Prueba Backend Junior

Jhoanatan Jerez De Felipe.

Instrucciones de ejecución:

Se debe tener instalado Nodejs.

Se debe tener instalado un editor como Visual Studio Code.

Se deben instalar Express y body-parser mediante: **npm install express body-parser**

En la terminal para iniciar el servidor debemos hacerlo con: **npm start**

Se debe verificar que tu servidor esté activo con http://localhost:3000

Se recomienda ejecutar desde Postman las siguientes rutas:

Método	**Ruta**	**Descripción**
GET	/libros	Obtener todos los libros
GET	/libros/:id	Obtener un libro por su ID
POST	/libros	Crear un nuevo libro
PUT	/libros/:id	Actualizar un libro existente
DELETE	/libros/:id	Borrar un libro por su ID

Resolución de preguntas técnicas.

1. **¿Qué es Node.js y para qué se usa?**

   Node.js es un entorno de ejecución para JavaScript en el servidor que permite crear aplicaciones de forma más eficiente. Se usa principalmente para desarrollar servidores, APIs, aplicaciones web en tiempo real y herramientas de automatización, facilitando el uso del mismo lenguaje en frontend y backend.

2. **¿Qué es Express y qué ventajas tiene al usarlo en un proyecto con Node.js?**

   Express es un framework para Node.js que simplifica la creación de servidores y rutas. Usa una sintaxis sencilla, para tareas comunes que organiza el código y posee una comunidad activa que aporta módulos y buenas prácticas para acelerar el desarrollo.

3. **¿Qué función cumple app.listen() en Express?**

   La función app.listen() en Express inicia un servidor y lo pone a escuchar en un puerto específico. Es el punto de entrada para que el servidor comience a aceptar solicitudes entrantes y responderlas, habilitando la comunicación con los clientes.

4. **¿Qué diferencia hay entre req.params y req.body en una petición?**

   req.params contiene los parámetros de la ruta, como los tipo /books/:id.

   req.body incluye los datos enviados en el cuerpo de una petición, como en formularios o solicitudes POST, generalmente en formato JSON o form-data.

5. **¿Qué es un middleware en Express? Da un ejemplo simple.**

   Un middleware en Express es una función que se ejecuta en el proceso de manejo de una solicitud, permitiendo modificar la petición, la respuesta o realizar tareas previas. Ejemplo simple: un middleware que registra la URL de cada solicitud.

6. **¿Cómo puedes manejar errores cuando algo falla dentro de una ruta?**

Para manejar errores en una ruta, se puede usar try-catch para capturar excepciones y responder con mensajes de error.

7. ¿Qué significa que una API sea RESTful?

Que una API sea RESTful significa que sigue los principios de la arquitectura REST, usando métodos HTTP, URLs claras, sin estado y recursos identificados mediante URIs, facilitando la interacción y la escalabilidad en la comunicación entre cliente y servidor.

8. ¿Qué método HTTP usarías para:
    - Obtener todos los libros
    - Crear un nuevo libro
    - Eliminar un libro existente

Para obtener todos los libros: método GET, para crear un nuevo libro: método POST y para eliminar un libro existente: método DELETE.

9. ¿Qué debe hacer el servidor si un cliente pide un libro con un ID que no existe?

El servidor debe responder con un código de estado 404 (No encontrado) y un mensaje indicando que el libro con ese ID no existe, informando al cliente de la situación.

10. ¿Cómo se define una ruta dinámica como /books/:id en Express?

En Express, una ruta dinámica como /books/:id se define usando parámetros en la ruta (req,resp) es decir petición y respesta.

11. ¿Por qué es una buena práctica separar rutas, controladores y lógica del negocio en carpetas distintas?

Es una buena práctica separar rutas, controladores y lógica del negocio en carpetas distintas para mejorar la organización del código, facilitar su mantenimiento, promover la escalabilidad y permitir que diferentes desarrolladores trabajen en diferentes partes del proyecto de forma más eficiente.

12. ¿Dónde colocarías la lógica para validar que un campo no esté vacío al crear un libro?

Se colocaría en un controlador específico dedicado a la validación, en una carpeta separada de las rutas principales, asegurando así una estructura clara y coherente del proyecto.

13. ¿Qué datos nunca deberías guardar en texto plano en un servidor?

Datos como contraseñas, claves de cifrado, información bancaria, datos personales sensibles y tokens de autenticación nunca deberían guardarse en texto plano en un servidor.

Es fundamental cifrar estos datos para proteger la privacidad y la seguridad de los usuarios, evitando riesgos ante posibles brechas de información.

14. ¿Qué medidas básicas tomarías para validar datos que llegan desde el cliente?

Se deben aplicar medidas básicas como verificar que los campos requeridos no estén vacíos, validar el formato de los datos (como correos electrónicos o números de teléfono), limitar la longitud de los campos.

Estas prácticas garantizan que la información recibida sea coherente, segura y útil para el procesamiento posterior.