PROTOCOLO DE LAS APLICACIONES O SISTEMAS QUE, PARA SU FUNCIONAMIENTO, DEBEN DE GARANTIZAR EL CONTROL DE MENORES Y LA NO INCLUSIÓN EN EL FICHERO DE REGISTRO DE INTERDICCIONES DE ACCESO AL JUEGO DEL PAIS VASCO

DIRECCIÓN DE JUEGO Y ESPECTÁCULOS

Fecha: 18 de enero de 2022

ÍNDICE

1 OBJETO

El presente procedimiento tiene por objeto describir las condiciones a cumplir por una aplicación o por un sistema que tiene que comprobar que una persona no sea menor y que no está incluida en el Registro de Interdicciones de acceso al juego de Euskadi.

2 ALCANCE

Este documento es aplicable a todos las aplicaciones y sistemas que deban de cumplir esas condiciones.

3 REFERENCIAS

Para la realización del protocolo se tienen como base los documentos siguientes:

a) Decreto 120/2016, de 27 de julio, por el que se aprueba el Reglamento General de Juego en la Comunidad Autónoma de Euskadi.

b) Orden de 17 de marzo de 2017, de la Consejera de Seguridad, por la que se regulan los requisitos y características técnicas de las máquinas de juego y sus condiciones de interconexión.

4 GENERAL

La empresa realizará el desarrollo de una invocación de Web-Service para comprobar que se accede a los servicios de autentificación en el servidor de pruebas, en un primer momento, y tras superar este paso, en el servidor de producción.

La empresa deberá, EN TODAS LAS OCASIONES, hacer la comprobación de que la persona que quiera acceder al local o a la aplicación no figura en el fichero del Registro de Interdicciones de acceso al juego de Euskadi.

Es imprescindible que el sistema, o personal del local, verifiquen la identidad de la persona que pretende acceder y confirmar que realmente es esa persona; ya que, si

solo se verifica la documentación, se podría acceder con un documento que no sea de la persona que accede.

Este fichero contiene actualmente las inscripciones de carácter voluntario, las inscripciones por resolución de la Dirección, las efectuadas por personal sanitario y las efectuadas por familiares debidamente justificadas, efectuadas en Euskadi, a la que se suman diariamente las inscripciones efectuadas con el mismo fin las de carácter estatal (Ministerio de Consumo).

La primera vez que se accede, se comprobará que no es menor y también que no figura en el registro de autoprohibidos. Las siguientes veces, si los datos previamente se han grabado en el fichero interno de la empresa, solo es necesario la comprobación de no inclusión en el fichero de autoprohibidos.

Para estas dos funciones, existen llamadas a realizar en el manual adjunto y, así, comprobar los resultados.

La Unidad de Juego de la Ertzantza (UJE) para poder comprobar “in situ” si una persona ha pasado por el control establecido, debe poder consultar el DNI o documento acreditativo y hora de paso por el control en la aplicación o sistema a instalar.

La empresa podrá guardar datos de registro e identificativos de las personas que acceden bajo su responsabilidad y con sus correspondientes declaraciones de ficheros de carácter personal. La pertenencia a este fichero no exime de su comprobación previa y obligatoria a través del diseño del Web-Service establecido.

Los sistemas a establecer en los locales de juego, basados en barreras, tornos o sistemas similares, tienen que ser desbloqueados ante una emergencia de desalojo. Este modo de desbloqueo será automático y unido al sistema de alarmas si cuando se

produce, por ejemplo, un incendio, el sistema de alarma se activa y puede proceder al desbloqueo automático. Si, por el contrario, el hecho causante no activa la alarma (inundación, atraco…), que se pueda realizar de forma manual por una persona.

Estos sistemas a instalar deberán de realizar el control del aforo del local.

No pueden disminuir la capacidad de evacuación del local.

Deben de cumplir el decreto 68/2000 de accesibilidad del Gobierno Vasco y el CTE-DB SUA.

5 DESARROLLO

La Dirección de Juego y Espectáculos pondrá a disposición de la empresa que lo solicite de un manual denominado “Web-Service de consulta del Registro de Autoprohibidos” que servirá como base para el desarrollo de ese Servicio Web por parte de la empresa.

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/ECB y una clave de 128 bits. Igualmente, la respuesta obtenida será devuelta cifrada bajo la misma clave.

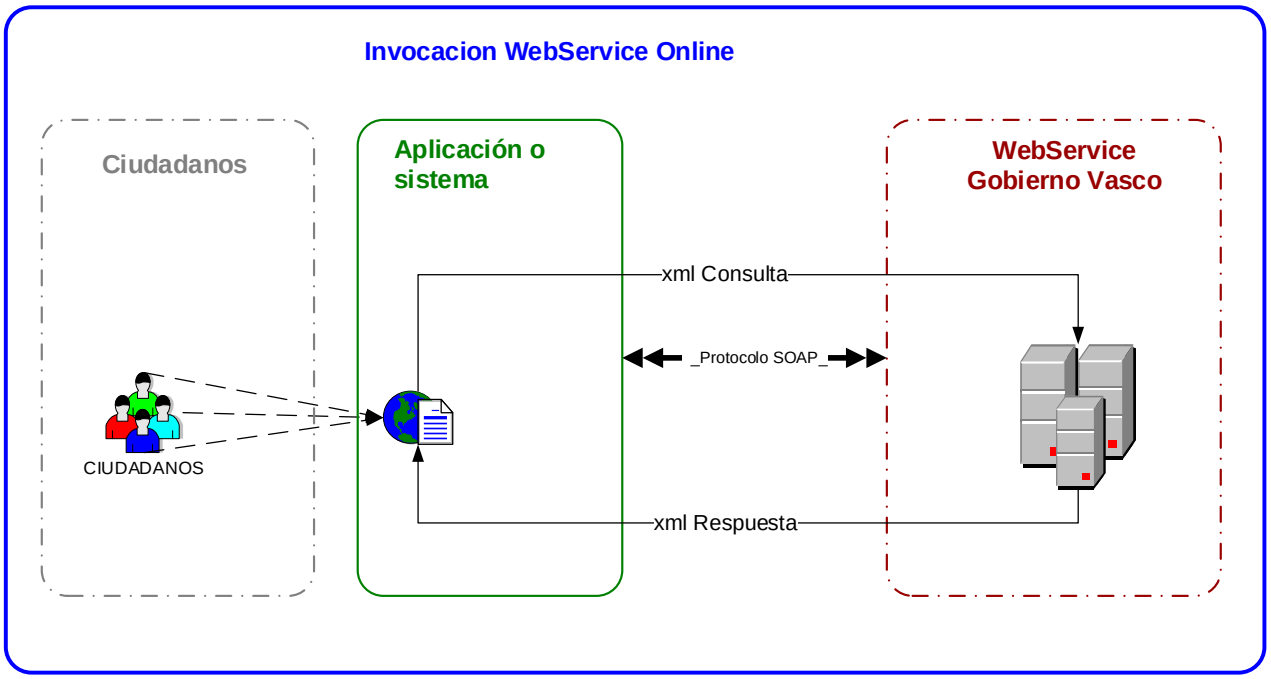
El contenido del manual es el siguiente:

6 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 dichos estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas.



7 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());
    }
}
```

7.1 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.

8 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 **temporal** para el entorno de **pruebas** es:

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

El servicio dispondrá de una nueva operación de nombre **procesarLlamadaEncrypt** que recibirá como parámetro un xml de consulta, el cual se enviará cifrado con una clave de 128 bits que habrá sido previamente facilitada 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 (que también ha de estar codificado previamente en base64)
  - **Metodo**: método que se desea consultar
  - **Dni**: DNI de la persona que la cual el servicio Web nos debe indicar si tiene acceso o no lo tiene
  - **TipoDocumento**: tipo documento de identificación DNI, NIF, NIE
  - **Fecha nacimiento:** Fecha de nacimiento con formato AAAAMMDD. Obligatorio para el *método* de ***validarRegistro***

El campo llamado *metodo* 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>validarAcceso</metodo>
    <dni>00000000T</dni>
    <tipoDocumento>DNI</tipoDocumento>
    <fechaNacimiento>19801201</fechaNacimiento>
</Consulta>
```

EJEMPLO DE PETICION COMPLETA CON EL XML DE CONSULTA CIFRADO:

```
<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:Userna
me>

<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">
74A0225ED5D106FD754B4C4626A9F0BB14CEA917BFD650868877E3213C37AB33
67B67200D6211EC093A4004255A048682B62CEE9C602249AEF28CC423DFAB151
77632D2F535505862AE0BC6170B043841AE1B7B2B5DBF5DFA19AB9749D76E818
63F12F20482A68E7F6AD7AF171776969B6F1DB7B1BF8697C9763A0E78A050FFC5
BB9EC9CB8714B07F958A03F8F031527A3BF076328D24E2E05EF198828952C9819
C77AD72CD28E6F8A0CD7833F4AE0EF1DAA9A4D5DD612FD1F496EF34F1A9B150
C0B29100BB2F9AEA0BC6F09A0FCFA213C9557C0F2B456528ABCB4F66F76C90C
                    </xml>
                    <ip xsi:type="xsd:string">
                        <![CDATA[192.168.2.2]]>
                    </ip>
                </ejie:procesarLlamadaEncrypt>
            </soapenv:Body>
        </soapenv:Envelope>
```

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

EJEMPLO DE LA RESPUESTA CON EL XML CIFRADO:

```
        <env:Envelope
            xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
```

```
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <env:Header/>
        <env:Body
          env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
          <m:procesarLlamadaEncryptResponse
            xmlns:m="http://www.ejie.es/">
            <result
      xsi:type="xsd:string">E03914AB31E7023B700840EEBA92C44FCC404178DE
B70F4766E30772F32E8EC0C5201BC67DF34A8DC860FFC8B77CF4DDFC96D33F5A
5CCA337C241AA03D6244DF9CC5EA924401F92D295E31659D6F8FDA7FF64C3DFC
71BF01E2F867EF3B465F6F4A720E49925E5C2C06EA9B1E8B84F3AC</result>
          </m:procesarLlamadaEncryptResponse>
        </env:Body>
      </env:Envelope>
```

Tras descifrar el cuerpo de la respuesta que nos devuelve el servicio Web, estructura del XML 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 >
```

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
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.xs
d">

      <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,
NIF</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: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>
```

9 PREGUNTAS Y RESPUESTAS FRECUENTES

P-¿Se puede consultar sobre el Webservice Autoprohibidos mediante documentos diferentes a DNI, NIF, NIE? En caso afirmativo ¿qué otros tipos de documentos se pueden consultar y qué denominación se debe enviar entre las etiquetas "<dni></dni>" del xml que se envía para cadauno de esos otros tipos de documentos?

R- En Validar Registro, la forma de hacer la consulta sobre si una persona es mayor de 18 años es contra un servicio de la Dirección General del Policía, que solo admite como tipos DNI", "NIE" y "PASAPORTE". Validar Acceso si puede hacerse con otros tipos, ya que será con el documento con el se ha realizado la autoprohibición.

P- ¿Se puede descargar el listado de prohibidos (con alguna modificación del endpoint) para tenerlo disponible cuando caiga la conexión con vuestro endpoint o si falla vuestra base de datos?

R- No se puede debido a que son datos sensibles protegidos especialmente por la LOPD.

P- ¿En caso de error cómo lo notificáis?

R-Se muestran mensajes de error al invocar al servicio descritos anteriormente.

P- ¿Cuándo tenemos que usar Validar Acceso y cuándo Validar Registro? ¿O sólo tenemos que usar Validar Acceso?

R -ValidarAcceso, consulta si está prohibida una persona en el registro de interdicciones de juego. Se debe realizar cuando la persona ya se ha dado de alta. Este servicio es mas rápido que validarRegistro

ValidarRegistro consulta si una persona está inscrita en el registro de interdicciones de juego y es mayor de 18 años. Solo se debe usar cuando se tiene que dar de alta una persona.

P- ¿Con qué periodicidad actualizáis el listado de prohibidos?

R- Diariamente

P-¿Que respuesta da el servicio web?

R- Se recibe la misma respuesta en cualquiera de los métodos. Devuelve S si está inscrito en el registro o no es mayor de 18 años o N si no está y es mayor a 18 años.