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 nativo 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 *restful* 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 *restful* 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 7 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.

- El path al que se suben las fotos de las entradas se indica en la variable $uploaddir que se encuentra en el fichero **/rest/funciones.php**. El valor que tiene asignado por defecto asume que la carpeta se llama *fotos* y que está en la misma carpeta que 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 entradas; 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 aparecerá un enlace a la página *nueva_entrada.html* 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.
   c) (0,75 puntos) Implementar una alternativa a los *input type="date"* para que en Firefox el usuario pueda elegir la fecha sin tener que escribirla.
3) Página ***index.html***:
   a) Mediante peticiones Ajax:
      i) (0,5 puntos) Pedir y mostrar las *últimas 6 entradas* creadas. Al pinchar en cada una de ellas para ir a *entrada.html*, el id de la entrada se añadirá a la url del enlace para poder recuperarlo en la página de destino.
      ii) (0,75 puntos) La zona de últimas entradas tendrá la típica botonera de paginación. En el pie de la zona 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*".
      iii) (0,5 puntos) Pedir y mostrar los 10 últimos comentarios escritos por los usuarios.

4) Página ***nueva_entrada.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 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. En todo momento, al pinchar en el elemento <img> también se mostrará el diálogo para poder seleccionar o cambiar el fichero de imagen.

ii) (0,5 puntos) Debe aparecer un mensaje avisando al usuario que el tamaño máximo de archivo es 500KB. 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 500KB. Si su tamaño es mayor de 500KB, se mostrará un mensaje sobre la imagen de la ficha indicando el error. Además, se destacará la ficha de las demás mediante css poniendo, por ejemplo, el borde de la ficha en rojo.

c) (0,75 puntos) Al pinchar en el botón para enviar los datos del formulario:
   i) No se podrán enviar los datos mientras haya fotos *no correctas*.
   ii) Para crear una nueva entrada primero se envían los datos de la entrada mediante AJAX. Si todo ha ido bien (respuesta correcta del servidor) y se ha creado la entrada, entonces se enviarán las fotos una a una. Todo ello se hace mediante llamadas ajax. El servidor *rest* copiará las fotos subidas a la carpeta *fotos*. Cada foto se guardará con su id como nombre y la extensión del fichero subido.
   iii) Al crear una nueva entrada 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 la entrada, además de los campos del formulario se debe enviar la clave obtenida al loguearse y el login del usuario. El login de usuario se envía como un campo más del formulario, pero la clave se debe enviar como una cabecera “Authorization”.

5) Página ***entrada.html***:
   a) Al cargar la página se debe obtener de la url el id de la entrada a mostrar, así como también el id del comentario, si fuera el caso.
      i) Si en la url no se encuentra el id de la entrada (porque se intenta acceder directamente), se debe redirigir a *index.html*.
      ii) (0,5 puntos) Se utilizará el id de la entrada para realizar una petición ajax al servidor y mostrar toda su información. Además, se harán otras dos peticiones ajax para mostrar tanto las fotos, como los comentarios de los usuarios sobre la entrada.
      iii) (0,5 puntos) Carga condicional de contenido utilizando JavaScript y AJAX.
         1) Si el usuario no está logueado, el formulario para dejar un comentario no estará disponible, ni siquiera estará oculto en el HTML. En su lugar aparecerá un mensaje con un texto similar

a: “Para dejar un comentario debes estar logueado”. En este mensaje, la palabra “logueado” será un enlace que lleve a *login.html*.

2) Cuando el usuario esté logueado, el html del formulario se obtendrá mediante una llamada ajax y se incluirá en el lugar correspondiente de la página.

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. El mensaje impedirá que se pueda interactuar con el formulario hasta que se cierre. Si el comentario se guardó correctamente se debe limpiar el formulario y refrescar la lista de comentarios. Si el mensaje no se pudo guardar, tras cerrar el mensaje se devolverá el foco al formulario.

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 al 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 (como cabecera de la petición), el login del usuario y el id del viaje.

6) Página ***buscar.html***.
   a) (0,75 puntos) Se debe implementar la llamada ajax al servidor para que realice la búsqueda con los valores indicados en el formulario de búsqueda. En el servidor la búsqueda se realiza utilizando el operador AND para combinar las condiciones.
      i) Se podrá hacer una búsqueda por título, descripción, autor de la entrada y fecha (desde - hasta).
      ii) 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, tras lo que será dirigido a *entrada.html* para ver toda la información de la misma.
   b) Los resultados de la búsqueda se mostrarán paginados. El funcionamiento de la paginación será el mismo que el implementado en *index.html* pero, lógicamente, con respecto a los resultados de la búsqueda.

7) (0,5 puntos) Página ***login.html*.**
   a) Si el usuario está 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. El registro se hace mediante la correspondiente petición ajax.
   a) (0,5 puntos) 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.
   b) (0,25 puntos) Si los campos password y repetir password no tienen el mismo valor, no se debe permitir la acción de registro y se debe mostrar un mensaje informativo junto al campo password o repetir password.
   c) (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*.

Entrega

- El plazo de entrega de la práctica finalizará el **domingo 23 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 al servicio *restful* 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 entradas**:
  - Con ID de entrada:
    - **rest/entrada/{ID_ENTRADA}**
      Devuelve toda la información de la entrada con el id indicado.
    - **rest/entrada/{ID_ENTRADA}/fotos**
      Devuelve todas las fotos de la entrada con el id indicado.
    - **rest/entrada/{ID_ENTRADA}/comentarios**
      Devuelve todos los comentarios de la entrada con el id indicado.
  - Con parámetros:
    - **rest/entrada/?u={número}**
      Devuelve las últimas (número) entradas más recientes.
    - **rest/entrada/?n={texto}**
      Devuelve las entradas que tengan la subcadena *texto* en el **nombre**.
    - **rest/entrada/?d={texto}**
      Devuelve las entradas que tengan la subcadena *texto* en la **descripción**.
    - **rest/entrada/?l={login}**
      Devuelve las entradas creadas por el usuario **login**.
    - **rest/entrada/?fi={aaaa-mm-dd}**
      Devuelve las entradas cuya fecha sea posterior a la **fecha fi**.
    - **rest/entrada/?ff={aaaa-mm-dd}**
      Devuelve las entradas cuya fecha sea anterior a la **fecha ff**.
    - **rest/entrada/?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_entrada={ID_ENTRADA}**
      Devuelve las fotos de la entrada con el id indicado.
    - **rest/foto/?pag={pagina}&lpag={registros_por_pagina}**
      Se utiliza para pedir los datos con paginación.

- **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_entrada={ID_ENTRADA}**
      Devuelve los comentarios de la entrada con el id indicado.
    - **rest/comentario/?u={número}**
      Devuelve los últimos comentarios (tantos como indique el número entero que se pasa) ordenados por fecha descendente.
    - **rest/comentario/?pag={pagina}&lpag={registros_por_pagina}**
      Se utiliza para pedir los datos con paginación.

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","CODIGO":"200","clave":"1225286697fbb5acbab
    746b2973de1e3", "login":"usu1","nombre":"Usuario 1",
    "email":"usu1@ph2.es", "ultimo_acceso":"2017-03-01 16:31:21"}
    ```
    Importante: El login y la clave que devuelve el servidor se deberán enviar en el resto de peticiones POST, junto a los parámetros correspondientes.

- **Creación de una entrada**: **rest/entrada/**
  Cabeceras de la petición:
  - **clave**: clave obtenida al hacer login. Se envía como cabecera “Authorization”.

  Parámetros de la petición:
  - **login**: login del usuario que crea la entrada.

- **nombre**: nombre de la entrada.
- **descripcion**: descripción de la entrada.

Respuesta:

- Si se ha podido crear la entrada:
  `{"RESULTADO":"ok", "CODIGO":"200", "id":"4"}`
  Devuelve el id de la entrada recién creada.

Subir las fotos de una entrada recién creada: **rest/foto/**

Cabeceras de la petición:

- **clave**: clave obtenida al hacer login. Se envía como cabecera "Authorization".

Parámetros de la petición:

- **login:** login del usuario que crea la entrada
- **id_entrada:** id de la entrada a la que pertenecen las fotos.
- **texto:** texto que acompaña a la foto.
- **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":"4"}`
  Devuelve el id de la foto recién subida. El fichero físico se guarda con el id como nombre y la extensión del fichero.

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

Cabeceras de la petición:

- **clave**: clave obtenida al hacer login. Se envía como cabecera "Authorization".

Parámetros de la petición:

- **login**: login del usuario que crea el comentario.
- **titulo**: título del comentario.
- **texto**: texto del comentario.
- **id_entrada**: id de la entrada sobre la que se hace el comentario.

Respuesta:

- Si se ha podido crear el comentario:
  `{"RESULTADO":"ok", "CODIGO":"200", "id":"17"}`
  Devuelve el id del comentario recién creado.

- **Registro de nuevo usuario**: **rest/usuario/**

Parámetros de la petición:

- **login**: login del usuario.
- **pwd**: contraseña.
- **pwd2**: contraseña 2. Es el valor del campo para repetir contraseña.
- **nombre**: nombre del usuario.
- **email**: correo electrónico del usuario.

Respuesta:

- Si se ha podido crear el usuario:
  `{"RESULTADO":"ok", "CODIGO":"200", "login":"usu4", "nombre":"Usuario 4"}`