PROGRAMACIÓN DEL CLIENTE WEB. 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 tecnologías como AJAX o el API Fetch para realizar peticiones a un servidor RESTful evitando, de esta manera, la recarga de la página web para actualizar su contenido.
- 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, así como también para realizar peticiones al servidor RESTful que se proporciona. Se recomienda crear una copia de la carpeta de la práctica 1 y nombrarla como práctica 2. De esta manera podréis empezar a realizar las modificaciones que se os piden en este enunciado y siempre tendréis el código de la práctica 1 tal cual lo entregasteis.

Para poder realizar esta segunda práctica es necesario utilizar el servidor web gratuito XAMPP (https://www.apachefriends.org/es/index.html). Junto al enunciado de la práctica se os proporciona un script sql que, importado mediante la herramienta phpMyAdmin de xampp, creará una base de datos llamada ***recetas***, en la que habrá una serie de datos de prueba, y concederá permisos de lectura/escritura al usuario ***pcw*** con contraseña ***pcw***.

Además, también se os facilita un archivo comprimido llamado `practica02.zip` que contiene dos carpetas: `fotos` y `rest`. La carpeta `fotos` contiene los archivos de imagen correspondientes a los datos de prueba de la BD, y la carpeta `rest` contiene una serie de ficheros php, organizados en subcarpetas, que proporcionan un servicio web de tipo *RESTful* mediante el que poder acceder a la base de datos. Estas dos carpetas deberán copiarse dentro de la carpeta en la que se encuentra el resto de archivos de la práctica 2 en el servidor XAMPP. Este servicio web *RESTful* será el destinatario de las peticiones ajax y/o fetch de la práctica. Al final de este enunciado se indican las peticiones que se pueden realizar junto a sus parámetros.

Entre los ficheros que se os proporcionan junto al enunciado, tenéis también un vídeo explicativo en el que se realiza todo el proceso.

Notas:

- El servidor RESTful está configurado suponiendo que la carpeta de la práctica 2 de pcw se llama `practica02` y está en la carpeta `htdocs/pcw`, por lo que el path de la

carpeta de la práctica 2 en el servidor XAMPP sería: `htdocs/pcw/practica02`

- Si el path en el que se encuentra la carpeta de la práctica 2 no es el indicado en el punto anterior, será necesario modificar la línea 9 del fichero `rest/.htaccess`.

- 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 de la propiedad **`sessionStorage`** del objeto `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. No obstante, entre los ficheros que se os proporcionan se encuentra un pdf explicativo al respecto.
- El path al que se suben las fotos de las recetas se indica en la variable `$uploaddir` que se encuentra en la línea 5 del fichero **`/rest/configbd.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. Atención, la validación del html para esta segunda práctica se hace de forma diferente a como se hizo para la primera ya que, en este caso, incluirá el contenido generado con JavaScript.

2) Para todas las páginas:
   a) (0,25 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; y para darse de alta como nuevo usuario.
      ii) Cuando el usuario está logueado, los enlaces para login y para alta de nuevo usuario deben desaparecer. Además, estando logueado aparecerá un enlace a la página `nueva_receta.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. Básicamente, si el usuario nos está logueado no podrá acceder a la página `nueva_receta.html`, y si el usuario está logueado no podrá acceder ni a la página de login, ni a la página para darse de alta como nuevo usuario.

3) Página **`index.html`**. Lo primero que se debe hacer en esta página es eliminar el control que se pedía en la práctica 1 para poder ordenar ascendente o descendente,

por valoración o fecha, ya que no se necesita más.

a) Mediante petición ajax/fetch:
   i) (0,25 puntos) Búsqueda rápida. En el cuadro de texto para la búsqueda rápida se podrá escribir el texto, o textos separados por comas, a buscar en los campos `nombre` y `elaboracion` de las recetas de la BD. Al pinchar en el botón para realizar la búsqueda, se redirigirá a la página `buscar.html` pasándole como argumento el contenido del campo de texto de búsqueda. De esta manera, será en la página `buscar.html` donde se realizará la búsqueda y se mostrarán los resultados.
   ii) (0,5 puntos) Pedir y mostrar las *últimas 6 recetas* creadas. Cada una de ellas permitirá al usuario, simplemente pinchando con el ratón, ir a `receta.html` para consultar toda la información de la receta. Hay que tener en cuenta que al redireccionar a la página de receta se debe añadir el `id` de la misma a la url del enlace para poder recuperarlo en la página de destino. Debéis tener en cuenta que al pinchar sobre el autor de la receta se debe redirigir a buscar.html, pasándole como parámetro el autor para que nos muestre todas las recetas del mismo.
   iii) (0,25 puntos) La zona de últimas recetas 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*".

4) Página **`nueva_receta.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) (0,25 puntos) En la zona de ingredientes, al pinchar el bóton para añadir nuevo ingrediente, se debe recoger el texto del campo de texto de nuevo ingrediente y añadirlo al final de la lista de ingredientes de la receta.
      ii) (0,5 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, como mínimo, 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.
      iii) (0,75 puntos) Si el tamaño del fichero de imagen seleccionado es mayor de 300kB no se debe cargar y, además, se debe mostrar un mensaje modal o emergente (no utilizar la función `alert()` de javascript) al usuario indicándoselo. Al seleccionar un fichero de imagen correcto, la imagen se mostrará en el elemento `<img>` de la correspondiente ficha.
   c) Al pinchar en el botón para crear la nueva receta,:

i) (0,25 puntos) si no se ha añadido ninguna foto, no se enviará la información al servidor y se mostrará un mensaje modal o emergente (no utilizar `alert()`) al usuario indicándole que, al menos, debe añadir una foto.

ii) (1 punto) Cuando al menos hay una foto en la receta, para enviar los datos al servidor, hay que seguir los siguientes pasos:

1) Se envían los datos de la receta mediante una petición ajax/fetch. Si se ha producido algún error al intentar crear la receta en la BD (se recibe mensaje de error del servidor), se mostrará un mensaje modal o emergente (no utilizar `alert()`) al usuario informándole del error.
2) Si todo ha ido bien (respuesta correcta del servidor) y se ha creado la receta, entonces se recogerán los ingredientes de la lista de ingredientes y se enviarán al servidor para darlos de alta.
3) Se enviarán las fotos de la receta **una a una**, es decir, hasta que no se tenga la confirmación por parte del servidor de que ha recibido y guardado correctamente la foto no se enviará la siguiente. Todo ello se hace mediante llamadas ajax/fetch. El servidor *rest* copiará las fotos subidas a la carpeta *fotos*. El servidor guardará cada foto con su id como nombre y la extensión del fichero subido, dentro de la carpeta *fotos*.
4) Al finalizar correctamente el proceso de creación de una nueva receta, se debe limpiar el formulario (lista de ingredientes y fotos incluidas) y mostrar un mensaje modal o emergente (no utilizar `alert()`) al usuario informándole del resultado. El mensaje mostrado tendrá un texto similar a "Se ha creado correctamente la receta **Sopa de caracol**" y deberá tener un botón para que el usuario pueda cerrarlo, tras lo que será redirigido a `index.html`.

Nota: Para poder crear la receta, 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 **`receta.html`**: Primero, eliminar los botones para la paginación de los comentarios que se pedía en la práctica 1. Por simplificar, no habrá paginación de comentarios.
   a) Al cargar la página se debe obtener de la url el `id` de la receta a mostrar.
      i) Si en la url no se encuentra el `id` de la receta (porque se intenta acceder directamente), se debe redirigir a `index.html`.
      ii) (0,75 puntos) Se utilizará el `id` de la receta para realizar una petición ajax/fetch y mostrar toda su información. Además, se harán otras tres peticiones ajax/fetch para mostrar tanto los ingredientes, como las fotos y los comentarios de la receta.
      iii) (0,25 puntos) Valoración de la receta. Si el usuario está logueado, deberán aparecer los botones de *Me gusta* y *No me gusta* para que pueda votar. Si el usuario no está logueado estos botones no deben

aparecer.

iv) (0,5 puntos) Para poder realizar el voto, ya sea negativo o positivo, es necesario realizar una petición ajax/fetch al servidor. Tras realizar el voto y recibir la confirmación del servidor, se deberá mostrar un mensaje (no utilizar `alert()`) al usuario informándole del voto correcto. El mensaje puede ser modal, emergente o simplemente un texto que aparezca debajo del botón con un icono/botón/etc que permita cerrarlo.

v) (0,5 puntos) Carga condicional de contenido utilizando JavaScript y ajax/fetch.
   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/fetch a un fichero html que contendrá el código html del formulario y se incluirá en el lugar correspondiente de la página. En ningún caso se generará el html desde javascript.

vi) (0,5 puntos) Guardar comentario. Se debe utilizar una petición ajax/fetch para enviar los datos del comentario al servidor. Se debe mostrar un mensaje modal o emergente (no utilizar `alert()`) al usuario indicándole el resultado. El mensaje tendrá un botón que permitirá cerrarlo y, a continuación,:
   1) si el comentario se guardó correctamente, se limpiará el formulario para dejar comentario;
   2) si el comentario no se pudo guardar, tras cerrar el mensaje se devolverá el foco al formulario.

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` de la receta.

6) Página **`buscar.html`**. Al igual que en `index.html`, hay que eliminar el control que se pedía en la práctica 1 para poder ordenar ascendente o descendente, por valoración o fecha, ya que no se necesita más. El formulario de búsqueda deberá tener dos botones: uno para limpiar el formulario (botón `reset`) y otro para enviar el formulario de búsqueda (botón `submit`).
   a) (0,5 puntos) Al cargar la página se consultará el posible parámetro de búsqueda que vendrá en la url. Recordad que puede venir como argumento un login de usuario, o un texto, o textos separados por comas, por el que buscar recetas. Si viniera alguno de estos parámetros, se debería realizar la búsqueda (utilizando la misma función que se pide implementar en el siguiente apartado) y mostrar los resultados en zona de resultados. Si no

hubiera ningún parámetro en la url, se mostrarán las últimas 6 recetas (misma petición que en `index.html`)

b) (0,5 puntos) Se debe implementar la llamada ajax/fetch 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 nombre, ingredientes, tiempo de elaboración (desde - hasta), dificultad, número de comensales y autor de la receta.
   ii) Los resultados de la búsqueda se mostrarán en la zona de resultados. Las fichas de las recetas se mostrarán igual que en la página `index.html` y con las mismas funcionalidades.

c) (0,25 puntos) 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/fetch.
   c) Si el proceso de login es incorrecto se debe mostrar un mensaje modal o emergente (no utilizar `alert()`) informativo al usuario. El mensaje tendrá un botón que cerrará el mensaje, tras lo que volverá a colocar el foco en el campo de login.
   d) Si el proceso de login es correcto se debe utilizar el objeto `sessionStorage` para guardar la información del usuario que nos devuelva el servidor. Más tarde, se utilizará esta información para saber si el usuario está logueado y para poder crear recetas, comentarios, etc. Se recomienda guardar toda la información que nos devuelve el servidor. Una vez guardada la información en `sessionStorage`, se debe redireccionar a la página `index.html`.

   Nota: En ambos casos, el mensaje debe ser modal o emergente (no utilizar `alert()`).

8) Página **`registro.html`**. Funcionará como página para darse de alta como nuevo usuario. El registro se hace mediante la correspondiente petición ajax/fetch.
   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/fetch.
   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, primero, limpiar el formulario y, a continuación, mostrar un mensaje modal o

emergente indicando al usuario que el registro se ha efectuado correctamente. Este mensaje 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 29 de abril de 2018**, 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 el trabajo realizado: qué partes de la práctica se han implementado y qué partes no.
  - 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 web *RESTful* de la práctica 2

ERROR

Para todas las peticiones, si se produce un error se devuelve el siguiente texto en formato JSON:

```
 {"RESULTADO":"ERROR","CODIGO":"código del error",
"DESCRIPCION":"Descripción del error"}
```

Peticiones GET

- **Disponibilidad de login**: `rest/login/{LOGIN}`
  Respuesta:
  - Login disponible: `{"RESULTADO":"OK", "CODIGO":200,"DISPONIBLE": "true"}`

  - Login no disponible: `{"RESULTADO":"OK","CODIGO":200,"DISPONIBLE": "false"}`

- **Pedir información de recetas**:
  - Con ID de receta:
    - **`rest/receta/{ID_RECETA}`**
      Devuelve toda la información de la receta con el id indicado.
    - **`rest/receta/{ID_RECETA}/fotos`**
      Devuelve todas las fotos de la receta con el id indicado.
    - **`rest/receta/{ID_RECETA}/comentarios`**
      Devuelve todos los comentarios de la receta con el id indicado.
  - Con parámetros:
    - **`rest/receta/?u={número}`**
      Devuelve las últimas (número) recetas más recientes.
    - **`rest/receta/?t={texto1,texto2,...}`**
      Devuelve las recetas que tengan la subcadena *texto* en el **nombre** y/o la **elaboración**. Es la petición a utilizar para la búsqueda rápida de `index.html`.
    - **`rest/receta/?n={texto1,texto2,...}`**
      Devuelve las recetas que tengan en el **nombre** al menos una de las subcadenas de texto indicadas y separadas por comas.
    - **`rest/receta/?i={ingrediente1,ingrediente2,...}`**
      Devuelve las recetas que tengan al menos uno de los ingredientes indicados y separados por comas.
    - **`rest/receta/?e={texto1,texto2,...}`**
      Devuelve las recetas que contengan en la **elaboración** alguna de las subcadenas de texto indicadas y separadas por comas.
    - **`rest/receta/?a={login}`**
      Devuelve las recetas cuyo autor es el usuario **login**.
    - **`rest/receta/?d={número}`**
      Devuelve las recetas con la dificultad indicada con número: 0 - baja; 1 - media; 2 - alta.
    - **`rest/receta/?c={número}`**
      Devuelve las recetas para el número de comensales indicado.
    - **`rest/receta/?di={número_de_minutos}`**
      Devuelve las recetas cuya duración sea igual o mayor al número indicado.
    - **`rest/receta/?df={número_de_minutos}`**
      Devuelve las recetas cuya duración sea menor o igual al número indicado.
    - **`rest/receta/?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.

Peticiones 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":"16215f8a6564f85f32934cdb38a1c643","login":"usuario2","nombre":"Usuario 2","email":"usuario2@pcw.es","fnac":"2018-01-01"}`

    Importante: El login y la clave que devuelve el servidor se deberán enviar en el resto de peticiones POST, junto a los parámetros correspondientes.

- **Crear de una receta**: **rest/receta/**
  Cabeceras de la petición:
  - **clave**: clave obtenida al hacer login. Se envía como cabecera "Authorization".

  Parámetros de la petición:
  - **l**: login del usuario que crea la receta.
  - **n**: nombre de la receta.
  - **e**: elaboración de la receta.
  - **t**: tiempo de elaboración de la receta.
  - **d**: dificultad de la receta.
  - **c**: número de comensales de la receta.

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

- **Subir ingredientes de una receta**: **rest/receta/{ID_RECETA}/ingredientes**
  Cabeceras de la petición:
  - **clave**: clave obtenida al hacer login. Se envía como cabecera "Authorization".

  Parámetros de la petición:
  - **l**: login del usuario que crea la receta.
  - **i**: vector de ingredientes como texto en formato JSON. Ejemplo: `["ingrediente 1", "ingrediente 2", ...]`.

  Respuesta:
  - Si se han podido insertar los comentarios:
    `{"RESULTADO":"OK", "CODIGO":200, "DESCRIPCION":"INGREDIENTES GUARDADOS CORRECTAMENTE"}`

- **Insertar comentario de una receta**: **rest/receta/{ID_RECETA}/comentario**

Cabeceras de la petición:
- **clave**: clave obtenida al hacer login. Se envía como cabecera "Authorization".

Parámetros de la petición:
- **l**: login del usuario que crea la receta.
- **titulo**: titulo del comentario.
- **texto**: texto del comentario.

Respuesta:
- Si se han podido insertar los comentarios:
  `{"RESULTADO":"OK", "CODIGO":200, "ID":24}`
  Devuelve el `id` del comentario recién creado.

- **Subir foto de una receta**: **rest/receta/{ID_RECETA}/foto**

Cabeceras de la petición:
- **clave**: clave obtenida al hacer login. Se envía como cabecera "Authorization".

Parámetros de la petición:
- **l**: login del usuario que crea la receta.
- **t**: texto descriptivo de la foto.

Respuesta:
- Si se ha podido subir la foto correctamente:
  `{"RESULTADO":"OK","CODIGO":200,"ID":5,"FICHERO":"5.jpg"}`
  Devuelve el `id` y el nombre del fichero de la foto recién subida.

- **Emitir un voto para una receta**: **rest/receta/{ID_RECETA}/voto/{VALOR_VOTO}**

Si `{VALOR_VOTO}` es 1, el voto es positivo; si es 0, negativo.
Cabeceras de la petición:
- **clave**: clave obtenida al hacer login. Se envía como cabecera "Authorization".

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

Respuesta:
- Si se ha emitido el voto correctamente:
  `{"RESULTADO":"OK","CODIGO":200,"ID":5,"VOTO":1}` -> positivo
  `{"RESULTADO":"OK","CODIGO":200,"ID":5,"VOTO":0}` -> negativo

- **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.
- **fnac**: fecha de nacimiento en formato aaaa-mm-dd.

Respuesta:
- Si se ha podido crear el usuario:
  `{"RESULTADO":"OK", "CODIGO":200, "login":"usuario3",`
  `"nombre":"Usuario 3"}`