Examen

Final febrero 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 cuatro 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.js*

[2 pt] En la Figura 1 se muestra el diseño relacional de una base de datos que almacena artículos de revista. Cada artículo tiene asociado un identificador numérico, un título, una fecha de publicación y una lista de palabras clave, que se encuentran en una tabla separada (`palabrasclave`).

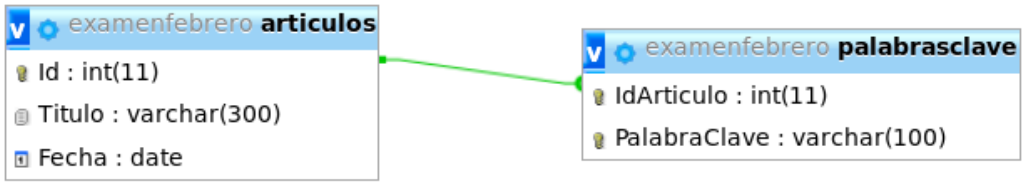

Figura 1: Diseño de la base de datos de artículos y palabras clave

Implementa una función `leerArticulos(callback)` que obtenga todos los artículos de la base de datos. Esta función debe construir un **array de objetos**, cada uno de ellos representando la información de un artículo mediante cuatro atributos: `id` (númerico), `titulo` (cadena de texto), `fecha` (objeto `Date`) y `palabrasClave` (array de cadenas). Por ejemplo:

```
{
  id: 1,
  titulo: "An inference algorithm for guaranteeing Safe destruction",
  fecha: 2008-07-19,  // ⟵ como objeto de la clase Date
  palabrasClave: [ 'formal', 'inference', 'memory' ]
}
```

La función `callback` pasada como parámetro a `leerArticulos` funciona de igual modo que las vistas en clase. Recibe dos parámetros: un objeto con información de error (en caso de producirse), y la lista con los artículos recuperados de la base de datos.

Instrucciones para el ejercicio 1

- Para realizar este ejercicio completa el fichero `ejercicio1.js` que se encuentra dentro del proyecto plantilla.
- Puedes ejecutar este mismo fichero para comprobar si la respuesta es correcta *[Shift+F6]*.

▷ 2. Aplicaciones web mediante *Express.js*

[2 pt] La constante matemática $e \approx 2{,}718281828459\ldots$ es un número irracional que puede aproximarse mediante la siguente expresión (donde $n \geq 0$):

$$e \approx \sum_{i=0}^{n} \frac{1}{i!} = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!}$$

donde la notación $i!$ indica "factorial de i". La aproximación es más precisa cuanto mayor sea el valor de n. Implementa una aplicación web que solicite el valor de n al usuario y calcule una aproximación de e mediante la fórmula anterior (Figura 2).

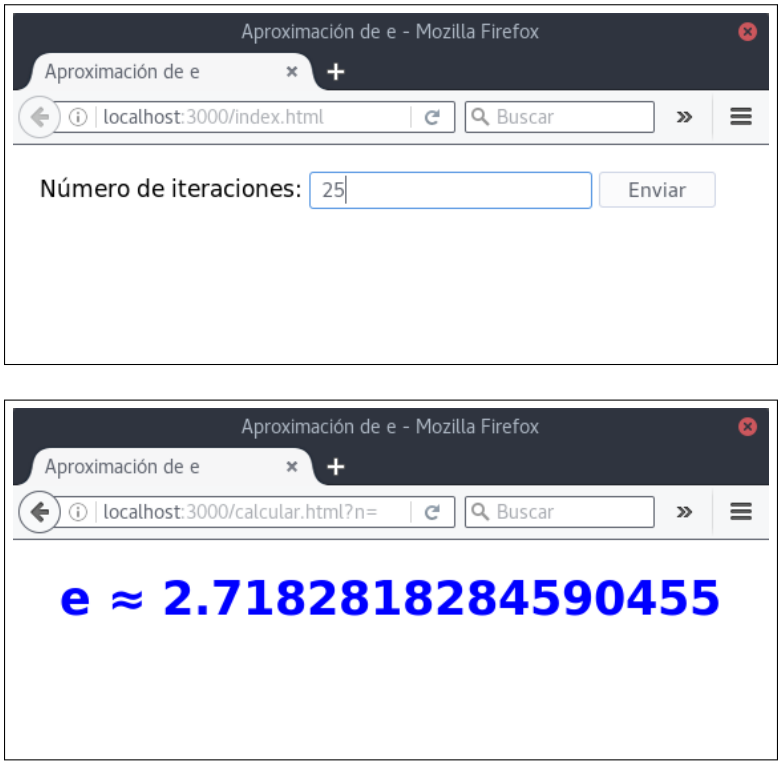

Figura 2: Funcionamiento de la aplicación para el cálculo de e

Si el usuario no ha introducido un número, se volverá a mostrar el formulario junto con un mensaje de error (Figura 3).

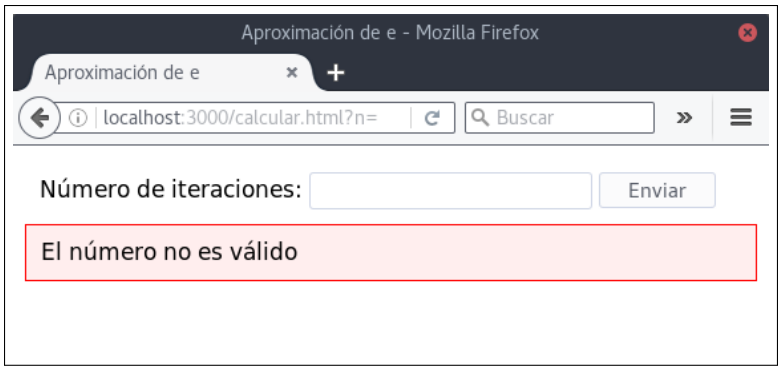

Figura 3: Mensaje de error en el formulario

Instrucciones para el ejercicio 2

- Realiza este ejercicio modificando el fichero `ejercicio2.js`.
- Añade las plantillas EJS necesarias en la carpeta `views`. Los ficheros `public/_calculoE.html` y `public/_result.html` pueden servirte como modelo para la elaboración de las plantillas EJS.
- Si quieres utilizar *nodemon*, sitúate con una línea de comandos en el directorio del proyecto y teclea: "`node node_modules/.bin/nodemon ejercicio2.js`"
- No puede utilizarse *jQuery* ni *AJAX* en este ejercicio.

▷ 3. Uso de sesiones en *Express.js*

[3.5 pt] El objetivo de este ejercicio es la realización de una aplicación web en la que el usuario pueda introducir una lista de tareas pendientes y marcar aquellas que ha finalizado (Figura 4). Las tareas han de guardarse **en la sesión del navegador** utilizando el *middleware* `express-session`.

(a). *[1.5 pt]* Implementa un manejador de ruta `/index.html` que muestre todas las tareas introducidas por el usuario, junto con un formulario invitando al usuario a introducir una nueva tarea. Implementa otro manejador para la ruta `/anyadirTarea` que recoja la información introducida en el formulario y añada la tarea introducida a la lista. De momento puedes ignorar la línea `Número de tareas pendientes` que se muestra en la Figura 4 y el enlace `Marcar como finalizada` mostrado a la derecha de cada enlace.

(b). *[1 pt]* Modifica la página anterior para que se muestre un enlace `[Marcar como finalizada]` a la derecha de cada tarea. Al hacer clic en ese enlace, la tarea se marcará como finalizada, y se mostrará de nuevo la página con la lista de tareas. Las tareas marcadas como finalizadas aparecerán tachadas y sin el enlace en el lado derecho (Figura 4).

(c). *[1 pt]* Añade un *middleware* a la aplicación anterior para que calcule el número de tareas de la lista que no hayan sido finalizadas y lo asigne a la variable `response.locals.numPendientes`. Modifica la página para que también muestre la lista de tareas pendientes.

Figura 4: Aplicación de lista de tareas

Instrucciones para el ejercicio 3

- Para realizar el ejercicio modifica el fichero `ejercicio3.js` del proyecto plantilla.
- Recuerda: las tareas se almacenan en la sesión del navegador, no en la base de datos. Además, no puede utilizarse *jQuery* ni *AJAX* en este ejercicio.
- Añade las plantillas EJS necesarias en la carpeta `views`. El fichero `public/_tareas.html` puede resultarte de ayuda para elaborar las plantillas.

▷ 4. AJAX y servicios web

[2.5 pt] Supongamos un juego en línea que almacena un array con las puntuaciones obtenidas por cada jugador, ordenadas de mayor a menor puntuación:

```
var records = [
    { nombre: "Fran", puntos: 955 },
    { nombre: "Rafael", puntos: 865 },
    { nombre: "Carmen", puntos: 563 },
    { nombre: "Rosario", puntos: 534 },
    { nombre: "Juan", puntos: 234 },
    { nombre: "Estela", puntos: 107 },
    ...
];
```

Figura 5: Tabla de puntuaciones más altas

(a). *[0.5 pt]* Implementa un servicio web que devuelva un JSON con los cinco primeros objetos (es decir, los de más puntuación) del array `records`. En caso de que el array contenga menos de cinco objetos, se devolverá todo el array.

Método: `GET`

URL: `/highestRecords`

Parámetros de entrada: Ninguno

Código de respuesta: `200` (OK).

Tipo resultado: JSON con un array de objetos. Cada uno de ellos tiene dos propiedades: `nombre` y `puntos`.

Resultado: Devuelve las cinco personas que han obtenido una puntuación más alta. Para cada una de ellas, se devuelve su nombre y su puntuación.

(b). *[1.5 pt]* Supongamos una página como la de la Figura 5, en la que se muestra una tabla con las cinco mejores puntuaciones. Escribe una función `actualizarLista()` para que, mediante `jQuery` y una petición AJAX al servicio implementado en el apartado anterior, actualice dicha tabla con los datos devueltos por la petición AJAX.

Indicación: La sentencia `$(selector).empty();` elimina del DOM todos los hijos que se encuentren debajo del elemento indicado por el `selector` dado.

(c). *[0.5 pt]* Suponemos que añadimos a la página anterior un formulario invitando al usuario introducir un nombre, junto con un botón `[Enviar]`. Modifica la aplicación para que, cuando se pulse dicho botón, se añada el usuario a la tabla de records con una puntuación aleatoria entre 1 y 1000 y actualice la tabla de records. Para ello puedes suponer implementado el siguiente servicio:

Método: `POST`

URL: `/newRecord`

Parámetros de entrada: En el cuerpo de la petición POST, un objeto JSON con un único atributo (llamado "`nombre`") que contenga el nombre introducido en el formulario.

Código de respuesta: `201` (*Created*).

Tipo resultado: Ninguno.

Resultado: Ninguno.

Instrucciones para el ejercicio 4

- Realiza el apartado (a) en el fichero `ejercicio4.js`.
- La página mostrada en la Figura 5 se encuentra en el fichero `public/records.html`. Completa los apartados (b) y (c) en el fichero `public/js/records.js`.

Para entregar:

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

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

☐ 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.