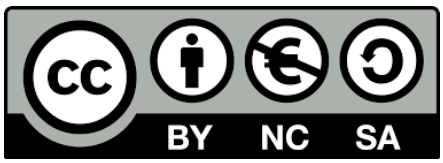
SERVICIOS WEB

Antonio Boronat Pérez

Dep. Informàtica

IES Joan Coromines (Bló)



Índice

Introducción

Un Servicio Web es un software basado en tecnologías estandarizadas que faciliten el intercambio de información entre aplicaciones que pueden estar desarrolladas usando diferentes plataformas y lenguajes de programación, conectadas en una red de ordenadores como puede ser Internet.

Los estándares de los Servicios Web están regulados por las organizaciones W3C y OASIS. Y en su desarrollo intervienen los protocolos siguientes:

- XML
- SOAP
- WSDL
- UDDI
- REST

Los Servicios Web usan el formato de texto (XML) para el intercambio de información sobre HTTP, esto tiene la ventaja de facilitar el acceso a contenidos posibilitando que aplicaciones desarrolladas con diferentes tecnologías puedan comunicarse de forma sencilla, ya que sólo deben interpretar contenidos codificados en XML y no en algún formado propietario. Por contra, la transmisión y procesado de estos contenidos en texto XML puede ser menos rápido que empleando tecnologías que usan modelos diseñados específicamente para el intercambio de información, aunque presenten mayor complejidad en el desarrollo de las aplicaciones que deban comunicarse.

Los Servicios Web suelen ofrecer una serie de métodos que son invocados por sus aplicaciones cliente a través de una red. Es posible, que se encadenen series de llamadas entre diferentes Servicios Web, actuando según sea el caso como cliente o servidor y así conseguir la ejecución de operaciones complejas. Por ejemplo, un Servicio Web puede atender una solicitud para comprar un coche, al mismo tiempo esta solicitud se puede dividir en dos solicitudes que ahora son cliente, de forma que una accede a una web con listados de morosos y otra a una entidad financiera para que prepare un plan de financiación para la compra. Cuando se procesen las solicitudes se devuelve la información y con éstas a la solicitud del cliente. Vemos que se pueden encadenar invocaciones a Servicios Web obteniendo resultados complejos procedentes de sistemas heterogéneos.

Para facilitar la interacción con los Servicios Web, estos deben disponer de una descripción de si mismos, de forma que los puedan usar las aplicaciones cliente. Además, los Servicios Web deben ser sencillos de identificar para poder localizarlos y aprovechar sus funcionalidades.

Resumiendo, un Servicio Web ofrece una serie de métodos a sus clientes con los que se comunicará mediante el uso de estándares, principalmente los mensajes viajarán con HTTP y usando el protocolo SOAP en el acceso al servicio, quien emplea XML para la codificación de la información intercambiada. A su vez, el Servicio Web describe sus características con el estándar WSDL y esta descripción será publicada para facilitar su localización en un servicio de registro basado en el estándar UDDI.

A continuación, describiremos los protocolos y estándares mencionados.

SOAP (Simple Object Access Protocol)

Consiste en un protocolo estandarizado donde se define la forma en que pueden comunicarse dos aplicaciones intercambiando información en formato XML. Y es uno de los elementos fundamentales para el desarrollo de aplicaciones Web. SOAP facilita el envío de mensajes donde la comunicación se establece sin estado, de forma que es un mecanismo básico con el que las aplicaciones pueden añadir la complejidad que necesiten para construir un sistema de comunicación sofisticado.

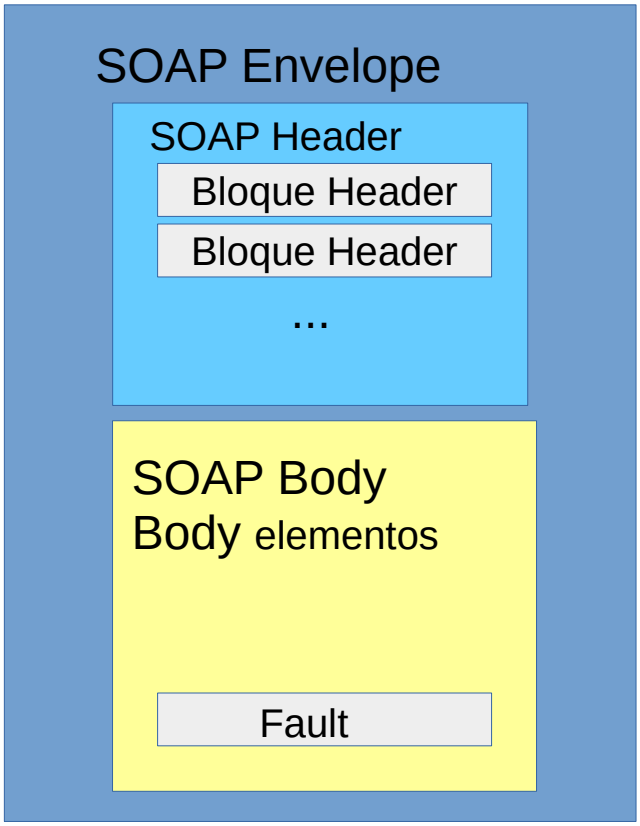
SOAP tiene tres partes:

- Sobre (envelope): define los mensajes, su contenido y utilización.
- Reglas de codificación: define los tipos de datos.
- Conversación: donde se establece la forma en que se llama a los procedimientos y su respuesta.

Los mensajes XML de SOAP tienen una estructura en la que se guarda la información a enviar. El mensaje tiene obligatoriamente un Sobre (Envelope) y un Cuerpo (Body), además podemos encontrar unas Cabeceras (Head) y una especificación de Errores (Fault).

Los bloques Header son opcionales y sirven para indicar cómo se debe procesar el mensaje, bien en su destino o durante su ruta.

Los elementos Body contienen los datos del mensaje y si se establecen los elementos Fault indicarán la forma de tratar los errores en el procesado o durante el envío del mensaje entre los elementos de la comunicación.



En el proceso de comunicación con SOAP intervienen una serie de elementos, donde los principales son SOAP Sender y SOAP Receiver aunque el conjunto lo forman:

- SOAP Sender: Elemento que envía el mensaje SOAP.
- SOAP Receiver: El que lo recibe.
- SOAP Message Path: Es el conjunto de elementos por los que debe pasar el mensaje desde el SOAP Sender, los intermedios, si los hay y finalmente el SOAP Receiver destinatario del mensaje.
- Initial SOAP Sender: Elemento que origina el mensaje, el primer SOAP Sender.
- SOAP Intermidiary: Son nodos intermedios que en un momento dado actúa como SOAP Receiver y a continuación como SOAP Sender para que el mensaje siga su camino.

- Ultimate SOAP Receiver: Es el destinatario final de los mensajes y quien ha de procesarlo.

Ejemplo de mensajes SOAP, primero la solicitud y después la respuesta:

```
<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <SOAP-ENV:Body>
            <SOAP-ENV:temporada>
                  <temporada xsi:type="xsd:string">Invierno</temporada>
             </SOAP-ENV:temporada>
      </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>


<?xml version="1.0" encoding="UTF-8"?>
  <SOAP-ENV:Envelope
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:ns1="http://localhost/toni/fruteria_soap_server_wsdl/"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <SOAP-ENV:Body>
         <ns1:temporadaResponse>
        <returnSOAP-ENC:arrayType="SOAP-ENC:Array[3]" xsi:type="SOAP-ENC:Array">
            <item SOAP-ENC:arrayType="xsd:string[1]" xsi:type="SOAP-ENC:Array">
            <item xsi:type="xsd:string">Naranja</item></item>
            <item SOAP-ENC:arrayType="xsd:string[1]" xsi:type="SOAP-ENC:Array">
            <item xsi:type="xsd:string">Piña</item></item>
            <item SOAP-ENC:arrayType="xsd:string[1]" xsi:type="SOAP-ENC:Array">
            <item xsi:type="xsd:string">Banana</item></item>
            </return></ns1:temporadaResponse>
      </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Finalmente, cabe destacar que SOAP puede ser transportado por diferentes protocolos, pero el habitual es HTTP, tanto por su implantación como por estar habilitado en la mayoría de cortafuegos.

Para facilitar el uso de Servicios Web basados en SOAP se usa una especificación según el estándar WSDL, donde se describe el servicio ofrecido.

WSDL (Web Service Description Languaje)

Este es un estándar basado en XML para la descripción de acceso a Servicios Web. WSDL presenta la interfaz para acceder a Servicios Web por parte de sus posibles clientes. Define los protocolos y formatos de los mensajes para poder usar dichos servicios. Normalmente, un cliente primero estudia la descripción de WSDL y después accede al servicio ofrecido por SOAP.

WSDL muestra cómo usar un Servicio Web, pero no lo que hace. Debemos destacar que WSDL, por ejemplo, no tiene porqué dar el puerto en el que invocar un servicio, así, podemos encontrar que una misma interfaz definida por WSDL sea implementada por diferentes aplicaciones. De forma que un cliente podrá acceder a cualquier puerto que la implemente y acceder al Servicio Web indistintamente de uno u otro.

La estructura de WSDL consta de los siguientes elementos:

- Tipos de datos: <types> Declara los tipos de datos empleados en el mensaje, son tipos definidos en los esquemas XML.
- Mensajes: <message> Define los elementos del mensaje, su constitución lógica y sus tipos.
- Tipos de puerto: <port Type> Indica las operaciones ofrecidas y los mensajes a intercambiar en la comunicación.
- Bindings: <binding> Especifica los protocolos empleados en la comunicación.
- Servicios: <service> Detalla los puertos y su dirección, hace referencia a lo indicado en los apartados anteriores.

UDDI(Universal Description, Discovery and Integration)

UDDI representa un directorio de Servicios Web donde pueden consultar los clientes para encontrar lo que necesitan.

Los servicios que se añaden a UDDI lo hacen mediante XML.

UDDI tiene tres partes:

- Páginas Blancas: tiene la dirección, contacto y otros identificadores.

- Páginas Amarillas: tiene una distribución en categorías industriales.
- Páginas Verdes: contienen información técnica sobres servicios ofrecidos.

UDDI es consultado mediante mensajes SOAP y responde con documentos WSDL donde, como se ha comentado, se muestra la forma de acceder a los Servicios Web.

Las empresas pueden ofrecer sus Servicios Web, registrándose en el sistema que ofrece el UDDI.

Los directorios UDDI se almacenan replicados en servidores de empresas que se han comprometido a seguir las especificaciones del Consorcio UDDI.org y facilitar su acceso a los posibles clientes de Servicios Web. Algunas empresas que mantienen estos servidores son: Microsoft, IBM y se van añadiendo otras como HP.

UDDI se diseña para poder responder a cuestiones como las siguientes:

- ¿Qué interfaces de Servicio Web basadas en WSDL se han publicado y establecido para un sector determinado?
- ¿Qué empresas han escrito una implementación basada en una de estas interfaces?
- ¿Qué Servicios Web, categorizados de algún modo, se ofrecen actualmente?
- ¿Qué Servicios Web ofrece una determinada empresa?
- ¿Con quién se debe poner en contacto el usuario para utilizar los Servicios Web de una empresa?
- ¿Cuáles son los detalles de implementación de un Servicio Web concreto?

Fuente de documentación:https://desarrolloweb.com/articulos/1589.php

REST (Representational State Transfer)

Es un modelo de aplicación distribuido aplicable al entorno Web. Este modelo permite crear Servicios Web pero sin utilizar SOAP. Actualmente, la mayoría de grandes organizaciones en Internet ofrecen un API REST para la creación de aplicaciones (Twitter, YouTube, Facebook, eBay, etc.)

Las características principales de los desarrollos REST son:

- Protocolo Cliente / Servidor sin estado: cada petición HTTP es independiente, no se necesita guardar ninguna información ni estado entre el cliente y el servidor. En algunos casos se usa una caché, pero no es de uso general.
- Las operaciones principales empleadas en REST y HTTP son: POST(crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar). Con estas operaciones se debe poder implementar la aplicación basada en REST.
- Los objetos en REST se manipulan a partir de una URI (Uniform Resource Identifier). Cada recurso que forme parte de una aplicación dispone de su propia URI. Mediante ellas interactuamos con los recursos como por ejemplo para consultarlos, modificarlos o eliminarlos.
- Interfaz Uniforme: La transferencia de información se realiza aplicando las acciones comentadas (POST, GET, PUT, y DELETE) sobre los recursos identificados con su URI. Este modo de operar da uniformidad a las interfaces.
- Sistema de Capas: Se establece una jerarquía entre los componentes de la aplicación y cada nivel se encarga de una funcionalidad.
- Uso de Hipermedios: Se basa en los hiperenlaces HTTP para que las aplicaciones REST permitan el acceso a sus clientes a los recursos apropiados.

Por tanto, como vemos los Servicios Web que implementan REST se basan prácticamente en HTTP, siendo más sencillos que los implementados con SOAP.

Fuente de documentación: Wikipedia y bbvaopen4u.com

Programación SOAP PHP

PHP dispone de clases para facilitar el desarrollo de aplicaciones basadas en Servicios Web SOAP, las clases que usaremos serán:

- SoapServer, permite crear un servicio Web al que se añaden las funciones o clases que usarán los clientes
- SoapFault, establece un objeto derivado de Exception que nos permite mostrar las incidencias producidas al usar un Servicio Web que no ha funcionado correctamente.
- SoapClient, esta clase permite crear objetos que se conectan a un Servicio Web y usar las funcionalidades ofrecidas. Estos objetos se pueden crear usando descripciones WSLD del servicio o indicando la URI y URL de acceso.

Nota: En Debian / Ubuntu con PHP 7.0 se debe instalar el soporte para SOAP como un paquete aparte: `sudo apt-get install php7.0-soap` , si usamos LAMPP no es necesario.

Servidor SOAP

El servicio Web lo podemos ofrecer bien creando las diferentes funciones que desempeñarán los servicios o bien creando una clase y sus métodos darán el servicio.

Las instrucciones para la creación del servicio:

```
include_once 'fruteria.php';//Clase métodos del Servicio Web
require 'WSDLDocument.php'; //Crear el fichero WSDL, si lo usamos

// La URI en la que se encuentra instalada la aplicación.
$uri = "http://localhost/fruteria_soap_server_wsdl/";
$server = new SoapServer(null, array("uri"=>$uri));//Crea el servicio
$server→setClass("fruteria"); // Si usamos una clase, en este caso
                              // fruteria
o bien, se añade cada función que formará el servicio

$server→addFunction("temporada");
$server→addFunction("fruta");
...
$server->handle(); // Añade el manejador que activa el servicio
```

Con el código anterior tendríamos funcionando el servicio Web basado en SOAP, ahora debemos decidir si accedemos a él directamente, es decir que el cliente conoce los nombres de las funciones ofrecidas y los datos devueltos por ellas o si creamos un fichero WSDL donde se describen las características necesarias para su utilización.

Si no usamos WSDL, el cliente debe conocer los valores de la URI y URL, así como los nombres de las funciones y tipos devueltos por estas para poder ser usados en la aplicación cliente del servicio Web.

En caso de crear el fichero WSDL, debemos buscar alguna herramienta que genere este fichero a partir de la clase que ofrece el servicio Web. En Internet se encuentra diferentes aplicaciones para crear estos ficheros, nosotros usaremos *WSDLDocument.php* que podemos obtener de: https://code.google.com/p/wsdldocument/ en downloads podemos descargar el fichero comprimido. Para crear el fichero WSDL debemos introducir comentarios de los métodos, donde se indiquen los parámetros que necesitan y sus tipos siguiendo el formato *PHPDocumentor,* puedes encontrar información sobre esta forma de comentar las aplicaciones en http://www.phpdoc.org/

Siguiendo el ejemplo anterior, la clase quedaría como:

```
class fruteria {
    /**
     * Busca las frutas de la temporada que recibe como parámetro
     * @param string $temporada
     * @return string
     **/
    public function temporada($temporada){
        $con_bd = conexion_bd(SERVIDOR, USER, PASSWD,  BASE_DATOS);
        if($con_bd){
            $sql="SELECT fruta FROM precios WHERE temporada='".$temporada."'";
            if($res = mysqli_query($con_bd, $sql)) {
                if(mysqli_num_rows($res) >= 1){ // Ha encontrado las frutas
                    $res_array = mysqli_fetch_all($res, MYSQLI_NUM);
                }
                else {
                  $res_array=new SoapFault("1","Error Temporada no encontrada");
                }
            }
            $cierre_bd = @mysqli_close($con_bd);
            return $res_array;
        }
    }
    /**
     * Busca los datos de la fruta y temporada indicada
     * @param string $temporada
     * @param string $fruta
     * @return string
     */
```

```
    public function fruta ($temporada, $fruta){
        $con_bd = conexion_bd(SERVIDOR, USER, PASSWD,  BASE_DATOS);
        if($con_bd){
          $sql="SELECT * FROM precios WHERE temporada='".$temporada."' and fruta= '" .$fruta."'";
          if($res = mysqli_query($con_bd, $sql)) {
                if(mysqli_num_rows($res) >= 1){ // Ha encontrado las frutas
                    $res_array = mysqli_fetch_all($res, MYSQLI_ASSOC);
                }
                else {
                   $res_array = new SoapFault("1", "Error fruta no encontrada");
                }
            }
            $cierre_bd = @mysqli_close($con_bd);
            return $res_array;
        }
    }
}// Class fruteria
```

Para crear el fichero WSDL, pondremos el código siguiente en el fichero PHP donde creamos el objeto SoapServer:

```
//Crea el contenido del fichero WSDL, necesita que en la clase
// se comente los parámetros de los métodos.
// Los parámetros son el nombre de la clase, la URL y la URI.
$wsdl = new WSDLDocument("fruteria",
      "http://localhost/fruteria_soap_server_wsdl/index.php",
      "http://localhost/fruteria_soap_server_wsdl/");

header('Content-Type: text/xml');

echo $wsdl->saveXML();
```

Una vez ejecutada la aplicación servidora, en el navegador aparecerá el código XML con la información que debe contener el fichero WSDL, así que la guardamos en un fichero con formato XML (en Firefox: Guardar como... → Página Web, sólo XML). Cuando dispongamos del fichero, podemos comentar las instrucciones anteriores ya que no es necesario volverlas a ejecutar.

El acceso a un servicio Web usando su fichero WSDL, lo podemos realizar accediendo a la URL de este fichero, por ejemplo:

```
$wsdl = "http://localhost/fruteria_soap_server_wsdl/fruteria.wsdl";
```

Pero lo más habitual es que al fichero WSDL se acceda usando una URL con formato de *query string* como:

```
$wsdl = "http://localhost/fruteria_soap_server_wsdl?wsdl";
```

Para que funcione esta opción debemos establecer una redirección de esta URL hacia el fichero WSDL. Para esto en Apache2 necesitamos configurar en el directorio de la aplicación el fichero *.htaccess* siguiendo los pasos:

1. Habilitar el módulo de Apache2 *mod_rewrite* si no lo está.
2. En la configuración del servidor de Apache2, dentro de la directiva

   <Directory path_aplicación>
   AllowOverride All
   </Directory>

3. Crear el fichero .htaccess, siguiendo el ejemplo de la URL:

```
RewriteEngine On
RewriteCond %{QUERY_STRING} wsdl
RewriteRule (.*) /fruteria_soap_server_wsdl/fruteria.wsdl? [R=302,L]
```

Para NGINX añadimos en el fichero de configuración del sitio web, en la directiva *location /* :

```
location / {
    if ($query_string ~ wsdl) {
        return 302 /fruteria_soap/fruteria_soap_server_wsdl/fruteria.wsdl;
    }
    try_files $uri $uri/ /index.php?_url=$uri&$args;
}
```

Y en la aplicación cliente creamos el objeto cliente:

```
$client = new SoapClient($wsdl);
```

Con esta orden ya tiene acceso a los métodos ofrecidos por el servicio Web sin necesidad de indicar nada más, tal y como veremos más adelante.

Cliente SOAP

Los clientes de un servicio Web SOAP deben crear objetos de la clase *SoapClient* pasándole como parámetro el fichero WSDL o la URI y URL del servicio según sea el caso. Con este objeto podemos ejecutar los métodos del servicio y obtener sus resultados.

```
$url = "http://localhost/fruteria_soap_server/index.php";
$uri = "http://localhost/fruteria_soap_server/";
$client = new SoapClient(null, array('location'=>$url, 'uri'=>$uri));
```

o bien:

```
$wsdl = "http://localhost/fruteria_soap_server_wsdl?wsdl";
$client = new SoapClient($wsdl, array("trace" => 1));
```

El constructor de la clase SoapClient, además de los parámetros indicados acepta otros opcionales que modifican su funcionamiento expresados mediante un *array*, como por ejemplo *trace*, que permite obtener los valores XML SOAP de la última solicitud y respuesta.

El objeto *SoapClient* da acceso a funciones que permiten conocer las características del servicio Web, por ejemplo *$obj_SoapClient→ __getTypes()*, que devuelve los tipos de datos empleados en los parámetros y respuestas de los métodos del servicio o *$obj_SoapClient→ __getFunctions()*, que muestra los métodos disponibles. Con *$obj_SoapClient→__getLastRequest()* y *$obj_SoapClient→__getLastResponse()* tenemos una cadena con el código XML SOAP de la última consulta y la respuesta.

```
var_dump($client->__getTypes());
echo "<br>";
var_dump($client->__getFunctions());
echo "<br><br>";
echo "<br>Request: " . htmlentities($client->__getLastRequest()) . "<br>";
echo "<br>Response: " . htmlentities($client->__getLastResponse()) . "<br>";
```

Para acceder a los métodos del servicio se realiza como si fuesen locales, conviene hacerlo dentro de una estructura try – catch para poder gestionar las posibles excepciones:

```
try{
  $frutas = $client->temporada($temporada);
  foreach ($frutas as $fruta){
    foreach ($fruta as $nombre){
       echo $nombre . "<br>";
    }
  }
}
catch (SoapFault $e){
 echo "Mensaje error: ".$e->faultstring  "<br>Code: " . $e->faultcode . "<br>";
}
```

La clase SoapFault se verá a continuación.

SoapFault

Entre las facilidades que ofrece PHP para desarrollar servicios Web basados en SOAP, tenemos la clase *SoapFault* (http://php.net/manual/es/class.soapfault.php), que permite tratar y documentar excepciones producidas al atender solicitudes de clientes. El funcionamiento se basa en crear la

clase en el servidor en el lugar donde se detecta el error y después en el cliente se ejecuta el acceso al servicio Web dentro de una sentencia *try – catch* donde se evalúa el resultado de la operación. Veamos un ejemplo:

En el servidor:

```
$res_array = new SoapFault("1", "Error Temporada no encontrada");
```

Al crear la clase debemos suministrar al menos los parámetros *faultcode* y *faultstring* que corresponden a la código de error y el mensaje asociado.

En el cliente:

```
try{
 $frutas = $client->temporada($temporada);
 foreach ($frutas as $fruta){
  foreach ($fruta as $nombre){
     echo $nombre . "<br>";
  }
 }
}
catch (SoapFault $e){
   echo "Mensaje error: " . $e→faultstring . "<br>Code: ".$e→faultcode. "<br>";
}
```

En el cliente, si se produce la excepción podemos acceder a los atributos establecidos al crear el objeto SoapFault, en el ejemplo anterior, a *faultstring* y *faultcode*.

Ejemplo completo de Servicio Web SOAP

Servidor:

index.php:

```
<?php
include_once 'fruteria.php'; //Clase con los métodos del Servicio Web
// Una vez generado el fichero WSDL se debe comentar todas las instrucciones
// necesarias para generarlo
require 'WSDLDocument.php'; // Crear el fichero WSDL,


$uri = "http://localhost/toni/fruteria_soap_server_wsdl/";
$server = new SoapServer(null, array("uri"=>$uri));
$server->setClass("fruteria");
$server->handle();

// Crea el contenido del fichero WSDL, para ello necesita que la clase
// se comente los parámetros de los métodos.
```

```
$wsdl = new WSDLDocument("fruteria",
       "http://localhost/toni/fruteria_soap_server_wsdl/index.php",
       "http://localhost/toni/fruteria_soap_server_wsdl/");
header('Content-Type: text/xml');
echo $wsdl->saveXML();

?>
```

fruteria.php

```
<?php

/*
 * Clase para dar servicio Web SOAP con WSDL
 */

/**
 * Description of fruteria
 * Estos comentarios son necesarios para generar el documento WSDL
 * Class fruteria ofrece los dos métodos del Servicio Web
 *
 * @author aboronat
 */
include 'funcion_conexion_bd.php';

DEFINE ("SERVIDOR", "localhost");
DEFINE ("USER", "root");
DEFINE ("PASSWD", "");
DEFINE ("BASE_DATOS", "fruteria");
/**
 * Class fruteria
 *
 */

class fruteria {
    /**
     * Busca las frutas de la temporada que recibe como párámetro
     * @param string $temporada
     * @return string[]
     */
    public function temporada($temporada){
        $con_bd = conexion_bd(SERVIDOR, USER, PASSWD,  BASE_DATOS);
        if($con_bd){
            $sql = "SELECT fruta FROM precios WHERE temporada = '" . $temporada  . "'";
            if($res = mysqli_query($con_bd, $sql)) {
                if(mysqli_num_rows($res) >= 1){ // Ha encontrado las frutas
                    $res_array = mysqli_fetch_all($res, MYSQLI_NUM);
                }
                else {
                    $res_array = new SoapFault("1", "Error Temporada no encontrada");
                }
            }
            $cierre_bd = @mysqli_close($con_bd);
            return $res_array;
        }
    }
```

```
    /**
     * Busca los datos de la fruta y temporada indicada
     * @param string $temporada
     * @param string $fruta
     * @return string[]
     */
    public function fruta ($temporada, $fruta){
        $con_bd = conexion_bd(SERVIDOR, USER, PASSWD,  BASE_DATOS);
        if($con_bd){
              $sql = "SELECT * FROM precios WHERE temporada = '" . $temporada  . "' and fruta =
'" . $fruta . "'";
            if($res = mysqli_query($con_bd, $sql)) {
                if(mysqli_num_rows($res) >= 1){ // Ha encontrado las frutas
                    $res_array = mysqli_fetch_all($res, MYSQLI_ASSOC);
                }
                else {
                    $res_array = new SoapFault("1", "Error fruta no encontrada");
                }
            }
            $cierre_bd = @mysqli_close($con_bd);
            return $res_array;
        }
    }
}// Class fruteria
```

Fichero .htaccess en el directorio de la aplicación en Apache2

```
RewriteEngine On
RewriteCond %{QUERY_STRING} wsdl
RewriteRule (.*) /fruteria_soap_server_wsdl/fruteria.wsdl? [R=302,L]
```

En NGINX en /*etc/*nginx/sites-available/defauls (si es el fichero de configuración del site)

```
location / {
    if ($query_string ~ wsdl) {
        return 302 /fruteria_soap/fruteria_soap_server_wsdl/fruteria.wsdl;
    }
    try_files $uri $uri/ /index.php?_url=$uri&$args;
}
```

Fichero fruteria.wsdl, generado con las instrucciones de index.php del servidor

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soap-env="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://localhost/fruteria_soap_server_wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://localhost/fruteria_soap_server_wsdl/">
<wsdl:types><xsd:schema targetNamespace="http://localhost/fruteria_soap_server_wsdl/">
<xsd:complexType name="stringArray">
<xsd:complexContent><xsd:restriction base="soap-enc:Array">
```

```
<xsd:attribute ref="soap-enc:arrayType" wsdl:arrayType="xsd:string[]"/>
</xsd:restriction></xsd:complexContent>
</xsd:complexType></xsd:schema>
</wsdl:types><wsdl:message name="temporadaRequest">
<wsdl:part name="temporada" type="xsd:string"/>
</wsdl:message><wsdl:message name="temporadaResponse">
<wsdl:part name="temporadaReturn" type="tns:stringArray"/>
</wsdl:message>
<wsdl:message name="frutaRequest">
<wsdl:part name="temporada" type="xsd:string"/>
<wsdl:part name="fruta" type="xsd:string"/>
</wsdl:message><wsdl:message name="frutaResponse">
<wsdl:part name="frutaReturn" type="tns:stringArray"/>
</wsdl:message><wsdl:portType name="fruteriaPortType">
<wsdl:operation name="temporada">
<wsdl:documentation>Busca las frutas de la temporada que recibe como
párámetro</wsdl:documentation>
<wsdl:input message="tns:temporadaRequest"/>
<wsdl:output message="tns:temporadaResponse"/>
</wsdl:operation>
<wsdl:operation name="fruta">
<wsdl:documentation>Busca los datos de la fruta y temporada indicada</wsdl:documentation>
<wsdl:input message="tns:frutaRequest"/>
<wsdl:output message="tns:frutaResponse"/>
</wsdl:operation></wsdl:portType>
<wsdl:binding name="fruteriaBinding" type="tns:fruteriaPortType">
<soap-env:binding xmlns="http://schemas.xmlsoap.org/wsdl/soap/" style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="temporada"><soap-env:operation
xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
soapAction="http://localhost/fruteria_soap_server_wsdl/index.php?method=temporada"
style="rpc"/>
<wsdl:input>
<soap-env:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</wsdl:input>
<wsdl:output><soap-env:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="fruta">
<soap-env:operation xmlns="http://schemas.xmlsoap.org/wsdl/soap/"
soapAction="http://localhost/fruteria_soap_server_wsdl/index.php?method=fruta"
style="rpc"/>
<wsdl:input>
<soap-env:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</wsdl:input>
<wsdl:output>
<soap-env:body xmlns="http://schemas.xmlsoap.org/wsdl/soap/" use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="fruteria">
<wsdl:documentation>Class fruteria</wsdl:documentation><wsdl:port name="fruteriaPort"
binding="tns:fruteriaBinding">
<soap-env:address location="http://localhost/fruteria_soap_server_wsdl/index.php"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Cliente:

index.php

```
<!DOCTYPE html>
<!--
Ejemplo Cliente SOAP para el Servicio Web frutería
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
<body>
<form name="fruteria" method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>" >
    <label>Temporada: </label> <input type="text" name="temporada" value="Invierno"><br>
    <label>Fruta:    </label> <input type="text" name="fruta"><br>
    <input type="submit" name="validar" value="Temporada">
    <input type="submit" name="validar2" value="Fruta">
    <input type="reset" name="cancelar" value="Cancelar">
</form>
<?php

/*
 * Cliente SOAP de Servio Web de la fruteria
 */
// Servidor sin WSDL
// Servidor con clase fruteria sin WSDL
//$url = "http://localhost/toni/fruteria_soap_server_wsdl/index.php";
//$uri = "http://localhost/toni/fruteria_soap_server_wsdl/";
//Con WSDL
//$wsdl = "http://localhost/fruteria_soap_server_wsdl/fruteria.wsdl";
$wsdl = "http://localhost/fruteria_soap_server_wsdl?wsdl";

if(isset($_REQUEST["validar"])){
    if(isset($_REQUEST["temporada"])){
         $temporada= $_REQUEST["temporada"];
         //$client = new SoapClient(null, array('location'=>$url, 'uri'=>$uri));
         $client = new SoapClient($wsdl, array("trace" => 1));
         //Muestra los tipos y métodos definidos para el acceso al servicio Web SOAP
         var_dump($client->__getTypes());
         echo "<br>";
         var_dump($client->__getFunctions());
         echo "<br><br>";
         try{
            $frutas = $client->temporada($temporada);
            // Presenta el XML  SOAP de la solicitud y la respuesta
            echo "<br>Request: " . htmlentities($client->__getLastRequest()) . "<br>";
            echo "<br>Response: " . htmlentities($client->__getLastResponse()) . "<br>";

            foreach ($frutas as $fruta){
                foreach ($fruta as $nombre){
                    echo $nombre . "<br>";
                }
            }
         }
         catch (SoapFault $e){
           echo "Mensaje error: ".$e->faultstring."<br>Code: " . $e->faultcode . "<br>";
         }

    }
}
if(isset($_REQUEST["validar2"])){
    if(isset($_REQUEST["temporada"]) && isset($_REQUEST["fruta"])){
         $temporada= $_REQUEST["temporada"];
         $fruta = $_REQUEST["fruta"];
```

```
        //$client = new SoapClient(null, array('location'=>$url, 'uri'=>$uri));
        $client = new SoapClient($wsdl);

        try{
           $res = $client->fruta($temporada, $fruta);
           foreach ($res as $fr){
                foreach ($fr as $campo => $valores) {
                   echo $campo . "  " . $valores . "<br>";
               }
           }
        }
        catch (SoapFault $e){
          echo "Mensaje error: ".$e->getMessage()."<br>Code: ". $e->faultcode . "<br>";
        }
    }
}

?>
</body>
</html>
```

Programación REST PHP

En un servicio Web REST, cada recurso es identificado de forma única mediante su URI. Mientras que la URL es un mecanismo para localizar un recurso, por ejemplo:

http://localhost/tarea/getTarea.php donde *http://localhost/tarea/* será la URL y con *getTarea.php* será la URI que identifica un recurso.

Los servicios Web ofrecidos por REST normalmente no disponen de un documento que indique las sus características como hace WSDL para SOAP. La razón es que con REST las interfaces son más simples y las aplicaciones deberían poder usarse intuitivamente, aunque en algunos casos en los que esto no sea posible se puede usar un tipo de documento con la especificación WADL aunque es una especificación que no tiene demasiada evolución.

Como se ha comentado anteriormente, el servidor REST espera una de las cuatro peticiones (GET, POST, PUT y DELETE) de forma que usaremos la variable $_SERVER[‘REQUEST_METHOD’] para saber qué petición debe atender y a continuación podemos usar una sentencia *switch* en el que cada *case* tome uno de los posibles valores y llamar a la función apropiada para responder la solicitud del cliente.

Para realizar la petición del cliente al servidor, en PHP podemos usar varios mecanismos:

HttpRequest

Esta clase pertenece a la librería HTTP de PHP pero no se distribuye directamente, sino que se debe instalar con la extensión PECL (http://ir2.php.net/manual/es/http.install.php).

cURL

La biblioteca (*libcurl*) permite conectar con servidores mediante diferentes protocolos, tales como: http, https, ftp, telnet, file o ldap. Para poder usarla PHP debe haberse compilado con esta librería, que es lo más habitual y por tanto, se puede usar prácticamente en cualquier instalación.

La información sobre cURL está disponible en el manual de PHP (http://php.net/manual/es/book.curl.php)

Este sería un ejemplo de llamada a un servidor usando cURL :

```
$curl = curl_init();

curl_setopt($curl, CURLOPT_URL, "http://localhost/toni/tareas_REST/index.php");
curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE);
curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "PUT");
curl_setopt($curl,CURLOPT_POSTFIELDS, http_build_query($params));
curl_setopt($curl, CURLOPT_HTTPHEADER, array("cache-control: no-cache"));

$response = curl_exec($curl);

$err = curl_error($curl);

curl_close($curl);

if ($err) {
  echo "cURL Error #:" . $err;
} else {
  echo $response;
}
```

Como vemos para establecer la conexión se debe ejecutar varias funciones:

- *curl_init* se crea una sesión.
- *curl_setopt* estable las opciones de la sesión, por ejemplo la URL, el método de conexión (se puede indicar de diferentes formas, las hay propias para GET y POST y la que se ve en el ejemplo para PUT y DELETE), establecer los parámetros que se debe enviar al servidor, en el caso de GET, PUT y DELETE se puede enviar junto con la URL como se realiza con el método GET típico, pero también se pueden enviar en el cuerpo de la petición como se realiza con el método POST, y que es como se muestra en el ejemplo.

  En el caso del envió por el GET:

  ```
  curl_setopt($curl,CURLOPT_URL,"http://localhost/toni/tareas_REST/index.php?operario=" .
  $operario);
  ```

  Para recuperar estos parámetros en el servidor debemos usar las variables $_REQUEST, $_GET y $_POST, o bien la equivalente con *filter_input()*. Pero si los métodos son PUT y DELETE para recuperar los datos se usa la sentencia siguiente que recupera todos los parámetros:

  ```
  parse_sta(file_get_contents("php://input"), $put_params);
  $oper = $put_params['operario'];
  ```

  Una opción importante es curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE); que estable que se devuelva el string de respuesta o FALSE si falla la sesión, en lugar de TRUE o FALSE y mostrar el resultado que es el comportamiento por defecto.

Y algunas más que puedan ser de interés para la conexión, se pueden consultar en

http://php.net/manual/es/function.curl-setopt.php

- *curl_exec* que establece la conexión
- *curl_error* devuelve el último error de la sesión en curso.
- *curl_close* cierra la sesión

file_get_contents

Esta función PHP devuelve el contenido de un fichero como un *string*, la función devolverá el *string* o FALSE si falla. Su sintaxis básica es `file_get_contents(cadena)`donde `cadena` será la ruta al fichero o la URL al servidor del que esperamos datos.

Su sintaxis completa se puede consultar en http://php.net/manual/es/function.file-get-contents.php

```
$response = file_get_contents("http://localhost/toni/tareas_REST/index.php?id=" . $id);
```

Esta función la podemos usar para acceder a recursos con el método GET.

Ejemplo completo de Servicio Web REST

Servidor REST: (index.php)

```
<?php
/*
 * Aplicación de la fruteria pero usando servicios Web REST
 * Servidor REST
 */
include 'funcion_conexion_bd.php';
DEFINE ("SERVIDOR", "localhost");
DEFINE ("USER", "root");
DEFINE ("PASSWD", "");
DEFINE ("BASE_DATOS", "fruteria");

$metodo = $_SERVER['REQUEST_METHOD'];
$recurso = filter_input(INPUT_SERVER, 'REQUEST_URI',FILTER_SANITIZE_URL);

switch($metodo){
    case 'GET':
        if(isset($_REQUEST['temporada'])){
            $temporada = filter_input(INPUT_GET,'temporada',FILTER_SANITIZE_STRING);
            $temporada = strtoupper($temporada);
            $sql = "SELECT id, fruta FROM precios WHERE temporada = '" . $temporada  . "'";
        }
        if(isset($_REQUEST['tempo']) && isset($_REQUEST['fruta'])){
```

```
            $temporada = filter_input(INPUT_GET, 'tempo',FILTER_SANITIZE_STRING);
            $temporada = strtoupper($temporada);
            $fruta = filter_input(INPUT_GET, 'fruta',FILTER_SANITIZE_STRING);
            $fruta = strtoupper($fruta);
            $sql="SELECT * FROM precios WHERE temporada='".$temporada."'and fruta='".$fruta."'";
        }
        $con_bd = conexion_bd(SERVIDOR, USER, PASSWD,  BASE_DATOS, $sql);
        echo json_encode($con_bd, TRUE);
        break;
    case 'PUT':
        // Los parámetros en los métodos PUT y DELETE se pueden enviar como
        // con el GET y se reciben de la misma forma (como arriba), o como POST,
        // pero para recibirlos se usa la instrucción de abajo:
        parse_str(file_get_contents("php://input"), $put_params);
        $id = $put_params['id'];
        $precio_kg = $put_params['precio_kg'];
        $sql = "UPDATE precios SET precio_kg = '" . $precio_kg . "' WHERE id = '" . $id . "'";
        $con_bd = conexion_bd(SERVIDOR, USER, PASSWD,  BASE_DATOS, $sql);
        echo json_encode($con_bd, TRUE);
        break;
    case 'DELETE':
        if(isset($_REQUEST['id'])){
            // Parámetro como GET
            $id = filter_input(INPUT_GET, 'id',FILTER_SANITIZE_NUMBER_INT);
            $sql ="DELETE FROM precios WHERE id = '" . $id . "'";
            $con_bd = conexion_bd(SERVIDOR, USER, PASSWD,  BASE_DATOS, $sql);
            echo json_encode($con_bd, TRUE);
        }
        break;
    case 'POST':
        $temporada = filter_input(INPUT_POST,
'temporada',FILTER_SANITIZE_STRING);
        $fruta = filter_input(INPUT_POST, 'fruta',FILTER_SANITIZE_STRING);
        $precio_kg = filter_input(INPUT_POST,
'precio_kg',FILTER_SANITIZE_STRING);
        $sql = "INSERT INTO precios (temporada, fruta, precio_kg) VALUES ('" .
$temporada . "', '" . $fruta . "', '" . $precio_kg . "')";
        $con_bd = conexion_bd(SERVIDOR, USER, PASSWD,  BASE_DATOS, $sql);
        echo json_encode($con_db, TRUE);
        break;
    default:
        $respuesta ="Opción incorrecta!!!!";
}
```

Cliente REST:

(curl_conexion.php):

```
<?php
/*
 * Función de conexión mediante cURL con servidores REST
 * Toma la $URL y el método GET, POTS, PUT o DELETE
 * y opcionalmente parámetros para el paso tipo POST
 */

function curl_conexion($url, $metodo, $params = NULL){
    $curl = curl_init();
```

```
    curl_setopt($curl, CURLOPT_URL, $url);
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, TRUE);
    curl_setopt($curl, CURLOPT_CUSTOMREQUEST, $metodo);
    if($params != NULL){
        curl_setopt($curl,CURLOPT_POSTFIELDS, http_build_query($params));
    }
    curl_setopt($curl, CURLOPT_HTTPHEADER, array("cache-control: no-cache"));
    // Ejecuta la llamada al servidor y obtiene la respuesta
    $response = curl_exec($curl);
    $err = curl_error($curl);
    curl_close($curl);
    if ($err) {
           $response = json_encode("cURL Error #:" . $err);
    }
    return $response;
}
```

(index.php):

```
<!DOCTYPE html>
<!--
Ejemplo Cliente REST para el Servicio Web frutería
-->
<html>
    <head>
        <meta charset="UTF-8">
        <!-- Bootstrap CSS -->
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css"
integrity="sha384-WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3ewGhXQFZCSftd1LZCfmhk
tB" crossorigin="anonymous">
        <title>Cliente Fruteria REST </title>
    </head>
<body>
<div id="main" style="position:absolute; left: 10%">
<h3>Frutería</h3>
<!-- Carrusel de frutas -->
<div id="carousel" class="carousel slide" data-ride="carousel" >
  <div class="carousel-inner">
    <div class="carousel-item active">
        <img  src="images/cerezas.jpg" alt="Cerezas"  height="250" width="250">
    </div>
    <div class="carousel-item">
      <img  src="images/fresas.jpg" alt="Fresas"   height="250" width="250">
    </div>
    <div class="carousel-item">
      <img  src="images/melocotones.jpg" alt="Melocotones"  height="250"
width="250">
    </div>
    <div class="carousel-item">
      <img  src="images/peras.jpg" alt="Peras"  height="250" width="250">
    </div>
    <div class="carousel-item">
      <img  src="images/manzanas.jpeg" alt="Manzanas"  height="250" width="250">
    </div>
    <div class="carousel-item">
```

```
        <img  src="images/naranjas.jpeg" alt="Naranjas"  height="250"
width="250">
    </div>
  </div>
</div>
<form name="fruteria" method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
    <div style="border-style: solid; border-color: green; border-radius: 8px;">
    <div class="input-group temporada" style="border-style: solid; border-color:
green; border-radius: 4px;">
        <div class="input-group-prepend">
         <span class="input-group-text" id="inputGroup-sizing-
default">Temporada</span>
        </div>
        <input type="text" class="form-control" aria-label="Temporada" aria-
describedby="inputGroup-sizing-default" name="temporada" value="Invierno" >
        <input type="submit" name="validar" value="Temporada" class="btn btn-
success">
    </div>
    <div class="input-group fruta">
        <div class="input-group-prepend">
         <span class="input-group-text" id="inputGroup-sizing-default">Fruta</span>
        </div>
        <input type="text" class="form-control" aria-label="Fruta" aria-
describedby="inputGroup-sizing-default" name="fruta" >
    </div>
        <input type="submit" name="validar2" value="Fruta" class="btn btn-success">
    </div>
    <div style="border-style: solid; border-color: yellow; border-radius: 8px;">
    <div class="input-group id" style="border-style: solid; border-color:
yellow; border-radius: 4px;">
        <div class="input-group-prepend">
         <span class="input-group-text" id="inputGroup-sizing-default">ID</span>
        </div>
        <input type="text" class="form-control" aria-label="Fruta" aria-
describedby="inputGroup-sizing-default" name="id" >
        <input type="submit" name="eliminar" value="Eliminar" class="btn btn-warning">
    </div>
    <div class="input-group precio_kg">
        <div class="input-group-prepend">
         <span class="input-group-text" id="inputGroup-sizing-default">Precio Kg</span>
        </div>
        <input type="text" class="form-control" aria-label="Fruta" aria-
describedby="inputGroup-sizing-default" name="precio_kg" >
    </div>
        <input type="submit" name="modificar" value="Precio Kg" class="btn btn-warning">
    </div>
    <input type="submit" name="crear" value="Nueva Fruta" class="btn btn-primary">
    <input type="reset" name="cancelar" value="Cancelar" class="btn btn-danger">
</form>
<?php

/*
 * Cliente REST de Servicio Web de la fruteria
 */
include 'curl_conexion.php';

// Crea una CARD de Bootstrap para mostrar los resultados de las búsquedas
$res = '<div class="card" style="width: 18rem;">';
```

```
$res .= '<div class="card-body">';
$res .= '<p class="card-text">';
$nom ='';
if(isset($_REQUEST["validar"])){
    if(isset($_REQUEST["temporada"])){
        $temporada= filter_input(INPUT_POST, "temporada", FILTER_SANITIZE_STRING);
        $url = "http://localhost/toni/fruteria_rest_servidor/index.php?temporada=" . $temporada;
        $res .= '<h5 class="card-title">Frutas de ' . $temporada . '</h5>';
        $response = curl_conexion($url, "GET");
        $frutas = json_decode($response);
        if(count($frutas) > 1){
            for($i = 0 ; $i < count($frutas) ; $i++ ){
                $nom .= 'ID: ' . $frutas[$i][0] . '  Fruta: ' . $frutas[$i][1] . "<br>";
            }
            $res .= $nom;
        }
        else{
            $res .= $frutas;
        }
        $res .= '</p>';
        $res .= '<a href="index.php" class="btn btn-primary">Cerrar</a>';
        $res .= '</div>';
        $res .= '</div>';
        echo $res;
    }
}
if(isset($_REQUEST["validar2"])){
    if(isset($_REQUEST["temporada"]) && isset($_REQUEST["fruta"])){
        $temporada= filter_input(INPUT_POST, "temporada", FILTER_SANITIZE_STRING);
        $fruta = filter_input(INPUT_POST, "fruta", FILTER_SANITIZE_STRING);
        $res .= '<h5 class="card-title">Datos de ' . $fruta . '</h5>';
        $dato ='';
        $url = "http://localhost/toni/fruteria_rest_servidor/index.php?tempo=" .
$temporada . "&fruta=" . $fruta;
        $response = curl_conexion($url, "GET");
        $frutas = json_decode($response);
        $nom_columna = array('ID', 'FRUTA','PRECIO Kg','TEMPORADA');
        if(is_array($frutas)){
            foreach ($frutas as $fr){
                 foreach ($fr as $campo => $valores) {
                    $dato .=  $nom_columna[$campo] . ":   " . $valores . "<br>";
                }
            }
            $res .= $dato;
        }
        else{
            $res .= $frutas;
        }
        $res .= '</p>';
        $res .= '<a href="index.php" class="btn btn-primary">Cerrar</a>';
        $res .= '</div>';
        $res .= '</div>';
        echo $res;
    }
}
if(isset($_REQUEST["crear"])){ // Dar de alta nueva fruta
    header("Location: crear.php");
}
```

```
if(isset($_REQUEST["modificar"])){
    if(isset($_REQUEST["id"]) && isset($_REQUEST["precio_kg"])){
        $id = filter_input(INPUT_POST, "id", FILTER_SANITIZE_STRING);
        $precio_kg = filter_input(INPUT_POST, "precio_kg",
FILTER_SANITIZE_STRING);
        $params = array('id' => $id, 'precio_kg' => $precio_kg) ;
        $url = "http://localhost/toni/fruteria_rest_servidor/index.php";
        $response = curl_conexion($url, "PUT", $params);
        $resp = json_decode($response);
        $res .= $resp;
        $res .= '</p>';
        $res .= '<a href="index.php" class="btn btn-primary">Cerrar</a>';
        $res .= '</div>';
        $res .= '</div>';
        echo $res;
    }
}
if(isset($_REQUEST["eliminar"])){
    if(isset($_REQUEST["id"])) {
        $id = filter_input(INPUT_POST, "id", FILTER_SANITIZE_STRING);
        $url = "http://localhost/toni/fruteria_rest_servidor/index.php?id=" .$id;
        $response = curl_conexion($url, "DELETE");
        $resp = json_decode($response);
        $res .= $resp;
        $res .= '</p>';
        $res .= '<a href="index.php" class="btn btn-primary">Cerrar</a>';
        $res .= '</div>';
        $res .= '</div>';
        echo $res;
    }
}
?>
    <!-- Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6ji
zo" crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/
l8WvCWPIPm49" crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"
integrity="sha384-smHYKdLADwkXOn1EmN1qk/HfnUcbVRZyYmZ4qpPea6sjB/pTJ0euyQp0Mk8ck+
5T" crossorigin="anonymous"></script>
</div>
</body>
</html>
```

(crear.php):

```
<!DOCTYPE html>
<!-- Ejemplo Cliente REST para el Servicio Web frutería Alta nueva fruta -->
<html>
    <head>
        <meta charset="UTF-8">
        <!-- Bootstrap CSS -->
```

```
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css"
integrity="sha384-WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3ewGhXQFZCSftd1LZCfmhk
tB" crossorigin="anonymous">
        <title>Alta nueva fruta </title>
    </head>
<body>
<div id="main" style="position:absolute; left: 10%">
<h3>Alta en frutería</h3>
<form name="crear" method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>" >
    <div class="input-group temporada">
        <div class="input-group-prepend">
         <span class="input-group-text" id="inputGroup-sizing-default">Temporada</span>
        </div>
        <input type="text" class="form-control" aria-label="Temporada" aria-
describedby="inputGroup-sizing-default" name="temporada" value="Invierno" >
    </div>
    <div class="input-group fruta">
        <div class="input-group-prepend">
         <span class="input-group-text" id="inputGroup-sizing-default">Fruta</span>
        </div>
        <input type="text" class="form-control" aria-label="Fruta" aria-
describedby="inputGroup-sizing-default" name="fruta" >
    </div>
    <div class="input-group precio">
        <div class="input-group-prepend">
        <span class="input-group-text" id="inputGroup-sizing-default">Precio Kg</span>
        </div>
        <input type="text" class="form-control" aria-label="precio" aria-
describedby="inputGroup-sizing-default" name="precio" >
    </div>
    <input type="submit" name="crear" value="Nueva Fruta" class="btn btn-primary">
    <input type="reset" name="cancelar" value="Cancelar" class="btn btn-danger">
</form>
<?php

/*
 * Aplicación Servicio Web REST fruteria
 * Alta de nueva fruta método POST
 *
 */
include 'curl_conexion.php';

$res = '<div class="card" style="width: 18rem;">';
$res .= '<div class="card-body">';
$res .= '<p class="card-text">';

if(isset($_REQUEST["crear"])){
    if(isset($_REQUEST["temporada"]) && isset($_REQUEST["fruta"]) && isset($_REQUEST["precio"])){
        $temporada= filter_input(INPUT_POST, "temporada", FILTER_SANITIZE_STRING);
        $temporada = strtoupper($temporada);
        $fruta = filter_input(INPUT_POST, "fruta", FILTER_SANITIZE_STRING);
        $fruta = strtoupper($fruta);
        $precio_kg = filter_input(INPUT_POST, "precio", FILTER_SANITIZE_STRING);
        $res .= '<h5 class="card-title">Datos de ' . $fruta . '</h5>';
        $params = array('temporada' => $temporada, 'fruta' => $fruta, 'precio_kg' => $precio_kg) ;
        $url = "http://localhost/toni/fruteria_rest_servidor/index.php";
        $response = curl_conexion($url, 'POST', $params);
```

```
        $res .= json_decode($response);
        $res .=  $fruta . 'añadida a la BD</p>';
        $res .= '<a href="index.php" class="btn btn-primary">Cerrar</a>';
        $res .= '</div>';
        $res .= '</div>';
        echo $res;
    }
}
?>
<!-- Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6ji
zo" crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"
integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/
l8WvCWPIPm49" crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"
integrity="sha384-smHYKdLADwkXOn1EmN1qk/HfnUcbVRZyYmZ4qpPea6sjB/pTJ0euyQp0Mk8ck+
5T" crossorigin="anonymous"></script>
</div>
</body>
</html>
```

Ejercicio

Desarrolla el ejercicio de conversión de divisas como un Servicio Web SOAP. Haz una versión sin WSDL y otra con WSDL.

La misma pero usando el modelo REST.