EUSKO JAURLARITZA GOBIERNO VASCO

SEGURTASUN SAILA
Babes Zibileko Sailburuordetza
Joko eta Ikuskizunen Zuzendaritza

DEPARTAMENTO DE SEGURIDAD
Viceconsejería de Protección Civil
Dirección de Juego y Espectáculos

Web Service de consulta de autoprohibidos de Apuestas Online

ÍNDICE

ÍNDICE **2**

0.- Introducción **3**

1.- Arquitectura del servicio web **4**

2.- Invocación Web Service genérico **5**

2.1.- Tratamiento de Excepciones 11

3.- Utilización del Webservice Autoprohibidos **12**

3.1.- Tipos de errores controlados por el servicio Web 18

0.- Introducción

El presente documento explica el objetivo del servicio web de la Dirección de Juego y Espectáculos para la identificación y control de los usuarios autoprohibidos que accedan a las páginas web de apuestas On-Line, así como las técnicas para hacer uso de dicho servicio desde cualquier página web de apuestas.

El servicio proporcionará a las páginas web que hagan uso de él un método para verificar la mayoría de edad del jugador así como la localización del usuario apostante en los registros de autoprohibidos del Gobierno Vasco.

Se explicará en el documento, tanto los trámites a seguir para solicitar el uso del servicio al Gobierno Vasco, así como las técnicas necesarias para implementar en las páginas web el código necesario para invocar al servicio web.

La identificación de la persona apostante y la verificación de su edad se realizarán sobre los sistemas de gestión del DNI del Ministerio de Interior de España, mientras que el cotejo de autoprohibidos se efectuará sobre los sistemas de la Dirección de Juego y Espectáculos del Departamento de Interior del Gobierno Vasco.

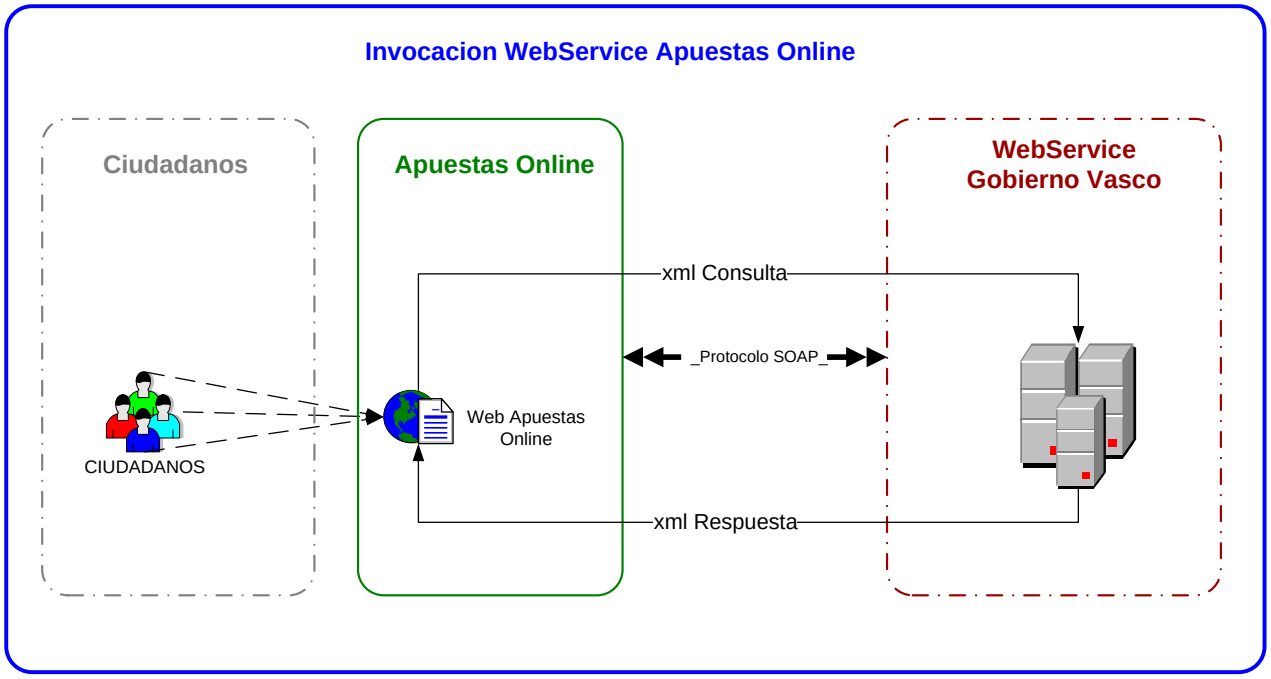
En la última versión del servicio, se ha añadido una capa adicional de seguridad, por la cual los parámetros que se envíen han de estar cifrados mediante el algoritmo AES y una clave de 128 bits. Igualmente, la respuesta obtenida será devuelta cifrada bajo la misma clave.

1.- Arquitectura del servicio web

Nuestro servicio web utiliza el protocolo SOAP.

SOAP estandariza el intercambio de mensajes entre diferentes aplicaciones. Por eso la función básica de SOAP es definir un formato de mensajes estándar (basado en XML) que encapsulará la comunicación entre aplicaciones.

Una de las Ventajas de SOAP es que permite la interoperabilidad entre múltiples entornos, SOAP se desarrolló sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dicho estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas.



2.- Invocación Web Service genérico

Para realizar la invocación Web Service necesitamos saber la url del WS y el nombre del método a invocar. Con esos dos datos podremos sacar cuantos y que parámetros debemos usar para realizar la llamada.

Para ello recordar que cualquier WS lleva asociada su WSDL donde están especificados todos estos parámetros (nombres de métodos, parámetros de entrada y de salida, sus nombres y tipos, etc.…).

Para realizar esta invocación (SOAP), aparte de los dos datos comentados (url y método), nos son necesarios los nombres de los parámetros que vamos a usar en la llamada, así como sus tipos y el tipo de dato de respuesta.

Estos parámetros vienen especificados dentro del WS que vamos a invocar. Podremos cargarlos dinámicamente o accediendo antes de codificar al WSDL. Dinámicamente se podría acceder de cualquiera de las maneras que permitan leer y obtener datos de un XML, ya que al fin y al cabo eso es lo que es.

Adjuntamos un ejemplo de WS en lenguaje Java en el que vamos a ver el método **runService** y su parámetro de entrada sería un **string** de nombre **strXML**. Mientras que el nombre del parámetro de vuelta nos da igual pero si nos interesa su tipo, **string**.

```
<?xml version="1.0" encoding="UTF-8"?>

 <s:schema xmlns:s="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified"
      targetNamespace="http://www.openuri.org/">

      <s:element name="runService">
        <s:complexType>
          <s:sequence>
            <s:element name="strXML" type="s:string" minOccurs="0"/>
          </s:sequence>
        </s:complexType>
      </s:element>

      <s:element name="runServiceResponse">
        <s:complexType>
          <s:sequence>
            <s:element name="runServiceResult" type="s:string" minOccurs="0"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="string" nillable="true" type="s:string"/>
</s:schema>
```

Ahora vamos con un ejemplo de cómo realizar su invocación:

- Por un lado, la aplicación que necesita invocar al webservice:

```
package src;

/**
 * Aplicación para probar los webservices del lote2 usando SOAP (la librería de
 * soap que viene con el weblogic)
 *
 * @author cd00040t
 */
public class AppWSTest {

    // url del web service
    private static final String URL_JWS =
"http://webldes53:7221/q99lCAT/com/ejie/services/Q99lWebServiceWS.jws";

    // nombre del servicio (método)
    private static final String METHOD_STRING = "runService";

    // array con los nombres de los parámetros.
    private static final String[] PARAM_NAMES = { "strXML" };

    // array con los parámetros.
    // (en este caso, es un String que contiene un xml)
    private static final String[] PARAM_VALUES = { "<servicerequest><data> "+
    "<parametros>"+
        "<parametro nombre=\"DNI\"
valor=\"java.lang.String\">000049</parametro>"+
        "<parametro nombre=\"CIF_EMPRESA\"
valor=\"java.lang.String\">A48035141</parametro>"+
    "</parametros></data></servicerequest>" };



    /**
     * Método main de la aplicación.
     *
     * @param args
     */
    public static void main(String[] args) {

        WebService webService = new WebService (URL_JWS,
                                                METHOD_STRING,
                                                PARAM_NAMES,
                                                PARAM_VALUES);

        try {
            String resultado = webService.invoke();

            System.out.println(". El resultado de la invocación es: " +
resultado);
        }
        catch (Exception e) {
```

```
            System.err.println("Exception: "+e.getMessage());
        }
    }
}
```

- Por otro lado, una clase que encapsula las invocaciones a web services y que es llamada desde la aplicación anterior:

```
public class WebService {

    // prefix - String giving the prefix of the namespace.
    // Siempre es "" para los webservices de integración
    private static final String PREFIX = "";
    // uri - a String giving the URI of the namespace
    // Siempre es "http://www.openuri.org/" para los webservices de integración
    private static final String TARGET_NAMESPACE = "http://www.openuri.org/";

    private String url;
    private String method;
    private String[] paramNames;
    private String[] paramValues;
    private String prefix;

    private String targetNamespace;

  /**
     * Constructor del R02SWebService.
     *
     * Define el prefix y el targetNamespace por defecto.
     *
     * @param url
     * @param method
     * @param paramNames
     * @param paramValues
     */
    public WebService (String url, String method, String[] paramNames, String[]
paramValues) {
        this.url = url;
        this.method = method;
        this.paramNames = paramNames;
        this.paramValues = paramValues;

        this.prefix = PREFIX;
        this.targetNamespace = TARGET_NAMESPACE;

    }
/**
     * Invoca al web service definido por this.
     *
     * @return el resultado de la invocación soap
     *
     * @throws Exception
     */
    public String invoke () throws Exception {
```

```
        String rtdo = "";
        try {

            // creamos el mensaje SOAP.
            MessageFactory mfactory = MessageFactory.newInstance();
            SOAPMessage message = mfactory.createMessage();

            // conseguimos la parte SOAP del mensaje
            SOAPPart soapPart = message.getSOAPPart();
            // conseguimos el envelope del soapPart
            SOAPEnvelope envelope = soapPart.getEnvelope();
            // conseguimos el body del envelope
            SOAPBody body = envelope.getBody();

            // conseguimos el tnsname del método al que se va a llamar
            Name name = envelope.createName(this.method, this.prefix,
this.targetNamespace);

            // creamos un new SOAPBodyElement objeto con el name especificado,
y
            // lo añadimos al SOAPBody. Conseguimos la referencia al
SOAPBodyElement
            // creado
            SOAPBodyElement element = body.addBodyElement(name);

            // añadimos al body los elementos con los parámetros. Hay que
conocer
            // el nombre del parámetro. (Aunque se podría conseguir con el
wsdl)
            for (int i=0; i < this.paramNames.length && i <
this.paramValues.length; i++) {
                SOAPElement param =
element.addChildElement(envelope.createName( this.paramNames[i]) );
                param.addTextNode( this.paramValues[i] );
            }

            System.out.println("----------------------");
            System.out.println("-- MESSAGE: ");
            System.out.println("----------------------");
            message.writeTo(System.out);
            System.out.println("\n----------------------");

            // conseguimos una conexión SOAP.
            SOAPConnectionFactory factory =
SOAPConnectionFactory.newInstance();
            SOAPConnection con = factory.createConnection();

            // hacemos la llamada SOAP y recuperamos la respuesta
            SOAPMessage response = con.call(message, this.url);

            // obtenemos el resultado en forma de String
            rtdo = getResultado(response);
            System.out.println("WebService: El resultado es: ["+rtdo+"]");

            System.out.println("SOAP Message recibido:");
```

```
            System.out.println("----------------------");
            response.writeTo(System.out);
            System.out.println("");
            System.out.println("----------------------");

        }
        catch (SOAPFaultException e) {
            System.err.println("SOAPFaultException: "
                    + e.getFaultActor() + " - "
                    + e.getFaultCode() + " - "
                    + e.getFaultString() + " - "
                    + e.getMessage());

            e.printStackTrace();
        }
        catch (SOAPException e) {
            System.err.println("SOAPException: "+ e.getMessage());

            e.printStackTrace();
        }

        return rtdo;

    }


    private static SOAPBodyElement getSOAPBodyElement (SOAPMessage response)
throws SOAPException {
        SOAPBodyElement rtdo = null;

        Iterator iterator =
response.getSOAPPart().getEnvelope().getBody().getChildElements();
        while (iterator.hasNext()) {
            Object oUndefined = iterator.next();

            if (oUndefined instanceof SOAPBodyElement) {
                rtdo = (SOAPBodyElement) oUndefined;
                break;
            }
        }
        return rtdo;
    }
    private static String getResultado (SOAPMessage response) throws
SOAPException, Exception {
        String rtdo = "";

        System.out.println("response:\n"+response.toString()+"\n");

        SOAPBodyElement soapBodyElement = getSOAPBodyElement(response);

System.out.println("soapBodyElement:\n"+soapBodyElement.toString()+"\n");

        if (soapBodyElement instanceof SOAPBodyElementImpl) {
            // todo ha ido bien... recogemos el resultado.
```

```
          // String responseElementName =
soapBodyElement.getElementName().toString();
          // String resultElementName =
responseElementName.replaceAll("Response$", "Result");
          String resultElementName = "result";

          Iterator iterator = soapBodyElement.getChildElements();
          while (iterator.hasNext()) {
              SOAPElement element = (SOAPElement) iterator.next();

              String elementName = element.getElementName().toString();

              if (elementName.equals(resultElementName)) {
                  // el element es el elemento resultado.
                  rtdo = element.getValue();
                  break;
              }
          }
        }
        else if (soapBodyElement instanceof SOAPFaultImpl) {
            // ha ocurrido un error

            SOAPFault fault = (SOAPFault) soapBodyElement;

            // tratamos el SOAPFault
            // nota: este método lanza una Exception
            tratarSOAPFault(fault);
        }
        else {
            throw new Exception("El SOAPBodyElement no es una instancia de
SOAPBodyElementImpl ni de SOAPFaulImpl");
        }

        return rtdo;
    }
    /**
     * Tratamiento del SOAPFault.
     *
     * @param fault
     *
     * @throws Exception
     */
    private static void tratarSOAPFault(SOAPFault fault) throws Exception {
        StringBuffer mensajeError = new StringBuffer();
        mensajeError.append("SOAPFault:\n");
        mensajeError.append("  faultCode: "+fault.getFaultCode()+"\n");
        mensajeError.append("  faultString: "+fault.getFaultString()+"\n");

        Detail detail = fault.getDetail();
        StringBuffer detailValue = new StringBuffer();
        Iterator it = detail.getDetailEntries();
        while (it.hasNext()) {
            DetailEntry detailEntry = (DetailEntry) it.next();
```

```
            detailValue.append("  detailEntry: "+detailEntry.getValue() +
"\n");
        }

        throw new Exception(mensajeError.toString());
    }
}
```

2.1.- Tratamiento de Excepciones

Cuando se produce un error en la ejecución de un servicio web, lo que el cliente recibe es un elemento **SOAP Fault** en el elemento SOAP Body del mensaje SOAP. Sólo puede haber un elemento SOAP Fault.

Un elemento SOAP Fault contiene información del error que se ha producido. Contiene los siguientes elementos:

- faultCode: obligatorio. Valores que puede tomar:
  - Client: el SOAPMessage no es correcto o no contiene la información suficiente. El error está provocado por lo que manda el cliente.
  - Server: el SOAPMessage no se ha procesado correctamente por un error de procesamiento en el servidor.
  - VersionMismatch: el namespace del SOAPEnvelope no es válido.
  - MustUnderstand: un elemento de un SOAPHeader tiene el atributo mustUnderstand a true, pero el servidor no lo reconoce.
- faultString: obligatorio. Texto explicativo del error.
- faultActor: opcional. Actor que ha causado el error. En nuestros servicios web no estamos definiendo actores. En la invocación de los servicios web de integración son irrelevantes.
- elemento Detail: opcional. Contiene los detalles del error.

En el código de ejemplo anterior se muestra cómo utilizar el SOAPFault en las invocaciones a web services.

3.- Utilización del Webservice Autoprohibidos

Para utilizar el servicio web de validación es necesario solicitar una autorización a la Dirección de Juegos y Espectáculos, quien asignará un **usuario** y **contraseña** para acceder al servicio.

También se deberá solicitar una contraseña de cifrado con el fin de poder encriptar los parámetros de consulta y desencriptar la respuesta obtenida, mediante el algoritmo de cifrado de AES

La url del servicio para el **entorno de pruebas** es:

http://svc.integracion.test.euskadi.net/ctxweb/v33aaApuestasWar

El servicio dispone de una operación de nombre **procesarLlamadaEncrypt** que en un futuro próximo **quedará en desuso**, que recibe como parámetro un xml de consulta, el cual se envía cifrado con una clave de 128 bits que habrá sido previamente facilitada por el departamento.

También dispondrá de una nueva operación llamada **procesarLlamadaEncryptV2** que **sustituirá** a **procesarLlamadaEncrypt,** la cual se envía con una clave de 128 bits y una clave de inicialización de vector de 128 bits, que ambas habrán sido facilitadas previamente por el departamento.

- El xml de consulta tiene la siguiente estructura:
  - **Usuario**: Usuario que intenta acceder a al servicio Web.
  - **Password**: Password del usuario que intenta acceder a al servicio Web (base64).
  - **Metodo**: método que se desea consultar.
  - **Dni**: Documento de la persona que la cual el servicio Web nos debe indicar si tiene acceso o no lo tiene. Obligatorio.
  - **TipoDocumento**: Tipo de documento de identificación. DNI, NIE. Obligatorio.
  - **Nombre:** Nombre de la persona titular que se consulta. Obligatorio para el *metodo* de ***validarRegistro.*** (Campo nuevo).
  - **Apellido1:** Primero apellido de la persona titular que se consulta. Obligatorio para el *metodo* de ***validarRegistro.*** (Campo nuevo).
  - **Fecha nacimiento:** Fecha de nacimiento con formato AAAAMMDD. Obligatorio para el *metodo* de ***validarRegistro.***

El campo llamado *metodo* que puede tomar estos dos valores para validar el control de acceso en las apuestas online.

- Método *validarRegistro* el cual deberá utilizarse en el momento que un usuario se registra en la aplicación de apuestas online, donde se comprueba que el usuario sea mayor de edad y que no exista en el registro de prohibidos de la Dirección de Juegos y Espectáculos.

- Método v*alidarAcceso* el cual deberá utilizarse en el momento que un usuario se realiza el login a la aplicación de apuestas online, donde se comprueba que el usuario no exista en el registro de prohibidos de la Dirección de Juegos y Espectáculos.

EJEMPLO DEL XML DE CONSULTA:

```
<?xml version="1.0" encoding="UTF-8"?>
    <Consulta>
        <usuario>AP265</ usuario >
        <password>ZXMvcGxhdGVhL2Rlc2NhcmdhR</ password > (base64)
        <metodo>validarRegistro</metodo>
        <dni>30000067V</dni>
        <tipoDocumento>DNI</tipoDocumento>
        <nombre>JUAN</nombre>
        <apellido1>GARCIA</apellido1>
        <fechaNacimiento>19801201</fechaNacimiento>
    </Consulta>
```

EJEMPLO DE PETICION CIFRADA DEL METODO PROCESARLLAMADAENCRYPT:

```
<soapenv:Envelope
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:ejie="http://www.ejie.es/">
<soapenv:Header>
    <wsse:Security
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <wsse:UsernameToken>
        <wsse:Username>481912E6A621206A0941714966ADA1DE</wsse:Username>
        <wsse:Password>C665450822C958794BB49508065E6D22</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
<soapenv:Body>
    <ejie:procesarLlamadaEncrypt
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <xml xsi:type="xsd:string">
74A0225ED5D106FD754B4C4626A9F0BB14CEA917BFD650868877E3213C37AB3367B67200
D6211EC093A4004255A048682B62CEE9C602249AEF28CC423DFAB15177632D2F53550586
2AE0BC6170B043841AE1B7B2B5DBF5DFA19AB9749D76E81863F12F20482A68E7F6AD7AF1
71776969B6F1DB7B1BF8697C9763A0E78A050FFC5BB9EC9CB8714B07F958A03F8F031527A
```

```
3BF076328D24E2E05EF198828952C9819C77AD72CD28E6F8A0CD7833F4AE0EF1DAA9A4D5
DD612FD1F496EF34F1A9B150C0B29100BB2F9AEA0BC6F09A0FCFA213C9557C0F2B456528
ABCB4F66F76C90C
          </xml>
          <ip xsi:type="xsd:string">
            <![CDATA[192.168.2.2]]>
          </ip>
        </ejie:procesarLlamadaEncrypt>
      </soapenv:Body>
    </soapenv:Envelope>
```

EJEMPLO DE PETICION CIFRADA DEL METODO PROCESARLLAMADAENCRYPTV2:

```
    <soapenv:Envelope
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ejie="http://www.ejie.es/">
    <soapenv:Header>
      <wsse:Security
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
        <wsse:UsernameToken>
      <wsse:Username>39554D71345542626252657347745071 72C8B46C191980ABD8C5
F6B2D03B7C440D0751EE383D45C8772809600108492F5FE2D0226638920426AC28B93D04
4343</wsse:Username>
        <wsse:Password>39554D71345542626252657347745071 4ACF7D4CFE815782A4390
0CBFA283B41C555400DD408ACF1BFA6CCAECCD305B1494A6FF3C869919ADB94A7449CEE
58F4</wsse:Password>
        </wsse:UsernameToken>
      </wsse:Security>
    </soapenv:Header>
    <soapenv:Body>
      <procesarLlamadaEncryptV2
        soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        xmlns="http://www.openuri.org/">
        <xml xsi:type="xsd:string">
    39554D71345542626252657347745071 1DB9FB8EAEC482CAA5AAADF645143FB7B946
916F4DC6EB7DD22F57F8F255D8002A1D017CA832269511BE03B9226079859DB87AB111FE
```

```
EEA9BAF2FD22CC18A9DA1922B04C6A79114A10FE58907220C0A115AB3E20BB2A7CCB48EB
6ED233436833626DCEF335993C6DF7A3F3739DB366D5A352BC5F38098C593866D5365D40
CE98278C377D3FC25FC1CEA9A0723B143A458EF15A9C478C6DD92E03D0FE095B446AC53D
ED66D7E3C136EEA371575BB4505DBBEF678F17F29D7776D883CB872F917C8814C68A0B52
F93B970ABE15D8E36C68829189D0DA7004237BA6A39A3F8D2C2274009183E408D81FCCF9
115BA132978D70DB7772B7F8854750052E731FF2F626
            </xml>
            <ip xsi:type="xsd:string">
                <![CDATA[192.168.2.2]]>
            </ip>
        </procesarLlamadaEncryptV2>
    </soapenv:Body>
    </soapenv:Envelope>
```

La respuesta del servicio Web del método **procesarLlamadaEncrypt** nos devolverá un texto cifrado en AES con la misma clave que la petición de consulta, con una estructura en formato XML.

La respuesta del servicio Web del método **procesarLlamadaEncryptV2** nos devolverá un texto cifrado en AES y la misma clave de inicialización de vector con la misma clave que la petición de consulta, con una estructura en formato XML.

La estructura del XML que nos devuelve el servicio Web seria de la siguiente manera:

- Si el usuario tiene acceso:

```
<?xml version="1.0" encoding="UTF-8"?>
    <Respuesta>
        <metodo>validarAcceso</metodo>
        <dni>00000000T</dni>
        <prohibida>N</prohibida>
        <errorConsulta/>
    </Respuesta>
```

- Si el usuario **NO** tiene acceso:

```
<?xml version="1.0" encoding="UTF-8"?>
    <Respuesta>
        <metodo>validarAcceso</metodo>
        <dni>00000000T</dni>
        <prohibida>S</prohibida>
        <errorConsulta/>
    </Respuesta>
```

- Cuando devuelve un **error controlado**

```
<?xml version="1.0" encoding="UTF-8"?>
    <Respuesta>
        <metodo>validarAcceso</metodo>
        <dni>00000000T</dni>
        <prohibida></prohibida>
        <errorConsulta>
            <IdError>5</IdError>
            <DescError>Introduzca su contraseña</DescError>
        </errorConsulta>
    </Respuesta >
```

3.1.- Tipos de errores controlados por el servicio Web

IdErrror	Descripción Error
5	Nombre del método no válido
6	DNI obligatorio
7	Tipo de documento obligatorio
8	Fecha de Nacimiento obligatorio
9	Nombre obligatorio
10	Apellido1 obligatorio
Interoperabilidad (servicio de Consulta o Verificación de Datos de identidad)	
[0901]	Servicio no disponible, por favor, inténtelo más tarde.
[0233]	Titular no identificado
[0231]	FORMATO DE DOCUMENTO ERRÓNEO

Tipos de errores que se devuelven dentro del ***SoapFault***

IdErrror	Descripción Error
XE3	Los datos del usuario o contraseña son incorrectos
XE4	Usuario y contraseña obligatorios

A continuación, se indica el schema que describe la estructura y las restricciones de los contenidos de los XML de intercambio del servicio web.

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:apu="com/ejie/v34bb/schemas/apuestasET/V34bbApuestasET.xsd"
  targetNamespace="com/ejie/v34bb/schemas/apuestasET/V34bbApuestasET.xsd">

  <xs:complexType name="ConsultaET">
    <xs:sequence>
      <xs:element name="usuario" type="xs:string"/>
      <xs:element name="password" type="xs:string">
        <xs:annotation>
          <xs:documentation>clave codificada en Base64</xs:documentation>
        </xs:annotation>
      </xs:element>
```

```
    <xs:element name="metodo" type="xs:string">
      <xs:annotation>
        <xs:documentation>validarRegistro: donde se comprueba que el usuario sea
mayor de edad y que no exista en el registro de prohibidos de la Dirección de
Juegos y Espectáculos.
        </xs:documentation>
        <xs:documentation>validarAcceso: donde se comprueba que el usuario no
exista en el registro de prohibidos de la Dirección de Juegos y Espectáculos.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="dni" type="xs:string"/>
    <xs:element name="tipoDocumento" type="xs:string">
      <xs:annotation>
        <xs:documentation>DNI, NIE</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="fechaNacimiento" type="xs:string">
      <xs:annotation>
        <xs:documentation>Fecha de nacimiento con formato AAAAMMDD. Obligatorio
para el metodo de validarRegistro.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="nombre" type="xs:string">
      <xs:annotation>
        <xs:documentation>Obligatorio para el metodo de validarRegistro.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="apellido1" type="xs:string">
      <xs:annotation>
        <xs:documentation>Obligatorio para el metodo de validarRegistro.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="RespuestaET">
  <xs:sequence>
    <xs:element name="metodo" type="xs:string"/>
    <xs:element name="dni" type="xs:string"/>
    <xs:element name="esProhibido" type="xs:string">
      <xs:annotation>
        <xs:documentation> S - El dni NO tiene acceso porque está
prohibido</xs:documentation>
        <xs:documentation> N - El dni SI tiene acceso </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="errorConsultaET" type="apu:ErrorConsultaET"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="ErrorConsultaET">
  <xs:sequence>
    <xs:element name="idError" type="xs:string"/>
```

```
    <xs:element name="descripcion" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
</xs:schema>
```