# PROGRAMACIÓN HIPERMEDIA II. PRÁCTICA 2.

## Objetivos de la práctica

- Aprender a sacar el máximo partido al lenguaje de programación JavaScript en el lado del cliente, para aplicar dinamismo e interacción con el usuario a un sitio o aplicación web.
- Aprender a hacer uso de la tecnología AJAX para realizar peticiones al servidor evitando, de esta manera, la recarga de páginas
- Aprender a trabajar de forma autónoma con JavaScript sin necesidad de utilizar ningún framework de terceros. Por ello **en esta práctica no se permite el uso de ningún framework JavaScript de terceros**, salvo que se indique lo contrario.

## Enunciado de la práctica

Partiendo del sitio web creado en la práctica 1 de la asignatura, se utilizará JavaScript para incorporar dinamismo e interacción con el usuario.

Para poder realizar esta segunda práctica es necesario utilizar el servidor XAMPP. Mediante phpMyAdmin se debe crear una base de datos llamada ***ph2*** y dar permisos al usuario ***ph2*** con contraseña ***ph2*** para que pueda acceder y manipular las tablas de dicha base de datos. Se proporciona un script sql para importar en phpmyadmin, que se encargará de crear la base de datos y todo lo necesario.

Además, también se facilita una serie de ficheros php, organizados dentro de una carpeta llamada *rest*, que proporcionan un servicio *rest* mediante el que acceder a la base de datos. Esta carpeta deberá copiarse dentro de la carpeta en la que se encuentra el resto del código de la práctica en el servidor XAMPP. Este servicio rest será el destino de las peticiones ajax de la práctica. Al final de este enunciado se indican las peticiones que se pueden realizar junto a sus parámetros.

**Notas**:

- Será necesario modificar la línea 5 del fichero *rest/.htaccess* para indicar el path completo de la carpeta *rest* dentro de la carpeta *htdocs*.
- Si el servidor de *mysql* no está configurado en el puerto por defecto (3306), será necesario modificar la línea 10 del archivo *rest/configbd.php* y añadirle el puerto. Por ejemplo, si el puerto en el que está instalado el servidor mysql es el 3307, la línea sería:

  $_server = "127.0.0.1**:3307**";

- Se necesitará hacer uso del atributo **sessionStorage** de la interfaz *Storage*, perteneciente al API *Web Storage* de HTML5, para llevar a cabo el control de sesiones en el navegador. El API *Web Storage* de HTML5 será convenientemente explicado en la clase de presentación de esta práctica.
- Los paths a los que se suben las fotos de usuario y las fotos de los viajes se

indican en las variables $uploaddir y $uploaddir_fotos_usu que se encuentran en el fichero **/rest/funciones.php**. Para los valores por defecto se supone que hay una carpeta fotos en la misma carpeta en la que se encuentra la carpeta rest.

## Trabajo a realizar

1) (0,5 puntos) Todas las páginas deben pasar la validación satisfactoriamente en http://validator.w3.org. El html a validar incluirá el contenido generado con JavaScript.

2) Para todas las páginas:
   a) (0,5 puntos) Se debe comprobar si el usuario está logueado (consultando *sessionStorage*) y hacer los cambios pertinentes en el menú de navegación, a saber:
      i) Cuando el usuario no está logueado deben aparecer los enlaces para ir a Inicio; para hacer login; para buscar viajes; y para darse de alta como nuevo usuario.
      ii) Cuando el usuario está logueado, los enlaces para login y para alta de usuario deben desaparecer. Además, estando logueado aparecerán los enlaces a la página *crear_viaje.html*, al perfil de usuario (realmente será la misma página *registro.html* pero preparada como perfil) y un enlace que permita al usuario cerrar la sesión (*limpiando* la información de sessionStorage), tras lo que será redirigido a *index.html*.
   b) (0,25 puntos) Comprobación de acceso en las páginas para las que se indique.
3) Página ***index.html***:
   a) Mediante peticiones Ajax:
      i) (0,5 puntos) Pedir y mostrar los *últimos 6 viajes* subidos al servidor, ordenados por fecha descendente. Al pinchar en cada uno de ellos para ir a *viaje.html*, el id del mismo se añadirá a la url del enlace para poder recuperarlo en la página de destino.
      ii) (0,5 puntos) Pedir y mostrar los 10 últimos comentarios escritos por los usuarios, ordenados de más a menos reciente. Debe haber un botón para actualizar la lista de comentarios, realizando de nuevo la petición ajax al servidor.

4) Página ***crear_viaje.html***.
   a) Si se intenta acceder sin estar logueado se debe redirigir a *index.html*.
   b) Añadir el código necesario javascript para lo siguiente:
      i) (0,75 puntos) Al pinchar el botón *Añadir foto*, se añadirá en la zona de fotos una ficha en blanco para subir una foto. Recordad que en cada ficha aparecerá: un elemento <img>, que mostrará la foto en pequeño; un elemento <textarea>, para escribir una descripción; un campo (o campos) para recoger la fecha en la que se tomó la foto; un botón que abrirá el típico diálogo para poder seleccionar el fichero de la foto; y un botón para poder eliminar la ficha del formulario. Cuando la ficha está vacía, debe

aparecer la típica imagen que indica que no hay foto seleccionada. Al pinchar en el elemento <img> también se mostrará el diálogo para poder seleccionar el fichero de imagen.

Tengo que terminar el metodo abrirFileDialog()

ii) (0,5 puntos) Al seleccionar un fichero la foto se mostrará en el elemento <img> de la correspondiente ficha. Se deberá comprobar que su tamaño no exceda los 2MB. Si su tamaño es mayor de 2MB, se mostrará un mensaje sobre la imagen de la ficha indicando el error. Además, el borde de la ficha se pondrá en rojo.

c) (0,75 puntos) Al pinchar en el botón para enviar los datos del formulario:
   i) No se podrá crear un nuevo viaje mientras haya fotos *no correctas*.
   ii) Para crear un nuevo viaje primero se envían los datos del viaje y si todo ha ido bien (respuesta correcta del servidor), entonces se enviarán las fotos una a una. Todo ello se hace mediante llamadas ajax. El servidor *rest* habrá creado, dentro de la carpeta fotos, una carpeta con el id del viaje. A esta carpeta es a la que se subirán las fotos del viaje. Cada foto se guardará con su id como nombre y la extensión del fichero subido.
   iii) Al crear un nuevo viaje y confirmar que todo ha finalizado correctamente, se debe mostrar el típico mensaje informativo sobre una *capa semitransparente*, superpuesta sobre toda la página, que impida interactuar con el formulario. El mensaje deberá tener un botón para que el usuario pueda cerrarlo, tras lo que será redirigido a *index.html*.

**Nota**: Para poder crear el viaje, junto a los campos del formulario se debe enviar la clave obtenida al loguearse y el login del usuario.

5) Página ***viaje.html***:
   a) Al cargar la página se debe obtener de la url el id del viaje a mostrar, así como también el id del comentario, si fuera el caso.
      i) Si en la url no se encuentra el id del viaje (porque se intenta acceder directamente), se debe redirigir a *index.html*.
      ii) (0,5 puntos) Se utilizará el id del viaje para realizar una petición ajax al servidor y mostrar toda su información. Además, se hará otra petición ajax para mostrar los comentarios de los usuarios sobre el viaje.
      iii) (0,5 puntos) Carga condicional de contenido utilizando JavaScript y AJAX. El formulario para dejar un comentario no estará disponible si el usuario no está logueado. Sólo estará en el HTML del documento si el usuario está logueado. Esto se debe hacer mediante una petición ajax del html del formulario.
      iv) (0,5 puntos) Guardar comentario. Se debe utilizar una petición ajax para enviar los datos del comentario al servidor. Se debe mostrar un mensaje al usuario mediante una transición/animación indicándole el resultado. Debe tener un botón que permita cerrar el mensaje. Si el comentario se guardó correctamente se debe limpiar el formulario y refrescar la lista de comentarios. Este

mensaje impedirá que se pueda interactuar con el formulario hasta que se cierre.

v) (0,5 puntos) Si el usuario está logueado, para cada comentario aparecerá un botón que permitirá al usuario responder al comentario. Al pinchar en este botón, en el campo "Título" del formulario aparecerá el mensaje "Re: " acompañado del título del mensaje que se está respondiendo y el foco se pasará al campo "Texto".

**Nota**: Para poder guardar el comentario, junto a los campos del formulario se debe enviar la clave obtenida al loguearse, el login del usuario y el id del viaje.

6) Página ***buscar_viaje.html***.
   a) (0,5 puntos) Se debe implementar la llamada ajax al servidor para que realice la búsqueda con los valores indicados en el formulario de búsqueda.
      i) Se podrá hacer una búsqueda global, que buscará un texto en los campos *nombre* y *descripción* del viaje.
      ii) Se podrá hacer una búsqueda más detallada en la que se podrán combinar distintos criterios de búsqueda.
      iii) Los resultados de la búsqueda se mostrarán en la zona de resultados. Se recomienda mostrar las fichas como en la página *index.html*. El usuario podrá pinchar en una ficha de viaje, tras lo que será dirigido a *viaje.html* para ver toda la información del mismo.
   b) (0,75 puntos) Los resultados de la búsqueda se mostrarán paginados. Al final de los resultados aparecerán unos botones que permitirán ir a la primera página, a la siguiente, anterior o última. Además, junto a estos botones aparecerá información que le dirá al usuario la página de resultados en la que se encuentra del total, algo así como "Página *X* de *N*".

7) (0,5 puntos) Página ***login.html.***
   a) Si el usuario logueado y se intenta acceder a esta página escribiendo la url en la barra de navegación, se debe redirigir a *index.html*.
   b) El login se hace mediante la correspondiente petición ajax.
   c) Si el proceso de login es incorrecto se debe mostrar un mensaje informativo al usuario. El mensaje informativo debe impedir la interacción con el formulario y tendrá un botón que cerrará el mensaje, tras lo que volverá a colocar el foco en el campo login.
   d) Si el proceso de login es correcto:
      i) Se debe utilizar *sessionStorage* para guardar información del usuario que nos devuelva el servidor. Más tarde, se utilizará esta información para saber si el usuario está logueado. Se recomienda guardar toda la información que nos devuelve el servidor.
      ii) Se debe mostrar un mensaje de bienvenida al usuario que le informe de la última vez que estuvo conectado. Este mensaje debe impedir interactuar con el formulario y tendrá un botón para que el

usuario lo pueda cerrar. Al cerrar este mensaje, se debe redireccionar a la página *index.html*.

**Nota**: En ambos casos, el mensaje debe mostrarse utilizando transiciones, animaciones y/o transformaciones CSS3.

8) Página ***registro.html***. Funcionará como página para darse de alta como nuevo usuario cuando no haya ningún usuario logueado; o bien funcionará como página de perfil cuando haya un usuario logueado. Tanto el registro como la actualización del perfil se hacen mediante la correspondiente petición ajax.
   a) (0,5 puntos) Cuando el usuario está logueado y la página funciona como la página de perfil del usuario:
      i) Se rellenarán todos los campos, menos los de password, con la información del usuario (podéis hacer uso de la información de usuario que tenéis en sessionStorage).
      ii) El campo de login debe deshabilitarse para que no se pueda modificar.
      iii) Los campos password y repetir password se podrán dejar en blanco.
   b) (0,5 puntos) Cuando la página funciona como página de registro, a la hora de introducir el login se debe ir comprobando si está disponible o no y mostrar un mensaje informativo al usuario junto al campo. Para comprobar si el login está disponible o no, se debe preguntar al servidor mediante ajax. Como lo hago? Qué peticion he de hacer con ajax?
   c) (0,25 puntos) Si los campos password y repetir password no tienen el mismo valor, no se debe permitir la acción de registro/actualización de perfil y se debe mostrar un mensaje informativo junto al campo password.
como subo la foto
   d) (0,5 puntos) Al seleccionar la foto de perfil, ésta se debe mostrar en la zona de la foto. Se deberá comprobar que su tamaño no exceda los 500KB. Si su tamaño es mayor de 500KB, se mostrará un mensaje indicando el error y no se podrá enviar el formulario.
   e) (0,25 puntos) Al registrarse un usuario correctamente, se debe mostrar un mensaje en una capa semitransparente con una transición/animación indicando al usuario que el registro se ha efectuado correctamente. Este mensaje debe impedir interactuar con el formulario de registro y tendrá un botón para poder cerrarlo, tras lo que será redirigido a la página *login.html*.
en una capa semitransparente? wtf

## Entrega

- El plazo de entrega de la práctica finalizará el **domingo 17 de abril de 2016**, a las 23:55h.
- La práctica debe ir acompañada de una **documentación** en la que figure, como mínimo, los siguientes apartados:
  - Nombre, DNI y grupo del autor o autores de la práctica.
  - Información sobre la parte optativa: indicar si se ha realizado en su totalidad, o bien qué partes se han realizado.

- Apartado de posibles incompatibilidades y problemas de los navegadores. Acompañar esta lista de posibles incompatibilidades y problemas de la solución utilizada para solventarlos (si se ha podido).

**Nota**: La documentación se puede hacer en un documento independiente, o bien incluirla en la página ***acerca.html***.

- **La entrega se realizará a través de la plataforma Moodle** mediante la opción habilitada para ello y consistirá en un único fichero comprimido que **deberá incluir lo siguiente**:
  - Documentación de la práctica (si se ha realizado en un documento independiente).
  - Código completo de la práctica. Se debe comprimir la carpeta completa del sitio web.

# Peticiones del servidor *rest* de la práctica 2

## ERROR

Para todas las peticiones, si se produce un error se devuelve:
{"RESULTADO":"error", "CODIGO":"*código del error*", "DESCRIPCION":"*Descripción del error*"}

## Método GET

- **Disponibilidad de login**: **rest/login/{LOGIN}**
  Respuesta:
  - Login disponible: {"RESULTADO": "ok", "CODIGO", "DISPONIBLE": "true"}
  - Login no disponible: {"RESULTADO": "ok", "CODIGO", "DISPONIBLE": "false"}

- **Pedir información de viajes**:
  - Con ID de viaje: **rest/viaje/{ID_VIAJE}**
    Devuelve toda la información del viaje con el id indicado.
  - Con parámetros:
    - **rest/viaje/?u={número}**
      Devuelve los últimos (número) viajes más recientes.
    - **rest/viaje/?bt={texto}**
      Realiza la **búsqueda global**. Devuelve los viajes que tengan la subcadena *texto* en el **nombre** o la **descripción**.
    - **rest/viaje/?n={nombre}**
      Devuelve los viajes que tengan la subcadena *texto* en el **nombre**.
    - **rest/viaje/?d={texto}**
      Devuelve los viajes que tengan la subcadena *texto* en la **descripción**.
    - **rest/viaje/?l={login}**
      Devuelve los viajes creados por el usuario **login**.

- **rest/viaje/?fi={aaaa-mm-dd}**
  Devuelve los viajes cuya fecha de inicio o fecha de fin sea posterior a la **fecha fi**.
- **rest/viaje/?ff={aaaa-mm-dd}**
  Devuelve los viajes cuya fecha de inicio o fecha de fin sea anterior a la **fecha ff**.
- **rest/viaje/?vi={número}**
  Devuelve los viajes cuya valoración sea mayor o igual a **vi**.
- **rest/viaje/?vf={número}**
  Devuelve los viajes cuya valoración sea menor o igual a **vf**.
- **rest/viaje/?pag={pagina}&lpag={registros_por_pagina}**
  Se utiliza para pedir los datos con paginación. Devuelve los registros que se encuentren en la página **pagina**, teniendo en cuenta un tamaño de página de **registros_por_pagina**.

Los parámetros se pueden combinar y utilizar varios en la misma petición. El resultado es una combinación de condiciones mediante AND.

- **Pedir información de fotos**:
  - Con ID de foto: **rest/foto/{ID_FOTO}**
    Devuelve toda la información de la foto con el id indicado.
  - Con parámetros:
    - **rest/foto/?id_viaje={ID_VIAJE}**
      Devuelve las fotos del viaje con el id indicado.
    - **rest/viaje/?pag={pagina}&lpag={registros_por_pagina}**
      Se utiliza para pedir los datos con paginación. Devuelve los registros que se encuentren en la página **pagina**, teniendo en cuenta un tamaño de página de **registros_por_pagina**.

- **Pedir información de comentarios**:
  - Con ID de comentario: **rest/comentario/{ID_COMENTARIO}**
    Devuelve toda la información del comentario con el id indicado.
  - Con parámetros:
    - **rest/comentario/?id_viaje={ID_VIAJE}**
      Devuelve los comentarios del viaje con el id indicado.
    - **rest/comentario/?u={ENTERO}**
      Devuelve los últimos comentarios (tantos como indique el entero que se pasa) ordenados por fecha descendente.
    - **rest/viaje/?pag={pagina}&lpag={registros_por_pagina}**
      Se utiliza para pedir los datos con paginación. Devuelve los registros que se encuentren en la página **pagina**, teniendo en cuenta un tamaño de página de **registros_por_pagina**.

## Método POST

- **Hacer login**: **rest/login/**
  Parámetros de la petición:

- **login**: login de usuario
- **pwd**: contraseña

Respuesta:

- Si se ha podido realizar el login:
  {"RESULTADO":"ok","CLAVE":"1225286697fbb5acbab746b2973de1e3", "LOGIN":"usu1","NOMBRE":"Usuario 1", "EMAIL":"usu1@ph2.es", "ULTIMO_ACCESO":"2016-03-01 16:31:21", "FOTO":"usu1.jpg"}
  **Importante**: El login y la clave que devuelve el servidor se deberán enviar para el resto de peticiones POST, junto a los parámetros.

- **Creación de un viaje**: **rest/viaje/**

  Parámetros de la petición:
  - **clave**: clave obtenida al hacer login
  - **login**: login del usuario que crea el viaje.
  - **nombre**: nombre del viaje.
  - **descripcion**: descripción del viaje.
  - **valoración**: valoración del viaje.
  - **vi**: valoración del viaje: valor inicial.
  - **vf**: valoración del viaje: valor final.
  - **fi**: fecha de inicio del viaje.
  - **ff**: fecha de finalización del viaje.

  Respuesta:
  - Si se ha podido crear el viaje:
    {"RESULTADO":"ok", "CODIGO":"200", "ID_VIAJE":"4"}
    Devuelve el id del viaje recién creado. Tras la creación correcta del viaje, se crea una carpeta cuyo nombre es el ID del viaje. Esta carpeta se crea en otra llamada *fotos* que debe estar en la misma carpeta que la carpeta *rest*. En la carpeta recién creada se guardarán los ficheros de las fotos del viaje.

  Subir las fotos de un viaje recién creado: **rest/foto/**

  Parámetros de la petición:
  - **clave:** clave obtenida al hacer login.
  - **login:** login del usuario que crea el comentario.
  - **id_viaje:** id del viaje sobre la que se hace el comentario.
  - **descripcion:** descripción de la foto.
  - **fecha:** fecha en la que se tomó la foto. El formato debe ser aaaa-mm-dd.
  - **foto**: fichero de la foto. Debe ser el contenido de un input type="file".

  Respuesta:
  - Si se ha podido subir la foto:
    {"RESULTADO":"ok", "CODIGO":"200", "ID_FOTO":"4", "FICHERO":"4.jpg"}

  Devuelve el id de la foto recién subida. El fichero físico se guarda en la carpeta creada con el ID del viaje correspondiente.

- **Creación de un comentario**: **rest/comentario/**

  Parámetros de la petición:
  - **clave**: clave obtenida al hacer login.
  - **login**: login del usuario que crea el comentario.

- **titulo**: título del comentario.
- **texto**: texto del comentario.
- **id_viaje**: id del viaje sobre la que se hace el comentario.

Respuesta:

- Si se ha podido crear el comentario:
  {"RESULTADO":"ok", "CODIGO":"200", "ID_COMENTARIO":"4"}
  Devuelve el id del comentario recién creado.

- **Registro de nuevo usuario / modificación del perfil de usuario**: **rest/usuario/**

Parámetros de la petición:

- **login**: login del usuario.
- **pwd**: contraseña.
- **nombre**: nombre del usuario.
- **email**: correo electrónico del usuario.
- **foto**: fichero de la foto de perfil del usuario. Debe ser el contenido de un input type="file".

En el caso de **modificación del perfil de usuario** se debe enviar, como mínimo, el campo **login** y el campo **clave**, además del resto de campos a modificar.

Respuesta:

- Si se ha podido crear el usuario:
  {"RESULTADO":"ok", "CODIGO":"ok", "LOGIN":"usu4", "FOTO":"usu4.jpg"}