Examen

Final septiembre 2017

Aplicaciones Web
Año 2016/2017
Grado en Ing. Software
Fdi - UCM

Instrucciones generales:

- La duración del examen es de **tres horas**. Su peso en la calificación de la asignatura es del 60 %. Para obtener la calificación de APTO es necesaria la nota mínima de 4 sobre 10.
- No se permite ningún tipo de material sobre la mesa, salvo un bolígrafo y este enunciado.
- El examen consta de tres ejercicios que se realizan en el ordenador. Para su realización se proporciona un proyecto plantilla. Tras el enunciado de cada ejercicio se indica qué ficheros hay que modificar o añadir.

▷ 1. Acceso a BD mediante *Node* y *Express*

[4.5 pt] Suponemos una base de datos que almacena las canciones contenidas en una lista de reproducción (*playlist*). Esta base contiene una única tabla (`playlist`) cuyo contenido es el siguiente:

id	title	author	album	year
1	Eternal Odyssey	Delerium	Chimera	2003
2	Somebody to love	Queen	A day at the races	1976
3	Believe	DB Boulevard	Frequencies	2004

(a). *[0.75 pt]* Implementa una función `getPlayList(callback)` que obtenga todas las canciones de la base de datos. Recibe como parámetro una función `callback` con dos parámetros: El primero de ellos sirve para propagar el objeto error (si lo hay) y el segundo parámetro, en caso de éxito, debe contener una lista de objetos, cada uno de ellos con cinco atributos: `id`, `title`, `author`, `album`, `year`.

(b). *[0.75 pt]* Implementa una función `insertInPlaylist(title, author, album, year, callback)` que inserte una fila en la tabla `playlist` con la información pasada como parámetro, y después llame a la función `callback` pasándole un objeto error si se produce algún fallo, o `null` en caso contrario.

(c). *[1.5 pt]* Diseña una aplicación *Express.js* que utilice el método del apartado (a) para visualizar el contenido de la tabla `playlist`. Para ello implementa el siguiente router,

```
app.get("/showPlaylist", function(request, response) {
  // ...
});
```

en el que se muestre una vista que contenga una tabla HTML (`<table>`) con la información de la tabla de la base de datos (Figura 1).

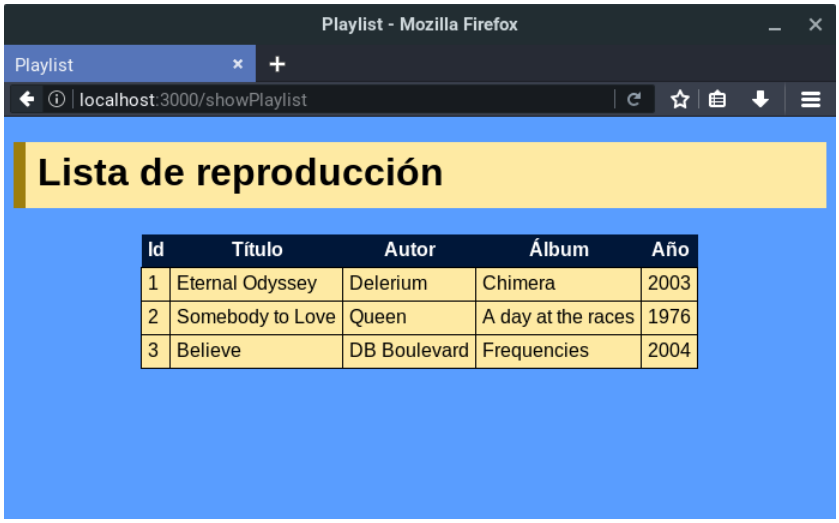

Figura 1: Visualización de la lista de reproducción.

(d). *[1.5 pt]* Supongamos que la ruta `/newEntry.html` muestra un formulario que contiene cuatro campos de texto para introducir el título, autor, álbum y año de una canción (Figura 2). Al pulsar el botón `Enviar` del formulario se realiza una petición POST a la ruta `/insertEntry`. Implementa el manejador de esta ruta para que se inserte la información introducida en el formulario en la base de datos:

```
app.post("/insertEntry", function(request, response) {
  // ...
});
```

En caso de éxito se debe redirigir a la ruta `/showPlaylist`. Si alguno de los campos del formulario está vacío, o el campo año no contiene un número, se debe volver a la página del formulario, introduciendo antes de este la cadena `Información incorrecta` (Figura 3).

Instrucciones para el ejercicio 1

- Para realizar los apartados (a) y (b) completa el fichero `dao.js` que se encuentra dentro del proyecto plantilla.
- Dentro del proyecto se encuentra un fichero `dao_test.js`, que hace uso de las dos funciones especificadas en los apartados (a) y (b), y que te permitirá comprobar la corrección de tu solución. Puedes ejecutar este fichero en *Netbeans* mediante *Shift+F6*.
- Completa los apartados (c) y (d) en el fichero `ej1.js`. Los ficheros `plantillaNewEntry.html` y `plantillaShowPlaylist.html` en el directorio `public` pueden ser de ayuda para la creación de las plantillas EJS correspondientes.

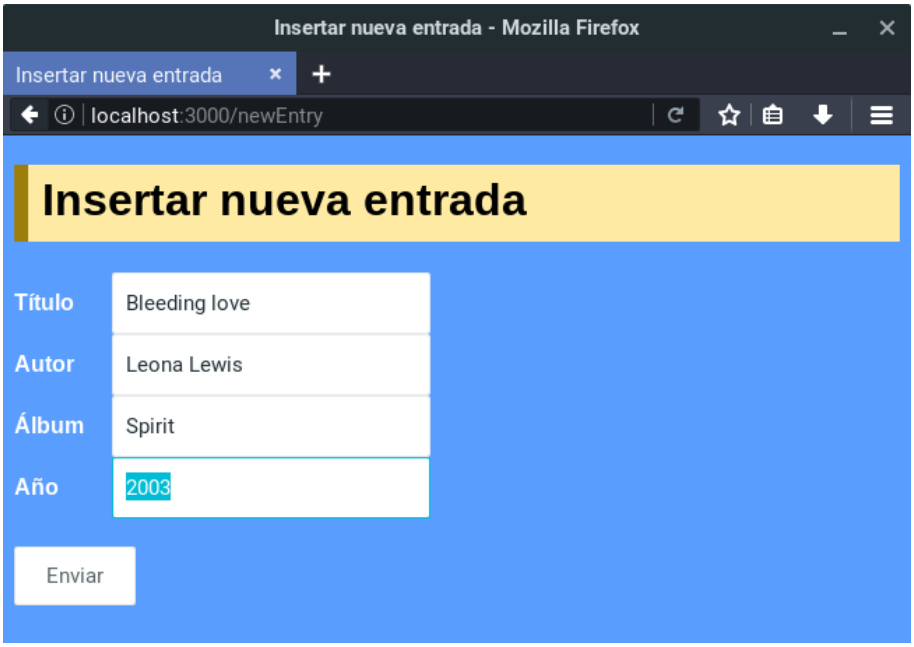

Figura 2: Inserción de una nueva entrada en lista de reproducción.

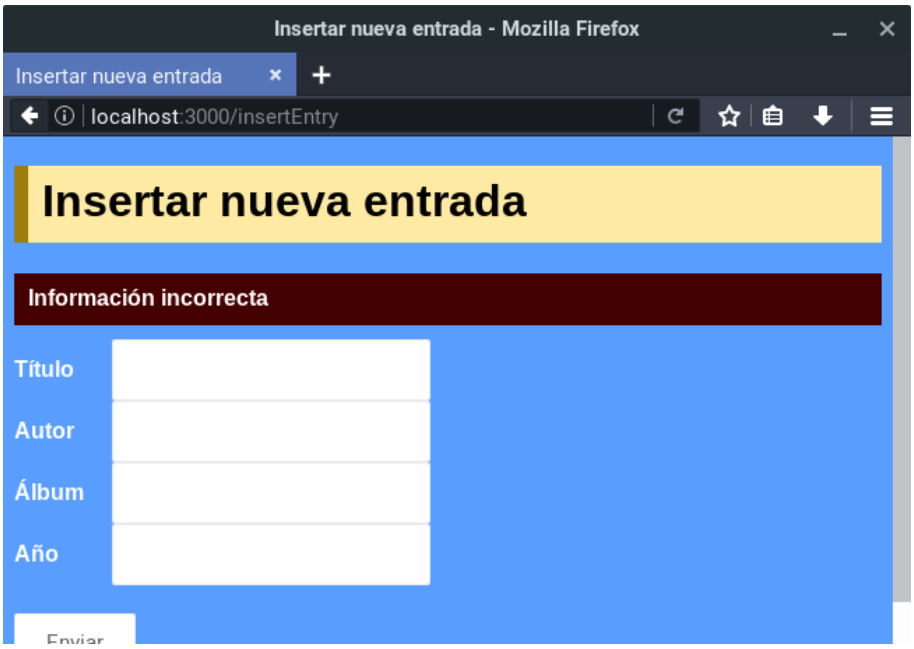

Figura 3: Mensaje de error al introducir información no válida.

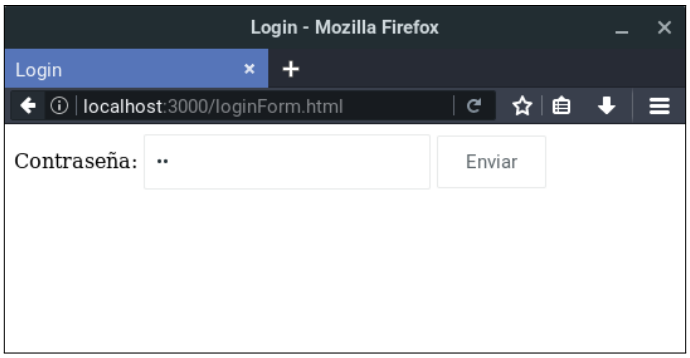

Figura 4: Introducción de contraseña.

▷ 2. Manejo de *cookies* en *Express*

[2.5 pt] Suponiendo una aplicación *Express.js*:

(a). *[1 pt]* Escribe un manejador de ruta "/" que muestre una página con el texto "`Has visitado esta página XX veces`", donde `XX` es un contador que se incrementa cada vez que el usuario accede a dicha ruta. Utiliza *cookies* para almacenar el valor del contador y una plantilla EJS para mostrar este valor al usuario.

(b). *[1 pt]* Modifica la ruta anterior para que solo se permita acceder al valor del contador si el usuario ha introducido previamente la contraseña "`aw`" en un formulario como el de la Figura 4. Si la contraseña es correcta, se inicializará el contador de visitas a cero y se mostrará dicho contador. Si la contraseña no es correcta, se redirigirá a una página `wrongPassword.html` indicándolo. Si el usuario intenta acceder al contador mediante la ruta "/" sin haber introducido la contraseña previamente, se le redirigirá al formulario de introducción de contraseña. Para realizar este ejercicio implementa un *middleware* que compruebe si el usuario ha introducido la contraseña previamente, y redirija al formulario de introducción de contraseña en caso contrario.

(c). *[0.5 pt]* Añade a la página del contador un enlace `[Desconectar]` que, al ser pulsado, elimine la información sobre la contraseña introducida previamente por el usuario, de modo que este último tenga que volver al formulario de la Figura 4 para acceder al contador.

Instrucciones para el ejercicio 2

- Modifica el fichero `ej2.js` para realizar este ejercicio.
- En la carpeta `public` tienes los ficheros HTML necesarios para la realización de este ejercicio. El fichero `plantillaShowCounter.html` puede servirte como base para la plantilla EJS correspondiente.

▷ 3. AJAX y servicios web

[3 pt] Partimos de una página como la de la Figura 5. Esta página contiene dos elementos `<div>` que contienen, respectivamente, una lista de productos disponibles (izquierda) y un carro de la compra inicialmente vacío (derecha).

Figura 5: Carro de la compra.

```
<div id="stockD">
    <ul id="stock">
        <li>Aceite</li>
        <li>Huevos</li>
        ...
    </ul>
</div>
<div id="carroD">
    <ul id="carro">
    </ul>
</div>
```

(a). *[1.5 pt]* Utilizando *jQuery* añade el manejador de eventos necesario para que al hacer clic en uno de los `<li>` de la lista de productos disponibles, se elimine el elemento seleccionado de la lista de productos disponibles y se añada a la lista del carro de la compra.

(b). *[1.5 pt]* Modifica la página anterior para que la lista inicial de productos disponibles se obtenga mediante una petición AJAX al servidor mediante la ruta `/stock`. Implementa dicha ruta en el servidor suponiendo que los productos están almacenados en el siguiente array:

```
var stock = ["Aceite", "Galletas", "Pan", "Cebolla", "Cereales", "Huevos",
             "Tomate", "Mantequilla", "Leche", "Café"];
```

Tras realizar la petición AJAX, se debe rellenar el `<ul>` de la lista de productos disponibles a partir del resultado devuelto por el servidor.

Instrucciones para el ejercicio 3

- Para realizar este ejercicio arranca el fichero `ej3.js` y accede a la siguiente ruta: `http://localhost:3000/ej3.html`.
- Modifica el fichero `static/ej3_client.js` para realizar el apartado (a).
- Modifica los ficheros `static/ej3_client.js` y `ej3.js` para realizar el apartado (b).

Para entregar:

☐ Comprueba que has rellenado el fichero `Alumno.txt`.

☐ Crea un fichero comprimido (`.zip`) con el proyecto `ExamenSeptiembre`.

☐ El nombre del fichero tiene que ser de la forma *DNI_Apellido*`.zip`. Incluye la letra del `DNI` y solamente el primer apellido.

☐ Entrega el fichero `.zip` utilizando el enlace del escritorio *Examenes en LABs - Entregas*. En la carpeta que aparece, seleccionar *ALUMNOS entrega de prácticas y exámenes* y subir el `.zip`.

☐ Antes de cerrar sesión, dirígete al puesto del profesor para comprobar que el fichero se ha subido correctamente.