Seminario de JavaScript

Práctica 4

Esta práctica tiene como propósito fortalecer la comprensión y habilidad en la manipulación de datos, interacción con APIs externas, y dinamismo en la web. Para ésto se va a utilizar el proyecto en `express` provisto por la cátedra.

Entrega Obligatoria

Junto a los demás integrantes de su grupo, implemente el ejercicio 9 y suba el código al repositorio git asignado por la cátedra.

Consideraciones sobre la entrega

- Se debere realizar la entrega sobre la rama `entrega-03` .
- Se debe realizar en grupo.
- Se debe subir al repositorio git asignado al grupo en gitlab.
- Cada integrante del grupo debe hacer sus commits con su aporte a la solución.

Objetivos

- Familiarizarse con el entorno de ejecución node.js y el gestor de paquetes npm.
- Aprender a servir archivos estáticos utilizando Express en un proyecto de Node.js.
- Implementar y manejar formularios en el lado del servidor.
- Gestionar la autenticación básica de usuarios y redirecciones de páginas.
- Trabajar con archivos JSON para la persistencia de datos.
- Desarrollar APIs REST sencillas y consumirlas desde el cliente.

Ejercicio 1: Preparando el Entorno de Desarrollo

Antes de comenzar a trabajar en los ejercicios, asegúrate de tener Node.js y npm instalados en tu sistema. Debés seguir la guía en el README.md del repositorio del grupo.

Ejecuta el comando para inciar el servidor:

```
npm run dev
```

Ejecticio 2: Preguntas de comprensión

Respondé las siguientes preguntas:

1. Explicar la función del archivo `package.json` .
2. Indicar qué versión de la biblioteca `express` se está utilizando.
3. Justificar la elección del puerto 3000 para la aplicación.
4. Describir cómo se especifica en Express la carpeta de archivos estáticos.
5. Analizar el bloque de código que define una ruta al raíz en `index.js` .
6. Aclarar por qué la URL de la imagen `shark.png` es directa y no contiene un subdirectorio `static` .

Ejercicio 3: Servir Archivos Estáticos

Modifique la aplicación para sirva archivos estáticos siguiendo este tutorial. Agregá una imagen, servila y accedela desde el navegador.

Ejercicio 4: Formulario de Login y Registro en el Servidor

Implemente un formulario de login y agregue una ruta en el servidor para recibir y mostrar los datos del formulario en la consola del servidor. Debe validar que los campos sean correctos.

Además responde las siguientes consignas:

1. ¿Qué método HTTP se utiliza para enviar los datos del formulario?
2. Describir cualquier configuración especial en Express necesaria para procesar datos de formulario.

Ejercicio 5: Validación de Formulario en el Cliente

Modifique el formulario de login para realizar validaciones en el lado del cliente, asegurando que los campos no estén vacíos y que el nombre de usuario sea un correo electrónico válido.

Además responde la siguiente consigna:

- ¿Cuál es una buena manera de validar el formato de una dirección de e-mail?

Ejercicio 6: Autenticación y Redirección

Modifique el formulario de login para autenticar usuarios en el servidor y realizar redirecciones según el éxito o fracaso de la autenticación. En caso de ser correctos, se debe **redirigir** a una página que informe el éxito y en caso de no serlo, a la misma página de login.

Además responde las siguientes consignas:

1. ¿Qué es una redirección en el contexto del protocolo HTTP?
2. ¿Cómo se realiza una redirección utilizando `express` ?

Ejercicio 7: Persistencia de Usuarios en JSON

Cambie la autenticación para que en lugar de usar constantes, los datos de los usuarios se almacenen y lean de un archivo `usuarios.json` .

Además responde la siguiente consigna:

- Indicar dónde se guarda el archivo `usuarios.json` y por qué.

Ejercicio 8: API REST para Consulta de Usuarios

Implemente un controlador que permita consultar el nombre de un usuario por su `username` mediante una solicitud GET, devolviendo la información en formato JSON.

Además responde las siguientes consignas:

1. Enumerar qué método HTTP se utiliza para esta consulta.

2. Explicar cómo devolver respuestas en formato JSON usando Express.
3. Describir qué se devuelve en caso de no encontrar el `username` o no enviar el parámetro.

Ejercicio 9: APIs REST Basadas en Archivo JSON

Este ejercicio propone construir una API que calcule y devuelva estadísticas sobre las distancias recorridas por diferentes vehículos. La implementación básica se realiza utilizando un archivo vehicles.json. Como desafío opcional, se puede optar por utilizar MongoDB para una gestión de datos más avanzada.

Datos de los Vehículos

Los vehículos se almacenan en un archivo llamado `vehicles.json` y deben tener atributos como el tipo de vehículo (por ejemplo, automóvil, bicicleta, motocicleta), la distancia total recorrida (en kilómetros) y el consumo de combustible (para vehículos que usen combustible).

Estructura de ejemplo para vehicles:

```
[
    {
        "id": 1,
        "type": "automóvil",
        "distance": 15000,
        "fuelConsumption": 10
    },
    {
        "id": 2,
        "type": "bicicleta",
        "distance": 300
    },
    {
        "id": 3,
        "type": "motocicleta",
        "distance": 5000,
        "fuelConsumption": 3
    }
]
```

Desarrollo de la API

- *Distancia Total Recorrida*: Implemente un endpoint que devuelva la distancia total recorrida por todos los vehículos registrados.
- *Distancia Promedio por Tipo de Vehículo*: Cree un endpoint que calcule y devuelva la distancia promedio recorrida agrupada por tipo de vehículo.
- *Mayor Consumidor de Combustible*: Desarrolle un endpoint |que identifique el vehículo con el mayor consumo de combustible, basándose en la distancia recorrida y el consumo de combustible. Considere solo vehículos que utilizan combustible.
- *Vehículo con Mayor Distancia Recorrida*: Implemente un endpoint que devuelva los datos del vehículo que ha recorrido la mayor distancia, obteniendo los datos del archivo JSON o de MongoDB.

Desafío Opcional: Uso de MongoDB

Puedes optar por almacenar y gestionar los datos de los vehículos utilizando un archivo vehicles.json o una base de datos MongoDB.

En este ejercicio, se trabajará con un archivo vehicles.json que contiene información sobre varios vehículos, incluyendo el tipo de vehículo (por ejemplo, automóvil, bicicleta, motocicleta), la distancia total recorrida (en kilómetros) y el consumo de combustible (para vehículos que usen combustible).