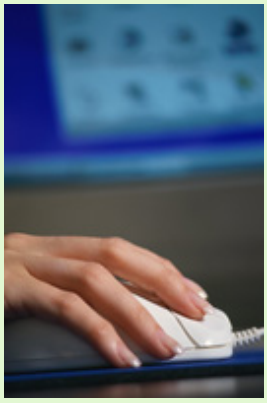
1.- Descripción de la tarea

Caso práctico



Un ayuntamiento se ha puesto en contacto con BK Programación para un proyecto muy interesante. Ada está muy interesada porque cree que le podrán sacar bastante partido al proyecto.

El ayuntamiento de "*Muchos Coches Pocas Calles*" necesita conocer cuales son las calles con mayor densidad de tráfico. Su idea es poder redistribuir el tráfico de forma que el movimiento por la ciudad sea más fácil y ágil.

El proyecto que **Ada** ha propuesto al ayuntamiento consiste en pequeñas estaciones repartidas a lo largo del pueblo encargadas de medir la cantidad de coches que pasan por ciertas calles. Ya ha puesto en manos de un equipo de desarrollo interdisciplinar el diseño de esas estaciones de medida de tráfico.

Sin embargo, tiene un gran problema. ¿Cómo puede recopilar la información de tráfico obtenida por cada estación? Ciertamente no lo tiene claro, pero Juan le da la idea: "Las estaciones podrían conectarse a Internet y enviar la información a un servicio web, de esa forma podríamos hacer uso de los datos recopilados por las estaciones y que desde el ayuntamiento puedan ver la evolución del tráfico día a día."

A **Ada** se le ha iluminado la cara y enseguida le ha dicho a Juan que se encarge de hacer un prototipo funcional del servicio web.

En esta tarea pondrás en práctica tus conocimientos sobre servicios webs e un servicio web que use SOAP sencillo. Tendrás que implementar tanto l parte de cliente, como la parte de servidor.

La idea es la que comenta Juan en el caso práctico (échale un vistazo ante de continuar). Se trata de que diferentes estaciones de tráfico autonoma puedan registrar su información en un servidor central a través un servici web sencillo. Cada estación de tráfico se encarga de contabilizar el volume

de tráfico en una calle concreta en diferentes tramos horarios (identificado como **`tramo1`**, **`tramo2`**, **`tramo3`** y **`tramo4`**).

En esta tarea tienes que hacer un prototipo de servicio web que permit almacenar dicha información y rescatarla posteriormente. Tienes que pensa en todo momento que las aplicaciones cliente (aplicaciones que usan servicio web y que llamaremos clientes SOAP) y el servicio web en si mism (servidor SOAP) estarán en máquinas diferentes, aunque aquí desarrollarás todo dentro de un mismo proyecto (es un prototipo).

El servicio SOAP a desarrollar está descrito por el siguiente descripto WDSL (no te asustes, es extenso pero no muerde):

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

<definitions
  name="trafico"
  targetNamespace="http://localhost/dwes05/tarea"
  xmlns:tns="http://localhost/dwes05/tarea"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
>
<types>
    <xsd:schema targetNamespace="http://localhost/dwes05/tarea">
        <xsd:complexType name="medicion">
            <xsd:all>
                <xsd:element name="day" type="xsd:int" />
                <xsd:element name="month" type="xsd:int"/>
                <xsd:element name="year" type="xsd:int"/>
                <xsd:element name="tramo" >
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="tramo1"/>
                            <xsd:enumeration value="tramo2"/>
                            <xsd:enumeration value="tramo3"/>
                            <xsd:enumeration value="tramo4"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:element>
                <xsd:element name="estacion" type="xsd:string"/>
                <xsd:element name="recuento" type="xsd:int"/>
            </xsd:all>
        </xsd:complexType>
    </xsd:schema>
</types>
<message name="nuevaMedicionRequest">
    <part name="datosMedicion" type="tns:medicion"/>
</message>
<message name="nuevaMedicionResponse">
    <part name="result" type="xsd:int"/>
</message>
<message name="getMedicionRequest">
    <part name="day" type="xsd:int"/>
    <part name="month" type="xsd:int"/>
    <part name="year" type="xsd:int"/>
    <part name="estacion" type="xsd:string"/>
```

```
</message>
<message name="getMedicionResponse">
    <part name="tramo1" type="xsd:int"/>
    <part name="tramo2" type="xsd:int"/>
    <part name="tramo3" type="xsd:int"/>
    <part name="tramo4" type="xsd:int"/>
</message>
<portType name="medicionPortType">
    <operation name="nuevaMedicion">
        <input message="tns:nuevaMedicionRequest"/>
        <output message="tns:nuevaMedicionResponse"/>
    </operation>
    <operation name="getMedicion">
        <input message="tns:getMedicionRequest"/>
        <output message="tns:getMedicionResponse"/>
    </operation>
</portType>
<binding name="medicionBinding" type="tns:medicionPortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc
    <operation name="nuevaMedicion">
        <soap:operation soapAction="http://localhost/dwes05/ws/mediciones.php
        <input>
            <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.or
        </input>
        <output>
            <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.or
        </output>
    </operation>
    <operation name="getMedicion">
        <soap:operation soapAction="http://localhost/dwes05/ws/mediciones.php
        <input>
            <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.or
        </input>
        <output>
            <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.or
        </output>
    </operation>
</binding>
<service name="nuevaMedicionService">
    <documentation>Servicio para mediciones de tráfico.</documentation>
    <port name="nuevaMedicionService" binding="tns:medicionBinding">
        <soap:address location="http://localhost/dwes05/ws/mediciones.php" />
    </port>
</service>
</definitions>
```

Es el momento de que analices con calma el descriptor de servicio we anterior y que busques en los apuntes todos aquellos aspectos qu necesites. No es necesario que lo modifiques, solo que lo entiendas.

De cara a realizar la tarea ten en cuenta que:

- El descriptor WDSL anterior debe ser accesible a través de la UR http://localhost/dwes05/ws/tarea05.wdsl
- Fíjate que en el descriptor anterior se describen dos operaciones:
  - **`nuevaMedicion`**: a través de esta operación las estaciones de tráfic envían los datos de una nueva medición de tráfico en un tram

dado.
    - `getMedicion`: a través de esta operación se obtienen la mediciones en los cuatro tramos definidos para un día concret de una estación concreta.
- Fíjate, además, que dichas operaciones tienen unos datos concreto de entrada y otros de salida, y que existe un tipo de datos compuest (`tns:medicion`) que se expresará en nuestro código como un objeto.

Vamos a abordar este proyecto paso a paso, para que sea productivo aprendas con él.

Se valorará en todos los casos la corrección ortográfica y gramatical de los mensajes para comunicarnos con el usuario, así como la presentación clara de cualquier información que se muestre al usuario.

1.1.- Ejercicio 1: activar PHP SOAP y crear la estructura de proyecto

Antes de comenzar tu proyecto deberás realizar varias acciones. Por un lado los datos del servicio web se almacenarán en una tabla en una tabla como la siguiente:

```
CREATE TABLE medicion (
  fecha DATE NOT NULL,
  tramo VARCHAR(10) NOT NULL,
  estacion VARCHAR(45) NOT NULL,
  recuento INT NOT NULL,
  PRIMARY KEY (fecha, tramo, estacion));
```

Deberás crear la tabla anterior en la base de datos como parte de este paso aunque dicha tabla no la usarás hasta el ejercicio 6.

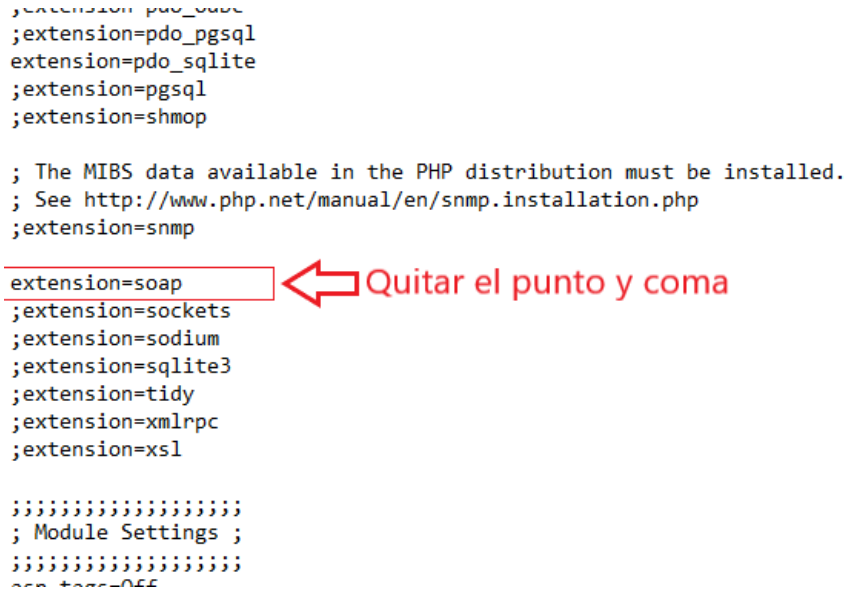
Por otro lado, deberás crear la estructura del proyecto (se indican carpetas subcarpetas y los archivos finales indicados en los ejercicios):

- **dwes05/**
  - **ejercicio1.php**: código del ejercicio a realizar en este apartado.
  - **ejercicio4.php**: código del ejercicio a realizar en el apartado 4.
  - **ejercicio5.php**: código del ejercicio a realizar en el apartado 5.
  - **ws/**: carpeta en la que irá alojado todo el código correspondiente a servicio web.
    - **config/** : carpeta que contendra la configuración apar acceder a la base de datos.
      - **settings.php**: Archivo que contiene la configuración d acceso a la base de datos usando **PDO**.
    - **tarea05.wsdl** : descriptor WSDL del servicio SOAP (s proporciona en la descripción)
    - **mediciones.php** : archivo principal del servicio web SOAP.
    - **MedicionModel.php**: archivo que implementará una clase co los métodos para dar soporte a los servicios web.
    - **verlog.php**: archivo que se añadirá más adelante y qu permitirá ver que eventos y errores han ocurrido en e servicio web.

Ten en cuenta que el proyecto debe ser accesible a través de l URL **http://localhost/dwes05/**, y que el servicio web deberá ser accesible través de la url **http://localhost/dwes05/ws/mediciones.php** (que es la UR

indicada en el descriptor WSDL). En sucesivos ejercicios irás desarrollando e servicio web, no te preocupes.

Otro aspecto, el más importante, a realizar en este ejercicio es **la activació de PHP SOAP**. Normalmente, en una instalación de desarrollo de XAM suele estar deshabilitado. Para habilitarlo tendrás que editar el archivo **`php.in`** y activar la extensión quitando el punto y coma:



Una vez activada la extensión, en el **`dwes05/ejercicio1.php`** tendrás que hacer un script que verifique que las clases **`SoapClient`** y **`SoapServer`** existen usando e método **`class_exists`**. Solo tienes que mostrar un mensaje que indique si las clases existen ono, ya está.

¡Importante! Recuerda que después de activar una extensión o realizar un cambio en el archivo php.ini tienes que reiniciar el servidor web.

1.2.- Ejercicio 2: creamos el esqueleto mínimo.

Cuando hablamos de esqueleto, hablamos de un conjunto de clases métodos que no tienen una implementación completa, sino un implementación de partida de modo que se puede usar como base par luego ir complementando nuestro código.

Antes de empezar, si te has fijado en el archivo WSDL, habrás visto que ha dos operaciones: **`nuevaMedicion`** y **`getMedicion`**. Veamos a que corresponden:

La operación **`nuevaMedicion`** es usada para notificar una nueva medición. Est operación tiene como entrada (input) un objeto compuesto por 6 elementos:

- **`day`**: día en el que se produce la medicion.
- **`month`**: mes en el que se produce la medicion.
- **`year`**: año al que corresponde la medicion.
- **`estacion`**: estación en la que se produce la mecición (no puede esta vacío)
- **`tramo`**: tramo horario de la medicion, el cual puede ser uno de l siguientes 4 valores; **`tramo`**1, **`tramo2`**, **`tramo3`** o **`tramo4`**.
- **`recuento`**: recuento de coches que han pasado en esa fecha por l estación. Es un valor que debe ser mayor o igual a 0.

Como resultado la operación **`nuevaMedicion`** retornará un número entero co uno de lo siguientes valores:

- **`-1`**: existe un error en los datos y no se puede procesar. El error pued ser debido a que la fecha, la estación, el tramo o el recuento son n válidos.
- **`-2`**: existe un error en la base de datos y no se ha podido almacenar dato.
- **`0`**: operacion efectuada correctamente, los datos se han guardado en l base de datos (previamente no existian).
- **`1`**: operación efectuada correctamente, los datos ya existian en la bas de datos y se han modificado.

Por otro lado la operación **`getMedicion`** es usada para que un cliente SOA pueda recuperar los datos medidos en una fecha concreta, por lo que com entrada (input) adimitirá 4 datos separados:

- **`day`**: día del que se necesitan los datos.
- **`month`**: mes del que se necesitan los datos.
- **`year`**: año del que se necesitan los datos.
- **`estacion`**: estación de la que se necesitan los datos.

El resultado de esta operación será un array asociativo con cuatro llave (**`tramo1`**, **`tramo2`**, **`tramo3`** y **`tramo4`**), y los recuentos almacenados en la base d

datos para cada tramo. Al realizar esta operación pueden pasar situacione que hay que manejar adecuadamente:

- a) En la base de datos no hay ningún dato para la fecha dada. En est caso, todos los tramos (`tramo1`, `tramo2`, `tramo3` y `tramo4`) tendrá com valor `-1` para así la existencia de un error.
- b) En la base de datos no haya ningún dato almacenado para l estación indicada. En este caso, todos los tramos también tendrá como valor `-1`, al igual que el caso anterior.
- c) En la base de datos hay datos para una estación y fecha, pero ha tramos sin información. En este caso, solo los tramos no existente tendrán como valor `-1`.
- d) Se ha producido un error en la base de datos y no se ha podid realizar la operación. En este caso, todos los tramos tendrán com valor `-2`.

Una vez que ya sabes como funciona el servicio web y el resultado esperad de cada operación, vamos a codificar un poco. Todavía no es necesario qu implementes las operaciones de acceso a la base de datos, eso se hará má adelante. De momento vamos a crear la clase `MedicionModel` (archiv `dwes05/ws/MedicionModel.php`) para manejar las operaciones definidas en archivo WSDL. Esta clase tendrá, de momento, los siguientes métodos y siguiente comportamiento:

- Constructor sin argumentos y que de momento no hará nada.
- Método público `nuevaMedicion` que admitirá un único parámetr (`$medicion`). Ese único parámetro será un objeto que contendrá lo atributos antes indicados (`day`, `month`, `year`, etc.). De momento, est método recien creado se limitará a retornar un número aleatorio.
- Método público `getMedicion` que admitirá 4 parámetros: `day`, `month`, `yea` y `estacion`. Este método retornará el array asociativo indicado (llave `tramo1`, `tramo2`, `tramo3` y `tramo4`) cuyos valores serán números aleatoric (en vez de datos obtenidos de la base de datos).

Como puedes observar, los métodos de esta clase dan soporte a la operaciones definidas en el descriptor `WSDL`.

Si necesitas crear alguna clase adicional, siente libre de hacerlo. Lo que se describe aquí es lo mínimo a realizar.

1.3.- Ejercicio 3: vamos a por el SOAP server.

Una vez creada la clase **MedicionModel**, vamos a codificar el archivo que har de punto de acceso al servicio web. Este archivo es el archivo **mediciones.ph** (**dwes05/ws/mediciones.php**).

Una de las dificultades de trabajar con servicios SOAP y otros web services es depurar la aplicación y encontrar errores. El problema reside en que s PHP emite errores o cualquier tipo de texto, el mensaje SOAP no ser correcto y no funcionará el servicio web.

Una solución para esto es desviar los mensajes de error a un archivo d texto, de forma que luego puedan ser consultados después de la ejecució cómodamente, y esto es lo que vamos a hacer. Esto, junto con el depurado te servirá para analizar que ocurre en el servicio web y si es todo correcto.

Para hacer esto, en el archivo **mediciones.php**, vamos a desviar los errores un archivo con el siguiente código:

```
ini_set("log_errors", 1);
ini_set("error_log", "ws-errors.log");
ini_set("display_errors", 0);
error_reporting(E_ALL);
```

El código anterior hará que los errores se vuelquen a un archivo (el archiv **ws-errors.log**) y que no se envíen al navegador.

Aquí es importante insistir en el hecho de que no se puede usar **echo**, **print** similar en ningún momento, dado se producirian errores en el servicio we SOAP al producir mensajes SOAP incorrectos. Si necesitas genera información para depurar tu aplicación debes utilizar otro mecanismo, p ejemplo:

- Escribir en el archivo **ws-errors.log** con la función **file_put_contents** co el flag **FILE_APPEND** para que añada contenido al final del archivo.
- Forzar la generación de un error PHP con el método **trigger_erro** Esta es posiblemente la opción más sencilla.
- Haz uso del segundo parámetro del método print_r y así obtener resultado en un string que puedas volcar al archivo de log.

Una vez hecho esto, vamos a crear un script para poder ver má cómodamente el contenido del archivo **ws-errors.log**. El script s llamará **verlog.php** (**dwes05/ws/verlog.php**) y puedes basarte en el siguient código para su realización:

```php
<?php
header( "refresh:5;url=verlog.php" );
if (isset($_GET['cleanlog'])) {
    unlink('ws-errors.log');
}
echo '<h3> Last call '.date('d-m-Y H:i:s').'</h3>';

if (file_exists("ws-errors.log")) {
    echo '<pre>';
    readfile("ws-errors.log");
    echo '</pre>';
}
else
{
    echo '<h1>No log file!</h1>';
}
```

Por supuesto el código anterior es personalizable, adaptalo a tu gusto.

Después de esto, ya solo nos falta completar el código de **mediciones.ph** para que se comporte como un servicio web (este es el aspecto má importante, obviamente). Lo que faltaría es:

- Crear una instancia de **SoapServer** usando como parámetro la url d descriptor WSDL que estamos desarrolland (**http://localhost/dwes05/ws/tarea05.wsdl**).
- Establecer la clase manejadora de las peticiones SOAP a la clas "**MedicionModel**" (creada en el paso anterior). Para esto deberás usar método **setClass** de la clase **SoapServer**.
- Invocar el método **handle** para manejar la petición y redireccionarla método esperado.

El código anterior es sencillo, son solo 3 líneas de código, pero es la bas para que esto empieze a funcionar.

1.4.- Ejercicio 4: registrar una medición con SOAPClient.

Ahora es el momento de empezar a usar el servicio web. ¡¡Llego la hora d la verdad!! No te preocupes, es fácil.

Este ejercicio se realizará en el script **`dwes05/ejercicio4.php`** y será accesibl a través de la URL **`http://localhost/dwes05/ejercicio4.php`**.

En este script debes realizar una interfaz web que solicite al usuario lo datos necesarios para la operación **`nuevaMedicion`** del servicio web qu tenemos entre manos. Una vez que el usuario introduzca los datos, s invocará el servicio web usando **`SoapClient`**.

Veamos en detalle como debe comportarse el script **`dwes05/ejercicio4.php`**:

- Al consultarse por primera vez, debe mostrar un formulario con lo datos necesarios para crear una nueva medición.
- Cuando se envía el formulario vía POST al script **`ejercicio4.php`**, deb realizarse lo siguiente:
  - Crea una instancia de **`SoapClient`** pasandole por parámetro l URL del servicio web que estamos desarrolland (**`http://localhost/dwes05/ws/tarea05.wsdl`**).
  - Invoca la operación correspondiente para registrar una nuev medición pasandole un único argumento (tipo objeto) con todo los datos recibidos del usuario. Aquí tienes que tener en cuent dos cosas:
    - Los nombres de los atributos en el objeto deben coincid con el nombre indicado en el descriptor WSDL (**`day`**, **`mont year`**, etc).
    - Los datos que envía el usuario no se validarán de form completa. Solo se verificarán que se han proporcionad (**`isset`**) y que tienen el contenido correcto (entero o caden según corresponda).
  - Analiza el resultado de la operación recibido del servicio web:
    - Muestra un mensaje en consonacia en función del códig de error recibido.
    - Si el código de error indica que la operación fue llevada co éxito, almacena la fecha y la estación en la sesión, de form que luego se pueda usar dicha información en el script d ejercicio 5.

Por su parte, en el servicio web, tenemos que hacer algunas modificacione concretamente en el método **`nuevaMedicion`** de la clase **`MedicionModel`** (que e el que da soporte a la operación SOAP **`nuevaMedicion`**). En dicho métod debes validar los datos recibidos, comprobando que:

- El día, el mes y el año corresponden a una fecha válida y que exist (recuerda que php tiene funciones para validar fechas), y que no se

"del futuro".
- Que el tramo sea uno tramo válido: **`tramo`**1, **`tramo2`**, **`tramo3`** o **`tramo4`**.
- Que la estación no sea una cadena vacía.
- Que el recuento no sea un número negativo (puede ser cero).

Este método todavía no almacenará información en la base de datos, por l
que, si los datos son correctos deberá retornar **`-2`** (error en la base de datos
Vamos ampliando funcionalidad poco a poco.

1.5.- Ejercicio 5: consultar una medicion con SOAPClient.

Siguiendo un proceso análogo al del ejercicio 4, pero con algunas evidente diferencias, vamos ahora a usar la otra operación del servicio wel **`getMedicion`**.

Para realizar este ejercicio deberás crear un script llamado **`ejercicio5.ph`** (**`dwes05/ejercicio5.php`**) accesible a través de la u **`http://localhost/dwes05/ejercicio5.php`**. En este script debe hacerse l siguiente:

- En primer lugar debe comprobar si existe información en la sesió (fecha y estación) procedente de la última medición registrada a travé de **`ejercicio4.php`**:
  - En caso de que exista dicha información, debe usarse dich información en la operación SOAP **`getMedicion`**.
  - En caso de que no exista dicha información, debe solicitarse a usuario dicha información a través de un formulario. En cualquie caso, siempre existirá un formulario que permitirá cambiar l fecha y la estación a consultar.
  - Si el usuario cambia la fecha y estación a través el formulario, s cambiará la fecha y estación almacenada en la sesión, de form que al volver a invocar este script se "recuerde" la última fecha estación consultadas.
- En segundo lugar, si se dispone de la información necesaria (fecha **`day`**, **`month`**, **`year`**- y **`estacion`**), ya provenga de la información de sesión del formulario, se realizará lo siguiente:
  - Se verificará la existencia de la información necesaria para usa el servicio web y que los tipos de datos corresponden con lo esperados (entero o cadena según corresponda). No se realizar ninguna otra validación.
  - Se invocará la operación **`getMedicion`** del servicio web qu estamos desarrollando a través de **`SoapClient`**, de forma similar como se describe en el ejercicio anterior.
  - Se mostrará el resultado de realizar dicha operación, mostrand los datos recibidos. Si se trata de un error, deberá mostrar pertinente mensaje de error.

Y como es de esperar, en el servicio web tenemos también que hace algunas modificaciones, ahora en el método **`getMedicion`** de la clas **`MedicionModel`** (que es el que da soporte a la operación SOAP **`getMedicion`** En dicho método debes validar los datos recibidos, comprobando que:

- El día, el mes y el año corresponden a una fecha válida y que exist (recuerda que php tiene funciones para validar fechas), y que no se "del futuro".
- Que la estación no sea una cadena vacía.

Nuevamente, este método todavía no almacenará información en la base d
datos, por lo que, si los datos son correctos deberá retornar `-2` (error en l
base de datos). En el siguiente ejercicio nos encargamos de eso.

1.6.- Ejercicio 6: almacenar los datos en la base de datos.

En este apartado modificaremos ahora el servicio web (principalmente l clase `MedicionModel`) para hacer que guarde la información de las medicione en la base de datos.

Todas las modificaciones a este respecto se realizarán siempre dentro de l carpeta `dwes05/ws`. El motivo es que el servicio web SOAP es normalment una aplicación web independiente del cliente SOAP. En nuestro desarroll deberíamos ser capaces de llevarnos solo la parte del servicio web (carpet `ws`) a un servidor independiente y que este pueda seguir siendo usado desd otros clientes SOAP. Tenlo en cuenta.

Para desarrollar esta parte solo hay cuatro restricciones:

- Debes situar la configuración de acceso a la base de datos en archivo `dwes05/ws/config/settings.php`.
- Debes crear un atributo privado en la clase `MedicionModel` dond almacenes la instancia de PDO a usar en el resto de los ejercicios.
- En el constructor de la clase `MedicionModel` debes crear una instanci de PDO, almacenarla en el atributo antes especificado y usarla dond necesites. No crear más instancias de PDO innecesariamente, dad que esto hará que el servicio web baje mucho su rendimiento.
- **Por seguridad, debes usar obligatoriamente consulta preparadas**.

Dicho esto, ya solo falta que pongas el broche final a la tarea almacenand la información en la base de datos.

2.- Información de interés

Recursos necesarios y recomendaciones

Recomendaciones específicas de esta tarea

- Es recomendable que configures el depurador de PHP, te facilitar bastante depurar el servidio web.
- Es muchas instalaciones de php los errores está configurados para n mostrarse al usuario. Mientras estas desarrollando una aplicación e imprescindible que los errores de php se muestre, de forma qu podamos corregir cualquier aplicación antes de que llegue al usuari final. En el siguiente enlace se explica como:

¿Cómo mostrar errores php?

Recomendaciones generales

- Como ya sabes, para escribir aplicaciones en PHP necesitarás u entorno XAMPP, LAMPP o similar. No te vamos a pedir que instales u entorno u otro, o que trabajes con un sistema operativo concreto, per si que todos los **archivos vayan codificados en UTF-8** y que **la rutas a los archivos sean siempre relativas**.
- Como plataforma de desarrollo (para escribir código), te aconsejamo **NetBeans 12**. Es la misma herramienta que se usa en otros módulos no tendrás que cambiar de herramienta.
- No abordes la tarea hasta que hayas repasado todos los contenidos, hecho al menos uno de los intentos para el examen en línea, de form que hayas tenido ocasión de consultar y resolver en los foros cualquie duda que te haya podido surgir.

Indicaciones de entrega

Una vez realizada la tarea, el envío se realizará a través de la plataforma. Comprime la carpeta del proyecto en un fichero .zip y nómbralo siguiendo las siguientes pautas:

Apellido1_Apellido2_Nombre_DWES_Tarea04

3.- Evaluación de la tarea

Criterios de evaluación implicados

- Se han reconocido las características propias y el ámbito de aplicación de los servicios Web.
- Se han reconocido las ventajas de utilizar servicios Web para proporcionar acceso a funcionalidades incorporadas a la lógica de negocio de una aplicación.
- Se han identificado las tecnologías y los protocolos implicados en la publicación y utilización de servicios Web.
- Se ha programado un servicio Web.
- Se ha creado el documento de descripción del servicio Web.
- Se ha verificado el funcionamiento del servicio Web.
- Se ha consumido el servicio Web.

Peggy_Marco ((Pixabay License)

¿Cómo valoramos y puntuamos tu tarea?

En esta puedes observar la puntuación máxima asignada a cada ejercicio d la tarea:

Rúbrica de la tarea	
Se ha implementado el ejercicio 1 apropiadamente y funciona correctamente.	**hasta 0,5 puntos**
Se ha implementado el ejercicio 2 apropiadamente y funciona correctamente.	**hasta 0,5 puntos**
Se ha implementado el ejercicio 3 apropiadamente y funciona correctamente.	**hasta 1 puntos**
Se ha implementado el ejercicio 4 apropiadamente y funciona correctamente.	**hasta 2,5 puntos**

Se ha implementado el ejercicio 5 apropiadamente y funciona correctamente.	**hasta 2,5 puntos**
Se ha implementado el ejercicio 6 apropiadamente y funciona correctamente.	**hasta 3 puntos**