

# Aplicaciones web híbridas.



## Caso práctico

Después de semanas de trabajo, **Carlos** da por finalizado el desarrollo de la aplicación en la que ha estado trabajando. Ha tenido que reescribir el código en varias ocasiones, cambiar la apariencia de algunas pantallas del interfaz, y retocar el esquema de la base de datos que habían diseñado en un principio, pero finalmente parece que el trabajo ha dado sus frutos. Habla con **Juan** y entre los dos revisan el resultado.



**Juan** está muy contento con lo que ve, y le propone hablar con **Ada**, para mostrarle la aplicación web. Aunque la directora ha seguido el progreso de la misma, en gran parte ha sido en las conversaciones que ha mantenido con **Juan**, y hace ya tiempo que no le informan directamente sobre los últimos avances.



[Ministerio de Educación y Formación Profesional](#) (Dominio público)

**Materiales formativos de FP Online propiedad del Ministerio de Educación y Formación Profesional.**

[Aviso Legal](#)

# 1.- Reutilización de código e información.



## Caso práctico

**Ada** revisa la aplicación web y les propone retocar un par de detalles, sobre todo relativos a la apariencia. Con los años de experiencia que tiene en el desarrollo de aplicaciones, les ofrece también algunos consejos sobre la usabilidad de los interfaces, y le indica que es muy importante seguir algunas normas básicas que garanticen la accesibilidad de la aplicación.

Por último, tienen que hablar con su amigo **Esteban** para concretar los detalles de la implantación de la aplicación en las dependencias del cliente. Parece que están a punto de finalizar el proyecto.




La **unidad 6** de este módulo tenía por título "Servicios Web". En ella aprendiste a crear y utilizar servicios web, empleando una arquitectura orientada a servicios (SOA). Los principales temas que practicaste en esa unidad son:

- ✓ A utilizar el protocolo SOAP para comunicarte con un servicio web. Con ayuda de la clase "**SoapClient**", intercambiabas peticiones y respuestas en formato SOAP con un servicio web existente.
- ✓ A crear tu propio servicio web. Mediante la clase "**SoapServer**" podías publicar tus propias funciones para que fueran accesibles como servicio web mediante SOAP.
- ✓ A procesar los documentos WSDL de descripción de los servicios web, y a crear los documentos WSDL de descripción de tus propios servicios.

Los servicios web permiten a tus aplicaciones comunicarse con otras utilizando la web (el protocolo HTML) como medio de transmisión. Sin embargo, los mecanismos que has 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.

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:  
`/articulo/KSTDTG332GBR`  
  
`/tienda/CENTRAL`
- ✓ Usar los distintos  métodos HTML para las peticiones. Por ejemplo, se podría utilizar el método HTML 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 XML o JSON en sus comunicaciones (o incluso ambos).



## Para saber más

REST no es un estándar, pero muchos servicios web actuales se implementan utilizando arquitecturas de tipo REST. Existen en Internet varios documentos sobre REST y ejemplos de su utilización que conviene revisar.

[Artículo sobre REST.](#)

En la presente unidad crearás aplicaciones que utilicen diversos servicios web.

## 2.- Características de las aplicaciones web híbridas.



### Caso práctico

En poco tiempo, **Carlos** modifica los detalles que había apreciado **Ada**. Está contento con el resultado obtenido, pero con la experiencia que ha adquirido en programación web durante su desarrollo, sabe que habría algunos detalles que se podrían añadir a la aplicación y que influirían positivamente en la experiencia del usuario.



**Ada** le llama y le comenta que la próxima semana **Esteban** vendrá a BK Programación a conocer la aplicación. **Carlos** guarda una copia de la aplicación en su estado actual, y decide tomarse esa semana para intentar mejorarla en algunos aspectos. Hace una lista de los detalles que le podría añadir, y al revisarla se da cuenta de que otras aplicaciones que conoce ya los implementan; pero en muchos casos, no son desarrollos propios sino que se basan en servicios ofrecidos por terceros: mapas de **Microsoft**, imágenes de **Flickr**,... ¿Podrá aprovechar algún servicio existente e integrarlo en su propia aplicación?

Una aplicación web híbrida, también conocida por su nombre en inglés (**mashup**), 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** o **AJAX**. En ocasiones el proveedor del servicio ofrece también un interface de programación (**API**) para facilitar el acceso a los datos. Es el caso de las **API** de compañías como **Google**, **Yahoo!**, **Flickr**, **Microsoft** o **Amazon**.

En otras ocasiones los datos se ofrecen de forma pública utilizando protocolos de redifusión web (también conocido como "syndication web") como **RSS** o **Atom**, y puede ser necesario procesarlos para extraer la información necesaria.

#### [Redifusión web.](#)

- ✓ 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**.

Por ejemplo, podrías montar una aplicación web híbrida que utilice la API de **Bing Maps**, e información de ubicación geográfica de las franquicias de una empresa para mostrar la localización de las tiendas en un mapa.

En esta unidad, vas a programar una aplicación web híbrida para la tienda web con la que has venido trabajando. En este caso se trata de facilitar la gestión de los envíos de las compras.



## Autoevaluación

Las siglas REST hacen referencia a un estándar que se utiliza en la implementación de servicios web

- ☐ Verdadero.
- ☐ Falso.

Incorrecta. Revisa el enunciado.

Efectivamente. Aunque el enunciado pueda parecer correcto, REST no es un estándar sino un conjunto de normas.

## Solución

1. Incorrecto
2. Opción correcta

## 3.- Utilización de repositorios de información.



### Caso práctico

**Carlos** se informa sobre los servicios web que pueden serle útiles, y decide añadir a la aplicación una funcionalidad que no tiene: un servicio de gestión de envíos para los productos que se vendan. En realidad nadie ha solicitado esa funcionalidad, pero cree que si es capaz de programarla la empresa de **Esteban** la llegará a utilizar. En ocasiones los clientes no piden una característica simplemente porque desconocen que es posible realizarla. Y cuando comenzó este proyecto en particular, no había nadie en BK Programación con la experiencia suficiente como para orientar correctamente al cliente.



e pone manos a la obra. Si en la semana que tiene le da tiempo a finalizarla, se la mostrará a **Esteban**. Y si no le da tiempo, echará mano de la versión anterior. De cualquier modo, no es tiempo perdido. Este proyecto le está sirviendo para adquirir experiencia que a buen seguro aprovechará en el futuro inmediato.

Cuando utilices servicios de terceros para desarrollar una aplicación web híbrida, deberás tener en cuenta que en ocasiones existen condiciones y límites al uso que puedes hacer del mismo.

La mayoría de las grandes compañías que proveen servicios web al usuario, como **Google** o **Yahoo!**, requieren un registro previo y ofrecen unas condiciones para su uso gratuito que dependen del servicio al que necesites 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 quieres 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 API 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');
```

De la misma forma, si quieres acceder al servicio del tiempo **Weather** de **Yahoo!**, tendrás que indicar en la consulta un identificador de aplicación.



[Faager](#) (Dominio público)

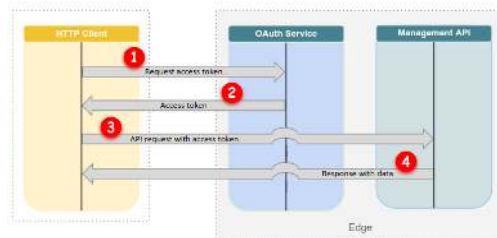
Existen ciertos servicios web que nos permiten acceder a información privada que almacenan de sus usuarios. Por ejemplo, la API de **Google Calendar** posibilita gestionar la información que cada usuario mantiene en sus calendarios personales. Si vas a usar un servicio de este tipo (como **Google Tasks**), 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.

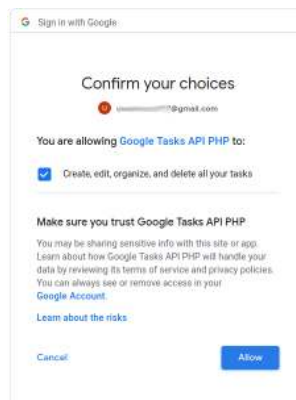
Por ejemplo, si una aplicación "X" solicita a **Google** acceso a los calendarios del usuario "gestor", Google pedirá a "gestor" permiso indicándole qué aplicación es la que solicita el acceso y a qué información. Si el usuario "gestor" otorga permiso, la aplicación "X" podrá acceder a los datos que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



[apigee](#) (CC BY-SA)

- ✓ 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.



Google y Firefox (Elaboración propia)

- ✓ 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.



## Para saber más

Existe una extensión **PHP** para programar las partes cliente y servidor del protocolo **OAuth**. No obstante en las páginas de documentación de los distintos servicios nos suele aparecer información de como acceder a los mismos en distintos lenguajes de programación.

[Extensión OAuth.](#)



## Autoevaluación

**Al utilizar OAuth2, cuando tu aplicación solicite información privada de un usuario, deberá acreditar su autorización utilizando:**

- ☐ Un código de autorización.
- ☐ Un código de acceso.

Incorrecta. El código de autorización lo recibe del servidor **OAuth2** cuando se certifica que tu aplicación está autorizada a acceder a información privada de un usuario, pero ¿utiliza ese directamente para acceder a la información?

Efectivamente. Antes de acceder a información privada de un usuario, tu aplicación deberá estar en posesión de un código de autorización, pero tendrá que utilizarlo para conseguir un código de acceso, que será el que finalmente utilice.

## Solución

1. Incorrecto
2. Opción correcta

## 3.2.- JSON y XML.

Muchas de las operaciones que se llevan a cabo cuando utilizas un servicio web implican la obtención de cierta información por parte del mismo. La información obtenida 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 utilizas un servicio como **Bing Maps Routes** para averiguar la ruta a seguir entre dos puntos, la respuesta que obtienes es una ruta compuesta por una serie de indicaciones a seguir para llegar al destino.



[kreatikar](#) (([Pixabay License](#)))

Los formatos más utilizados por los servicios web para dar formato a esas respuestas son dos: **JSON** y **XML**. En algunos casos tendrás que adaptar tu código al tipo de respuesta que ofrezca el servicio. Otros servicios permiten que escojas el tipo de respuesta que prefieras. 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**: [Extensión JSON](#).

- ✓ **json\_decode**. Decodifica una cadena de texto **JSON** y la transforma en un objeto **PHP** (opcionalmente también se podría convertir en un array si le pasamos la opción **true**).

```
$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 **JSON** a partir del contenido de una variable **PHP**.

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

Así como las opciones para trabajar con **JSON** desde **PHP** están bien definidas, para utilizar **XML** hay más variedad de herramientas para escoger. En la versión **4** de **PHP** podías utilizar dos formas para procesar documentos en formato XML: **DOM** y **SAX**. La extensión **SimpleXML**, habilitada por defecto a partir de **PHP** 5.1.2, facilita la tarea de extraer información de un documento **XML**. Vamos a ver su funcionamiento.

[Extensión SimpleXML](#).

La extensión convierte un documento XML en un objeto de la clase `SimpleXMLElement`. Puedes cargar el documento:

- ✔ desde una cadena de texto, utilizando la función `simple_load_string`.

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

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

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

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



## Debes conocer

Es importante que conozcas cómo procesar documentos XML para extraer información de los mismos. En el manual de PHP puedes encontrar información sobre la utilización de la extensión `SimpleXML`.

[Extensión SimpleXML.](#)

## 4.- Creación de aplicaciones web híbridas.



### Caso práctico

Pasa la semana y **Carlos** consigue tener a punto la nueva versión de la aplicación. Para la presentación, pone las dos versiones en funcionamiento, y no le comenta a nadie los nuevos cambios. Si hay algún problema con las últimas modificaciones, echará mano de la versión anterior.



uando en BK Programación muestran a **Esteban** la aplicación, éste queda asombrado por el resultado. El nuevo servicio de gestión de envíos sorprende a todos positivamente, especialmente a **Ada** y a **Juan**, que empiezan a darse cuenta de las nuevas capacidades que ha adquirido **Carlos** en las últimas semanas. **Esteban** comenta que muy posiblemente les sea útil, especialmente al poder consultar la información de los envíos desde un dispositivo móvil. Pasarán un par de meses probando la aplicación, y a continuación plantea la posibilidad de tener una nueva reunión para hablar de posibilidades de ampliación y de otros proyectos. Ha quedado muy contento con el trabajo de **Carlos** y quiere seguir contando con él próximamente.

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

Tanto si vas a generar tu propio código para acceder a un servicio web, como si utilizas una API ya programada, es posible que necesites utilizar la librería cURL.

#### [Manual de PHP](#)

cURL es una librería (y también una herramienta para utilizar desde la 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. Es importante que te asegures de que tu instalación de PHP incluye dicha librería. Si no la tienes activada en tu instalación de PHP, en Ubuntu puedes instalarla ejecutando: `sudo apt-get install php-curl`

Recuerda reiniciar Apache tras la instalación, y comprueba que la ejecución de `phpinfo()` muestra algo como lo siguiente





## Autoevaluación

La librería `cURL` debe utilizar una lista válida de autoridades de certificación:

- ☐ Para poder acceder a servidores web utilizando `HTTPS`.
- ☐ Para poder acceder a servidores web utilizando `HTTP`.

Efectivamente. Para poder certificar la validez del certificado digital que envía el servidor cuando se utiliza `HTTPS`, `cURL` necesita una lista válida de autoridades de certificación.

Incorrecta. Al utilizar `HTTP` no se utilizan certificados digitales.

## Solución

1. Opción correcta
2. Incorrecto

## 4.1.- Yahoo! Weather.

Existen muchas fuentes de datos disponibles en Internet que puedes utilizar para construir una aplicación web híbrida. En esta unidad vamos a centrarnos en la información accesible a través de servicios web, concretamente en los que ofrecen las empresas **Google**, **Yahoo!** y **Microsoft**.

Veamos como funciona el servicio **REST** de **Yahoo! Weather**. Puedes encontrar toda la información necesaria sobre su utilización en la web de desarrollo de **Yahoo!**.

[Yahoo! Weather.](#)



[Mudassar Iqbal \(Pixabay License\)](#)

Como has podido observar, si has visitado el enlace superior para usar este servicio debemos disponer de una "API KEY". En dicho enlace nos explican como podemos hacernos de una.

El servicio **Yahoo! Weather** se basa en **REST** y los datos relativos a la petición se envían mediante parámetros **GET**. Es necesario indicar como mínimo un parámetro de localización.

La forma más sencilla de indicar una localización es utilizando el parámetro **location** (o su equivalente **q**).

Existe otro tipo de parámetros, los de control, que permiten indicar otra información no directamente relacionada con la localización. Es obligatorio el parámetro **appid** para indicar el ID de tu aplicación web. Otros parámetros de control son:

### Algunos parámetros de control del servicio de geocodificación Yahoo! PlaceFinder.

Parámetro de control	Significado
<b>location</b>	Nombre de la ciudad por ejemplo 'almeria-es'
<b>lat</b>	Latitud
<b>lon</b>	Longitud
<b>format</b>	Formato de la respuesta, los valores son: <b>format=xml (default)</b> or <b>format=json</b>
<b>u</b>	Unidades <b>u=f</b> (imperial default) o <b>u=c</b> (sistema métrico)
<b>woeid</b>	id único de la localización

En la documentación del servicio tienes una lista completa de los parámetros de control que puedes emplear.

[Parámetros de control.](#)



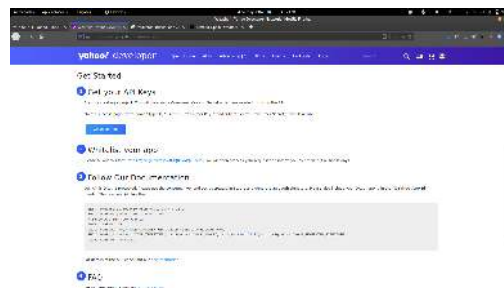
Por ejemplo, una petición simple podría tener la siguiente forma, fíjate que aunque los parámetros se pase por GET se usan 📄 URL amigables:

<https://www.yahoo.com/news/weather/spain/almeria/almeria-752212>

Recuerda incluir en las peticiones a los servicios de **Google**, **Microsoft** y **Yahoo!**, tus propios identificadores (los que te han asignado al registrarte) allí donde sea necesario.

Como ya hemos visto, para poder usar este servicio, es necesario registrarnos y crear una aplicación, una vez creada nos darán el **id** de nuestra aplicación, nuestro **id** de cliente y nuestro "**client secret**" que necesitaremos:

### Creándonos una App



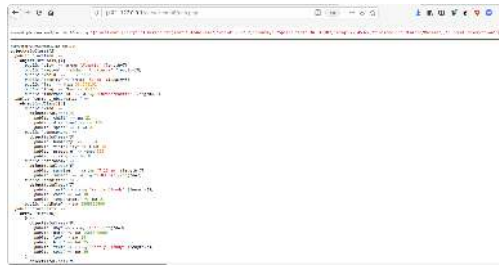
Yahoo y Firefox (Elaboración propia)

### Datos de la App creada



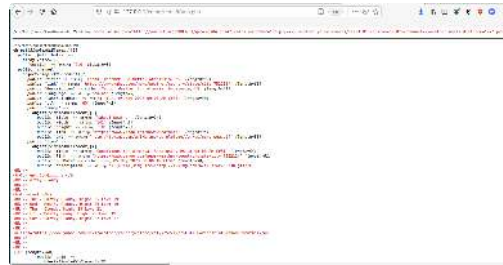
Yahoo y Firefox (Elaboración propia.)





Firefox (Captura de pantalla elaboración propia)

Salida en XML (`new SimpleXMLElement()`)



Firefox (Captura de pantalla elaboración propia)



## Autoevaluación

Relaciona las palabras con su significado relativo al servicio "Yahoo! Weather":

### Ejercicio de relacionar.

Palabra	Relación	Significado
woeid	<input type="checkbox"/>	1. Clave del usuario.
consumer_secret	<input type="checkbox"/>	2. Unidades en las que nos devolverá la información
u	<input type="checkbox"/>	3. Id único de la localización
format	<input type="checkbox"/>	4. Formato de la respuesta <u>JSON</u> o <u>XML</u>

Enviar

En la pregunta figuran tanto parámetros que se utilizan al hacer las peticiones, como elementos que se incluyen en las respuestas.

## 4.2.- Bing Maps

Bing Maps es una web de mapas creada por Microsoft para su buscador Bing. Su principal competidor es Google Maps. En Windows 10 viene preinstalada con el nombre de Windows Maps. Puedes encontrar información de todos los servicios REST que nos ofrece Bing Maps en el enlace siguiente: [Servicios Bing Maps](#).

Como habrás podido ver en la documentación, puedes utilizar, entre otros, los siguientes parámetros:



[Mudassar Iqbal](#) (Pixabay License)

### Algunos parámetros del servicio Bing Maps.

Parámetro	Significado
locality	Localidad de la que quieres obtener las coordenadas.
postalCode	Código postal ( <b>opcional</b> )
point	Las coordenadas de la ubicación que desea hacer reverse geocoding. Un punto se especifica por una latitud y una longitud. Son obligatorias si queremos obtener una dirección desde una coordenadas. El formato es (lat, lon) 47.64054, -122.12934
addressLine	Dirección ( <b>opcional</b> )
countryRegion	Código <u>iso</u> del país (por ejemplo ES) ( <b>opcional</b> )
inclnb	Incluye el barrio, si está disponible ( <b>opcional</b> ) Los valores son: 1 (incluye el barrio) 0 (no incluye el barrio, valor por defecto)
incl	El único valor para este parámetro es <u>ciso2</u> . Cuando especifica <code>incl = ciso2</code> , el código de país <u>iso</u> de dos letras se incluye para las direcciones en la respuesta. ( <b>opcional</b> )

Parámetro	Significado
maxResults	Especifica el número máximo de localizaciones de la respuesta, se espera un número entre <b>1</b> y <b>20 (opcional)</b> . El valor por defecto es <b>5</b>
o	Formato de salida por defecto es <b>JSON</b> , si se indica "o=xml" nos dará la salida en <b>XML</b>
c	se intenta dar la respuesta en el idioma/cultura especificada ( <a href="#">Lista de culturas soportadas</a> )

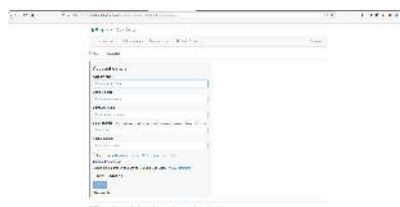
Para usar el servicio **Bing Maps** necesitas crearte una "**KEY**" de desarrollador registrado en **Microsoft**. Puedes crearte una usando las instrucciones del enlace siguiente. Con una key de tipo **Basic Key** es suficiente. [Página para crearte una Bing Maps Keys.](#)

### Creando key



Firefox (Elaboración propia)

### Creando key



Licencia: Eleboración propia

key creada



Captura de pantalla Firefox (Elaboración propia)

El formato de una petición de esta API será:

```
http://dev.virtualearth.net/REST/v1/Locations/ES/04005/almeria/addressLine?inclnb=0&incl=ciso2&maxResu
```

Por defecto la salida nos la da en JSON, la respuesta obtenida a esa petición es:

```
{
  "resourceSets": [
    {
      "estimatedTotal": 1,
      "resources": [
        {
          "__type": "Location:http://schemas.microsoft.com/search/local/ws/rest/v1",
          "bbox": [
            36.836570739746094,
            -2.4601500034332275,
            36.85546875,
            -2.4520499706268311
          ],
          "name": "04005, Andalusia, Spain",
          "point": {
            "type": "Point",
            "coordinates": [
              36.842872619628906,
              -2.4560999870300293
            ]
          },
          "address": {
            "adminDistrict": "Andalusia",
            "adminDistrict2": "Almería",
            "countryRegion": "Spain",
            "formattedAddress": "04005, Andalusia, Spain",
            "locality": "Almería",
            "postalCode": "04005",
            "countryRegionIso2": "ES"
          },
          "confidence": "Medium",
          "entityType": "Postcode1",
          "geocodePoints": [
            {
```

```

        "type": "Point",
        "coordinates": [
            36.842872619628906,
            -2.4560999870300293
        ],
        "calculationMethod": "Rooftop",
        "usageTypes": [
            "Display"
        ]
    }
],
"matchCodes": [
    "Ambiguous",
    "UpHierarchy"
]
}
]
}
],
}
}

```

Si queremos la salida en XML nuestra consulta sería:

```
http://dev.virtualearth.net/REST/v1/Locations/ES/04005/almeria/addressLine?inclnb=0&incl=ciso2&o=xml&k
```

Si queremos hacer una búsqueda a partir de la longitud y latitud (**reverse geocoding**):

```
https://dev.virtualearth.net/REST/v1/Locations/36.835236,%20-2.463084?o=json&c=ES&key=tu_key
```

Fíjate que en la URL el espacio en blanco se codifica como "%20" y "," sería "%2C".



## Autoevaluación

**El servicio Bing Maps devuelve la información en un formato u otro:**

- ☐ Dependiendo de la URL que se utilice en la petición.
- ☐ En función de un parámetro **GET** que se debe utilizar en la petición.

Revísate las opciones

Efectivamente con el parámetro "`o=xml`", pasado por **GET** indicamos que la salida la queremos en este formato.

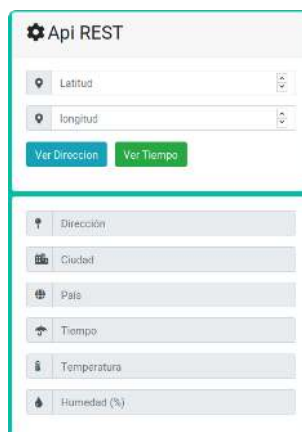
## Solución

1. Opción correcta
2. Incorrecto



## 4.3.- Aplicación web híbrida, tiempo y localización.

Vamos a crear una aplicación sencilla que utilice los dos servicios web que acabas de ver de **Bing Maps** y **Yahoo! Weather**. Se trata simplemente de ver cómo se pueden utilizar desde **PHP**. Al igual que en **Yahoo! Weather**, **Bing Maps** ofrece información de como recuperar los datos en **PHP**, esa información la tienes disponible en el enlace siguiente: [Bing Maps y PHP](#).



Captura de pantalla Firefox  
(Elaboración propia)

El formulario está dividido en dos zonas. En la superior, el usuario podrá introducir unas coordenadas (latitud y longitud) y en la inferior los datos de una dirección y el tiempo en la misma. Se trata de utilizar los dos botones del formulario para realizar consultas a ambos servicios, de tal forma que:

- ✓ Al utilizar el botón "**Ver dirección**", se realiza una consulta inversa al servicio "*Find a Location by Point*" de **Bing Maps** y se cubren los datos de la zona inferior del formulario relativos a Dirección Ciudad y País con la respuesta obtenida.
- ✓ Al pulsar sobre el botón "**Ver Tiempo**" se lanza una petición a **Yahoo! Weather** y se cubren los campos de Tiempo, Temperatura y Humedad.

Para realizar las llamadas mediante **Ajax** utilizarás la librería **xajax**, vista en la unidad anterior. En ambas peticiones hemos recuperado los datos en formato **JSON**, para eso hemos utilizado la función **PHP** : "`json_decode($resp, true)`" (el parámetro "**true**" es para convertir los datos **JSON** en un array asociativo)

La función que se encarga de realizar la llamada a **Yahoo! Weather** incluye el siguiente código:

```
function getTiempo($la, $lo)
{
    $resp = new xajaxResponse();
    if (!validar($la, $lo)) {
        $resp->setReturnValue(false);
        return $resp;
    }
    $datos = new Weather($la, $lo);
    $tiempo = $datos->getTiempo();
```

```

$temp = $tiempo['condition']['temperature'];
$shumedad = $tiempo['atmosphere']['humidity'];
$tiem = $tiempo['condition']['text'];
$resp->assign('tie', 'value', $tiem);
$resp->assign('tem', 'value', $temp . "°");
$resp->assign('hum', 'value', $shumedad . "%");
$resp->setReturnValue(true);
return $resp;
}

```

Y la que utiliza **Bing Maps** para obtener la dirección de unas coordenadas (también en formato JSON):

```

function getLocalizacion($la, $lo)
{
    $resp = new xajaxResponse();
    if (!validar($la, $lo)) {
        $resp->setReturnValue(false);
        return $resp;
    }
    $datos = new Weather($la, $lo);
    $ubicacion = $datos->getLocalizacion();
    $dir = $ubicacion['addressLine'];
    $ciudad = $ubicacion['locality'];
    $pais = $ubicacion['countryRegion'];
    $resp->assign('dir', 'value', $dir);
    $resp->assign('ciu', 'value', $ciudad);
    $resp->assign('pai', 'value', $pais);
    $resp->setReturnValue(true);
    return $resp;
}

```

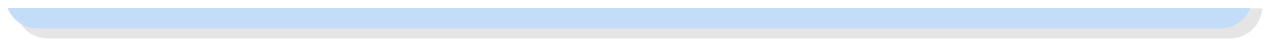
Ambas funciones se apoyan en la clase **Weather** (que tendrás que implementar) y sus métodos que son los que se encargan de recuperar en formato JSON los datos pedidos. Para esta clase te puedes ayudar en la documentación de como trabajar con PHP en **Bing Maps** y **Yahoo! Weather**.

Examina el código completo de la aplicación que se incluye en el siguiente fichero. Asegúrate de ajustar la ruta a la librería **xajax** y de configurar tus propias credenciales para que todo funcione:

 [Código de la aplicación.](#) (zip - 0,19 MB)

HTML5 incluye entre sus nuevas características una API de geolocalización, que permite a las aplicaciones web utilizar código JavaScript para obtener las coordenadas en que se encuentra el usuario. La aplicación del código anterior utiliza esa funcionalidad. Para conocer más detalles sobre el funcionamiento de la geolocalización en HTML5 y ver cómo se puede integrar con Google Maps, puedes consultar la siguiente página web.

[Geolocalización con HTML5.](#)



## 4.4.- Bing Maps. Calcular rutas.

Bing Maps Routes API es un servicio web de Microsoft, que al igual que el visto en apartados anteriores también forma parte de Bing Maps, y cuya principal utilidad es el cálculo de rutas para llegar desde una ubicación origen a otra ubicación destino. Las rutas pueden incluir además puntos intermedios (hitos), y tanto ellos como el origen o el destino pueden especificarse mediante direcciones reconocibles por el usuario o mediante coordenadas (latitud y longitud).



[Documentación sobre Bing Maps Routes API.](#)

[Mudassar Iqbal \(Pixabay License\)](#)

Los únicos parámetros que deben figurar obligatoriamente en una petición son:

- ✓ El punto origen (`wp.0`).
- ✓ El punto destino (`wp.1`).
- ✓ La `api key`

Entre los parámetros opcionales están:

- ✓ Se pueden indicar puntos intermedios de origen a destino (`wp.0`, `wp.1`, . . . , `wp.n`)
- ✓ La "cultura (idioma, unidades de medida....)" en que se devuelven los resultados (`c`).
- ✓ El formato de salida (JSON o AJAX) ("`o`").
- ✓ El modo de desplazarse (*walking*, *driving* o *transit*). Por defecto el valor es: *driving*.

Por ejemplo, una consulta de Madrid a Almería, en coche, formato de salida JSON, sistema métrico y castellano podría tener la siguiente forma:

```
http://dev.virtualearth.net/REST/V1/Routes/Driving?c=es&o=json&wp.0=madrid&wp.1=almeria&key=tuKey
```

La respuesta obtenida en formato JSON mostrará las indicaciones necesarias para llegar desde el origen al destino. El código necesario en PHP para utilizar el servicio será:

```
$url="http://dev.virtualearth.net/REST/V1/Routes/Driving?c=es&o=json&wp.0=madrid&wp.1=almeria&key=tu_k  
$json=json_decode(file_get_contents($url), true);  
var_dump($json);
```

Una vez decodificado el formato JSON obtenido, el acceso a la información se realiza utilizando los elementos del objeto `$respuesta`. Revisa la documentación del servicio para ver cómo se estructuran las respuestas, o accede a la url desde el navegador y obsérvalo. Por ejemplo, si del ejemplo anterior queremos la distancia, podríamos hacerlo así:

```
$distancia=$json['resourceSets'][0]['resources'][0]['travelDistance'];  
echo "$distancia klmts.";
```

Se utiliza la función "`file_get_contents()`" de **PHP** para almacenar en una variable la respuesta obtenida del servicio. Básicamente lo que haces es convertir en un **string** el contenido de un fichero o una **URL** en este caso. Para que funcione asegúrate de tener la directiva "`allow_url_fopen`" de "`php.ini`" activada

[Función `file\_get\_contents`.](#)

Cuando crees una aplicación que utilice servicios proporcionados por terceros, deberás tener en cuenta siempre sus **licencias de uso**.



## Autoevaluación

En una petición al servicio Bing Maps Routes , deben figurar obligatoriamente los parámetros:

- ☐ origen, destino y api key.
- ☐ origen, destino, formato de salida.

Efectivamente, si no se indica el modo de desplazarse se supone el coche (driving) por defecto.

Incorrecto. el formato de salida no es necesario se supone **JSON** por defecto.

## Solución

1. Opción correcta
2. Incorrecto

## 4.5.- Google Tasks.

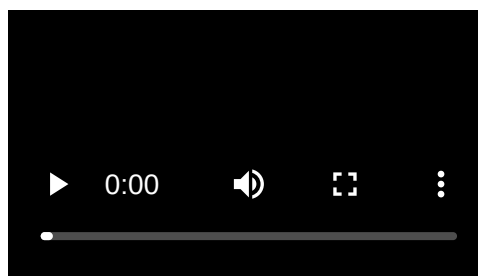
---

Los servicios que has utilizado hasta ahora recibían peticiones por parte del usuario (en nuestro caso una aplicación web) y generaban, y devolvían, respuestas a las mismas en formatos JSON o AJAX. El servicio **Google Tasks** necesita, además, una autorización para devolver información privada de un usuario. Para obtener esa autorización, nuestra aplicación deberá usar el protocolo **OAuth2**.



El servicio de **Google Tasks** nos permite gestionar las tareas personales del usuario. El enlace siguiente es una guía rápida para empezar a usarlo: [Inicio rápido de Google Tasks con PHP](#).

[Mudassar Iqbal \(Pixabay License\)](#)



Google, Firefox y VokoScreenNg (Elaboración propia uso educativo no comercial.)

### [Descripción Textual del Vídeo.](#)

Una vez creada la autorización podemos descargarla en un archivo "**credentials.json**" que será el que utilizemos.

Básicamente existen dos tipos de recursos:

- ✓ **Listas de tareas.** Una de ellas es la lista de tareas por defecto; existe siempre y no se puede eliminar.
- ✓ **Tareas.** Es cada uno de los elementos que contiene una lista de tareas. Puede contener información como el título de la tarea, notas, o fechas.

Hay dos formas de utilizar el servicio:

- ✓ Mediante llamadas **REST** directamente, al igual que hicimos cuando utilizamos los servicios anteriores.
- ✓ Mediante una librería cliente, disponible para múltiples lenguajes (entre ellos **PHP**). Será la que veremos.

Para poder usar **Google Task** deberemos instalarnos las librerías del cliente de **Google**, lo podemos hacer con **Composer** como ya sabemos, una vez iniciado **Composer** en nuestro proyecto solo tenemos que teclear en consola desde la carpeta del proyecto:

```
composer require google/apiclient:^2.0
```

Puedes encontrar documentación y gran cantidad de ejemplos del uso de estas librerías en el enlace siguiente: [Documentación y ejemplos Api PHP Google](#).

Para poder utilizar el servicio **Google Tasks** tendrás que añadir las siguientes líneas en tu código **PHP**:

```
session_start();
require 'vendor/autoload.php';
$redirect_uri = 'http://' . $_SERVER['HTTP_HOST'] . $_SERVER['PHP_SELF'];
```

Asegúrate de ajustar correctamente la ruta a la librería. Aquí se está haciendo uso del "autoload" de **Composer**

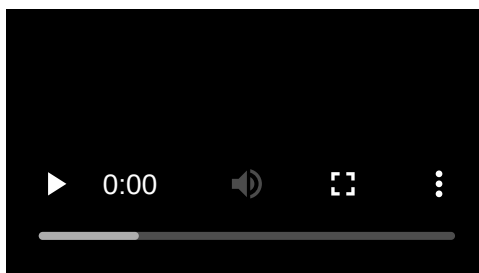
Será necesario crear dos objetos; uno de tipo **cliente**, que utilizará OAuth2 para gestionar las autorizaciones de acceso a los servicios:

```
1 $client = new Google_Client();
2 $client->setApplicationName('Google Tasks API PHP');
3 $client->setAuthConfig('credentials.json');
4 $client->setRedirectUri($redirect_uri);
5 $client->setScopes(Google_Service_Tasks::TASKS); //TASKS_READONLY
6 $client->setPrompt('select_account consent');
```

Es importante señalar que la **URI** de redirección debe ser válida y estar dada de alta como tal en la configuración de la aplicación web, tal y como se mostró en el videotutorial anterior (por ejemplo, puede ser la misma dirección de la página que estamos programando).

En la línea **5** estamos dando acceso para, no solo ver las tareas si no para crearlas borrarlas, modificarlas. Si queremos solo acceso para ver las tareas podemos cambiar **TASKS** por **TASKS\_READONLY**.

Una vez creado el cliente si es la primera vez que entramos nos redirigirá a la página donde el usuario Google al que deseamos gestionar las tareas se le pedirá los permisos pertinentes. Hasta que este usuario marque la aplicación como de confianza nos aparecerá una advertencia de seguridad diciéndonos que la aplicación no es segura. Fíjate en el vídeo siguiente:



Google, Firefox, VokoScreenNG (Elaboración propia uso educativo no comercial)

[Descripción textual del vídeo.](#)

## 4.5.1- Google Tasks (I).

El primer paso para acceder al servicio es autenticarse vimos en un vídeo anterior que si es la primera vez que entramos o el **token** ha caducado nos mandará a la página de **Google**, donde deberemos validarnos con un usuario y dar permiso a la aplicación a acceder a nuestras tareas (esto se hace en las líneas **19** y **20** del código que tienes abajo). La clave de acceso obtenida se debe almacenar utilizando la función **setAccessToken**. Puede almacenarse en una variable de sesión para futuras llamadas al servicio.



[Auisis](#) (Dominio público)

```
1  if (isset($_GET['code'])) {
2      $token = $client->fetchAccessTokenWithAuthCode($_GET['code']);
3      $client->setAccessToken($token);
4
5      // Guardamos el token en una variable de sesion
6      $_SESSION['token'] = $token;
7
8      // redirigimos a esta misma página
9      header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));
10 }
11
12 // set the access token as part of the client
13 if (!empty($_SESSION['token'])) {
14     $client->setAccessToken($_SESSION['token']);
15     if ($client->isAccessTokenExpired()) {
16         unset($_SESSION['token']);
17     }
18 } else {
19     $authUrl = $client->createAuthUrl();
20     header("Location:$authUrl"); //nos manda a la página de google
21 }
```

Una vez autenticado el cliente, puedes emplear el objeto de la clase **Google\_Service\_Tasks** para gestionar las listas de tareas y las tareas del usuario.

- ✓ Para Listar las 10 primeras Listas de Tareas (**id** y nombre (**title**)): (en principio solo hay una):

```
$service = new Google_Service_Tasks($client);
$optParams = ['maxResults' => 10];
$results = $service->tasklists->listTasklists($optParams);
if (count($results->getItems()) == 0) {
    echo "Ninguna Lista de tareas encontrada";
} else {
    echo "Listas de Tareas:<br>";
    foreach ($results->getItems() as $tasklist) {
        printf("%s (%s)\n", $tasklist->getTitle(), $tasklist->getId());
    }
}
```



```
}  
}
```

- ✓ Para Listar las tareas de una lista de tarea (hay que pasar el id de la lista de tareas):

```
$res1 = $service->tasks->listTasks("YjNUcFR3NDk4a01SanZNTQ"); //cambia el id por el de la lista  
echo "Tareas:<br>";  
foreach ($res1->getItems() as $tasklist) {  
    printf("%s (%s)\n", $tasklist->getTitle(), $tasklist->getId());  
}
```

- ✓ Para crear una lista de tareas nueva:

```
$opciones=["title"=>"Lista de Tareas 3"];  
$taskList = new Google_Service_Tasks_TaskList($opciones);  
$service->tasklists->insert($taskList);
```

- ✓ Para crear una tarea nueva en una lista de tareas:

```
$op=["title"=>"Entregar Trabajo DWES", "notes"=>"Formato pdf", "due"=>"2020-08-10T10:57:00.000-  
$tarea=new Google_Service_Tasks_Task($op);  
$service->tasks->insert('YjNUcFR3NDk4a01SanZNTQ', $tarea); //pon el id de tu lista de tareas
```

- ✓ Para borrar una tarea de una lista de tareas:

```
$service->tasks->delete("YjNUcFR3NDk4a01SanZNTQ", "bU9hVFBdWlFPdFhnbVJGaQ"); //id_tasklist, idt
```

La documentación sobre los parámetros y los métodos la puedes encontrar en el enlace siguiente: [Documentación](#).



## Autoevaluación

Para utilizar el servicio Google Tasks desde PHP, puedes emplear:

- ☐ La API que ofrece Google.
- ☐ La API que ofrece Google o llamadas REST.

Incorrecta. En la documentación de **Google** figuran los posibles métodos que se pueden utilizar, y también se ha comentado con anterioridad.

Efectivamente. Puedes utilizar uno u otro método, pero la **API** que ofrece **Google** es el más sencillo de ambos.

## Solución

1. Incorrecto
2. Opción correcta

## 4.6.- Aplicación web híbrida de gestión de repartos.

Vamos a ver cómo puedes crear una aplicación web híbrida que utilice los servicios que acabas de ver, vamos a hacer una pequeña aplicación para gestionar los pedidos de una tienda. El supuesto del que se parte es el siguiente:

Como la zona de influencia de la tienda aún es pequeña, solo una localidad (en el ejemplo se trabaja con Almería, tu puedes hacerlo con la Localidad que quieras) y pensando también en mantener la relación con los clientes, se ha decidido hacer reparto directo de los productos que se compran en la tienda online. Para ello se ha pensado en crear una aplicación web híbrida con las siguientes características:

- ✓ Se utilizará la API del servicio de tareas de **Google (Google Tasks)** para almacenar como listas de tareas la información de los repartos. De esta forma la información podrá ser consultada desde cualquier lugar utilizando un dispositivo con conexión a Internet. Cada lista de tareas se corresponde en la aplicación con una lista de reparto, y cada una de sus tareas con un envío. Para diferenciar una lista de otra, se le pone como parte del título la fecha del día en que se hará el reparto.
- ✓ Para cada producto que se reparte se creará una tarea en la lista correspondiente. Esa tarea almacenará la dirección de envío y sus coordenadas. Para obtenerlas, se utilizará el servicio **Bing Maps de Microsoft**.
- ✓ Para optimizar la ruta que se ha de recorrer, se utilizará **Routes de Bing Maps**. La idea es mostrar el orden de los productos que se van a repartir cada día de forma que se minimice la distancia recorrida.
- ✓ La función para que nos muestre en un mapa la dirección de envío se pedirá en la práctica de la unidad. No hace falta implementarla

La apariencia de la aplicación será:




Firefox (Elaboración propia.)

Cuando se cree una nueva tarea (un nuevo envío), se pedirá la dirección y se mostrará una pantalla como la siguiente para que el usuario complete los datos necesarios. Los campos longitud y latitud son de solo lectura, se rellenarán cuando le demos al botón "Ver Coordenadas".

 Crear Envio

 Dirección

Ver Coordenadas

 Latitud

 longitud

 Producto

Nuevo Envio

Volver

Captura de Firefox (Elaboración propia.)

Se utilizara **Bing Maps** para obtener las coordenadas.

## 4.6.1.- Aplicación web híbrida de gestión de repartos (I).

Para elaborar una tarea nueva en una lista de tareas, lo que haremos será procesar el formulario "**Crear Envío**" que vimos en la página anterior, el **action** de dicho formulario será la página principal de la aplicación de la aplicación. En la llamada al formulario pasamos por **get** el **ID** de la Lista de Tareas donde vamos a crear la tarea y lo volvemos a mandar de vuelta junto con la latitud, longitud y el producto. El **título** de la tarea será el nombre del producto y la dirección y el campo "**notes**" de la tarea las coordenadas.

```
<?php
if (isset($_POST['lat'])) {
    $note = $_POST['lat'] . "," . $_POST['lon'];
    $title = ucwords($_POST['pro']) . " " . ucwords($_POST['dir']) . ", Almería. ";
    $idLt = $_POST['idLTarea'];
    unset($_SESSION[$idLt]);
    //guardamos la tarea
    $op = ['title' => $title, 'notes' => $note];
    $tarea = new Google_Service_Tasks_Task($op);
    try {
        $res = $service->tasks->insert($idLt, $tarea);
    } catch (Google_Exception $ex) {
        die("Error al guardar la tarea: " . $ex);
    }
    unset($_POST['lat']);
}
```

Para el resto de operaciones: crear y borrar listas de tareas, borrar una tarea y ordenar los envíos puedes des utilizar parámetros **GET** y recargar la misma página. Si está presente un parámetro '**action**', se realizará un procesamiento determinado. Por ejemplo para crear una nueva Lista de Tareas:

```
if (isset($_GET['action'])) {
    switch ($_GET['action']) {
        case 'nlt':
            $opciones = ["title" => $_GET['title']];
            $taskList = new Google_Service_Tasks_TaskList($opciones);
            try {
                $service->tasklists->insert($taskList);
            } catch (Google_Exception $ex) {
                die("Error al crear una lista de tareas: " . $ex);
            }
            break;
    }
}
```

Para obtener las coordenadas de una dirección concreta, una solución es utilizar **Xajax** para llamar a los servicios **REST** de **Bing Maps**, de forma similar a como ya hiciste en la aplicación web anterior. Nos creamos la clase **Coordenadas** con unos métodos de apoyo que utilizaremos después.

```

class Coordenadas
{
    public static $iniciourl = "http://dev.virtualearth.net/REST/v1/Locations/ES/Almeria/";
    public static $finurl = "?include=ciso2&maxResults=1&c=es&key=tu_Api_Key";
    public $coordenadas;
    public $url;

    public function __construct()
    {
        $num = func_num_args();
        if ($num == 1) {
            $dir = str_replace(" ", "%20", func_get_arg(0));
            $this->url = self::$iniciourl . "$dir" . self::$finurl;
        }
    }

    public function getCoordenadas()
    {
        $salida = file_get_contents($this->url);
        $salida1 = json_decode($salida, true);
        return $salida1["resourceSets"][0]["resources"][0]["point"]["coordinates"];
    }

    public function ordenarEnvios($dato)
    {
        . . .
    }
}

```

Haremos uso del método "getCoordenadas()" en el método PHP que llamaremos desde xajax

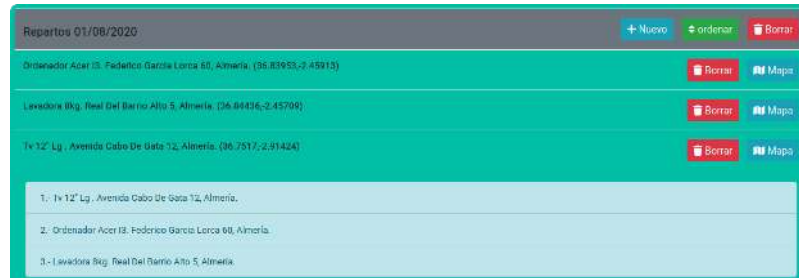
```

function getCoordenadas($dir){
    $resp=new xajaxResponse();
    $dir=trim($dir);
    if(strlen($dir)<4){
        $resp->setReturnValue(false);
        return $resp;
    }
    $c=new Coordenadas($dir); //hacemos uso del método getCoordenadas de la clase Coordenadas
    $lat=$c->getCoordenadas()[0];
    $lon=$c->getCoordenadas()[1];
    $resp->assign('lat', 'value', $lat);
    $resp->assign('lon', 'value', $lon);
    $resp->setReturnValue(true);
    return $resp;
}

```

Básicamente se obtienen las coordenadas y se ponen en los campos correspondientes del formulario.

## 4.6.2.- Aplicación web híbrida de gestión de repartos (II).



Captura pantalla Firefox (Elaboración propia)

Un caso especial es el método a utilizar para optimizar las distintas entregas de un pedido. Una solución es realizar el proceso en dos pasos. En primer lugar puedes utilizar **ajax** para llamar al servicio **Bing Maps** y obtener el orden óptimo de entrega. El proceso lo realizamos en dos partes como las coordenadas. En la clase Coordenadas que vimos en el apartado anterior tenemos el método "**ordenarEnvios()**". Fíjate que para calcular una ruta necesitamos un principio y un fin, se han puesto unas coordenadas ficticias de la Localidad en las que estamos trabajando que será el principio y el fin de la ruta. Ponemos que la ruta se optimiza en función a la distancia "**optimize=distance**"

```
public function ordenarEnvios($dato){
    //Ponemos las coordenadas del almacen por ejemplo '36.86071,-2440779' como inicio y fin de la
    $base = "http://dev.virtualearth.net/REST/v1/Routes/driving?c=es&wayPoint.0=36.86071,-2.440779";
    $puntos = explode("|", $dato);
    $num = 1;
    $trozo = "";
    for ($i = 0; $i < count($puntos); $i++) {
        $trozo .= "wayPoint." . $num++ . "=" . $puntos[$i] . "&";
    }
    $trozo .= "wayPoint." . $num . "=36.86071,-2.440779&optimize=distance&optWp=true&routeAttribut
    $url = $base . $trozo;
    $salida = file_get_contents($url);
    $salida1 = json_decode($salida, true);
    $wayp = $salida1["resourceSets"][0]["resources"][0]["waypointsOrder"];
    //quitamos el primero y el ultimo (inicio y fin) (El almacen)
    array_shift($wayp);
    array_pop($wayp);

    for ($i = 0; $i < count($wayp); $i++) {
        $resp[] = substr(strstr($wayp[$i], '.'), 1);
    }
    return $resp; //array con la ruta ya optimizada
}
```

Puede encontrar información sobre el servicio **Routes de Bing Maps** en el enlace siguiente: [Información Routes](#).

Al pulsar en el botón 'ordenar' de una lista, se ejecuta la siguiente función **JavaScript** en ella cojemyos todas las coordenadas previamente para cada tarea las hemos almacenado en un campo oculto "`<input type='hidden' value='{$starea->getNotes()}'>`", acuérdate que el campo "notes" almacenaba las coordenadas

```
function ordenarEnvios(id) {
    var puntos = $("#"+ id + " input:hidden").map(function () {
        return this.value;
    }).get().join("|");
    var respuesta = xajax.request({ xjxfun: "ordenarEnvios" }, { mode: 'synchronous', parameters: [puntos] });
    if (respuesta == false) {
        alert("No se pudo ordenar el envio");
        return respuesta;
    }
    // Si obtuvimos una respuesta, reordenamos los envíos del reparto
    // Cogemos la URL base del documento
    var url = "http://127.0.0.1/curso/tema8/repartos/public/repartos.php";
    // Añadimos el código de la lista de reparto
    url += '?action=oEnvios&idLt=' + id;
    // Y un array con las nuevas posiciones que deben ocupar los envíos
    for (var r in respuesta) url += '&pos[' + respuesta[r];
    window.location = url;
}
```

Esta función ejecuta mediante **AJAX** el siguiente código **PHP**, que obtiene el orden óptimo de reparto:

```
function ordenarEnvios($puntos){
    $resp=new xajaxResponse();
    if(strlen(trim($puntos))==0){
        $resp->setReturnValue(false);
        return $resp;
    }
    $c=new Coordinadas();
    $datos=$c->ordenarEnvios($puntos); //hacemos uso del método ordenarEnvios de la clase Coordinadas
    $resp->setReturnValue($datos);
    return $resp;
}
```

El array obtenido se envía en parámetros **GET** a la misma página, que lo debe procesar para ordenar las tareas según indica.



## Autoevaluación



Para obtener una ruta optimizada utilizando el servicio de Bing Maps, solo es necesario poner los puntos de la ruta.

- ☐ Cierto, solo es necesario el inicio y fin de la ruta
- ☐ Hace falta el parámetro opcional: `optimize (optmz)`

Incorrecta. El parámetro `optimize (optmz)` nos permite optimizar la ruta en función de distintos parámetros como la distancia, (`optmz=distance`), el tiempo (`optmz:time`) . . .

Efectivamente. El parámetro `optimize (optmz)` nos permite optimizar la ruta en función de distintos parámetros como la distancia, (`optmz=distance`), el tiempo (`optmz:time`) . . .

## Solución

1. Incorrecto
2. Opción correcta

## 4.6.3.- Aplicación web híbrida de gestión de repartos (III).

Una vez obtenido el array que indica el orden óptimo de los puntos intermedios, se vuelve a cargar la página y en ese momento se deberán procesar los parámetros `GET` recibidos que indican cómo reordenar las tareas de la lista.

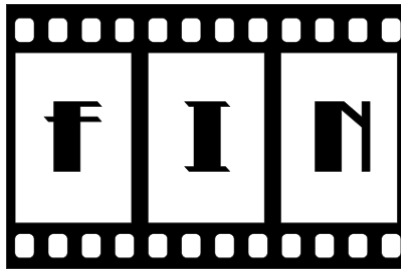
```
case 'oEnvios':
    $apos = $_GET['pos'];
    $id_lista = $_GET['idLt'];
    unset($_SESSION['idLt']);
    //Obtenemos todas las tareas de esta lista de tareas
    $tareas = getTareas($id_lista);
    foreach ($apos as $k => $v) {
        //los envios me los manda ordenados del 1 al n
        //en php los array empiezan por cero, por eso restamos 1
        //asi el envio 1 pasa a ser el 0, el 2 el 1 ...
        $p = $v - 1;
        $array0[$k] = $tareas->getItems()[$p]->getTitle();
    }
    $_SESSION[$id_lista] = $array0;
```

En nuestra página hacemos que nos aparezcan el orden de reparto si le hemos dado a ordenar, de la forma siguiente. Cada vez que borre o cree un nuevo envío se hará un `unset`, de la variable de sesión creada, para ocultar el orden calculado pues deberíamos calcularlo otra vez en función a las tareas resultantes.

```
if (isset($_SESSION[$lista->getId()])) {
    echo "<div class='container mt-2 mb-2' style='font-size:0.8rem'>";
    echo "<ul class='list-group'>";
    foreach ($_SESSION[$lista->getId()] as $k => $v) {
        echo "<li class='list-group-item list-group-item-info'>" . ($k + 1) . ". - " . $v .
    }
    echo "</div>";
}
```

Intenta completar por ti mismo la programación de la aplicación web. Puedes descargar y revisar el código de la solución completa en el siguiente enlace. Recuerda que debes ajustar el código para indicar las rutas correctas a las librerías de **Xajax** y de **Google**, y poner en el mismo tus propias claves de uso de los servicios web de **Google** y **Bing Maps**. La ruta de redirección, que debe estar registrada. Por razones obvias se han omitido el archivo de credenciales de **Google Task** y la key de **Bing Map**. Se ha dejado el archivo "`composer.json`", acuérdate de hacer "`composer install`" para que se cree la carpeta "`vendor`" con las librerías necesarias (por ejemplo las de **Google**).

 [Aplicación web.](#) (zip - 0,19 MB)



[studio hades](#) (Dominio público)