UNIVERSIDAD COMPLUTENSE
MADRID

APLICACIONES WEB
2018-2019
Grado en Ingeniería del Software
Facultad de Informática

Práctica guiada - Parte 7

Fecha de entrega: 2018

Las aplicaciones web desarrolladas en esta práctica guiada y en la práctica de la red social, obedecen a un modelo particular de aplicaciones web que es el llamado MPA (*Multiple Page Application*). En este modelo, cada vez que el usuario de la web realiza una acción, el servidor devuelve al navegador una página HTML con el resultado de esta operación. Es decir, la interacción con aplicaciones de tipo MPA web se basa en la navegación por una sucesión de páginas.

En esta última parte de la práctica guiada se desarrollará una aplicación web utilizando otro modelo distinto: el modelo SPA (*Single Page Application*). En este modelo, la interacción entre cliente y el servidor se realiza a través de una única página web, la cual el navegador irá transformando dinámicamente. Cada vez que el usuario realiza una acción sobre la página, el cliente realiza una petición al servidor. Sin embargo, la respuesta del servidor ya no consiste en una página HTML entera con el resultado de la acción, sino en una representación JSON de dicha respuesta. El cliente, al recibir esta respuesta, actualiza los elementos de la página HTML actual conforme al resultado de la operación.

La temática de este ejercicio es la misma que en los ejercicios anteriores: un gestor de tareas. Sin embargo, ahora se implementará este gestor mediante una aplicación SPA haciendo las siguientes simplificaciones con respecto a las entregas anteriores:

- Se elimina la posibilidad de que una tarea tenga asociada una o varias etiquetas. Una tarea consiste, solamente, en un texto.
- Tampoco se realizará distinción entre tareas completadas y no completadas, de modo que el atributo **done** de las tareas desaparece. Al lado de cada tarea habrá un botón **[Eliminar]** que borrará directamente el elemento correspondiente de la lista de tareas.
- No es necesario realizar gestión de usuarios. Se supone que existe una única lista de tareas compartida entre todos los usuarios que accedan a la aplicación.
- Las tareas no se guardarán en una base de datos MySQL, sino en un array de objetos, que será accesible mediante una variable global en el servidor. El valor inicial del array será el siguiente:

```
let tasks = [
    {
        id: 1,
        text: "Comprar billetes de avión"
    },
    {
        id: 2,
        text: "Hacer las maletas"
    },
    {
        id: 3,
        text: "Comprar regalos de reyes"
    },
    {
        id: 4,
        text: "Reservar coche"
    }
];
```

El array **tasks** inicialmente contiene cuatro objetos. Cada uno de ellos representa una tarea, que consiste en un texto y un identificador numérico **id**, que es el que se utilizará para realizar operaciones sobre la misma (por ejemplo, para eliminarla). Se supone también que existe una variable global **idCounter** que contiene el identificador que se asignará a la próxima tarea que se inserte. Esta variable se incrementará con cada inserción. Con esto se pretende simular el comportamiento de los campos AUTO_INCREMENT en MySQL.

En la Figura 1 se muestra la nueva interfaz simplificada del gestor de tareas. El código HTML correspondiente debe construirse a partir de la página HTML desarrollada en entregas anteriores (se llamará **tasks.html**). Este fichero debe hacer referencia, mediante una etiqueta <script>, a un fichero **index.js** que contendrá el código a ejecutar en el lado del cliente. La funcionalidad en el lado del servidor se implementará en el fichero **app.js**.

Desarrollo en el lado del servidor: app.js

El servidor proporciona tres servicios al navegador:

Servicio 1: Devolver todas las entradas contenidas en la lista de tareas:
Método: **GET**
URL: **/tasks**
Parámetros de entrada: Ninguno
Códigos de respuesta: **200** (OK)
Tipo de resultado: JSON
Resultado: Array con las tareas de la lista. Cada tarea es un objeto con dos atributos: **id** y **text**.

Servicio 2: Añadir una nueva entrada a la lista de tareas.
Método: **POST**
URL: **/tasks**
Parámetros de entrada: Un objeto JSON con un único atributo (**text**), que contendrá el texto de la tarea a insertar.
Códigos de respuesta: **200** (OK)
Tipo de resultado: JSON
Resultado: Objeto tarea insertado, con dos atributos: **id** y **text**. El identificador de la tarea será generado automáticamente por el servidor. El texto será el mismo que el especificado en el parámetro de entrada.

Servicio 3: Eliminar una entrada a la lista de tareas.
Método: **DELETE**
URL paramétrica: **/tasks/:id**
Parámetros de entrada: El parámetro **:id** de la URL paramétrica contiene el identificador de la tarea a eliminar.
Códigos de respuesta: **200** (OK) si el **:id** indicado es un número, **404** (Not found) si no existe el elemento, o **400** (Bad request) si el **:id** indicado no es un número.
Tipos de resultado: JSON
Resultado: Un objeto vacío **{}**.

Antes de continuar con la siguiente apartado, es recomendable comprobar el funcionamiento de estas tres operaciones mediante un cliente REST como, por ejemplo, *Advanced REST Client*.

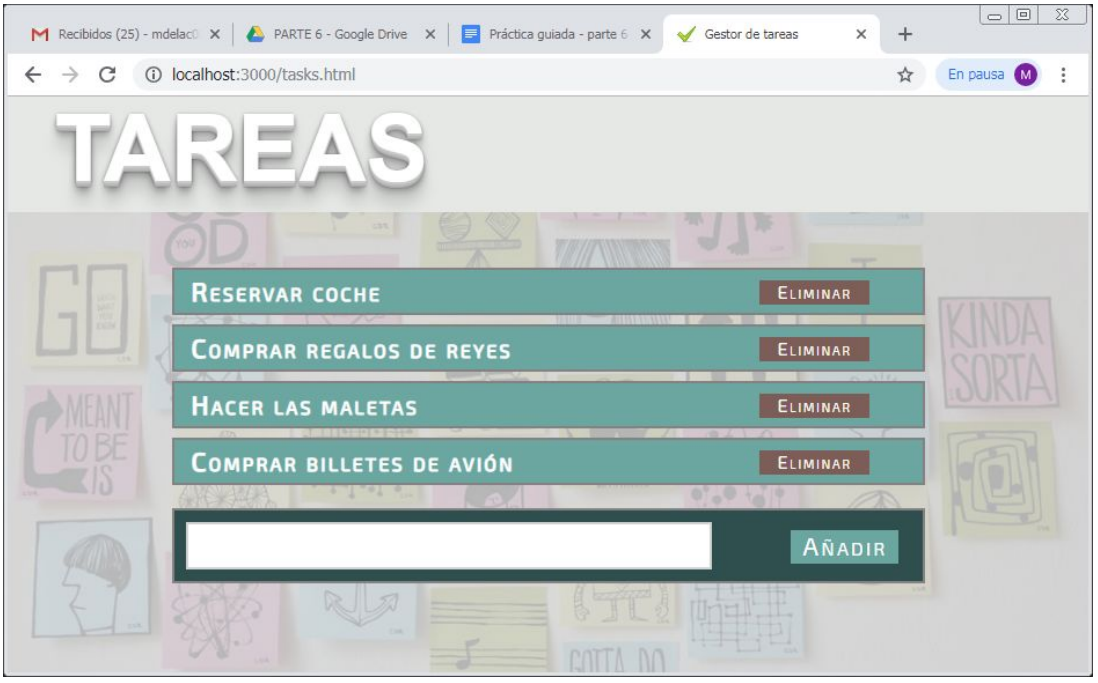

Figura 1

Desarrollo en el lado del cliente: public/js/index.js

El fichero **index.js** debe incluir las siguientes funciones:

- **taskToDOMElement(task)** A partir de un objeto con dos atributos (**id** y **text**) representando la información de una tarea, construye un elemento <x> con el contenido de dicha tarea y devuelve una selección al mismo. Este elemento contiene, además, un atributo personalizado con el identificador de la tarea. En esta descripción, <x> representa el elemento HTML que se haya utilizado para representar cada tarea en el fichero **tasks.html**.
- Manejador de inicialización del DOM, que se limita a llamar a la función **loadTasks()** y a asignar sendos manejadores de eventos a los botones **[Eliminar]** y al botón **[Añadir]** de la página.
- **loadTasks()** Solicita al servidor la lista de tareas (mediante una petición AJAX) e inserta cada una de ellas en el DOM para que se visualicen en la página.
- **onRemoveButtonClick(event)** Maneja los eventos de clic en un botón **[Eliminar]**. Debe enviar una petición AJAX para eliminar la tarea correspondiente del servidor y eliminar dicha tarea del DOM.
- **onAddButtonClick(event)** Maneja los eventos de clic en el botón **[Añadir]**. Debe obtener la cadena introducida en el cuadro de texto situado a la izquierda de dicho botón. Si esta cadena es no vacía, realizará una petición AJAX al servidor para insertar la tarea en la lista de tareas. Además, debe añadir el elemento del DOM correspondiente.

En estas dos últimas operaciones (eliminación e inserción), la actualización del DOM debe hacerse modificando únicamente los elementos involucrados, es decir, eliminando el <x> correspondiente en el primer caso, y añadiendo un nuevo <x> en el segundo caso. No se permite eliminar todos los <x> y llamar a **loadTasks()** para obtener la lista actualizada.

Entrega de la práctica

La fecha límite para entregar el trabajo es el 2018 (el enlace para la entrega en el campus virtual se cierra a las 23:55).

La entrega se realizará en el campus virtual y deberá contener en un fichero **pxx.zip** todos los elementos necesarios para la reconstrucción del proyecto (xx es el número de pareja de laboratorio).