MIDDLEWARE PARA ACCESO A BASES DE DATOS A TRAVÉS DE WEB SERVICES BASADOS EN EL MODELO DE SEGURIDAD AXIS2

MIDDLEWARE FOR DATABASE ACCESS THROUGH WEB SERVICES BASED IN AXIS2 SECURITY MODEL

AUTORA
MARÍA GUADALUPE ELÍAS GUZMÁN
Ing. en Sistemas Computacionales
*IT Tlajomulco
Estudiante
Academia de Sistemas
mguzman@ittlajomulco.edu.mx
MÉXICO

AUTOR
LUIS ANTONIO GAMA MORENO
Doctor en Ciencias de la Computación
*IT Tlajomulco
Profesor
Academia de Sistemas
lgama@ittlajomulco.edu.mx
MÉXICO

AUTOR
JOSE LUIS TORRES RODRÍGUEZ
Maestro en Ciencias
*IT Tlajomulco
Profesor
Academia de Sistemas
jltrmr@ittlajomulco.edu.mx
MÉXICO

AUTOR
CHRISTIAN GUILLERMO MURGUÍA VADILLO
Ing. en Sistemas Computacionales
*IT Tlajomulco
Profesor
Academia de Sistemas
ing_tlajomulco@tecnm.mx
MÉXICO

***INSTITUCION**
Instituto Tecnológico De Tlajomulco, Jal.
Ittj
Tecnológico Profesional
Km. 10 Carretera Tlajomulco - San Miguel Cuyutlán
S/N, Tlajomulco De Zúñiga, Jalisco. C.P. 45640
Direccion@Ittlajomulco.Edu.Mx
México

RECEPCIÓN: Noviembre 29 de 2016 **ACEPTACIÓN:** Febrero 16 de 2017

TEMÁTICA: Convergencia de servicios y redes de telecomunicaciones

TIPO DE ARTÍCULO: Artículo de Investigación Científica e Innovación

Forma de citar: Elías, M., Gama, L., Torres, J. y Murguía, C. (2017). Middleware para acceso a bases de datos a través de web services basados en el modelo de seguridad AXIS2. En R, Llamosa Villalba (Ed.). Revista Gerencia Tecnológica Informática, 16(44), 17-24. ISSN 1657-8236.

RESUMEN ANALÍTICO

En este artículo se presenta el diseño de un middleware para el acceso a bases de datos a través de Web Services (WS) basado en el modelo de seguridad Axis2. El WS denominado MABDA (Middleware para Acceso a Bases de Datos con Axis2), es una capa de software entre la base de datos y las aplicaciones que soliciten acceso a la misma. MABDA accede al Sistema Integral de Información (SII) de la institución, sin comprometer la seguridad. Esto se logra gracias a que las aplicaciones cliente envían únicamente: 1) un código relacionado con la consulta a realizar en la base de datos y 2) un certificado cifrado. Una vez autenticados estas credenciales, se obtiene como resultado un archivo en formato XML con el resultado de la consulta. El protocolo de seguridad Axis2 está orientado a XML y soporta estándares como WS-Security y WS-SecurityPolicy. Axis2 verifica el contenido de los mensajes entrantes y salientes para no comprometer la seguridad. Con el uso de MABDA, los desarrolladores podrán crear aplicaciones y solicitar datos del sistema legado "SII", sin importar la plataforma o lenguaje de programación.

PALABRAS CLAVES: Web-Services, Seguridad, XML, Apache Axis2.

ANALYTICAL SUMMARY

In this paper, a design of a middleware to access databases through Web Services (WS) based on the Axis2 security model is presented. The WS called MABDA (Middleware for Access to Databases based on Axis2) is a software layer between the database and the applications that request access to it. MABDA is able to connect to the Integral Information System (IIS) of the institution without compromising the security. This is achieved because client applications only send: 1) a code related to the query to be performed in the database and 2) an encrypted certificate. Once these credentials have been authenticated, an XML file with the data requested is returned. The Axis2 security protocol is XML oriented and supports standards such as WS-Security and WS-SecurityPolicy. Axis2 verifies the content of the incoming and outgoing messages in order to preserve the data security. With the use of MABDA, developers will be able to develop applications and request data stored in the "IIS" legacy system, in any platform or programming language.

KEYWORDS: Web-Services, Security, XML, Apache Axis2.

INTRODUCCIÓN

La tecnología de Web Services (WS) permite generar un ambiente distribuido en el cual, cualquier número de aplicaciones puedan interactuar entre ellas o entre organizaciones de manera independiente, neutral, y sin importar la plataforma ni el tipo de lenguaje de programación usado [3]. Los WS han transformado la web, ya que se pueden publicar, ubicar e invocar por otras aplicaciones, incluso por otros WS (Composición de WS) [9] y han permitido a los usuarios consumir servicios desde cualquier tipo de proceso sin importar el lenguaje. Por otro lado, la seguridad es hoy en día una característica y también requisito relevante para cualquier aplicación distribuida y en particular para los WS.

En este artículo se presenta el diseño de un WS que actúa como middleware (para las aplicaciones cliente-servidor) entre las aplicaciones cliente y la base de datos a acceder. El WS denominado Middleware para Acceso a Bases de Datos con Axis2 (MABDA) está desarrollado en la plataforma de Java. MABDA, diseñado para interactuar con el Sistema de Información Integral (SII) del Instituto Tecnológico de Tlajomulco (ITTJ). El proyecto SII, en un sistema propietario del TecNM, diseñado para ser usado por los Institutos Tecnológicos que pertenecen al sistema. El proyecto SII está desarrollado para operar a través de internet y de la red local de cada instituto. Soportado por el sistema operativo Linux y la base de datos de SyBASE. MABDA proporciona acceso (sólo de lectura y a ciertas tablas de datos) a aplicaciones desarrolladas por cada institución sin importar el lenguaje de programación utilizado. MABDA, actúa como una capa de seguridad para la base de datos debido al uso del protocolo Apache Axis2 junto con el módulo de seguridad Rampart porque soporta WS-Security y WS-SecurityPolicy. Como Rampart es implementado como un módulo, se acopla en la

infraestructura del procesamiento Axis2 y realiza su trabajo al interceptar mensajes en puntos particulares del procesamiento entrante y saliente.

Existen actualmente diversos esfuerzos para desarrollar aplicaciones que permitan el acceso a recursos como las bases de datos, de manera transparente y segura. En [4] se presenta una visión general del middleware de WS. Este middleware debe proporcionar soporte para el desarrollo de aplicaciones distribuidas basadas en WS. En [14] se describe el diseño de un middleware para realizar consultas mediante WS, el cual consta de dos partes, un módulo de consulta para los diseñadores de sistemas y otro para los usuarios finales, reduciendo la duplicación de código y mejorando la eficiencia del desarrollo. En [1] se presenta la integración e intercambio de datos basándose en WS y XML para la eliminación de información aislada, argumentando la importancia de la integración de importantes datos empresariales.

El resto del artículo está organizado de la siguiente manera. En la sección 1 se presenta el fundamento teórico sobre seguridad en los WS. En la sección 2 se encuentra la seguridad con Axis2. En la sección 3 se describe la problemática de acceso a BD. En la sección 4 se muestra el diseño del WS. En la sección 5 se presentan las pruebas y resultados, y finalmente en la sección 6 se presentan las conclusiones.

1. SEGURIDAD

Con el desarrollo de Internet, la tecnología de los WS se ha convertido en la dirección de desarrollo del comercio electrónico, gobierno electrónico y otros campos, mientras que la forma de diseñar una arquitectura integrada para mejorar la seguridad de los WS sigue siendo un problema difícil de resolver.

Los WS de Seguridad (WS-Security) engloban diversos requerimientos de seguridad tales como: la integridad, un mensaje debe permanecer inalterado durante la transmisión del mismo; confidencialidad, el contenido de un mensaje no se puede ver mientras están en tránsito, excepto por el personal autorizado de servicios; y disponibilidad, un mensaje se entrega inmediatamente a su destinatario, asegurando así que los usuarios legítimos reciben los servicios de los que tienen derecho [7]. Por otra parte, cada WS debe proteger a sus propios recursos contra el acceso no autorizado. Esto a su vez requiere de medios adecuados para: *identificación*, por lo que el destinatario de un mensaje tiene que verificar la identidad reclamada del remitente y *autorización*, por lo que el destinatario de un mensaje tiene que aplicar políticas de control de acceso para determinar si el remitente tiene derecho a utilizar los recursos necesarios.

Los WS permiten un acceso fácil a los datos y las conexiones son muy dinámicas, garantizando la confidencialidad e integridad de los datos que se transmiten a través de los protocolos de los WS [6]. Si un solo WS no cumple con las exigencias de servicios por los consumidores, es necesario aplicar la composición de WS, esto es: componer varios WS que en conjunto satisfacen las necesidades de los usuarios. Por otra parte, cuando se producen ataques de seguridad en los mensajes SOAP que se comunican entre WS mientras se accede a un servicio o durante la composición de un servicio en la Computación Orientada a Servicios (SOC), los desarrolladores pueden utilizar los servicios como elementos fundamentales en el proceso de desarrollo de aplicaciones [8]. En [13] se presentan soluciones a los problemas de modelado, composición, ejecución y verificación de WS. Para la composición de WS se basan en la inferencia orientada a objetivos de planificación. La mayoría de los WS-Security existentes han proporcionado soluciones para asegurar la autenticación del cliente, confidencialidad e integridad de la información en la capa de red y no en la capa de aplicación. Por lo tanto, gracias a que la tecnología de Axis2, que divide sus funcionalidades en módulos asociados con flujos de entrada y de salida, para notificar cuando se produce un evento, por ejemplo, la llegada de un mensaje SOAP. Además, es el responsable de codificar los mensajes en el formato común XML, asegurando la autenticación del cliente.

Los estándares de XML y los WS son ampliamente utilizados en sistemas distribuidos actuales. La seguridad de la comunicación basada en XML y los WS es de gran importancia para el conjunto de seguridad de estos sistemas. Por otra parte y con el fin de facilitar la interoperabilidad de los WS, los mecanismos de seguridad pueden basarse principalmente en estándares como XML Encryption y WS-Security para proteger el contenido [10].

1.1 WS-SECURITY

Los WS-Security basados en XML también proporcionan una capa de mensajes de seguridad en la capa de red y no en la capa de aplicación [5]. La especificación WS-Security, describe la forma de asegurar los WS en el nivel de los mensajes, en lugar del nivel del protocolo de transferencia o en el de la conexión. Para ello, tiene como objetivo principal describir la forma de firmar y de encriptar mensajes de tipo SOAP.

WS-Security se basa en estándares y certificaciones digitales para dotar a los mensajes SOAP de los criterios de seguridad necesarios. Se definen cabeceras y usa XML signature para el manejo de las firmas en el mensaje, las firmas digitales proporcionan de extremo a extremo

garantías de integridad del mensaje e información de la autenticación al emisor del mensaje. La encriptación de la información la realiza mediante XML Encryption, haciendo uso del intercambio de credenciales de los clientes.

WS-Security define la forma de conseguir integridad, confidencialidad y autenticación en los mensajes SOAP. Se realiza de la siguiente manera:

- La autenticación identifica al remitente.
- WS-Security utiliza token de seguridad para mantener esta información mediante un encabezado de seguridad del mensaje SOAP.
- La integridad del mensaje se consigue mediante firmas digitales XML, que permiten encabezado de seguridad del mensaje SOAP.
- La confidencialidad del mensaje se basa en la especificación XML Encryption y garantiza que sólo el destinatario o los destinatarios a quien va destinado el mensaje podrán comprender las partes correspondientes.

Se apoya en WS-Adressing para asegurar el no repudio.

1.2 WS-POLICY

La infraestructura de los WS puede ser mejorada para entender ciertas políticas y forzar su uso en tiempo de ejecución, es decir, que las políticas que se determinaron puedan ser utilizadas por otros desarrolladores para determinar si pueden o no utilizar el servicio, todo en tiempo de ejecución sin la necesidad de alguna otra intervención. WS-Policy [2] define un modelo genérico y una sintaxis para describir y comunicar las políticas de los WS.

2. SEGURIDAD CON AXIS2

Rampart es el módulo de seguridad Axis2, el cual soporta WS-Security para WS basados en SOAP, WS-SecurityPolicy, WS-SecureConversaton y WS-Trush. Como Rampart es implementado como un módulo (en realidad un par de módulos – Rampart.mar y Rahas.mar), este se acopla en la infraestructura de procesamiento Axis2 y hace su trabajo al interceptar mensajes en puntos particulares del procesamiento entrante y saliente verificando o haciendo cambios en los mensajes según sea apropiado [11].

La seguridad es crucial cuando los WS intercambian datos de negocios. Si los datos son interceptados por terceros esto puede tener consecuencias financieras o legales negativas, o si aceptan datos fraudulentos como válidos. Siempre es posible diseñar e implementar el manejo de seguridad propio de cada aplicación para WS – como para cada forma de intercambio de datos – pero ese es un enfoque arriesgado, porque una pequeña y oscura omisión puede conducir a serias vulnerabilidades.

Uno de los principales beneficios de SOAP es que permite extensiones modulares. La seguridad ha sido un enfoque principal para extensiones desde los inicios de SOAP, dando como resultado la estandarización de WS-Security y las tecnologías relacionadas que permiten que la seguridad sea configurada para cada servicio.

Los requisitos de seguridad en el intercambio de información generalmente cubren tres aspectos:

- Confidencialidad: Solo el destinatario pretendido de un mensaje tiene el acceso al contenido del mensaje.
- Integridad: El mensaje recibido sin notificación.
- Autenticidad: El origen del mensaje puede ser verificado.

WS-Security permite responder fácilmente a estos tres aspectos [12].

2.1 MEJORANDO LA SEGURIDAD CON AXIS2

La nueva arquitectura sobre la que se basa Axis2 es más flexible, eficiente y configurable en comparación con la arquitectura Axis1. Apache Axis2 no sólo es compatible con SOAP 1.2 y SOAP 1.3, también ha integrado el soporte para el estilo de WS. Es más eficiente, modular y orientado en XML que la versión anterior. Axis2 contiene nuevas características, mejoras e implementaciones como:

- Velocidad: utiliza su propio modelo de objetos y logra una velocidad mayor a las versiones anteriores.
- AXIOM: viene con su propio modelo de objetos más ligero, para el procesamiento de mensajes que es extensible.
- MEP soporte: más útil con la flexibilidad necesaria para soportar los patrones de mensajes de Exchange (MEPS) con soporte en WSDL 2.0.

3. PROBLEMÁTICA

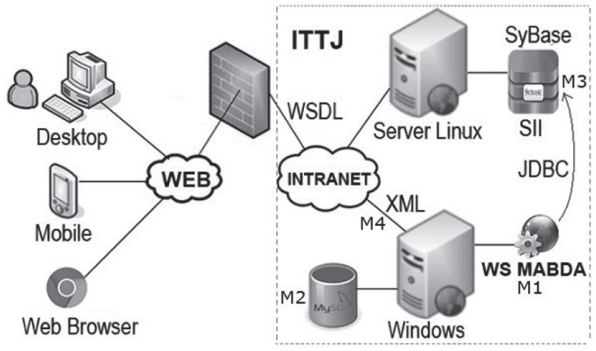
Para tener acceso a sistemas de información "legados" como lo es el SII del instituto, el desarrollador tendría que conocer la arquitectura, plataforma de desarrollo (lenguaje de programación), diseño de la base de datos, entre otros detalles técnicos propios del sistema. Además que el desarrollador deberá tener amplia experiencia en las tecnologías usadas. Si el desarrollador no cuenta con los respectivos diagramas

y/o documentación apropiados, la tarea sería muy difícil de completar. A esto le agregamos los problemas de seguridad, que implica el acceso a datos de manera remota. Ya que conectarse a un servidor de BD implica enviar a través de la red datos sensibles tales como: la dirección IP y el puerto del servidor, el nombre de la BD, el nombre de usuario y la contraseña de acceso. Si esto se realiza sin las medidas de seguridad básicas como cifrado por SSL, encriptado de contraseñas, etc., estos datos comprometerán la seguridad de la BD, ya que pueden ser interceptados y darles un mal uso, y aun peor, acceder y/o modificar datos sensibles en una BD. Por otro lado, si se encontraran modificaciones o robo de información por falsos usuarios, sería difícil identificar cuál usuario fue el responsable de tal evento. Para resolver esta problemática se plantea la implementación de un WS aplicado para este caso de estudio, donde el proceso de conexión con la BD del sistema sólo será proporcionado a un único usuario administrador del WS, impidiendo así que varios usuarios tengan tales accesos. La información almacenada en el WS estará encriptada para reafirmar la seguridad de la información que viaja por la red.

4. DISEÑO DEL WS MABDA

El WS MABDA está desarrollado bajo la plataforma de Java EE (Enterprise Edition), y consta de los siguientes módulos (M): 1) el núcleo principal es un Web Service basado en SOAP, el cual permite recibir las peticiones de los clientes, esta petición es autenticada por el 2) módulo de datos local, el cual, autentifica las credenciales del solicitante en una base de datos de usuarios previamente registrados, quienes son los únicos que tendrán acceso al servicio.

FIGURA 1. Arquitectura general del Web Service MABDA.



Una vez autenticado el solicitante, el módulo 3) de conexión a la base de datos del SII, que está desarrollado bajo la API JDBC (Java Database Connectivity) y un driver JDBC tipo 2 (nativo), este realiza una conexión con la base de datos del SII que se encuentra alojada en un servidor Linux y el DBMS de Sybase. Este driver permite a las aplicaciones desarrolladas en Java acceder de manera nativa a SyBase.

Cuando dicha solicitud es procesada, el módulo 4) transforma el resultado (ResultSet) de la consulta en un archivo XML, el cual es finalmente retornado como resultado al solicitante.

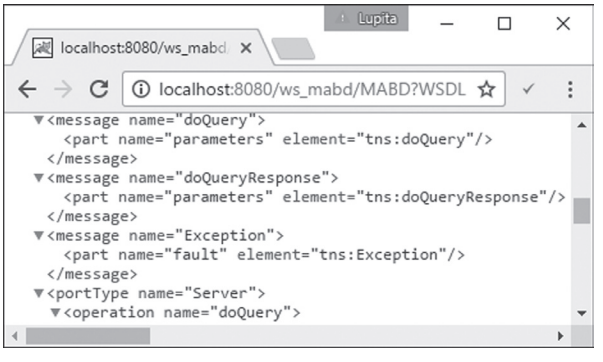
En la Figura 1, se muestra el proceso de una petición de un cliente. Cuando el cliente consume el WS MABDA, éste le devuelve un resultado XML. Un solicitante podrá consumir el WS MABDA desde plataformas de escritorio (C++, Java, C#, Delphi), móviles (iOS, Android) o navegadores de internet a través de páginas dinámicas (PHP, JS, Servlets, JSP, ASP, etc.).

4.1 DESCRIPCIÓN DEL WS CON WSDL

Un WSDL (Web Services Definition Languaje – Lenguaje de Descripción de Servicios Web) describe el WS cuando éste es publicado. Es el lenguaje XML que los proveedores emplean para describir sus WS.

Una vez publicado el WS en Internet, éste genera un WSDL el cual proporciona una interfaz para las aplicaciones y los métodos a los que tendrán disponibilidad. A continuación, se muestra el ejemplo del WSDL generado para el WS MABDA. El siguiente código (ver Figura 2.) muestra el método doQuery, el cual es usado para realizar una consulta en la base de datos del SII. Después de publicar el WS MABDA, el cliente puede consumir el WS en diferentes lenguajes por medio del WSDL con la ruta del mismo.

FIGURA 2. WS publicado.



4.2 PARÁMETROS DE CONEXIÓN

Se creó un archivo de configuración (ini) que contiene los parámetros que se requieren para acceder a la BD de MySQL como a la de SyBASE. Entre estos valores se encuentran almacenadas las contraseñas encriptadas,

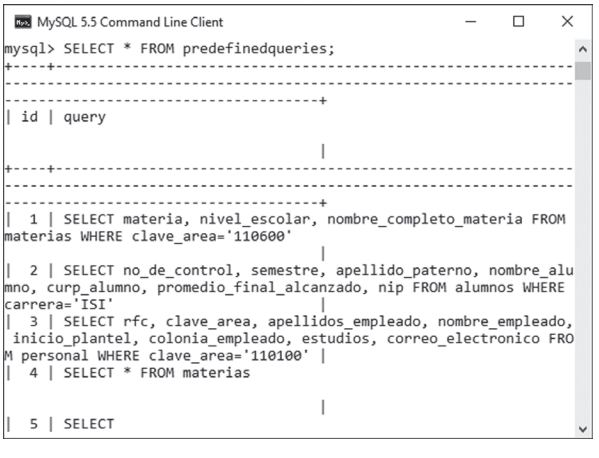
como se muestra en la Figura 3. Este archivo debe ubicarse dentro de la carpeta del proyecto del WS.

FIGURA 3. Archivo de configura para acceso.

5. PRUEBAS Y RESULTADOS

Las pruebas del WS se han desarrollado en las instalaciones del ITTJ por medio de la base de datos de control escolar (SII). El WS tiene el acceso a la base de datos por medio del protocolo de seguridad Axis2. El solicitante sólo debe indicar dos parámetros para realizar una consulta: 1) el código de la consulta y 2) un certificado que lo autentifica como usuario validado para el servicio.

FIGURA 4. Consultas predefinidas en a base de datos local.



El WS MABD tiene una base de datos local donde almacena la información de los usuarios validados (previamente registrados) y las consultas ya predefinidas como se puede observar en la Figura 4.

Tabla 1. Consumo de WS

Plataformas	Protocolos	Frameworks
Java	SOAP WSDL XML	Apache Axis 2 J2EE
C++	SOAP XML-RPC WSDL	Apache Axis 2 gSOAP
C#	SOAP WSDL	NuSOAP .NET
PHP	XML-RPX REST JSON	NuSOAP

5.1 CONSUMIENDO EL WS CON JAVA

A continuación se describe la implementación de un cliente para consumir el WS MABDA a través del lenguaje Java. Se empleó el IDE de programación NetBeans ver. 8.0.2, basado en el JDK ver 1.8.0_25. El proceso consiste en crear una aplicación Cliente, al cual se le indica la ruta del WSDL del WS MABDA. Al importar el WSDL en el proyecto, aparecerán los métodos que tiene publicados el WS y son los que se manipulan para consultar, en este caso el método `doQuery`. En el siguiente código se encuentra la implementación del programa para consumir el WS MABDA llamado DemoWSSII.java.

```
public class DemoWSSII {
public static void main(String[a] args) {
  try {
    //(ID, Certificado)
    String xml = doQuery(4, "0137");
    System.out.println(xml);
    } catch (Exception_Exception ex) {
      System.out.println(ex.toString());
    }
}

private static String doQuery(int codigo,
java.lang.String certificado) throws
Exception_Exception {
   demowssii.MABD service =
              new demowssii.MABD();
    demowssii.Server port =
              service.getServerPort();
    return port.doQuery(codigo,
                        certificado);
    }
}
```

Al método `doQuery` se deben pasar los valores respectivos a los siguientes parámetros: parámetro "`int codigo`", indica el identificador de consulta a ejecutar (estas consultas están previamente registradas); y el parámetro "`String certificado`", el cual es un identificador que el sistema le otorga a la aplicación, tanto para autenticarse como usuario registrado, como para tener el privilegio de acceso a dicha consulta. La ejecución del programa genera la salida que se muestra en la Figura 5., el cual es el resultado de la consulta (`codigo = 4`) devuelto por el WS MABDA en formato XML. La consulta referente al `codigo` muestra la tabla materias.

FIGURA 5. Datos obtenidos del MABDA desde Java

```
<materia>SCA1026</materia>
<nivel_escolar>L</nivel_escolar>
<tipo_materia>1</tipo_materia>
<clave_area>110600</clave_area>
<nombre_completo_materia>TALLER DE SISTEMAS
 OPERATIVOS</nombre_completo_materia>
<nombre_abreviado_materia>TALL SIST
 OPER</nombre_abreviado_materia>
```

5.2 CONSUMIENDO EL WS CON PHP

A continuación se describe la implementación de un cliente para consumir el WS MABDA a través del lenguaje PHP. El proceso consiste en crear una aplicación Cliente, importar la librería nusoap, la cual conecta al lenguaje Java con PHP y así ingresar la dirección del WSDL en el proyecto. También se requiere del servidor WAMP para mostrar los datos resultantes de la consulta. El siguiente código muestra la implementación del programa para consumir el WS MABDA llamado `DemoCliente.php`. Así como en la sección 4.2, se deben de suministrar los valores a los parámetros del WS MABDA. En el siguiente código se muestra la implementación del consumo del WS.

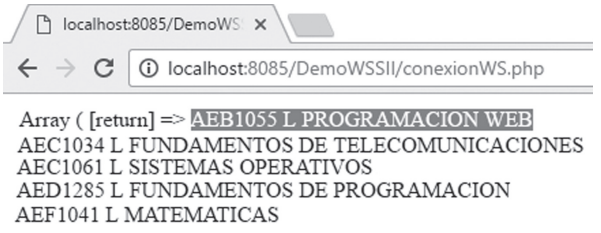
```
<?php
 include('lib/nusoap.php');
 $cliente = new nusoap_client('http://lo
calhost:8080/ws_mabd/MABD?WSDL','wsdl');

 //Nombre del método y parámetros.
 $resultado = $cliente->call('doQuery',
array('codigo' => '1',
      'certificado' => '0125'));

 //Imprimir resultados de la llamada al
método.
  error_reporting(E_COMPILE_ERROR |
          E_ERROR | E_CORE_ERROR);
  print_r($resultado);
?>
```

La ejecución del programa en el servidor WAMP genera la salida como se muestra en la Figura 6., el cuál es el resultado de la consulta (`codigo = 1`). La consulta para este `codigo` hace referencia a la tabla materias en cierta área.

FIGURA 6. XML recibido del MABDA desde una solicitud en PHP.



6. CONCLUSIONES

En este artículo se presentó el diseño de un WS para el acceso a una base de datos, y la realización de diferentes consultas al SII (Sistema Integral de Información). El WS denominado MABDA puede responder a las peticiones de los clientes a través del servidor de aplicaciones donde se hospeda el WS en Internet, y que generó un WSDL que el solicitante (cliente) puede consumir en diferentes lenguajes de programación tales como: Java, C++, C#, PHP, entre otros. El solicitante envía el código de la consulta y un certificado que lo autentifica como usuario válido, y el WS retorna los datos resultantes de la consulta en formato XML. El WS MABDA, no compromete la seguridad en el acceso a los datos, porque el WS es un intermediario entre la base de datos y el cliente gracias al protocolo Axis2 que se utilizó.

Se presentaron dos ejemplos del consumo de WS, uno se realizó con el lenguaje Java donde se creó un proyecto como WS-Client para la importación del WSDL publicado y el segundo ejemplo mediante PHP, donde se importó una librería para indicar la ruta del WSDL publicado por el WS. Éste segundo ejemplo muestra los resultados mediante un navegador.

Finalmente, con el uso del WS MABDA, se está garantizando la seguridad de los datos para éste caso de estudio ya que utiliza el protocolo de seguridad Axis2 que verifica los mensajes de entrada y de salida, y que sólo los usuarios previamente registrados tienen acceso para realizar consultas, así los usuarios tienen la flexibilidad para obtener datos del SII, ya que, al acceder al WS y obteniendo el WSDL pueden consumir el WS desde el lenguaje de programación que mejor les convenga.

7. AGRADECIMIENTOS

Este trabajo está desarrollado dentro del proyecto de investigación del TecNM denominado "Composición de Servicios Web para identificación de personas a través de sus dispositivos móviles" con clave: ITTLAJ-PTC-002. Instituto Tecnológico de Tlajomulco, Jal.

8. REFERENCIAS

[1] Decheng Qiu, Junning Liu, Guoying Zhao: Design and application of data integration platform based on web services and XML. Electronics Information and Emergency Communication (ICEIEC), 6th International Conference (2016).

[2] Elsafie Abeer, Schwenk Jörg: Semi-automated Fuzzy Mcdm and Lattice Solutions for WS-Policy Intersection. IEEE World Congress (2015).

[3] Gama-Moreno, L.A., Dávalos S., Martínez-Hernández, C., Ramírez A.: Capa de conexión basada en Servicios Web para la emisión de documentos electrónicos. Congreso Internacional de Computación Colombia-México. CICOM- 2015. ISSN, (2015).

[4] Gustavo Alonso, Fabio Casati, Harumi A. Kuno, Vijay Machiraju: Web Services - Concepts, Architectures and Applications. Data-Centric Systems and Applications, Springer, ISBN 978-3-540-44008-6 (2004).

[5] Kanchana Rajaram A., Chitra Babu B.: API Based Security Solutions for Communication among Web Services. Advanced Computing (ICoAC), 2013 Fifth International Conference (Dec 20, 2013).

[6] Lei Gao, Shu-Fen Lui, Lu Han: A Solution Of Axis2 Message Routing And Web Services Security. IEEE pervasive computing and applications (ICPCA), 2011.

[7] Martino L., Bertino E.: Tutorial 6: Security in SOA and Web Services. Web Services. ICWS '06. International Conference (2006).

[8] Milanovic Nikola, Malek Miroslaw: Current Solutions for Web Service Composition. IEEE Internet Computing 8(6): 51-59 (2004).

[9] Moo F., Hernández R., Uc V.: Web Service composition using the bidirectional Dijkstra algorithm. IEEE Latin America Transactions (Volume: 14, Issue: 5, May 2016).

[10] Nils Agne Nordbotten: XML and Web Services Security Standards. IEEE Communications Surveys and Tutorials 11(3): 4-21(2009).

[11] Sosnoski Dennis: Java Web Services: Axis2 WS-Security basics. Architecture Consultant and Trainer, Sosnoski Software Associates Ltd (2012).

[12] Sosnoski Dennis: Servicios Web Java: Firma y cifrado de WS-Security de Axis2. Architecture Consultant and Trainer, Sosnoski Software Associates Ltd (2012).

[13] Srivastava Biplav, Koehler Jana: Web Service Composition - Current Solutions and Open Problems. ICAPS Workshop on Planning for Web Services (2003).

[14] Yongzhen Ke, Fan Qin, Zhenwei Chen: A general query middleware based on web service. Mechanic Automation and Control Engineering (MACE), Second International Conference (2011).