Práctica 3. Aplicación web en el servidor

TITULACIÓN: Grado en Ingeniería de tecnologías de telecomunicación (14312020)

TITULACIÓN: Doble Grado Ing. de tecnologías de la telecomunicación e Ing. telemática (15212007)

TITULACIÓN: Grado en Ingeniería telemática (14512016)

CENTRO: ESCUELA POLITÉCNICA SUPERIOR (LINARES)

CURSO ACADÉMICO: 2021-2022

Descripción de las tareas a realizar

En esta práctica se deberán desarrollar algunos de los servicios consumidos en la práctica 2 sobre Node.js para dar soporte a la gestión de los exámenes. El profesor aportará un código de base a partir del cual realizar la aplicación.

Las características que deberá cumplir la aplicación servidora sobre Node.js serán las siguientes:

1. Soporte empleando Express al servicio `/exam` con los métodos PUT (crear nuevo examen).
2. Soporte empleando Express al servicio `/exam/id` con el método DELETE para borrar un examen.
3. Soporte empleando Express al servicio `/exam/user` con el método GET para mostrar solamente los exámenes del usuario “user”. La salida se deberá mostrar en un HTML generado basándose en una plantilla Mustache con el mismo aspecto que el resto de páginas de la práctica 2.
4. Soporte para el acceso a datos empleando MongoDB.

Tareas

Tarea previa. Instalación y configuración de Node.js y MongoDB.

Previo al inicio de la práctica, se deberá tener **descargado e instalado Node.js** (incluye NPM) y **la versión Community de MongoDB**. A continuación, se detallan los pasos a seguir.

Instalación de Node.js

Node.js está disponible en varias versiones en https://nodejs.org/en/download/. También está disponible a través de gestores de paquetes como Homebrew en macOS (más en https://nodejs.org/en/download/package-manager/).

Una vez instalado, tendremos disponible en el intérprete de comandos (*shell*) el comando `node`.

Para ejecutar un script con **Node.js**, simplemente se ejecuta:

```
node nombre-script.js
```

Instalación y uso de npm

El gestor **npm** es instalado junto con Node.js. Se puede comprobar que está instalado ejecutando el comando `npm -v`, que mostrará la versión disponible si se instaló correctamente.

Una vez instalado, para actualizar **npm** a nuevas versiones se puede ejecutar:

```
npm install npm@latest -g.
```

Actualizar módulos con npm

Con `npm outdated`, ejecutado en el directorio de la aplicación, se muestran los paquetes que están desactualizados.

Para actualizar los módulos desfasados: ejecutar en el directorio donde está la aplicación `npm update` y así los módulos que tengan disponible una nueva versión se descargarán.

Vea: https://docs.npmjs.com/updating-packages-downloaded-from-the-registry

Instalación y uso de MongoDB

Para instalar la **versión Community** de MongoDB, en función de su sistema operativo, deberá seguir las indicaciones que se encuentran en:

https://docs.mongodb.com/manual/administration/install-community/

Al instalar MongoDB también se instala la **aplicación MongoDBCompass** que permite gestionar a través de su interfaz de usuario las bases de datos locales, facilitando así la creación de datos de inicio y borrado de registros.

Iniciar una base de datos

El servicio de MongoDB se instala en el sistema operativo y **no es necesario iniciarlo**, pero si se quiere iniciar el servidor MongoDB en un directorio concreto y con una carpeta para la base de datos concreta:

```
 "C:\Program Files\MongoDB\Server\4.2\bin\"mongod --dbpath
.\data\db
```

La ruta hasta la instalación puede ser esa o similar. También se puede agregar **mongod** al *path* del sistema operativo. La ruta para los ficheros de la base de datos (por ejemplo **".\data\db**) se puede elegir por el administrador según su criterio.

Para iniciar MongoDB en una dirección IP concreta:

```
"C:\Program Files\MongoDB\Server\4.2\bin\"mongod --dbpath
.\data\db --bind_ip 192.168.1.160
```

Nota: con MongoDBCompass nos podremos conectar al servidor de MongoDB y gestionar la base de datos para crear bases de datos, crear, editar y borrar registros manualmente.

Tarea 1. Creación del proyecto Node.js e instalación de módulos y dependencias

1. Abrir una ventana del intérprete de comandos (ejecutar “cmd” en Windows).
    a. Navegar a la carpeta donde se desea realizar la práctica.
    b. Teclear: **`npm init`** y completar la información que se requiere. Como fichero de aplicación elija como nombre **`app.js`**. Esto creará el fichero de configuración **`package.json`**.
2. En la misma ventana del intérprete de comandos instalar los módulos necesarios (con **`--save`** actualiza la configuración de **`package.json`** y los módulos se copian en una nueva carpeta **`node_modules`** y se crea también un nuevo archivo **`package-lock.json`** con toda la información de los módulos instalados):
    a. Instalar Express: **`npm install express –save`**
    b. Instalar controlador para MongoDB: **`npm install mongodb --save`**
    c. Instalar soporte a las plantillas Mustache:
        i. **`npm install mustache –g --save`**
        ii. **`npm install mustache-express –g --save`**
3. Descomprimir, en el mismo directorio creado, el fichero zip proporcionado para la práctica.

Tarea 2. Copiar el código de la práctica 2 en el nuevo proyecto

Para que las peticiones de la aplicación sean dentro del mismo dominio, copiar todo el contenido de la práctica 2 a una nueva carpeta **`/public`** del nuevo proyecto. Reemplace cualquier fichero allí existente y actualice, si fuera necesario las URL de los servicios incluidos en el index.html de la práctica 2:

- Las peticiones se deben quedar del tipo "`/login`" o "`/exam`".

Recuerde que si intenta acceder a los servicios de la práctica 3 desde otra IP o puerto deberá tener configurado el CORS en el servidor (ver Tema 4).

Tarea 3: servicio PUT /exam

A partir del código facilitado, crear un servicio para recibir los datos enviados desde el cliente en el formato especificado en la práctica 2. El servicio está definido según las siguientes características:

- Método a emplear: PUT.
- URL del servicio: `/exam`.
- Datos: los datos que se deberán recibir en formato JSON en el servidor son el usuario que lo crea ("`user`") y el título ("`title`") del examen, la fecha de celebración del examen (“date”) repositorio. Ejemplo del JSON:

```
{
     "user": "user1",
     "title": "Tema 1",
     "date": "2020-25-02",
     "repository": " 5ec545d80dbb5c60c04ea"
}
```

- Respuestas que debe dar el servidor al cliente:
  - 201. En el caso de recibir los datos correctamente y después de insertar el repositorio correctamente en la base de datos, se enviará al cliente una respuesta 200.
  - Tipo de datos: `application/json`.
  - 400. Se enviará esta respuesta si el formato recibido es incorrecto.
  - 500. Se enviará esta respuesta si ha habido un error al guardar el repositorio en la base de datos.

Pasos:

1. Crear la entrada en la aplicación express para el método PUT /exam.
2. Añadir los eventos para la recepción de datos desde el cliente.
3. Hacer la conexión con la base de datos MongoDB. La base de datos se deberá llamar “examenes” y la colección “exams”.
4. Añadir un elemento a la base de datos.
5. Responder al cliente con el resultado de la operación.

Tarea 4: servicio DELETE /exam/id

Este servicio deberá ser creado de manera autónoma por el alumno.

El servicio **`DELETE /exam/id`** buscará en la base de datos un repositorio concreto, el especificado en la ruta "id", y lo eliminará. El servicio está definido según las siguientes características:

- Método a emplear: DELETE.
- URL del servicio: `/exam/id`.
- Datos: no se envían datos.
- Respuestas que debe dar el servidor al cliente:
  - 200. En el caso de recibir una petición correcta y después de borrar el repositorio.
  - 404. En el caso de que el documento no se encuentre en la base de datos.
  - 500. Se enviará esta respuesta si ha habido un error al borrar el repositorio de la base de datos.
- **Nota: recuerde especificar la ruta como /exam:id para obtener el valor de id con req.params.id.**

Pasos:

1. Crear la entrada en la aplicación express para el método DELETE /exam/:id.
6. Hacer la conexión con la base de datos MongoDB. La base de datos se deberá llamar “examenes” y la colección “exams”.
2. Borrar el documento de la base de datos cuyo _id coincida con el pasado en el servicio.
3. Responder al cliente con el JSON resultado de la operación.

Tarea 5: servicio GET /exam/*user*

Generar una salida HTML, empleando una plantilla Mustache para mostrar la lista de exámenes directamente en HTML, pero solamente los de un usuario.

Este servicio no hace falta que se integre en la página creada para la práctica 2, se podrá probar directamente a través de una pestaña del navegador, por ejemplo con: http://localhost:8083/exam/usuario.

Para mostrar todos los exámenes, se deberá emplear una estructura en HTML basado en listas no ordenadas. Además, se deberán añadir, al menos, un estilo con tres propiedades para los elementos del listado.

Pasos:

1. Crear la entrada en la aplicación express para el método GET /exam/user.
2. Hacer la conexión con la base de datos MongoDB. La base de datos se deberá llamar "examenes" y la colección "exams".
3. Buscar todos los documentos en la base de datos. Para que el motor de plantilla Mustache pueda recorrer los elementos, el resultado de la búsqueda se debe asignar a una propiedad nueva en un objeto JSON: {"exams":*result*}.
4. Añadir el soporte a los motores de plantilla a la aplicación.
5. Crear una plantilla HTML con Mustache en una nueva carpeta /views.
6. Pasar al motor de plantillas (render) los datos y la plantilla y devolver el htnl resultante al cliente.

Tarea 6: Soporte en la aplicación (front-end) del servicio DELETE /exam/id

Añadir en la página index.html de la práctica 2, en la sección de listado de exámenes, la funcionalidad necesaria para que, a través de un botón para cada entrada de la lista mostrada, se haga una petición al servicio `DELETE /exam/id` (Tarea 4) y se pueda borrar un examen.

Además, si se recibe una respuesta correcta del servicio, eliminar ese elemento del DOM.

Competencias a desarrollar

La realización del presente ejercicio intenta desarrollar las siguientes competencias:

- **C.7. Conocimiento y utilización de los fundamentos de la programación en redes, sistemas y servicios de telecomunicación.**
  - *En esta práctica se profundiza en la programación de servicios en el ámbito del a World Wide Web, concretamente en la creación de aplicaciones en el servidor con la tecnología Node.js que permite aprovechar los conocimientos previos adquiridos de JavaScript además del manejo de datos con bases de datos no-SQL como es MongoDB.*
- **CB.4. Que los estudiantes puedan transmitir información, ideas, problemas y soluciones a un público tanto especializado como no especializado.**
  - *Se busca desarrollar esta competencia a través de la interacción con el alumno para que comente sus desarrollos dentro de la parte de validación de su trabajo.*
- **CG.3. Conocimiento de materias básicas y tecnologías, que le capacite para el aprendizaje de nuevos métodos y tecnologías, así como que le dote de una gran versatilidad para adaptarse a nuevas situaciones.**

  - *Con esta práctica se busca que el alumnado complemente su conocimiento en el desarrollo de servicios y aplicaciones telemáticos a partir de información técnica, como la mostrada en los seminarios, bibliografía, así como la disponible en portales especializados en Internet. Con esto además se busca lo siguiente:*
    - *Afianzamiento del conocimiento del lenguaje HTML.*
    - *Afianzamiento del conocimiento del lenguaje CSS.*
    - *Afianzamiento del conocimiento de la sintaxis de transferencia JSON.*
    - *Afianzamiento del conocimiento del lenguaje JavaScript.*
    - *Afianzamiento del conocimiento del DOM.*
    - *Familiarización con el desarrollo de aplicaciones de servidor con Node.js.*
    - *Familiarización con el gestor de paquetes npm.*
    - *Familiarización con Express sobre Node.js.*
    - *Familiarización con MongoDB sobre Node.js.*
    - *Familiarización con el uso de plantillas Mustache sobre Node.js.*
    - *Afianzamiento del conocimiento del entorno de desarrollo de aplicaciones usado en el curso.*
    - *Afianzamiento del uso de los recursos de referencia de Internet.*
- **CG.9. Capacidad de trabajar en un grupo multidisciplinar y en un entorno multilingüe y de comunicar, tanto por escrito como de forma oral, conocimientos, procedimientos, resultados e ideas relacionadas con las telecomunicaciones y la electrónica.**
  - *Esta competencia está en sí contenida en todas las tareas dado que la mayoría de las fuentes estarán en inglés, buscado potenciar el desarrollo en la comprensión de este idioma. Asimismo, a través de los comentarios del código se busca que el alumno sea capaz de sintetizar y explicar los objetivos de su trabajo de forma clara para que puedan ser entendidos por otros.*

Resultados de aprendizaje

La realización del presente ejercicio persigue la consecución de los siguientes resultados:

- Resul-05. Fomentar los trabajos grupales, así como la **transmisión de procedimientos, resultados e ideas** en el ámbito de las telecomunicaciones.
- Resul-10. El alumno distinguirá y relacionará los **conceptos de servicio y aplicación en el ámbito de las comunicaciones**; además asimilará los detalles vinculados con los **servicios de telecomunicación más populares y las aplicaciones y protocolos que los soportan**. Describir los principales servicios y aplicaciones de Internet.
- Resul-21. **Conocer materias básicas y tecnológicas** que capaciten para el **aprendizaje de nuevos métodos y tecnologías**
- Resul-22. Dotar de una gran **versatilidad para adaptarse a nuevas situaciones**.
- Resul-23. **Analizar y valorar el impacto social y medioambiental** de las soluciones técnicas.

- Resul-24. Trabajar en un grupo multidisciplinar y en un **entorno multilingüe**.
- Resul-25. Comunicar, tanto por escrito como **de forma oral**, **conocimientos, procedimientos, resultados e ideas relacionadas con las telecomunicaciones** y la electrónica

Detalles de la práctica.

Recursos necesarios

- Ordenador PC.
- Conexión a Internet.
- Recursos: transparencias de clase.
- Recursos online: W3schools, Mozilla Developer Network o Stack Overflow, entre otros.
- Entorno de desarrollo: Visual Studio Code.
- Navegador Mozilla Firefox o Google Chrome.
- Servidor Node.js.
- Gestor de bases de datos MongoDB.
- Gestor de paquetes npm.

Programación de la práctica

Esta práctica se realizará en cuatro sesiones durante otras cuatro semanas.

- Trabajo en laboratorio: 8 horas.
- Trabajo autónomo del alumno: 12.

Sesiones en laboratorio:

- Primera sesión. Preparación de una aplicación.
  - Completar la tarea preliminar, así como la tarea 1 y 2.
- Segunda sesión. Desarrollar servicios de la Tarea 3 y 4.
- Tercera sesión: Desarrollar servicios de la Tarea 5.
- Cuarta sesión: Modificar la vista para completar la Tarea 6 y prueba de la práctica en conjunto.

Trabajo autónomo:

- Repaso de tecnología AJAX: 1 hora.
- Repaso de Node.js y expess: 1 hora.
- Repaso de MongoDB: 1 hora.
- Revisar estructura del proyecto Node.js creado: 1 hora.
- Revisar la especificación de los servicios: 1 hora.
- Revisar y probar la Tarea 3: 1 hora.
- Revisar y probar la Tarea 4: 1 hora.
- Revisar y probar la Tarea 5: 1 hora.
- Revisar y probar la Tarea 6: 1 hora.
- Revisar y probar la práctica: 3 horas.

Material a entregar en la plataforma de docencia de la UJA

Un fichero comprimido con los archivos generados por la práctica para la aplicación Node.js, con la misma estructura que en el servidor, para todos los ficheros generados en todas las tareas, pero **sin incluir la carpeta node_modules** o cualquier otra creada por el entorno de desarrollo.

- El nombre del fichero deberá comenzar por **práctica3_20212022_** y estar seguido por el usuario del autor o autores del fichero. Ejemplo: **práctica3_20212022_alumno1** o **práctica3_20212022_alumno1_alumno2** para dos autores.
- **No entregue las carpetas dedicada a guardar información del proyecto del entorno de desarrollo**, solamente las carpetas que tengan ficheros js, html y css (public) y vistas (view).
- El código JS **deberá estar comentado** y las Tareas identificadas claramente en él.

Nota: En el caso de que el ejercicio se realice por parejas, ambos alumnos deberán entregar el mismo material por separado en la plataforma de docencia.

Evaluación

Después de la entrega de la práctica 4 se realizará **una prueba de evaluación práctica** para evaluar la adquisición de las competencias y resultados de ambas prácticas. Esta prueba será práctica, y se deberá desarrollar una página web y un servicio con la funcionalidad que se describa en el enunciado.

La calificación de esta prueba será el 50% de la calificación del aspecto de S4-Prácticas de laboratorio.

Es obligatorio entregar todo el material descrito en la práctica, cumpliendo con los requisitos establecidos, para que la prueba de evaluación sea tenida en cuenta.

El no cumplir con los requisitos de entrega de la práctica podrá conllevar penalizaciones en la prueba de evaluación de hasta 5,0 puntos.

Téngase en cuenta que el plagio, o la copia, conllevará la calificación de 0 en todo el aspecto de S4-Prácticas de laboratorio, además de la posible aplicación de las medidas previstas en la normativa de fraude en la evaluación.