U08. Aplicaciones Web Híbridas

1.- Reutilización de código e información

La **unidad 7** de este módulo tiene por título "Servicios Web". En ella se aprende a crear y utilizar servicios web, empleando una arquitectura orientada a servicios (`SOA – Service Oriented Architecture`). Los principales temas que se tratan en esa unidad son:

- Utilizar el protocolo `SOAP` para comunicarse con un servicio web. Con ayuda de la clase `"SoapClient"`, se intercambian peticiones y respuestas en formato `SOAP` con un servicio web existente.
- Crear un servicio web. Mediante la clase `"SoapServer"` se pueden publicar funciones propias para que estén accesibles como servicio web mediante `SOAP`.
- Procesar los documentos `WSDL` de descripción de los servicios web.
- Crear los documentos `WSDL` de descripción de servicios propios.

Los servicios web permiten a las aplicaciones comunicarse con otras utilizando la web (el protocolo HTTP) como medio de transmisión. Sin embargo, los mecanismos que se han utilizado hasta ahora no son la única forma de implementar y utilizar servicios web. Desde hace un tiempo han ido apareciendo servicios web que utilizan arquitecturas basadas en `REST` (REpresentational State Transfer).

`REST` hace referencia a un conjunto de normas que se pueden utilizar para establecer mecanismos de comunicación con servicios web. Concretamente, un servicio web implementado mediante `REST` debería al menos:

- Utilizar una estructura de **URIs** para acceder a los recursos gestionables mediante el servicio web, por ejemplo:
  `/articulo/KSTDTG332GBR`
  `/tienda/CENTRAL`
- Usar los distintos métodos HTML(*GET*, *HEAD*, *POST*,…) para las peticiones. Por ejemplo, se podría utilizar el método `GET` para obtener información de un artículo:
  `GET /articulo/KSTDTG332GBR`
  Y el método `HTTP DELETE` para borrarlo:
  `DELETE /articulo/KSTDTG332GBR`
- No almacenar información sobre el estado: todas las peticiones se tratarán de forma independiente y deben incluir toda la información necesaria para poder atenderla.
- Utilizar JSON o `XML` en sus comunicaciones (o incluso ambos).

Muchos servicios web actuales se implementan utilizando arquitecturas *REST*.

2.- Aplicaciones web híbridas

Una aplicación web híbrida se caracteriza por combinar datos y/o funcionalidades procedentes de diversos orígenes para formar un nuevo tipo de aplicación o servicio.

Los tipos de fuentes de información más habituales que se utilizan en una aplicación web híbrida son:

- Información proveniente de servicios web, disponible mediante diversos protocolos y estructurada utilizando formatos de intercambio como **JSON**. En ocasiones el proveedor del servicio ofrece también un interface de programación (API) para facilitar el acceso a los datos. Es el caso de compañías como `Google, Microsoft, Amazon`...

  En otras ocasiones los datos se ofrecen de forma pública utilizando protocolos de redifusión

web (también conocido como "sindication web") como ***RSS*** o ***Atom***, y puede ser necesario procesarlos para extraer la información necesaria.

- Información generada y gestionada por el propietario de la aplicación web híbrida, como pueden ser datos internos de una empresa.

De forma menos habitual, podemos encontrarnos aplicaciones web que utilicen técnicas de ingeniería inversa para extraer los datos que se muestran en algunos sitios web, como puede ser el caso de los precios de los productos en las tiendas web. Estas técnicas se conocen por su nombre en inglés: `web scraping (ver ejemplo api.ipify.org/www.cualesmiip.es)`.

3.- Repositorios de información

Cuando se utilizan servicios de terceros para desarrollar una aplicación web híbrida, se debe tener en cuenta que en ocasiones existen condiciones y límites al uso que puedes hacer de los mismos.

La mayoría de las grandes compañías que proveen servicios web al usuario, como `Google`, requieren un registro previo y ofrecen unas condiciones para su uso gratuito que dependen del servicio al que se necesite acceder. Algunos de estos servicios ofrecen una versión adicional de pago con mejores condiciones.

En muchas ocasiones, el proveedor del servicio (aunque también puede ser un tercero) ofrece además librerías que facilitan la utilización del servicio desde un lenguaje de programación determinado y ejemplos de utilización del mismo. Por ejemplo, si se quiere utilizar la API de `Google Tasks` existen librerías de programación para los lenguajes `Java, Python, PHP` y para la plataforma `.Net de Microsoft`.

Para que se pueda verificar la utilización que hace cada usuario de un servicio determinado, es necesario incluir dentro del código que interactúa con el mismo una clave personal que le identifica en el sistema. Por ejemplo, si quieres utilizar la de `Google Books`, necesitarás indicar tu código de desarrollador al hacer una consulta de forma similar a:

```
$client->setDeveloperKey('la clave de desarrollador va aquí');
```

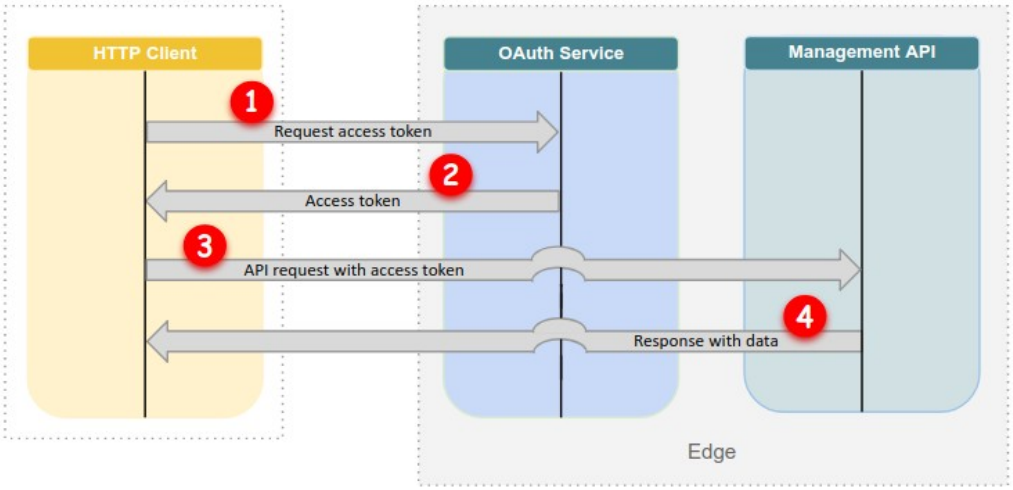
Existen ciertos servicios web que nos permiten acceder a información privada que almacenan de sus usuarios. Por ejemplo, la de `Google Calendar` posibilita gestionar la información que cada usuario mantiene en sus calendarios personales. Si vas a usar un servicio de este tipo (como `Google Tasks`), necesitarás que tu aplicación solicite permiso a los propietarios de la información antes de poder acceder a la misma. Para ello muchos proveedores de servicios web utilizan un protocolo llamado `OAuth` (la versión actual es la 2.0).

3.1. OAuth2

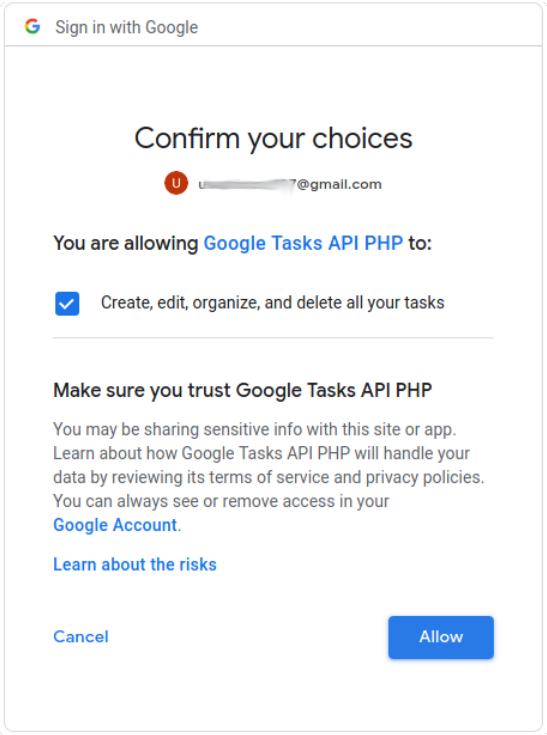
El protocolo estándar de autorización `OAuth2`, permite a una aplicación externa obtener acceso a información de carácter privado a través de un servicio web. Para ello establece un acuerdo de acceso a la misma entre la aplicación externa, el servicio web y el propietario de los datos a los que se solicita el acceso.

Por ejemplo, si una aplicación "X" solicita a `Google` acceso a los calendarios del usuario "`gestor`", Google pedirá a "`gestor`" permiso indicándole qué aplicación es la que solicita el acceso y a qué información. Si el usuario "`gestor`" otorga permiso, la aplicación "X" podrá acceder a los datos que solicitó a través del servicio de `Google`.

OAuth2 funciona de forma similar pero ligeramente distinta dependiendo de quién solicite acceso a la información. En nuestro caso supondremos que el solicitante será siempre una aplicación web. Veamos por ejemplo qué sucede cuando nuestra aplicación necesita acceder a información personal del usuario a través del servicio de `Google Tasks`. En este caso, los pasos que se seguirán son los siguientes:



- La aplicación web se comunica con el servidor de autorización `OAuth2`, indicando la información a que quiere acceder y el tipo de acceso a la misma.
- El servidor de autorización `OAuth2` requiere al usuario de la aplicación web a que inicie sesión con su cuenta de `Google` (si aún no lo ha hecho), y le redirige a una página en la que le pide su consentimiento para otorgar acceso a su información privada.



- Si el usuario da su consentimiento, el servidor de autorización `OAuth2` devuelve a la aplicación web un código de autorización.
- Antes de poder acceder a la información privada del usuario, la aplicación web debe

intercambiar ese código de autorización por otro código de acceso.

- Utilizando el código de acceso, la aplicación puede utilizar el servicio de `Google Tasks` para gestionar la información privada del usuario, dentro de los límites de acceso que se han otorgado.
- Los códigos de acceso tienen un tiempo de vida limitado. Cuando caducan, la aplicación ha de comunicarse de nuevo con el servidor de autorización `OAuth2` para obtener un código de refresco.

En las páginas de documentación de los distintos servicios suele aparecer información de como acceder a los mismos en distintos lenguajes de programación.

3.2. JSON y XML

La información obtenida de un servicio web puede ser bastante sencilla o tener cierto grado de complejidad. Por ejemplo, cuando utilizas un servicio de geocodificación para averiguar las coordenadas concretas de una dirección, obtienes básicamente esas coordenadas. Pero cuando se utiliza un servicio como `Bing Maps Routes` para averiguar la ruta a seguir entre dos puntos, la respuesta que se obtiene es una ruta compuesta por una serie de indicaciones a seguir para llegar al destino.

Los formatos más utilizados por los servicios web para dar formato a esas respuestas son dos: JSON y XML. Puede que haya que adaptar el código al tipo de respuesta específica que ofrezca el servicio, también es posible que se nos permita escoger el tipo de respuesta que se prefiere. Veamos cómo se pueden procesar de forma sencilla desde PHP mensajes en ambos formatos.

A partir de la versión **5.2** de PHP, se incluye por defecto la extensión *JSON*. Su funcionamiento es muy sencillo. Incorpora dos funciones para tratar con cadenas de texto en notación:

- `json_decode`. Decodifica una cadena de texto y la transforma en un objeto (opcionalmente también se podría convertir en un array si le pasamos la opción `true`).

```php
$cadena_json = '{"negro":1,"rojo":2,"verde":3,"azul":4}';
$objeto_php = json_decode($cadena_json);
$array_php = json_decode($cadena_json, true);
```

- `json_encode`. Función inversa a la anterior. Devuelve una cadena de texto en notación a partir del contenido de una variable PHP.

```php
$arr = array('negro' => 1, 'rojo' => 2, 'verde' => 3, 'azul' => 4);
$cadena_json = json_encode($arr,JSON_UNESCAPED_UNICODE);
```

La extensión *SimpleXML*, habilitada por defecto a partir de PHP 5.1.2, facilita la tarea de extraer información de un documento XML. La extensión convierte un documento XML en un objeto de la clase SimpleXMLElement. Se puede cargar el documento:

- desde una cadena de texto, utilizando la función simplexml_load_string.

```php
$xml = simplexml_load_string($cadena);
```

- desde un fichero, utilizando la función simplexml_load_file. Puedes utilizar una dirección URI como origen, por ejemplo:

```php
$xml = simplexml_load_file('http://localhost/dwes/ut8/config.xml');
```

Los nodos y atributos del documento XML se convierten en miembros del objeto devuelto. Los nodos pasan a ser arrays que contienen a su vez como elementos los atributos y subnodos del

documento XML.

Veamos con un ejemplo el uso de la función anterior. Partimos de un fichero ***Alumnos.xml*** (obtenido de un servicio web) y queremos mostrar los datos obtenidos en formato XML en una tabla HTML.

Alumnos.xml:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Alumnos>
    <Alumno>
        <dni>12345678Z</dni>
        <nombre>Juan Manuel</nombre>
        <ap1>Carmona</ap1>
        <ap2>Velasco</ap2>
        <nota>8.5</nota>
    </Alumno>
    <Alumno>
        <dni>12345679S</dni>
        <nombre>Juana</nombre>
        <ap1>Núñez</ap1>
        <ap2>Álvarez</ap2>
        <nota>9.5</nota>
    </Alumno>
    <Alumno>
        <dni>12345679S</dni>
        <nombre>Alicia</nombre>
        <ap1>Martínez</ap1>
        <ap2>Alonso</ap2>
        <nota>9.75</nota>
    </Alumno>
</Alumnos>
```

Si ejecutamos el siguiente código, ***ProcesarAlumnos.php:***

```php
<?php declare(strict_types=1); ?>
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>Notas de la asignatura</title>
    <style>
        table { margin:0 auto;}
        th, td { border: 1px solid black;}
        th { background-color: orange;}
        tr:nth-child(odd) { background-color: aqua;}
        tr:nth-child(even) { background-color: lightcoral;}
    </style>
</head>
<body>
    <?php
        $alumnos=simplexml_load_file("./Alumnos.xml");
        echo "\t<table>\n";
        echo "\t\t<thead>
            <tr><th>DNI</th><th>Apellidos, Nombre</th><th>Nota</th></tr>
        </thead>\n";
        foreach($alumnos->Alumno as $alu) {
            echo "<tr><td>$alu->dni</td><td>$alu->ap1 $alu->ap2, $alu->nombre</td><td>$alu->nota</td></tr>\n";
        }
        echo "\t\t<thead>\n";
        echo "\t<table>\n";
    ?>
</body>
</html>
```

Obtendremos el siguiente resultado en el navegador:

DNI	Apellidos, Nombre	Nota
12345678Z	Carmona Velasco, Juan Manuel	8.5
12345679S	Núñez Álvarez, Juana	9.5
12345679S	Martínez Alonso, Alicia	9.75

Estamos recorriendo el árbol que representa al documento (D.O.M.) como una lista de nodos ***alumno***.

Hay que recordar que hay tecnologías alternativas para hacer esto, como ***XSL***, lo mismo podría haberse hecho con XSL.

Alumnos.xsl:

```
<?xml version='1.0' encoding='utf-8'?>
<xsl:stylesheet version='2.0' xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
    <xsl:output encoding='utf-8' indent='yes' method='html' doctype-system='about:legacy-compat' />
    <xsl:template match='/'>
        <html>
            <head>
                <meta charset="utf-8"/>
                <title>Notas de la asignatura</title>
                <style>
                    table { margin:0 auto;}
                    th, td { border: 1px solid black;}
                    th { background-color: orange;}
                    tr:nth-child(odd) { background-color: aqua;}
                    tr:nth-child(even) { background-color: lightcoral;}
                </style>
            </head>
            <body>
                <h1></h1>
                <table>
                    <thead>
                        <tr><th>DNI</th><th>Apellidos, Nombre</th><th>Nota</th></tr>
                    </thead>
                    <tbody>
                        <xsl:for-each select="Alumnos/Alumno">
                            <tr>
                                <td><xsl:value-of select="dni"/></td>
                                <td>
                                    <xsl:value-of select="ap1"/>
                                    <xsl:value-of select="ap2"/>,
                                    <xsl:value-of select="nombre"/>
                                </td>
                                <td>
                                    <xsl:value-of select="nota"/>
                                </td>
                            </tr>
                        </xsl:for-each>
                    </tbody>
                </table>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

4.- Aplicaciones web híbridas

Para crear aplicaciones web híbridas, habrá que acceder a diversos servicios para obtener información. En muchas ocasiones esos servicios estarán accesibles mediante HTML. En otras ocasiones, especialmente cuando se acceda a información privada de un usuario, se necesitará utilizar un protocolo seguro como HTTPS.

Tanto si se va a generar código para acceder a un servicio web, como si se utiliza una API ya programada, es posible que se necesita utilizar la librería *cURL*.

cURL es una librería (y también una herramienta de línea de comandos), que permite utilizar de forma sencilla varios protocolos de comunicaciones para transmitir información. Soporta, entre otros, los protocolos ***HTTP***, ***HTTPS***, ***FTP***, ***TFTP***, ***TELNET***, ***LDAP***, ***SMTP***, ***POP3*** e ***IMAP***. Viene ya preinstalada, pero si no la tuviésemos: "sudo apt install phpX.x-curl", donde X.x es la versión de PHP con la que estemos desarrollando nuestras aplicaciones. En caso de que tener que instalar hay que reiniciar ***Apache2***: ***sudo systemctl restart apache2***.

Muchos de los servicios REST disponibles en Internet, tienen una versión gratuita para permitirnos hacer pruebas y una de pago mucho más robusta para uso profesional. A continuación veremos algunos ejemplos de uso de APIs REST disponibles en Internet. Para hacer uso de estos servicios deremos registrarnos en la mayoría de los casos.

openweathermaps.org

En este servicio podemos abrir una cuenta gratuita que nos permite hacer un uso no profesional del servicio. Para ello, previo registro, nos suministran una ***appid*** que no es más que un token que nos permite acceder al servicio sin tener que pasar por una autenticación.

En el ejemplo siguiente se muestra un ejemplo de acceso al servicio, mediante cURL, para obtener los parámetros en tiempo real del estado del tiempo, en una serie de poblaciones de Tenerife:

```php
<?php declare(strict_types=1); ?>
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
    <title>Temperaturas Tenerife</title>
    <?php require_once("../../recursos/librerias/tablasCss.php"); ?>
</head>
<body>
    <h1>Temperatuas de las ciudades más importantes de Tenerife</h1>
    <?php
        function getOpts($ciudad,$pais="es") {
            $ciudadCodificada=str_replace(" ","+",$ciudad).",$pais";
            return [
                CURLOPT_URL => "https://api.openweathermap.org/data/2.5/weather?units=metric&q=$
{ciudadCodificada}&appid=ff5858920c3ec4557b28e843582402d8",
                CURLOPT_HEADER => false,
                CURLOPT_RETURNTRANSFER => true
            ];
        }
        function curlGet($ciudad,$pais="es") {
            $opts=getOpts($ciudad);
            $curlHandle=curl_init();
            curl_setopt_array($curlHandle,$opts);
            $respuesta=curl_exec($curlHandle);
            curl_close($curlHandle);
            return json_decode($respuesta);
        }
        $ciudades=[
            "Puerto de la Cruz",
            "San Cristóbal de La Laguna",
            "Santa Cruz de Tenerife",
            "La Orotava",
            "Candelaria",
            "Icod de los Vinos",
            "Garachico",
            "Los Cristianos"];
        //"units"=>"metric","appid"=>"465f3183260a033f5ac5ca9397e5fda8";
        $campos=["Longitud"=>"coord->lon","Latitud"=>"coord->lat","Tmin"=>"main->temp_min","Tmax"=>"main-
>temp_max","Presión"=>"main->pressure","Humedad"=>"main->humidity","Visibilidad"=>"visibility","Viento"=>"wind-
>speed","Nubes"=>"clouds->all"];
        echo "\t<table>
        <thead>
            <tr><th>Ciudad</th>";
        foreach($campos as $clave=>$campo) {
            echo "<th>$clave</th>";
        }
        echo "</tr>
        </thead>
        <tbody>\n";
        //https://api.openweathermap.org/data/2.5/weather?
units=metric&q=Puerto+de+la+Cruz,es&appid=465f3183260a033f5ac5ca9397e5fda8
        foreach($ciudades as $ciudad) {
            $datos=curlGet($ciudad);
            if (isset($datos->cod) && $datos->cod!=200) { //200 significa respuesta Ok
                continue;
            }
            echo "<tr>";
            echo "<th>$ciudad</th>";
            foreach($campos as $campo=>$propiedad) {
                echo "<td class='derecha'>",eval("return \$datos->$propiedad;"),"</td>";
            }
            echo "</tr>";
        }
        echo "\t\t</tbody>\n";
        echo "\t<table>\n";
    ?>
</body>
</html>
```

- `CURLOPT_URL`: especifica la URL a la que deseas hacer una solicitud.

- `CURLOPT_RETURNTRANSFER`: establece que el resultado de la solicitud se debe devolver como una cadena.
- `CURLOPT_FOLLOWLOCATION`: establece que se deben seguir las redirecciones.
- `CURLOPT_HEADER`: establece que se deben incluir los encabezados de la respuesta en la cadena resultante.
- `CURLOPT_POST`: establece que la solicitud se hace con el método POST.
- `CURLOPT_POSTFIELDS`: especifica los campos del formulario que se enviarán con la solicitud POST.
- `CURLOPT_SSL_VERIFYPEER`: establece si se deben verificar los certificados SSL del servidor.
- `CURLOPT_SSL_VERIFYHOST`: establece si se deben verificar los nombres de host de los certificados SSL del servidor.
- `CURLOPT_HTTPHEADER`: especifica los encabezados HTTP adicionales que se deben incluir en la solicitud.
- `CURLOPT_USERAGENT`: establece el agente de usuario que se enviará en la solicitud.
- `CURLOPT_CONNECTTIMEOUT`: establece el tiempo de espera en segundos para establecer una conexión con el servidor.