Industrial Data
ISSN: 1560-9146
iifi@unmsm.edu.pe
Universidad Nacional Mayor de San Marcos
Perú

Santos López, Félix Melchor
Desarrollo de un servicio web de verificación vehicular en centrales de riesgos crediticios
Industrial Data, vol. 14, núm. 2, julio-diciembre, 2011, pp. 16-25
Universidad Nacional Mayor de San Marcos
Lima, Perú

Revista de la Facultad de Ingeniería Industrial
14(2): 16-25 (2011) UNMSM
ISSN: 1560-9146 (Impreso) / ISSN: 1810-9993 (Electrónico)

Desarrollo de un servicio web de verificación vehicular en centrales de riesgos crediticios

Recibido: 21/03/11 Aceptado: 05/09/11

([1]) Félix Melchor Santos López

RESUMEN

Las centrales de riesgo crediticio en el Perú son empresas receptoras de información de fuentes externas (AFP, bancos, seguros, municipalidades, etc.) que se encargan de brindar diversos servicios a las empresas, entidades y ciudadanos del país, por ende sus sistemas informáticos están constantemente actualizados y a la vanguardia de la tecnología. Debido a esto y la gran cantidad de tecnologías presentes en el mercado, recientemente se viene optando por brindar soluciones basados en Servicios Web, ya que permiten una comunicación, interoperabilidad e intercambio de información independiente del lenguaje de programación en que estén desarrollados los sistemas. Esto representa una gran ventaja por ejemplo para el sistema de "Verificación Vehicular" que brinda información de los vehículos registrados en el país. Adicionalmente, la utilización de UML (Unified Modeling Language) como el lenguaje de modelado del sistema y el lenguaje de programación Java en su versión J2EE (Java Second Enterprise Edition) permiten un adecuado desarrollo del software, y JMeter contribuye a realizar las pruebas del sistema lográndose deducir los parámetros de configuración para su óptimo desempeño, que para el presente sistema se concluye que no deberá sobrepasar los 5 000 usuarios concurrentes del servicio web.

Palabras clave: servicio web, verificación vehicular, central de riesgo crediticio, J2EE, UML

Development of a web service of vehicle inspection in central credit risk

ABSTRACT

The central credit risk companies in Peru collect much information from different external sources and they offer various services to the others enterprises, entities and citizens. Therefore, their information systems are updated constantly. In the world there are many different technologies developed in different language programming, thus it was considered a pain in the neck for software engineers. Fortunately, nowadays there is a framework called Web Service that allows communication, interoperability and exchange among different language programming. The "Vehicle Inspection" system gives all the information about vehicles registered in the country. In addition, this system was developed in J2EE (Java Second Enterprise Edition) and UML (Unified Modeling Language) because are reliable and free technologies; also JMeter is a good test tool that helps to establish the accurate measure of the parameters for a good system performance. As a result, 5 000 concurrent users should not exceed the capacity of this web service.

Keywords: web service, vehicle inspection, central credit risk, J2EE, UML

INTRODUCCIÓN

En la actualidad los servicios web o Web Services proporcionan una importante interoperabilidad e integración entre sistemas de información desarrollados bajo diferentes tecnologías. Por otro lado, las centrales de riesgos crediticios son las encargadas de la recepción de información crediticia de diferentes fuentes externas (AFP, bancos, seguros, municipalidades, etc.), en el Perú están reguladas por la Superintendencia de Banca, Seguros y AFP (SBS).

Estas centrales de riesgos a su vez proporcionan una serie de servicios en base al procesamiento y modelamiento de la información que acumulan. Uno de estos servicios es el de "verificación vehicular", consulta que se realiza a los sistemas de información por parte de usuarios externos (clientes). Estos usuarios externos son principalmente las empresas aseguradoras que venden el Seguro Obligatorio de Accidentes de Tránsito (SOAT). La información brindada por esta consulta se utiliza para validar los datos del vehículo proporcionados por un comprador del SOAT.

Un inconveniente presentado durante varios años es el problema de comunicación e incompatibilidad de tecnologías, debido a que las empresas e instituciones elegían diferentes plataformas para el desarrollo de sus sistemas de información y esto dificultaba la interoperabilidad entre tecnologías diferentes de dos empresas. Sin embargo, en junio del año 2001 el Gartnet Group, un centro de investigación de tecnologías de información y firma consultora formada por importantes empresas del sector de las Tecnologías de Información, documentó un cronograma para la adopción de Web Services desde el 2001 al 2005.

"La Web usa HTTP para correr sobre TCP/IP, que se convirtió en el estándar universal. La invención de XML fue lo que realmente permitió el camino para los Web Services. Con SOAP y WSDL, las compañías pudieron crear y describir sus propios Web Services." [1] [4] (traducido de la referencia). Estos conceptos serán explicados en las secciones siguientes. Así mismo, para el desarrollo de este Web Service se empleó el Lenguaje Unificado de Modelo UML, el lenguaje de programación Java en su versión J2EE, la plataforma SUN JAX-WS (ver definición en tabla 1) para le generación de los artefactos necesarios para la publicación del Web Service y JMeter como testeador para las pruebas del sistema.

1 Ingeniero Informático, Pontificia Universidad Católica del Perú.
E-mail: fsantos@pucp.edu.pe

Los principales objetivos de la investigación son establecer si las tecnologías seleccionadas funcionan correctamente, desarrollar los diagramas UML que permitirán una diagramación adecuada y ordenada para futuras modificaciones, y por último determinar el valor máximo de usuarios concurrentes para el correcto funcionamiento del Web Service.

DEFINICIONES

Web Service o servicio web permite la comunicación negocio a negocio B2B (Business to Business, que se denomina comercio en la red), permitiendo a las empresas compartir e integrar datos y servicios heterogéneos en una Arquitectura Orientada a Servicios SOA. *"En los últimos años, Web Service se ha convertido en una tendencia dominante y se está haciendo la tecnología "omnipresente" que prometía ser en sus orígenes"* [1].

Los Web Services permiten llamadas a procedimientos remotos y la mensajería asíncrona, soliendo implementarse mediante mensajes XML a través del protocolo de comunicación HTTP.

Adicionalmente, *los "Web Services son definidos por W3C que es el comité responsable de su arquitectura y reglamentación"* [2].

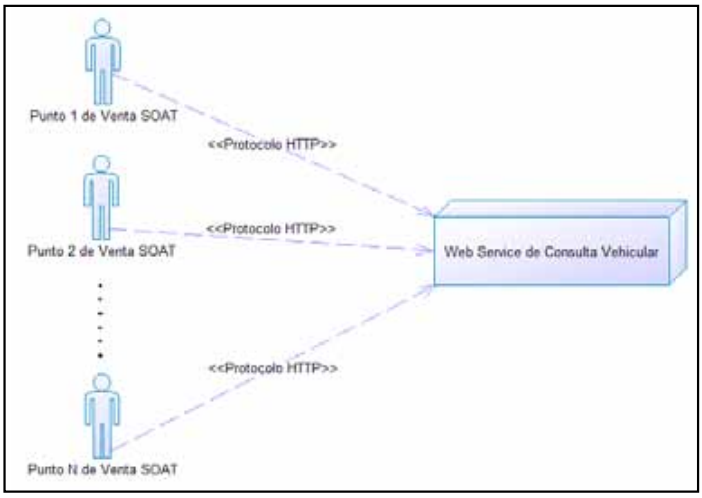
A continuación la definición de términos importantes para la compresión de los Web Services:

- **XML:** Extensible Markup Language, es el formato estándar para los datos que se intercambian.
- **SOAP:** Simple Object Access Protocol, protocolo sobre el que se establece el intercambio, principalmente HTTP.
- **WSDL:** Web Services Description Language, es el lenguaje de la interfaz pública para los servicios web, es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios web.
- **UDDI:** Universal Description Discovery and Integration, protocolo para publicar la información de los servicios web, permite a las aplicaciones comprobar qué servicios web están disponibles.
- **HTTP:** Hipertext Transfer Protocol, es el protocolo utilizado en cada transacción de la World Wide Web.
- **W3C:** World Wide Web Consortium, consorcio internacional que produce recomendaciones para la World Wide Web.

Para el caso particular de la implementación del Web Service de "Verificación Vehicular", se realiza mediante el protocolo HTTP por ser un estándar ampliamente utilizado. Se aprecia en la Figura 1 que los diversos "consumidores" del Web Service serán los puntos de venta de los SOAT, por ejemplo módulos en los supermercados, retails, grifos, universidades, etc. Inclusive BlackBerries u otros dispositivos móviles para vendedores de a pie.

La comunicación con el Web Service no se realiza a través de los usuarios humanos, sino los sistemas informáticos de la empresa, porque el consumo del Web Service se realiza mediante la invocación de otro sistema informático, por tanto la Figura 1 es un diagrama ilustrativo, siendo los "Puntos de Venta" sistemas de información.

Figura 1. Comunicación de los consumidores del Web Service y el protocolo HTTP.



Fuente: Elaboración propia

Tabla 1. Cuadro de tecnologías a utilizar y su descripción

Tipo de Tecnología	Descripción y sustento
Lenguaje de Programación Java. Versión: JDK 6.	J2EE (Java Second Enterprise Edition), lenguaje de programación ampliamente utilizado en las empresas, posee una gran variedad de documentación, una madurez en el mercado que lo hacen confiable, soporte de la corporación Oracle y es Open Source.
Enterprise Java Bean – EJB. Versión: EJB 3.	Perteneciente al J2EE, EJB es un API para el desarrollo de aplicaciones empresariales que permite almacenar y gestionar las clases encargadas de llevar a cabo la lógica de negocio de un sistema de información. Se utiliza la versión número 3 que permite la utilización de "anotaciones", lo cual disminuye el tiempo de desarrollo debido a que es más simple.
SUN JAX-WS. Versión: 2.2.1.	Permite la construcción de Web Services mediante la utilización de "anotaciones" proporcionados por el JDK 6 y genera automáticamente el archivo WSDL, UDDI y las clases necesarias para el despliegue del Web Service.
Servidor de Aplicaciones Jboss. Versión: Jboss 5.0.1GA.	Servidor de Aplicaciones encargado de almacenar la aplicación y ponerlo en marcha. Jboss es compatible para la puesta en producción de aplicaciones J2EE, su versión EJB3 y Web Services. Adicionalmente, la versión GA de Jboss es Open Source.
JMeter Versión: 2.4.	JMeter es un software OpenSource desarrollado en java como una aplicación de escritorio que permite realizar pruebas de esfuerzo de aplicaciones y servicios Web.

Fuente: Elaboración propia

Así mismo, para el desarrollo del Web Service se seleccionaron las tecnologías que se indican, de forma sustentada en la Tabla 1.

En la Figura 2 se aprecia cómo se desplegará y almacenará el aplicativo que resolverá el problema de la implementación y puesta en marcha del Web Service de verificación vehicular. Está compuesto por dos partes:

Aplicativo Web (Contenedor Web): donde se almacena en sí el WSDL, es decir el Web Service que se publica para su consumo.

Aplicación Empresarial EJB 3 (Contenedor de Aplicaciones): donde se almacena en sí la clase encargada de realizar la lógica del negocio. Esta no es visible por los consumidores, sino que es invocada por el "Aplicativo Web".

La aplicación de este Web Service permitirá su consumo por parte de los clientes, en este caso vienen a ser los sistemas de información de las empresas que adquieran el servicio. Estos sistemas no necesariamente tienen que estar desarrollados bajo el lenguaje de programación Java, debido a que se emplea el protocolo de comunicación HTTP y el estándar XML. Por ello, se pueden utilizar lenguajes de programación como .NET, Visual Basic, PHP, Ruby, Cobol, etc. para poder consumir el Web Service.

En la Figura 3 se muestra un ejemplo de la interacción de un "Consumidor" (Client Side) y un "Productor" (Server Side) desarrollados en Java.

Todo ello empieza del lado izquierdo, el "Consumidor" establece los valores de los parámetros (param) necesarios para realizar el consumo del Web Service y seguido invoca mediante un mensaje **Request: SOAP** y el "Productor" recibe la petición, lee los parámetros enviados y procesa. Por último, genera un retorno mediante el mensaje **Response: SOAP** el cual es enviado de regreso al "Consumidor", finalizando el ciclo.

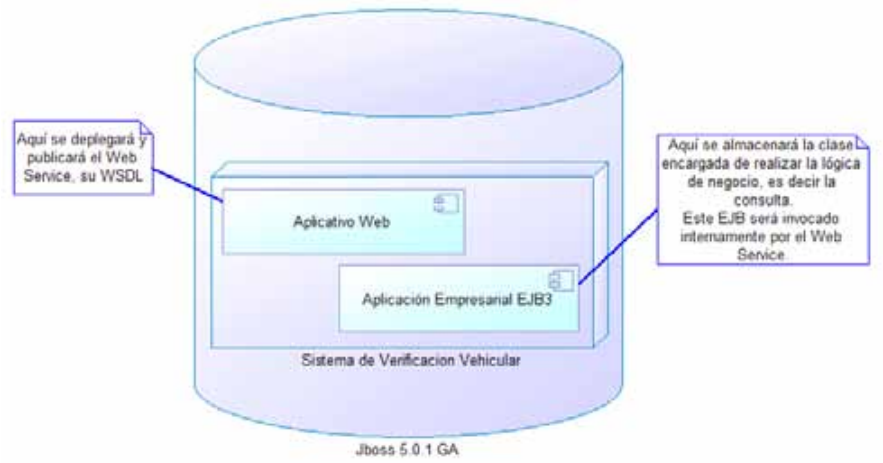
2. DESARROLLO DEL SISTEMA DE INFORMACIÓN

2.1. Análisis del sistema mediante la notación UML

Siendo Java el lenguaje de programación a utilizar y su principal característica es de ser un lenguaje orientado a objetos; el análisis del sistema se basará en la notación UML, es decir el Lenguaje Unificado de Modelado.

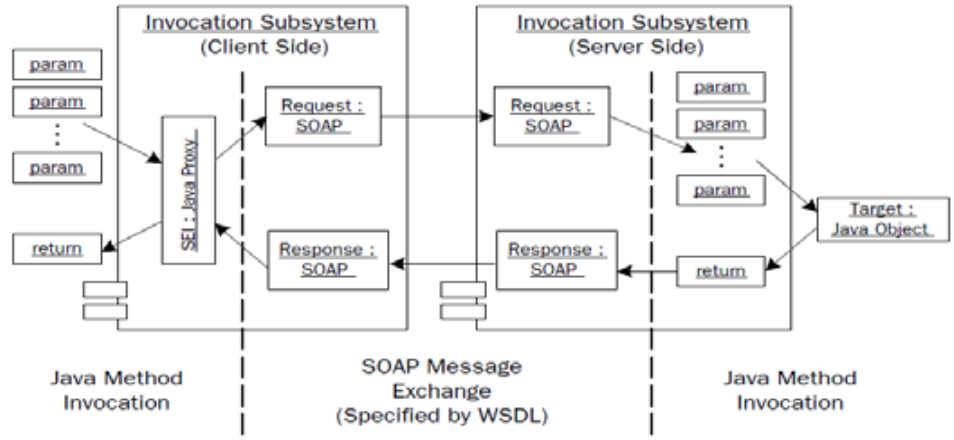
"UML es un lenguaje de modelado visual de propósito general que se utiliza para especificar, utilizar, construir y documentar los artefactos de un sistema software" [7]. Dada esta definición, se establece que un modelo capta los aspectos importantes de lo que

Figura 2. Gráfico del Servidor Jboss y el aplicativo a desarrollar.



Fuente: Elaboración propia

Figura 3. Invocación de un Subsistema (Client Side) a un Subsistema (Server Side) desarrollado en Java mediante mensajes emitidos por SOAP.



Fuente. Tomado de Mark Hansen [5]

se está modelando y simplifica u omite el resto.

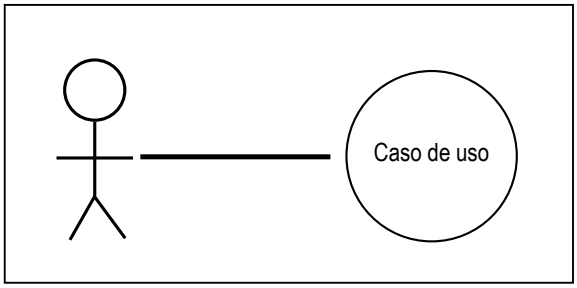
Uno de los artefactos más importantes de la notación UML son los casos de uso del sistema, los cuales modelan la funcionalidad del sistema tal como lo perciben los agentes externos que interactúan con este. Su funcionalidad se expresa como una transacción que se realiza entre un actor y el sistema; en la Figura 4 se aprecia un ejemplo.

Para el Sistema de Verificación Vehicular se identifica al actor externo como un “Sistema Consumidor Externo” y al proceso de la consulta como el caso de uso “Consultar Vehículo”, tal como se aprecia en la Fig. 5.

“Nótese que la caja del actor contiene el símbolo <<actor>>, denominado estereotipo UML, el cual trata de clasificarlo de un solo modo” [3]. Se decide utilizar este estereotipo como buena práctica, ya que el actor que interactúa con el sistema no es una persona en sí, sino un sistema informático externo. Cabe precisar que este caso de uso se debe de especificar, es decir escribir “literalmente” cómo actúa el actor y el caso de uso en un llamado al Web Service.

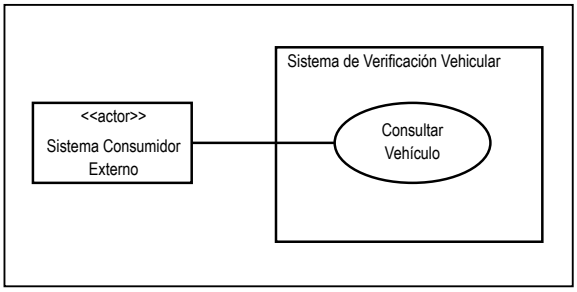
Adicionalmente, en la Figura 6 se presenta el diagrama de clase de análisis con las principales entidades a implementar. En este caso “Estructura-VehicularEntrada” representa los parámetros de

Figura 4. Diagrama de caso de uso



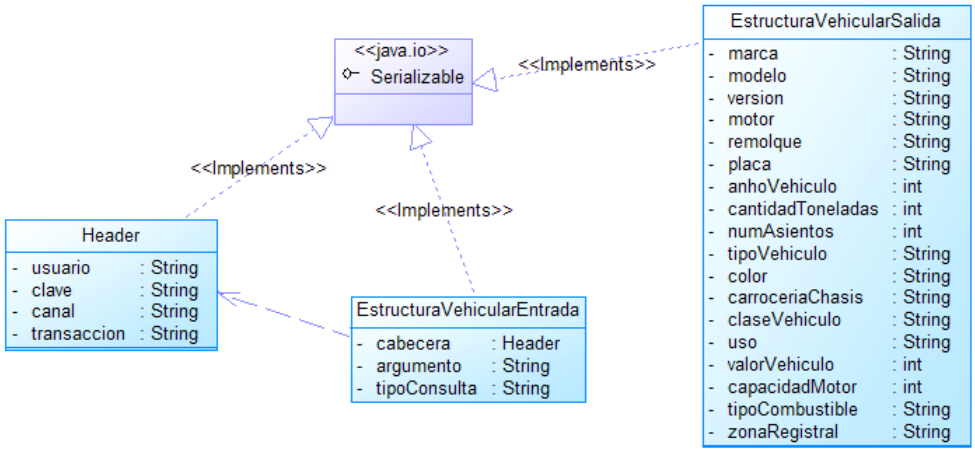
Fuente: Elaboración propia

Figura 5. Diagrama de caso de uso del sistema



Fuente: Elaboración propia

Figura 6. Diagrama de clases del sistema



Fuente: Elaboración propia

entrada a enviar al Web Service y "EstructuraVehicularSalida" representa a los parámetros a devolver producto de la consulta.

2.2. Diseño del sistema

"El diseño orientado a objetos transforma el modelo de análisis creado usando análisis orientado a objetos, en un modelo de diseño que sirve como anteproyecto para la construcción de software" [6].

En el presente diseño, se define en la Figura 7 el diagrama de clases de diseño que será el soporte de la implementación y lógica del negocio, basados en las estructuras definidas en la Figura 6.

Se aprecia que la clase "*VerificacionVehicularWS*" es la entrada del Web Service, seguido se invoca a la interfaz "*VerificacionCICSLocal*" que a su vez está implementada por el EJB Session "*VerificacionCICS*"; siendo esta última la encargada de realizar la lógica del negocio que implica una llamada a la función "*conectarCICS*" de la clase "*ConexionVehiculo*", que obtiene realmente los datos de un servidor del tipo CICS.

Así mismo, todo el aplicativo estará empaquetado en un archivo del tipo Enterprise Application Archive (EAR) que es un estándar propio de las aplicaciones J2EE y se utiliza para su despliegue en el servidor Jboss. En la figura 8 se observa un EAR y sus principales componentes internos que agrupan dentro de sí a las clases de las figuras 6 y 7.

2.3. Implementación de J2EE

La construcción del Web Service se desarrolla con

el lenguaje de programación Java en su versión J2EE. Así mismo, esta permite el desarrollo de los servicios web mediante el uso de anotaciones, los cuales son los siguientes:

@WebService → especifica que la clase es un Web Service a publicar.

@WebResult → especifica el método y el resultado del Web Service a publicar.

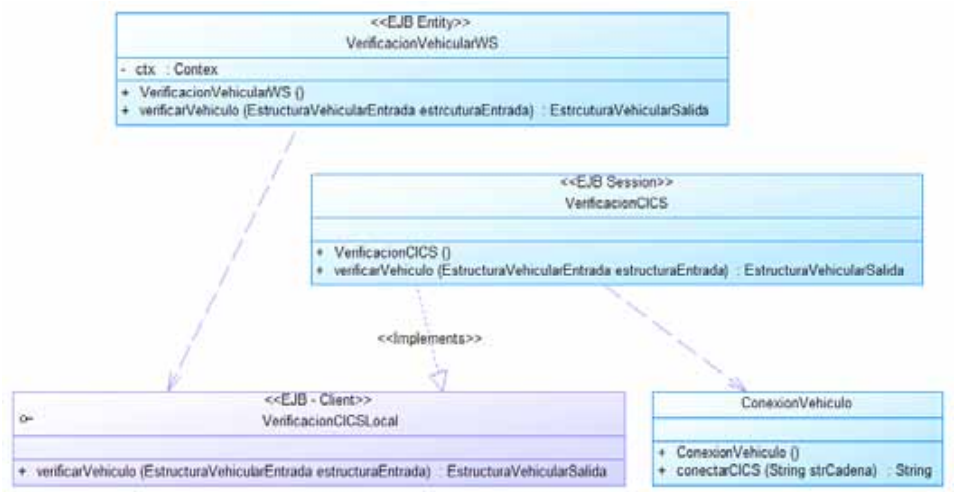
@WeParam → especifica el o los parámetro(s) que recibe el Web Service.

En la figura 9 se visualiza la clase VerificacionVehicularWS y la utilización de las anotaciones para su publicación como Web Service. Adicionalmente, se aprecia la utilización de las clases de diseño previamente realizadas en el punto 2.2 DISEÑO DEL SISTEMA y Figura 7.

Después de implementar y escribir el código fuente se procede a utilizar la herramienta de compilación JAX-WS en su versión 2.2.1, que proporciona los artefactos necesarios para la publicación y puesta en marcha del Web Service. Estos artefactos son principalmente los siguientes: la clase VerificacionVehicularWS compilada, el archivo de extensión .XSD (contiene las estructuras de entrada y salida del Web Service) y el archivo WSLD; siendo este último el principal y el encargado de publicar el Web Service en la Web y proveer la recepción y emisión de los mensajes a través del SOAP. En la figura 10 se aprecia el contenido del archivo WSDL del tipo XML generado.

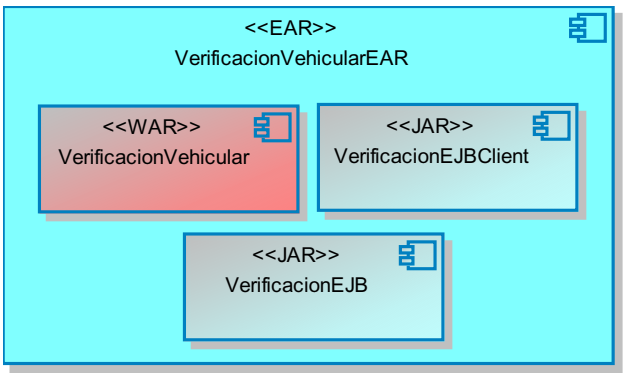
En la Figura 11 se visualiza los archivos XML de un consumo del Web Service de Verificación Vehicular. En la sección izquierda de la figura se aprecia el archivo de entrada con los campos respectivos a

Figura 7. Diagrama de Clases de Diseño



Fuente: Elaboración propia

Figura 8. Enterprise Application Archive (EAR) del sistema.



Fuente: Elaboración propia

Figura 9. Visualización del código la clase VerificacionVehicularWS.

```
package com.pe.verificacion.ws;

import javax.jws.WebParam;

@WebService(name="VerificacionService")
public class VerificacionVehicularWS {

    private Context ctx;

    public VerificacionVehicularWS(){

    public @WebResult(name="Salida") EstructuraVehicularSalida verificarVehiculo
    (@WebParam(name="Entrada") EstructuraVehicularEntrada estructuraEntrada)
    {
        EstructuraVehicularSalida estructuraSalida = null;

        VerificacionCICSLocal verificacionCICS;
        try {

            verificacionCICS = (VerificacionCICSLocal) ctx.lookup("java:comp/env/VerificacionCICSLocal");

            estructuraSalida = verificacionCICS.verificarVehiculo(estructuraEntrada);

        } catch (NamingException e) {
            e.printStackTrace();
        }

        return estructuraSalida;
```

Fuente: Elaboración propia

Figura 10. Archivo WSDL generado por el JAX-WS

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-
WS RI 2.2.1-b01-. -->
<definitions targetNamespace="http://ws.verificacion.pe.com/"
name="VerificacionVehicularWSService" xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:tns="http://ws.verificacion.pe.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://ws.verificacion.pe.com/"
schemaLocation="VerificacionVehicularWSService_schema1.xsd"/>
    </xsd:schema>
  </types>
  <message name="verificarVehiculo">
    <part name="parameters" element="tns:verificarVehiculo"/>
  </message>
  <message name="verificarVehiculoResponse">
    <part name="parameters" element="tns:verificarVehiculoResponse"/>
  </message>
  <portType name="VerificacionService">
    <operation name="verificarVehiculo">
      <input
wsam:Action="http://ws.verificacion.pe.com/VerificacionService/verificarVehiculo
Request" message="tns:verificarVehiculo"/>
      <output
wsam:Action="http://ws.verificacion.pe.com/VerificacionService/verificarVehiculo
Response" message="tns:verificarVehiculoResponse"/>
    </operation>
  </portType>
  <binding name="VerificacionServicePortBinding" type="tns:VerificacionService">
```

Fuente: Elaboración propia

Figura 11. Archivos XML de entrada (izquierda) y salida (derecha) de un consumo del WS.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.verificacion.pe.com/">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:verificarVehiculo>
         <Entrada>
            <argumento>II4939</argumento>
            <cabecera>
               <canal>IAX123</canal>
               <clave>MACRI1234</clave>
               <transaccion>0</transaccion>
               <usuario>RIMACSEGURO1</usuario>
            </cabecera>
            <tipoConsulta>1</tipoConsulta>
         </Entrada>
      </ws:verificarVehiculo>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:verificarVehiculoResponse
            xmlns:ns2="http://ws.verificacion.pe.com/">
         <Salida>
            <anhoVehiculo>2010</anhoVehiculo>
            <cantidadToneladas>2300</cantidadToneladas>
            <capacidadMotor>1600</capacidadMotor>
            <carroceriaChasis>Nueva de fabrica</carroceriaChasis>
            <claseVehiculo>Class EE</claseVehiculo>
            <color>Negro</color>
            <kilometraje>150000</kilometraje>
            <marca>Mercedez Benz</marca>
            <modelo>Coupe</modelo>
            <motor>1800</motor>
            <numAsientos>4</numAsientos>
            <placa>II4939</placa>
            <remolque>No aplica</remolque>
            <tipoCombustible>Premium</tipoCombustible>
            <tipoVehiculo>De lujo</tipoVehiculo>
            <uso>Ejecutivo</uso>
            <valorVehiculo>75000</valorVehiculo>
            <version>C1</version>
            <zonaRegistral>Lima - Lima - San Isidro</zonaRegistral>
         </Salida>
      </ns2:verificarVehiculoResponse>
   </S:Body>
</S:Envelope>
```

Fuente: Elaboración propia

enviar. En la sección derecha se observa el archivo de respuestas con los campos obtenidos producto de la consulta. Los mencionados archivos viajan por medio de la Web a través del protocolo HTTP y la definición del WSDL.

3. PRUEBA DEL SERVICIO WEB

"El objetivo de las pruebas, expresado de forma sencilla, es encontrar el mayor número de errores con una cantidad razonable de esfuerzo, aplicado sobre un lapso de tiempo realista" [6].

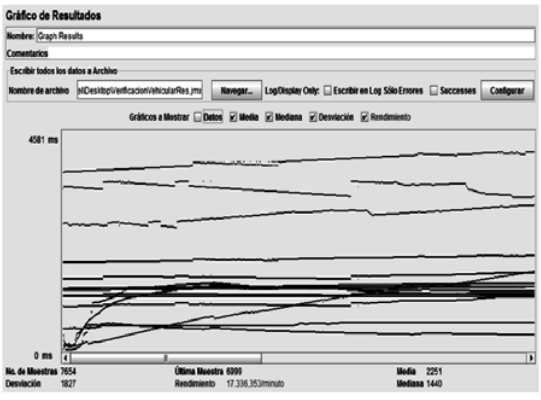
En el párrafo anterior se definió las pruebas para un sistema de información tradicional y que refleja el enfoque de las pruebas del tipo funcional. Sin embargo, la prueba más crítica e importante para un Web Service son las denominadas pruebas de estrés, las cuales someterán al servicio web a un número determinado e incremental de invocaciones.

Lo que se busca primordialmente es evaluar la respuesta y desempeño del Web Service ante un mayor número de consultas concurrentes. Por ello, se utilizó la herramienta de testeo de software JMeter especializada en pruebas para sistemas desarrollados bajo la tecnología J2EE.

En la Tabla 2 se muestra los datos de la simulación del sistema, empezando por 10 usuarios hasta llegar a los 10 000 con los incrementos como se aprecia en la tabla. Además, JMeter arroja un resultado denominado "Rendimiento" que se establece como el número de peticiones por minuto, el cual cuantifica precisamente el rendimiento del aplicativo. Como se aprecia en la gráfica a mayor número de usuarios concurrentes el valor de "Rendimiento" va en aumento, es decir el Web Service toma un mayor tiempo para procesar las solicitudes de consulta.

Todos los valores de la Tabla 2 son extraídos de las pruebas y reflejan una unidad de medida de JMeter denominada latencia que se expresa en ms (milisegundos). En la Figura 13 se muestra la gráfica de evolución del rendimiento, lo cual refleja que a

Figura 12. Gráfico de las pruebas en la herramienta JMeter.



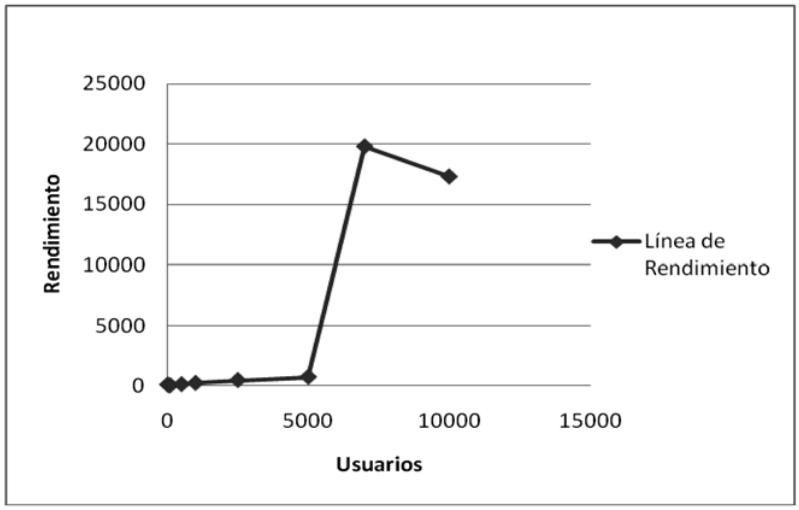
Fuente: The Apache Project [8].

Tabla 2. Cuadro de valores arrojados en las pruebas

Número de usuarios	Rendimiento (petición/minuto)	Media (ms)	Mediana (ms)	Desviación (ms)
10	85.167	3534	2040	2616
50	25.927	3337	3041	2446
100	37.490	3542	3077	2373
500	121.257	3556	3070	2594
1000	224.309	2937	1681	2423
2500	457.948	2245	1158	2169
5000	726.207	2077	1197	1923
7000	19839.271	2324	1432	1821
10000	17336.353	2251	1440	1827

Fuente: Elaboración propia

Figura 13. Gráfico de rendimiento del Web Service.



Fuente: Elaboración propia

mayor número de usuarios se requerirá un número mayor de tiempo de procesamiento y se verá reflejado en la lentitud del sistema. Cabe mencionar, que un pico de incremento considerable se obtiene al variar de 5 000 a 7 000 usuarios.

4. CONCLUSIONES

- El Web Service funcionó correctamente y su desarrollo e implementación con las tecnologías Java J2EE, EJB3, JAX – WS y JBoss fue una elección correcta y viable al ser Open Source y no tener la limitación de una licencia comercial.
- El análisis y diseño mediante la notación UML permitió un adecuado entendimiento del sistema así como una clara documentación para modificaciones o mantenimientos en el futuro.
- Se garantiza la compatibilidad del Web Service con los llamados de otros aplicativos desarrollados en diferentes lenguajes de programación, debido a que se utilizó el protocolo SOAP y el estándar XML para su implementación.

5. RECOMENDACIONES

- La aplicación desarrollada fue puesta en producción en una importante central de riesgos crediticios del país, del mismo modo se espera que otras centrales lo implementen ya que implica un negocio importante para ellos por ser el SOAT un seguro obligatorio, debido a que las compañías aseguradoras se ven en la obligación de comprobar la información brindada por sus clientes.
- Adicionalmente, las centrales de riegos deben tener en cuenta los resultados de las pruebas realizadas en esta investigación y a modo de recomendación se debería de configurar el servidor de aplicaciones JBoss para recibir un máximo de 5000 usuarios concurrentes, ya que hasta ese valor la aplicación trabaja con un buen rendimiento.

6. REFERENCIAS BIBLIOGRÁFICAS

[1] Clay W. Avondolio D. Schrager S. Mitchell M. Scanlon J. (2007). Profesional Java JDK. 6.ª ed. First Edition, Anaya Multimedia - WROX, Spain.

[2] Ferris C, Farrell J. – IBM Research Triangle Park, INC (2003). What are Web Services? Association for Computing Machinery - ACM Digital Library. Disponible en: http://portal.acm.org/citation.cfm?id=777335 (Visitado el 03/01/2011).

[3] Larman C. (2008). UML Y PATRONES – Una introducción al análisis y diseño orientado a objetos y al proceso unificado. 2.ª ed. Pearson – Prentice Hall, España.

[4] Levvit J (2001), From EDI to XML and UDDI: A brief history of Web Services, Information Week. Disponible en: http://www.informationweek.com/news/development/tools/showArticle.jhtml?articleID=6506480 (Visitado el 03/01/2011).

[5] Mark D. Hansen (2007). SOA Using Java Web Services, First Edition, Prentice Hall, U.S.A.

[6] Pressman R (2005). Ingeniería del Software – Un enfoque práctico. 6.ª ed. Mc Graw Hill, España.

[7] Rumbaugh J. Jacobson I. Booch G (2007). El lenguaje Unificado de Modelado, UML 2.0 – Manual de Referencia. 2.ª ed. Pearson, España.

[8] The Apache Jakarta Project (2010). Building a Web Service Test Plan, Apache JMeter. Disponible en: http://jakarta.apache.org/jmeter/usermanual/build-ws-test-plan.html_(Visitado el 03/01/2011).