo de XML a objetos y viceversa) y WSIT que garantiza la interoperabilidad, fiabilidad, seguridad, y transaccionabilidad de los servicios Web. GlassFish es un servidor de aplicación para J2EE que trae soporte para Metro. (Domínguez, et al).

NetBeans IDE desde la versión 5.5 soporta *WS-Security*, ya sea directamente a través de la herramienta o utilizando un Plugin como WSIT. Desde la versión 6.1 de NetBeans soporta la implementación de servicios Web con el Stack de Metro.

Visual Studio desde la versión 2003 soporta la especificación *WS-Security* utilizando el Plugin WSE. En la versión Visual Studio 2008 se pueden utilizar servicios Web WCF. También se pueden crear aplicaciones WCF utilizando Visual Studio 2005, ya que, Microsoft ha liberado un paquete de extensiones y un Kit de desarrollo que puede ser utilizado en esta versión (Domínguez, et al, 2005).

[2] Especificaciones de Metro 1.4 https://jax-ws.dev.java.net/guide/Metro_Specifications.html

WSE es un complemento de Visual Studio .NET que permite satisfacer los requisitos empresariales, dicho de otra forma es la implementación de Microsoft para servicios web avanzados. Desde la versión WSE 2.0 incluye el estándar *WS-Security* el cual permite a las organizaciones exponer servicios de forma segura que puedan ser empleados en ambientes heterogéneos. Después de la versión WSE 3.0 fue embebido por el WCF.

WCF es el modelo unificado de programación para aplicaciones distribuidas SOA sobre plataformas Windows y pensado para interoperar en forma transparente con tecnología J2EE (Domínguez, et al, 2005).

La seguridad ha sido uno de los aspectos más desarrollados en los últimos tiempos en el área de servicios Web. Sun y Microsoft se agruparon para trabajar en este tema, buscando con ello que además de que la seguridad sea interoperable, sea estándar en cualquiera de las dos plataformas tecnológicas.

Lo anterior evidencia que la seguridad ofrecida tanto por J2EE como por .NET abarca el mismo conjunto de especificaciones. Por ejemplo, ambas plataformas soportan las especificaciones de seguridad a nivel de mensaje como *WS-Security*, esto quiere decir que si en J2EE se implementa la encriptación del mensaje SOAP utilizando la especificación de *WS-Security* y un intruso logra desencriptar el mensaje SOAP, entonces también es posible violar la seguridad en .NET pues utiliza la misma especificación de *WS-Security* para encriptar mensajes SOAP.

En el momento de implementar ambas plataformas muestran cierto grado de simplicidad, eliminando en ese aspecto la ventaja de una plataforma frente a la otra.

2.2 Portabilidad

Se consigue una mejor portabilidad con la plataforma J2EE donde existe la máquina virtual de java para la mayoría de los Sistemas Operativos en cada una de sus versiones.

Generalmente los proyectos de servicios Web realizado en estos IDE generan una extensión (.WAR) que contiene toda la aplicación; este archivo se puede transportar a diferentes sistemas operativos y es posible desplegarlo para ser accedido mediante un cliente, utilizando algún servidor de aplicación que soporte las característica con que se creó el servicio Web.

La portabilidad de .NET a través del PE (Ejecutable Portable que contiene MSIL y los metadatos requeridos) es mucho menor a la obtenida con J2EE, ya que, no existen versiones del CLR para la mayoría de los sistemas operativos, solo para las versiones de Windows.

A través de CLR se consigue que .NET sea una plataforma de ejecución independiente del lenguaje, o comúnmente conocido como multilenguaje, lo que permite integrar desarrolladores de distintos perfiles. Aunque esto en ocasiones presenta ciertas ventajas en otras se convierte en una desventaja, ya que, mantener un proyecto en múltiples lenguajes es costoso. Si una aplicación está realizada en varios lenguajes se necesitan expertos en cada lenguaje para entenderla y mantenerla, aumentando los costos. No obstante, en la plataforma .NET las librerías o clases son comunes a los lenguajes, con lo que los desarrolladores no tienen que aprender una nueva librería cuando cambian de lenguaje.

Tabla 1. Pruebas a NetBeans y Visual .NET

Tipo de Prueba	Nombre	Tiempo [s]
Prueba de arranque	NetBeans	30.1
Prueba de arranque	Visual .NET	3.7
Prueba de arranque Servidor	PLAY GlassFish	42.7
Prueba de stop del servidor	STOP GlassFish	14.9
Prueba de arranque Servidor	PLAY IIS	7.4
Prueba de stop del servidor	STOP IIS	5.6
Prueba para crear un WS	NetBeans	53.7
Prueba para crear un WS	Visual .NET	8.7
Compilación y despliegue del WS	NetBeans	15.4
Compilación y despliegue del WS	Visual .NET	9.2
Número de instancias	NetBeans	1.0
Número de instancias	Visual .NET	43.0

En la tabla1 se presenta el tiempo promedio de cada una de las pruebas realizadas a NetBeans 6.5 para J2EE y Visual Studio 2008 para .NET. Para las diferentes pruebas de tiempos se realizaron tres mediciones.

En las diferentes pruebas de tiempos realizadas a los dos IDEs, demuestra una gran ventaja Visual Studio frente a NetBeans, en el momento de desarrollar operaciones básicas para construir Servicios y Clientes Web. Cabe resaltar que en la prueba en donde se mide el tiempo promedio que tarda en crearse un WS en cada IDE, es donde se aprecia una de las mayores ventajas de Visual Studio Frente NetBeans.
La anterior afirmación se basa en el hecho de que es posible realizar cinco servicios Web en Visual Studio mientras que en NetBeans se hace solamente un Servicio. Además, iniciar Visual Studio requiere un tiempo promedio mucho menor que NetBeans, lo cual genera un ahorro en el tiempo para realizar cualquier tipo de aplicación. Implicando con ello, un consumo mucho menor de los recursos del sistema de computo en el momento de realizar un arranque.

Un aspecto relevante de Visual Studio es que Microsoft solo tiene disponible esta herramienta para Windows, obligando así a desarrollar el Servicio Web bajo este IDE, a diferencia de Sun que posee diferentes IDEs de J2EE, de los cuales existe disponibilidad en diferentes sistemas operativos.

2.3 Interoperabilidad

Se inició este análisis realizando un prototipo de Servicio y Cliente Web que retorna un *String*, lo cual no presentó problema alguno. Al igual que otras pruebas realizadas con otros tipos de datos tanto simple (como: *int*, *float*, *boolean*, u otros) como complejos (*arraylist*, *vector*, otros). El único inconveniente se generó al realizar un WS en la plataforma .NET, al momento de retornar un *DataSet*, aplicado a la base de datos del prototipo .NET. El *DataSet* de ADO.Net representa un conjunto completo de datos, incluyendo las tablas que contienen, ordenamiento y restricción de los datos, así como las relaciones entre las tablas.

El tipo de dato retornado por el WS es un esquema donde no se conoce su estructura interna debido a que está acompañado por la etiqueta *any* (que significa cualquier). Solo se podrá conocer la estructura interna en tiempo de ejecución cuando se realice la búsqueda en la base de datos, añadiendo así los datos devueltos de acuerdo a la consulta realizada. Al utilizar *DataSet* se generan varios problemas, uno de ellos es que el desarrollador del cliente no conoce la estructura del objeto que el método devolverá, por consiguiente no podrán generar un *proxy* muy útil.

Otro problema, es la gran cantidad de datos que se envía al utilizar *DataSet*. Cuando se trata de acceder desde J2EE utilizando NetBeans genera error.

Al utilizar en .NET en el desarrollo de Servicios Web el tipo de dato *DataSet* se pierde o se dificultad hasta cierto punto la interoperabilidad con J2EE, ya que este tipo de datos es especifico de .NET, aunque se trabaje con un esquema especifico preestablecido. Es posible que se genere un error cuando se produzca algún cambio en el WS o en la consulta. La forma posible de consumir este WS en J2EE consiste en recibir el mensaje SOAP enviado por el Servicio .NET y convertirlo a un tipo de dato especifico de JAVA, utilizando alguna API para manipular documentos XML

2.4 Desempeño

A continuación se presenta el resumen de la prueba realizada al servidor GlassFish v2 de J2EE, y al Servidor de .NET IIS 5.1, utilizando la herramienta AdventNet QEngine[3] al prototipo de Servicio y Cliente Web J2EE y .NET propuesto.

La prueba se realizó con 25 usuarios virtuales con repeticiones por usuarios de 500. Las URLs que se utilizaron fueron las del prototipo propuesto.

J2EE recibió 12500 peticiones que equivalen a los 25 usuarios virtuales activados con repeticiones por cada uno de 500, el cual respondió todas las peticiones generando respuesta para cada usuario. A diferencia de .NET que aunque respondió 12500 peticiones, recibió más de esa cantidad, ya que, se generaron algunos errores de denegación de IP (error 403), que trae por defecto IIS para defenderse de ataque DoS. En otros casos, el tiempo de respuesta en algunas peticiones expiró, en respuesta a lo cual, el AdventNet QEngine generó peticiones adicionales, para completar las 12500 peticiones planteadas inicialmente.

[3] QEngine-Load Testing and Functional Testing Tool http://www.adventnet.com/products/qengine/index.html

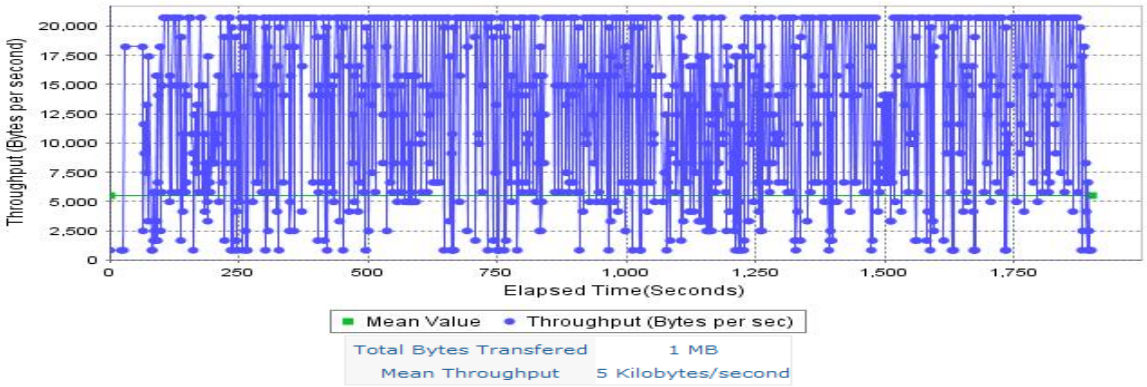

Fig. 2. Datos recibidos por Usuarios en J2EE.

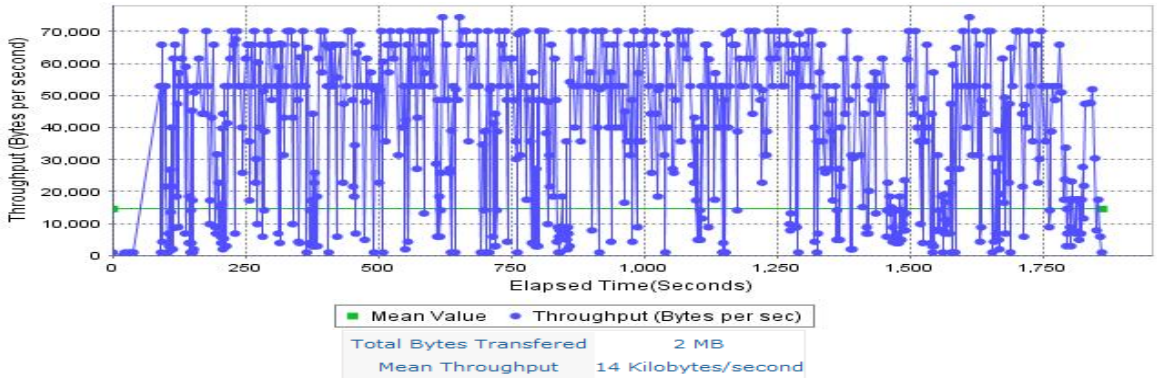

Fig. 3. Datos recibidos por Usuarios en .NET.

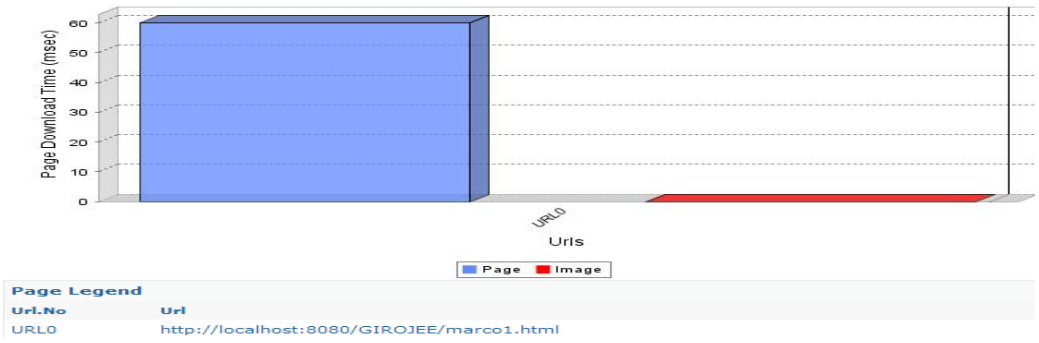

Fig. 4. Tiempo de descarga por páginas en J2EE

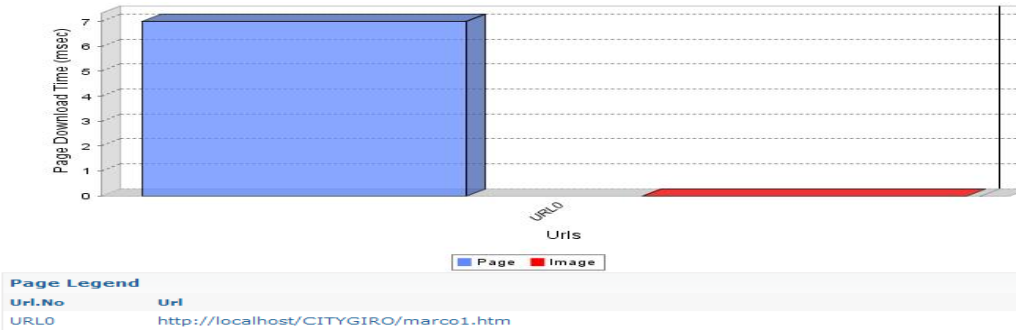

Fig. 5. Tiempo de descarga por páginas en .NET

Las figuras 2 y 3 muestran la cantidad de datos en bytes por segundo (bytes/s) que los usuarios virtuales recibieron del servidor.

El rendimiento de bytes/s presentado en GlassFish fue menor que el presentado en el IIS, ya que se presentaron valores máximos aproximados de 21000 bytes/s y de 72000 bytes/s respectivamente.

Las figuras 4 y 5 muestran el tiempo de descarga por página para cada una de las plataformas. El tiempo de descarga de las páginas .NET es mucho mejor frente a la de J2EE ya que existe una diferencia de 53 ms por páginas descargadas.

En (Rampally, 2006) se realizó un estudio similar utilizando la herramienta AdventNet QEngine, teniendo en cuenta los siguientes escenarios:

a. Clientes JSP accediendo a los servicios web EJB.
b. Cliente JSP accediendo a servicios web .NET.
c. Clientes .NET accediendo a Servicios Web EJB.
d. Clientes .NET accediendo a Servicios Web .NET.

El estudio concluye que un WS .NET responde más rápido que un servicio Web EJB. Entre los clientes que acceden a los WS .NET el tiempo de la página descargada de los clientes JSP es mayor que los clientes .NET Entre los clientes que acceden a los WS EJB, el tiempo de la página descargada de los clientes .NET es mayor que los clientes JSP.

3. CONCLUSIONES

Ambas plataformas se basan en especificaciones de seguridad comunes a nivel de mensaje. Así que la decisión de utilizar una plataforma en vez de la otra radica al final en los desarrolladores o en la organización que pretenda implementar los servicios Web, aunque según [15] se cree que existe una ventaja de seguridad de J2EE frente a .NET, ya que, desde un comienzo Java se fundamentó en un estricto modelo de seguridad a diferencia de .NET, y que por ende la experiencia Java en cuanto a seguridad es mucho mayor.

Cada plataforma se basa en una arquitectura de código intermedio y de máquinas virtuales para permitir la portabilidad, pero J2EE posee un mayor grado de portabilidad.

Si la portabilidad es un factor importante y fundamental es mejor decidirse por una solución J2EE, donde hay JVM para varios sistemas operativos. De otra parte, si lo que se pretende desarrollar exige varios lenguajes, es mejor una solución .NET debido a su independencia de lenguaje, proporcionada por el CLR.

La ventaja obtenida al comparar los dos IDE se encuentra a favor de Visual Studio. Aspectos como el IDE marca la diferencia cuando el factor tiempo de desarrollo, es de suma importancia en lo que se pretende realizar, aun cuando se trata de servicios Web o de cualquier clase de aplicación, decidiéndose así por optar por una solución .NET.

Por otro lado, se resalta el hecho que NetBeans es una de tantas herramientas para construir servicios en J2EE. Además cuando se cuenta con aplicaciones con diferentes sistemas operativos, la utilización de Visual Studio se convierte en un problema. Entonces en un escenario como este la mejor solución es J2EE, ya que, la mayoría de las herramientas para construir servicios se pueden instalar en cualquier sistema operativo.

Al implementar servicios Web J2EE y .NET y requerirlos desarrollando clientes en ambas plataformas, se comprobó en efecto que los servicios Web permiten la interoperabilidad. Sin embargo se puede perder cierto grado de interoperabilidad al utilizar servicios Web .NET, ya que, estos a veces devuelven datos específicos de la misma plataforma.

Si se requiere un máximo nivel de interoperabilidad en los servicios Web que se pretenden desarrollar es mejor optar por una solución J2EE. Por otro lado si se sabe que los servicios Web únicamente serán requeridos por clientes .NET, es una buena opción construir estos servicios en .NET utilizando *DataSet* en caso que así se requiera y ahorrar tiempo en el desarrollo de los mismos.

Aunque en las pruebas de desempeño realizadas se nota una superioridad de .NET, se plantea como recomendación realizar pruebas con otras características. No obstante, de acuerdo a los resultados obtenidos en las pruebas realizadas de desempeño, se recomienda utilizar servicios Web .NET para lograr el máximo desempeño y según Rampally tratar hasta donde sea posible de hacer los clientes en la misma plataforma donde se realizó el servicio Web como tal.

REFERENCIAS

[1]. Aquino Salvioni, Nathalie M., y Frutos Acosta, Juan Carlos. (2002). Fundamentos de la Máquina Virtual Java y el Entorno .NET. Universidad Católica Nuestra Señora de la Asunción. Facultad de Ciencias y Tecnología. Disponible en Internet en: http://www.jeuazarru.com/docs/Java_y_PuntoNET.pdf

[2]. Bostad, Geir. (2006). Supporting SAM: Infrastructure Development for Scalability Assessment of J2EE Systems. Disponible en Internet en: http://daim.idi.ntnu.no/masteroppgaver/IME/IDI/2002/3289/masteroppgave.pdf

[3]. Carpe García, Francisco. (2001). Estudio de la plataforma .NET. Disponible en Internet en: http://ditec.um.es/cgi-bin/dl/ProyectoNET.pdf Fecha de consulta: Abril 2008

[4]. Cheng Eric, Duff James y Chiesa Dino. (2007). Interoperabilidad de Servicios Web entre Microsoft .NET e IBM WebSphere. Disponible en Internet en: http://www.microsoft.com/

[5]. Domínguez Jiménez J. J., Estero Botaro A., Medina Bulo I., Palomo Duarte M. y Palomo Lozano F. El Reto De Los Servicios Web Para El Software Libre. Universidad de Cádiz. Departamento de Lenguas y Sistemas Informáticos. Disponible en Internet en: http://www.willydev.net/

[6]. Garrido Pino, Miguel Ángel. (2006). Evaluación Comparativa de aplicaciones Web entre J2EE y Microsoft. NET. Disponible en Internet en: http://biblioteca.uct.cl/tesis/miguel-garrido/tesis.pdf

[7]. Gutiérrez, E. Fernández-Medina, M. Piattini. (2005). Seguridad en Servicios Web. Disponible en Internet en: http://www.info-ab.uclm.es/descargas/ thecnicalreports/DIAB-05-01-2/Seguridad_en_Servicios_Web.pdf

[8]. Molinari, Lía. (2004). Arquitecturas Orientadas a Web Services. Disponible en Internet en: http://www.sedici.unlp.edu.ar/search/request.php Fecha de consulta: Julio de 2008

[9]. Organización para la Cooperación y el Desarrollo Económicos (OCDE). (2002). Guías para la Seguridad de los Sistemas de Información y Redes. Disponible en Internet en: http://www.uaslp.mx/PDF/2042_182.pdf

[10]. Rampally, Maneesh. (2006). Performance Comparison Of Interoperability Between J2EE And .NET Web Services. Disponible en Internet en: http://people.cis.ksu.edu/~maneesh/maneesh/Report_Final_Draft.doc

[11]. Romero Masis, Edgardo Alberto. (2004). Estado del Arte de la Tecnología de Web Services. Disponible en Internet en: http://www.juliux.org/tesis/webservices/Web_Services_final.doc Fecha de consulta: Julio de 2008

[12]. Samper C, Jose G. (2005). Integración Del Sistema De Gestión Administrativa y el Sistema De Comercio Electrónico mediante XML Web Services. Disponible en Internet en: http://www.tauniversity.org/tesis/Tesis_Jose_Samper1.pdf

[13]. Skonnard, Aaron. (2007) Mejorar la interoperabilidad de los servicios Web. Artículo de Microsoft. Disponible en Internet en: http://www.microsoft.com/

[14]. Vásquez Romero, William y Rojas, Juan Guillermo. (2004). Mecanismos de Control de Acceso en Web Services. Disponible en Internet en:http://www.javeriana.edu.co/biblos/tesis/ingenieria/Tesis208.pdf

[15]. Comparación entre J2EE y .NET. http://cek.blogia.com/2004/050301-comparacion-entre-j2ee-y-.net.php

LISTA DE ABREVIATURAS

CLR	*Common Language Runtime*
EJB	*Enterprise Java Beans*
IDE	*Integrated Development Environment*
IIS	*Internet Information Server*
JIT	*Just-In-Time*
J2EE	*Java 2 Enterprise Edition*
JVM	*Java Virtual Machina*
JSP	*Java Server Pages*
OASIS	*Organization for the Advancement of Structured Information Standards*
MSIL	*Lenguaje Intermedio de Microsoft*
SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocol*
UDDI	*Universal Description, Discovery and Integration*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
WCF	*Windows Communication Foundation*
WSIT	*Web Services Interoperability Technology*
WS	*Web Service*
WSDL	*Web Services Description Language*
WSE	*Web Services Enhancements*
XML	*Extended Markup Language*
W3C	*World Wide Web Consortium*