Ejercicios Unidad 10 – Mecanismos de Comunicación Asíncrona (AJAX).

EJERCICIO: u10e01_lectorFicherosPromesas

Crea una página web con las siguientes características:

- Tendrá un campo de texto en el que, nada más cargarse la página, se muestre la URL de la misma. Ese campo se puede editar por el usuario.
- Un botón junto al campo de texto que se llame “Mostrar contenido”. Al hacer clic sobre él, se cargará en un `textarea` el contenido del archivo indicado en el campo de texto.
- El `textarea`, inicialmente estará vacío.

EJERCICIO: u10e01_lectorFicherosXHR

Repite el ejercicio anterior, pero utilizando XMLHttpRequest en lugar de promesas.

EJERCICIO: u10e02_portafolioJSONPromesas

Partiendo del fichero JSON **portafolio.json**, muestra en una tabla toda la información que contiene ese fichero, utilizando AJAX.

EJERCICIO: u10e02_portafolioJSONXHR

Repite el ejercicio anterior, pero utilizando XMLHttpRequest en lugar de promesas.

EJERCICIO: u10e03_localidadGETPromesas

Crea un programa con las siguientes características:

- Una página con HTML que tenga un **`input`** de tipo **texto** y un **botón**: cuando el usuario introduzca el nombre de una localidad y pulse el botón obtendrá, en un `<div id="resultado">`, un mensaje que indicará si la ciudad está incluida dentro de una lista de ciudades o no. El mensaje será rojo si no está incluida y verde en caso afirmativo.
- Un archivo PHP proporcionado comprueba que la localidad recibida por parámetro está o no incluida dentro de una lista de 10 localidades (utiliza un `array` en PHP y lo recorre para comprobarlo).
- La petición debe realizarse de forma asíncrona, de modo que no se recargará la página, sino que se mostrará el resultado una vez finalizada la consulta al servidor.

EJERCICIO: u10e03_localidadGETXHR

Repite el ejercicio anterior, pero utilizando XMLHttpRequest en lugar de promesas.

EJERCICIO: u10e04_localidadPOSTPromesas

Repite el ejercicio anterior, utilizando el método POST en lugar de GET.

EJERCICIO: u10e04_localidadPOSTXHR

Repite el ejercicio anterior, pero utilizando XMLHttpRequest en lugar de promesas.

EJERCICIO: u10e05_seriesJSONPromesas

(*) Puedes comprobar si un archivo JSON está bien definido con el JSON Viewer http://jsonviewer.stack.hu/.

Crea un archivo **series.json** con información de series. Será un array de objetos:

- **Serie**: contendrá los datos de una serie en concreto, que serán:
  - **Título**: nombre de la serie.
  - **Plataforma**: nombre de la cadena que produce la serie (HBO, Netflix, etc.)
  - **Director**: nombre del director de la serie.
  - **Año**: año de estreno de la serie.
  - **Terminada**: podrá contener un valor “sí” o “no” en función si ha terminado o no su emisión.

Crea una página web que, al cargarse, haga una petición AJAX al servidor, que le devolverá el archivo JSON.

Si el archivo **series.json** no existe o no se ha podido cargar correctamente, se mostrará un texto indicándolo en la pantalla.

Si el archivo se carga correctamente, en la página se generará una tabla con los datos de las series, teniendo en cuenta las siguientes condiciones:

- **El título, la cadena y el director:** el título será negrita, y el director en cursiva.
- El **año** aparecerá en color rojo si la serie es anterior al año 2000, en amarillo si está entre el 2001 y el 2010 y en verde si es posterior al 2011. Estas variaciones se recogen en un archivo en CSS con reglas, como por ejemplo: `.rojo`, `.amarillo` o `.verde`.
- En la celda **terminada** habrá un icono que indique si la serie se ha terminado o no su emisión. Para ello, utiliza una fuente de iconos como por ejemplo *Font Awesome* (ver ejemplo de cómo hacerlo en la carpeta “prueba-fuentes”) o una imagen.

EJERCICIO: u10e06_billetes

Crea un archivo **viajes.json**, donde se registren los posibles viajes (origen y destino) que ofrece una empresa de autobuses.

Los orígenes y destinos posibles son:

- De Madrid: a Barcelona, Valencia y Sevilla.
- De Barcelona: a Madrid y Zaragoza.
- De Valencia: a Madrid.

Crea una página web formada por:

- Una lista desplegable donde se encuentre el listado de orígenes posibles y otro para los destinos.
- Al cargar la página web, se obtendrá del servidor el archivo **viajes.json** y se cargará en el primer desplegable, la lista de orígenes posibles.
- Una vez seleccionado uno de los orígenes, se mostrará la lista de destinos para ese origen.

EJERCICIO: u10e07_teatroJSON

Crea una página HTML teniendo en cuenta contendrá un desplegable donde estén cargados los teatros que se encuentran en el fichero **teatros.json**. La carga de los datos se ha de realizar de forma asíncrona mediante AJAX.

Su funcionamiento será el siguiente:

1. Al seleccionar un teatro del desplegable, se deben visualizar las obras que hay en cartelera en ese teatro. Para ello se hará uso del fichero **cartelera.json**. Se mostrará el título de la obra, el precio, la sinopsis y la imagen de la obra en campos separados. En principio se mostrará la información de la primera obra.
2. Para poder acceder al resto de las obras de dicho teatro se mostrarán 4 botones: **Primero**, **Anterior**, **Siguiente** y **Último**". Se debe controlar que cuando se esté en la primera obra solamente estén activos los botones de siguiente y último, etc.
3. Al cambiar de teatro no se visualizará la información del resto de teatros, es decir, se limpiará la pantalla.

EJERCICIO: u10e08_viajesBdPHP

En la carpeta correspondiente al ejercicio, encontrarás los siguientes ficheros:

- **dbcreacion.sql** lo utilizarás para crear una base de datos.
- **procesar.php** donde introducirás las sentencias necesarias para manipular la base de datos.
- **index.html**
- **viajes.json**

Nada más acceder a la página, se verá una tabla donde se mostrarán los diferentes viajes y su precio que cargaremos desde el fichero **viajes.json**.

Se podrá seleccionar un viaje haciendo clic sobre la fila correspondiente, en ese momento cambiará de color.

Además, deben estar disponibles una serie de botones: **Mostrar viajes comprados**, **Comprar** (desactivado) y **Anular viaje** (desactivado). Cuyo funcionamiento será el siguiente:

1. Al pulsar sobre **Mostrar viajes comprados** se visualizará una tabla con todos los viajes comprados que están almacenados en la tabla de la BD **viajescomprados**.
2. Al seleccionar un viaje de la tabla se habilitará el botón de **Anular viaje**. Si lo pulsamos borrará dicho viaje de la tabla **viajescomprados** y actualizará la tabla que se muestra.
3. Al seleccionar un viaje de la tabla se activará el botón de **Comprar**, si se pulsa se mostrarán unas cajas de texto donde introducir el nombre del comprador, la descripción del viaje (que se copiará de la tabla de viajes), email del destinatario, número de viajes comprados, precio final del viaje, número de tarjeta (que será de 16 números) y CSV que es un código de las tarjeta de 3 dígitos.

Todos los accesos a distintos archivos se deben realizar de forma asíncrona mediante AJAX.

EJERCICIO: u10e09_tareas (repaso)

Vamos a crear una nueva versión de la lista de tareas.

Dividiremos la pantalla en tres zonas:

- Tareas pendientes
- Tareas realizadas
- Tareas eliminadas

Las tareas se leerán de una base de datos que tendrá almacenada la tarea y su estado: pendientes, realizadas o eliminadas.

En cada una de las zonas de la pantalla, habrá una tabla con las tareas que están en ese estado. En la primera zona, estará la caja de texto para crear una nueva tarea, junto con los botones **Añadir** y **Eliminar** que creamos en la actividad correspondiente de la unidad anterior.

En la tabla de **Tareas pendientes**, junto a las tareas, estará botón **Eliminar tarea** que ya teníamos, y un nuevo botón **Finalizar,** para indicar que se ha realizado la tarea. Ambos botones se crearán a partir de una imagen creada con una fuente. Al pasar sobre el botón, aparecerá un `tooltip` (title) describiendo lo que hace el botón.

En la tabla de **Tareas realizadas**, cada tarea tendrá un botón **Eliminar tarea**.

Además, en el lado del servidor, tendremos un array con sugerencias de tareas. Cuando el usuario empiece a escribir una nueva tarea, se mostrarán las sugerencias en un `<datalist>`, para que pueda seleccionarlas. Las sugerencias que aparecerán serán las que contengan el texto que está escribiendo el usuario.

Algunas sugerencias pueden ser:

- Felicitar a…
- Médico…
- Comprar…
- Leer…
- Limpiar…

Se utilizarán llamadas asíncronas para:

- Cargar las tareas de la base de datos cuando se cargue la página.
- Presentar las sugerencias.
- Opcional: modificar la base de datos cada vez que el usuario elimine una tarea o la marque como finalizada.