Simulacro de examen

Duración: 1 hora y 30 minutos

Instrucciones Generales:

1. Crea un proyecto llamado **GestorDeServicios**. Configura el proyecto asegurándote de que **no tiene repositorio remoto (git repository)**.
2. Desarrolla el código únicamente en el archivo app.js. Si necesitas otras carpetas o archivos (vistas, archivos estáticos, etc.), organízalos adecuadamente dentro del proyecto.
3. Descarga e instala los paquetes necesarios para completar las funcionalidades de cada ejercicio. Es tu decisión determinar cuáles son esenciales.
4. Utiliza express-generator para generar la estructura de carpetas del proyecto.
5. Configura la carpeta public para los archivos estáticos y asegúrate de que se pueda cargar contenido en el navegador desde ahí.
6. Configura la carpeta views para las vistas de la aplicación y asegúrate de configurarlas de tipo ejs.

Ejercicio 1: Configuración básica y página de inicio

1. Configura un servidor con Express.js que escuche en el puerto 3500.
2. Define un middleware para registrar en la consola las peticiones que llegan al servidor (ruta, método, y hora de la petición).
3. Crea un gestor de rutas específico para la página de inicio (/).
    - Esta página debe mostrar un formulario con campos básicos para añadir un nuevo servicio.
    - Los campos mínimos son: nombre del servicio, descripción breve y precio.
4. Usa una plantilla o un archivo estático para diseñar la página de inicio. Personaliza el diseño utilizando estilos de Bootstrap.

- El middleware debe registrar cada acceso a cualquier ruta del servidor.
- La página inicial debe mostrarse correctamente y contener un formulario funcional.

Ejercicio 2: Gestión de servicios

1. Implementa una ruta POST para procesar el formulario de la página inicial. Al enviar los datos:
    - Los servicios enviados deben almacenarse en una estructura de datos en memoria (por ejemplo, un array).
    - Devuelve una respuesta con un mensaje confirmando la creación del servicio y muestra los datos enviados.
2. Define un middleware para validar los datos del formulario:
    - El nombre debe ser obligatorio y tener al menos 3 caracteres.
    - El precio debe ser un número mayor a 0.
    - Si los datos no son válidos, muestra un mensaje de error en la respuesta y no proceses el formulario.

   - Se debe comprobar que ningún campo es un intento de hackeo mediante la inyección SQL, en tal caso se utilizará una lista negra de IP y no se permitirá ninguna operación en la aplicación.
3. Crea una página adicional en la ruta /servicios para mostrar en una tabla todos los servicios añadidos.

- El formulario debe validar en el cliente que se han introducido datos.
- La tabla en /servicios debe reflejar dinámicamente los datos almacenados en memoria.

Ejercicio 3: Subida y visualización de imágenes

1. Implementa una funcionalidad para permitir a los usuarios subir una imagen asociada a cada servicio desde el formulario en la página de inicio.
   - Configura un middleware para manejar la subida de archivos (por ejemplo, usando multer) y almacénalos en una carpeta uploads.
2. Modifica la tabla en la página /servicios para mostrar las imágenes junto a los datos de cada servicio (si no se ha subido ninguna imagen, muestra un icono genérico o un mensaje indicando que no hay imagen disponible).

- La subida de imágenes debe ser funcional y válida. (máx 50Kb)
- La tabla debe mostrar imágenes junto a los servicios.

Entrega: El proyecto comprimido en un archivo ZIP, excluyendo la carpeta node_modules debe subirse antes de la finalización del examen al campo virtual en la tarea habilitada para ello.