

UT9 DESARROLLO DE APLICACIONES WEB HÍBRIDAS

Índice

- [UT9 DESARROLLO DE APLICACIONES WEB HÍBRIDAS](#)
 - [Índice](#)
 - [Reutilización de código e información](#)
 - [Arquitectura de una aplicación web híbrida](#)
 - [Comunicación en la arquitectura mashup](#)
 - [Características](#)
 - [Utilización de repositorios de información](#)

Reutilización de código e información

Los servicios web permiten a tus aplicaciones comunicarse con otras utilizando la web (el protocolo HTTP) como medio de transmisión.

En la UT 7 hemos visto servicios SOAP y los servicios REST. La mayoría de las APIs existentes en el mercado utilizan estos últimos.

Un servicio web implementado mediante REST puede:

- Utilizar una estructura de URIs para acceder a los recursos gestionables mediante el servicio web
 - `http://miweb.com/productos`
 - `http://miweb.com/productos?id=1`
- Usar los distintos métodos HTTP (POST, DELETE,...)
- Utilizar XML o JSON en sus comunicaciones (o incluso ambos)
- En esta unidad habrá que crear aplicaciones utilizando diversos servicios web

¿Y cómo consumíamos estos servicios web REST? Para obtener los datos de un servicio web REST vamos a utilizar la librería **cURL**.

Es una biblioteca que permite conectarse y comunicarse con diferentes tipos de servidores y diferentes tipos de protocolos.

Una aplicación web híbrida o mashup implica que debe acceder a datos o procesar información a través del sitio que nos ofrece el proveedor, ya que este servicio de nuestro proveedor nos va a dar un valor añadido a nuestra aplicación, y sin esta ayuda sería técnicamente imposible ofrecer estos servicios.

Un ejemplo ya visto era el siguiente:

```
$url_servicio = "http://zoologico.laravel/rest";  
$curl = curl_init($url_servicio);  
//establecemos el verbo http que queremos utilizar para la petición  
curl_setopt($curl, CURLOPT_CUSTOMREQUEST, "GET");  
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
```

```
$respuesta_curl = curl_exec($curl);
curl_close($curl);

$respuesta_decodificada = json_decode($respuesta_curl);
```

Arquitectura de una aplicación web híbrida

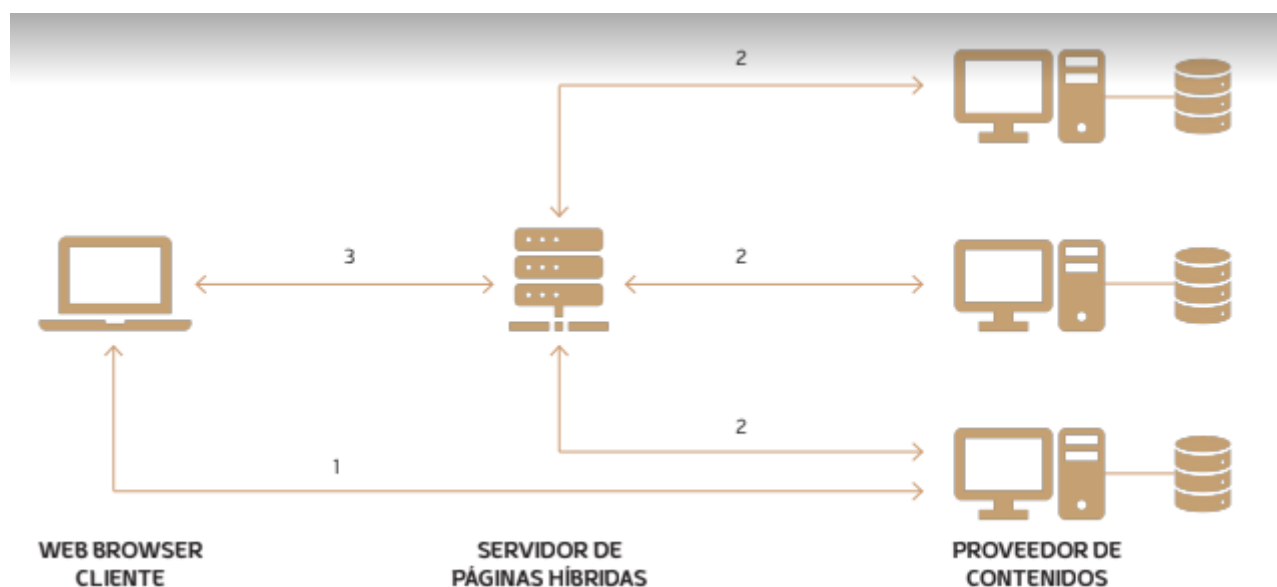
La arquitectura de una aplicación web híbrida está compuesta por tres partes principales. Por un lado, lo más importante es el **proveedor de contenidos** que ofrece servicios como mapas, geolocalizaciones, API para conectar con los servicios de AdWords o transcripciones de audio a texto, como es el gran y todo poderoso Google. En el centro estaría situado **el servidor** de la nueva página híbrida y por último, **el cliente**, que es quien accede a la página.

- **Proveedor de contenidos** Es la fuente de datos. Los datos suelen estar disponibles a través de una interfaz pública o utilizando una API con diferentes protocolos como RSS o ATOM o servicios Web.
- **Sitio Mashup:** es la aplicación nueva que provee de un nuevo servicio utilizando diferentes fuentes de información de las que no es dueño. El sitio mashup hará uso de las tecnologías de las que disponga el proveedor de contenidos para conseguir generar la aplicación híbrida
- **Cliente Web:** es la interfaz del usuario del mashup. Es una aplicación web. Aquí el contenido también puede ser mezclado en el mismo navegador del cliente. Se pueden utilizar lenguajes como JavaScript y AJAX.

Comunicación en la arquitectura mashup

Vamos a ver el esquema de las diferentes vías de comunicación entre el navegador web del cliente y el proveedor de contenidos, pasando o no por el servidor de aplicaciones o páginas híbridas.

- 1 - el navegador cliente accede directamente a la información del proveedor de contenidos mediante JavaScript
- 2 - el servidor de aplicaciones obtiene los datos directamente del proveedor de contenidos; estos se mezclan en el servidor y son enviados al navegador del cliente.
- 3 - el navegador del cliente accede al servidor web y extrae el contenido almacenado



Características

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 XML. En ocasiones el proveedor del servicio ofrece también un interface de programación (para facilitar el acceso a los datos Es el caso de las API de compañías como Google, Yahoo!, Flickr Microsoft o Amazon)
- Información generada y gestionada por el **propietario de la aplicación web híbrida**, como pueden ser datos internos de una empresa
- A veces los datos se ofrecen de forma pública utilizando protocolos de redifusión web (también conocido como **sindicación** web) como **RSS** o **Atom** y puede ser necesario procesarlos para extraer la información necesaria.
- **Web scraping**: técnica utilizada para extraer datos de sitios web que no ofrezcan ningún servicio web. Está en desuso, debido a los siguientes inconvenientes que presenta:
 - No todas las webs autorizan su uso
 - Complejidad
 - Sobrecarga de los servidores
 - Propensión a fallos inesperados

Utilización de repositorios de información

Para utilizar servicios de terceros, hay condiciones y limites. Por ejemplo, a partir de un cierto número de peticiones, ya nos empezarían a facturar: un ejemplo podría ser el servicio de Google de geolocalización, que tiene un límite de 1500 peticiones.

La mayoría de los proveedores de servicios web utilizan un protocolo llamado OAuth. Es un protocolo estándar de autorización.

OAuth2, permite a una aplicación externa obtener acceso a información de carácter privado de un tercero a través de un servicio Web. Para ello establece un acuerdo de acceso a ella 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 dwes, Google pedirá a dwes permiso indicándole qué aplicación es la que solicita el acceso y a qué información. Si el usuario dwes otorga permiso, la aplicación 'X' podrá acceder a los datos que solicitó a través del servicio de Google.

Veamos, por ejemplo, qué sucede cuando nuestra aplicación necesita acceder a la 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 la que quiere acceder y el tipo de acceso a esta.
- El servidor de autorización requiere al usuario de la aplicación web que inicie sesión con su cuenta Google (si aún no lo ha hecho), y lo 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.
- Los códigos de acceso tienen un tiempo de vida limitado. Cuando caducan, la aplicación ha de comunicarse de nuevo con el servidor de autorización para obtener un código de refresco.

Además, la mayoría de la información que nos suministran los proveedores de servicios web viene en formato XML y JSON.

Un ejemplo de aplicación web híbrida, es una aplicación que utilice la API de Google Maps y muestre información de ubicación geográfica de las franquicias de una empresa para mostrar la localización de las tiendas en un mapa.



Otro ejemplo podría ser una aplicación que lea distintos RSS de periódicos (Washington Post, Reuters, CNN, BBC, Yahoo News...) y se conecte a Calais para obtener de cada uno de los ítems ciudades y países que se citen en ellos.

Una vez obtenidas las ciudades y países, se utilizaría la API de Google Maps para posicionar cada una de las ciudades o países de los que se hayan encontrado noticias. En caso de que se pinche en cada marca del mapa se cargarían las noticias relacionadas con esta ciudad y utilizando Flickr se obtendrían 2 fotografías de este lugar.

Podeís encontrar información sobre apis en el siguiente artículo de [NORDIC APIS](#)

Obtener una api key de google maps en el siguiente artículo de [Maplink](#)

Información [Google Maps Platform](#)

Documentación sobre [Amazon API Gateway](#)

Documentación de la [API de Flickr](#)


Documentación de la [API de YouTube](#)


Ejemplo:


Utilización de Google Api para obtener la dirección a partir de las coordenadas de latitud y longitud
 mirar [artículo de firefox](#)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <p> Pincha el botón para conseguir tus coordenadas </p>
    <button onclick="getlocation()">Inténtalo</button>
    <p id="demo"></p>
    <script>
      var x=document.getElementById("demo");
      function getlocation(){
        if(navigator.geolocation){
          navigator.geolocation.getCurrentPosition(showPosition);
        }
        else {
          x.innerHTML="Geolocation no está soportada por este
navegador.";
        }
      }
      function showPosition(position){
        x.innerHTML="Latitud" + position.coords.latitude +
          "<br />Longitud" +position.coords.longitude;
      }
    </script>
  </body>
</html>
```

propuesta realizar una función que a partir de la latitud y longitud, nos dé los siguientes datos: población, código postal, calle, número y comunidad.

 Hoja09_Mashup_01

 Hoja09_Mashup_02

 Hoja09_Mashup_03