**CONSTRUCCIÓN Y PRUEBAS DE SOFTWARE**

**LABORATORIO N° 12**

**Pruebas de Integración**

**(Postman)**

**CODIGO DEL CURSO:**

|  |  |  |
| --- | --- | --- |
| ***Alumno(s)*** | | ***Nota*** |
|  | |  |
|  | |  |
|  | |  |
| ***Grupo*** |  | |
| ***Ciclo*** | ***IV*** | |
| ***Fecha de entrega*** |  | |

**I.- OBJETIVOS:**

* Conocer los aspectos básicos para realizar pruebas de integración con Postman.
* Usar herramientas incorporadas a las tecnologías de pruebas.

**II.- SEGURIDAD:**

|  |  |
| --- | --- |
|  | **Advertencia:**  **En este laboratorio está prohibida la manipulación del hardware, conexiones eléctricas o de red; así como la ingestión de alimentos o bebidas.** |

**III.- FUNDAMENTO TEÓRICO:**

Revise sus diapositivas del tema antes del desarrollo del laboratorio.

**IV.- NORMAS EMPLEADAS:**

No aplica

**V.- RECURSOS:**

* En este laboratorio cada alumno trabajará con un equipo con la plataforma de Postman e Internet.

**VI.- METODOLOGÍA PARA EL DESARROLLO DE LA TAREA:**

* El desarrollo del laboratorio es individual.

**VII.- MARCO TEORICO:**

1. Concepto

Postman es una herramienta gratuita y muy usada por desarrolladores y testers para trabajar con APIs REST. Permite enviar peticiones HTTP como GET, POST, PUT, DELETE, y ver la respuesta del servidor de forma organizada.

1. Uso

Postman se usa para:

* Probar endpoints: ver si las URLs responden correctamente.
* Validar respuestas: comprobar que el servidor responde con los datos correctos (código de estado, contenido JSON, etc.).
* Automatizar pruebas: usando scripts en JavaScript, Postman puede verificar automáticamente si los resultados son correctos.
* Simular flujos completos: como login → crear → consultar → eliminar.
* Repetir pruebas fácilmente: sin necesidad de programar un cliente o frontend.

1. Postman en pruebas de integración

Postman puede considerarse una forma de realizar pruebas de integración, siempre que estés probando cómo interactúan componentes reales a través de una API. Por ejemplo:

* El frontend comunica con el backend.
* El backend se conecta a una base de datos.
* Un servicio externo responde correctamente.

1. Guía de uso
   1. Collections

Una Collection es un conjunto organizado de peticiones que pertenecen a un mismo proyecto o flujo de pruebas. Son muy útiles para mantener pruebas ordenadas, reutilizables y automatizables.

Dentro de una colección puedes encontrar los siguientes campos:

* Peticiones (Requests): Permite crear peticiones HTTP con métodos como:
* GET: obtener datos
* POST: crear datos
* PUT: actualizar todos los datos
* PATCH: actualizar parcialmente
* DELETE: eliminar datos
* URL: Es la dirección del endpoint de la API que estás probando. Ejemplo: https://api.com/posts/1
* Params: Permite agregar parámetros a la petición. Existen varios tipos:
* Path params: forman parte de la URL (/users/:id).
* Query params: van en la URL como ?id=1&name=Juan.
* Header params: enviados en los encabezados.
* Body params: enviados en el cuerpo de la petición (usualmente JSON).
* Headers: Información adicional enviada con la petición, como:
* Content-Type: application/json
* Authorization: Bearer <token>
* Body: Contenido de la petición (para POST, PUT, PATCH). Puede estar en formato raw, form-data, x-www-form-urlencoded, o binary.
* Authorization Sección donde puedes especificar cómo se autoriza la petición. Tipos comunes:
* No Auth: sin autenticación
* API Key: clave en URL o encabezado
* Bearer Token: token en el header
* Basic Auth: usuario y contraseña en base64
* OAuth 1.0 / 2.0: protocolos de autorización avanzados
* Pre-request Script: Código JavaScript que se ejecuta antes de enviar la petición. Ideal para preparar variables o cálculos dinámicos.
* Tests: Código JavaScript que se ejecuta después de recibir la respuesta. Se usa para validar automáticamente el código de estado, datos, campos JSON, etc.
* Test Results: Muestra visualmente si las pruebas automáticas pasaron o fallaron.
  1. Environments

Un Environment (entorno) en Postman es un conjunto de variables que puedes reutilizar en múltiples peticiones, especialmente útil si trabajas con diferentes configuraciones (producción, pruebas, desarrollo).

* Nombre del Environment: Nombre que identifica el entorno. Ejemplo: Producción, Desarrollo, Test Local.
* Variables: Puedes definir variables como {{base\_url}}, {{token}}, {{userId}} que luego puedes usar en la URL, headers o body.
* Valores iniciales: Se usan al importar/exportar el entorno.
* Valores actuales: Son los valores activos durante la ejecución de peticiones. Estos pueden cambiar dinámicamente mediante scripts.
  1. Collection Runner

El Collection Runner de Postman permite ejecutar una colección de peticiones de manera automática y secuencial, como si fueran un flujo de pruebas. Es especialmente útil para probar APIs en series de pasos o con múltiples conjuntos de datos.

Se usa para:

* Para automatizar pruebas de extremo a extremo (end-to-end).
* Para ejecutar todas las peticiones de una colección sin hacerlo manualmente una por una.
* Para validar comportamientos completos como: Crear un recurso → Consultarlo → Modificarlo → Eliminarlo.
* Para realizar pruebas con múltiples datos (data-driven) desde un archivo .csv o .json.
* Usar variables entre peticiones: Puedes guardar datos (como un id) en una petición y usarlos en la siguiente.
* Repetir pruebas: Puedes configurar múltiples iteraciones del mismo flujo (ej. 10 veces).

**VIII.- PROCEDIMIENTO:**

1. Abrir la herramienta Postman, descargar la aplicación para computadora o abrir en entorno web.

Usaremos la entidad 'posts' de la API pública <https://jsonplaceholder.typicode.com>.

1. Operaciones CRUD

GET todos los posts

URL: https://jsonplaceholder.typicode.com/posts

Descripción: Obtiene una lista de 100 posts.

POST crear un post

URL: https://jsonplaceholder.typicode.com/posts

Método: POST

Body (raw, JSON):

{

"title": "Mi primer post",

"body": "Este es el contenido",

"userId": 1

}

PUT actualizar un post

URL: https://jsonplaceholder.typicode.com/posts/1

Método: PUT

Body:

{

"id": 1,

"title": "Post actualizado",

"body": "Nuevo contenido",

"userId": 1

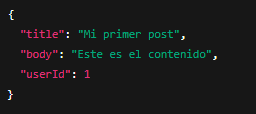
}

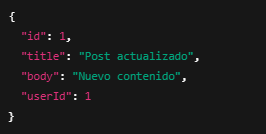
DELETE un post

URL: https://jsonplaceholder.typicode.com/posts/1

Método: DELETE

No requiere body.





1. Pruebas automáticas con JavaScript

Las pruebas automáticas en Postman permiten validar que las respuestas de la API sean correctas. Se escriben en JavaScript dentro de la pestaña 'Tests' y se ejecutan automáticamente al enviar la petición.

Ejemplo 1: Validar código de estado

Código que verifica que la respuesta sea 201 (creación exitosa):

pm.test("Status code es 201", function () {

pm.response.to.have.status(201);

});

Ejemplo 2: Validar un valor en el JSON

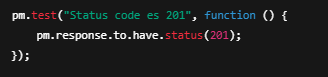
Verifica que el título del post creado sea el esperado:

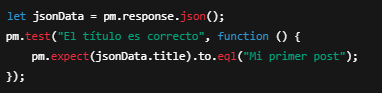
let jsonData = pm.response.json();

pm.test("El título es correcto", function () {

pm.expect(jsonData.title).to.eql("Mi primer post");

});

****

****

1. Pre-request Scripts

Los pre-request scripts son fragmentos de JavaScript que se ejecutan justo antes de enviar la petición. Sirven para preparar variables, autenticar, generar datos dinámicos, etc.

Ejemplo 1: Generar un título dinámico

En la pestaña 'Pre-request Script':

let randomId = Math.floor(Math.random() \* 1000);

pm.environment.set("tituloDinamico", "Post " + randomId);

Y en el body JSON usar:

{

"title": "{{tituloDinamico}}",

"body": "Este es un post aleatorio",

"userId": 1

}

Ejemplo 2: Guardar la fecha actual

Código en 'Pre-request Script':

let now = new Date().toISOString();

pm.environment.set("fechaCreacion", now);





1. Colección con flujo de datos entre peticiones (Collection Runner)

El Collection Runner permite ejecutar una serie de peticiones en secuencia, pasando datos entre ellas mediante variables de entorno. Ideal para probar flujos de autenticación, creación y consulta de recursos.

Flujo sugerido

1. Crear un post y guardar su ID en una variable:

let jsonData = pm.response.json();

pm.environment.set("postId", jsonData.id);

2. Consultar el post recién creado usando:

GET https://jsonplaceholder.typicode.com/posts/{{postId}}

3. Actualizar el post con PUT a:

[https://jsonplaceholder.typicode.com/posts/{{postId}}](https://jsonplaceholder.typicode.com/posts/%7b%7bpostId%7d%7d)



1. En otra API (de su elección y diferente a la utilizada en clase), crear una colección y realizar las operaciones CRUD, cada método con dos test. Finalmente correr la colección. Debe mostrar evidencia de los métodos y test realizados, además del resultado al correr la colección.

Se seleccionó otra API diferente (ejemplo: https://reqres.in).

Se creó una colección nueva con operaciones:

GET /users

Resultado: Código 200 OK, listado de usuarios recibido.

POST /users

Resultado: Código 201 Created, usuario creado.

Test: Validación del status y del campo name.

PUT /users/2

Resultado: Código 200 OK, datos del usuario actualizados

Test: Validación del campo job.

DELETE /users/2

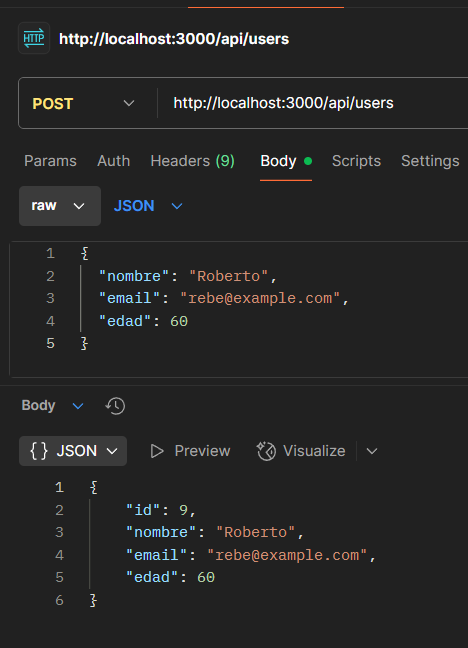
Resultado: Código 204 No Content.

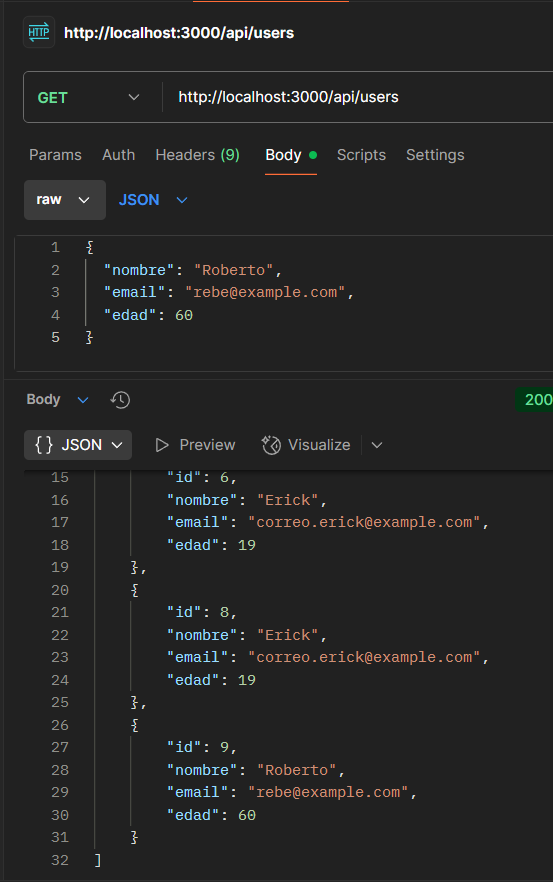
Test: Validación de status y respuesta vacía.

Collection Runner ejecutado con éxito.

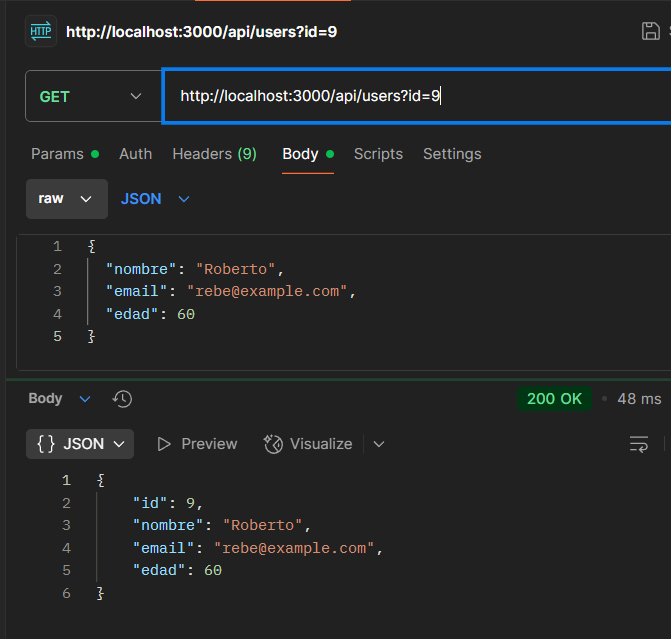
Evidencia de ejecución muestra todos los tests "Passed" y sin errores.

UNO DE LOS EJEMPLOS PUEDE SER LO QUE HICE CON EL PROFESOR USNAYO O PARECIDOS

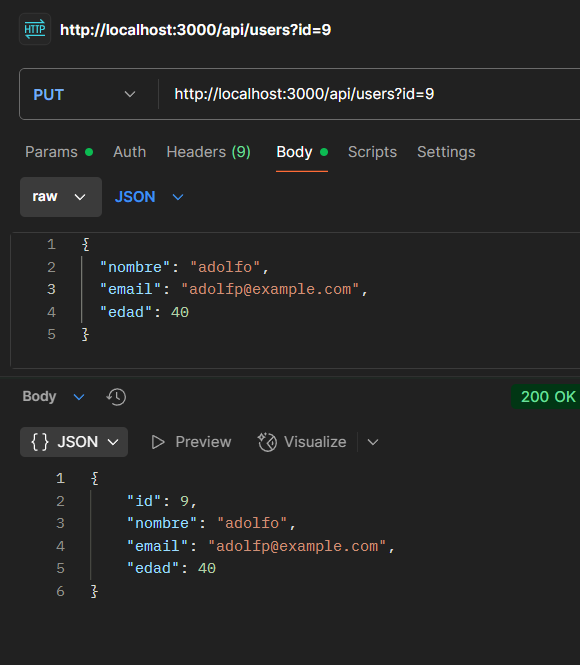
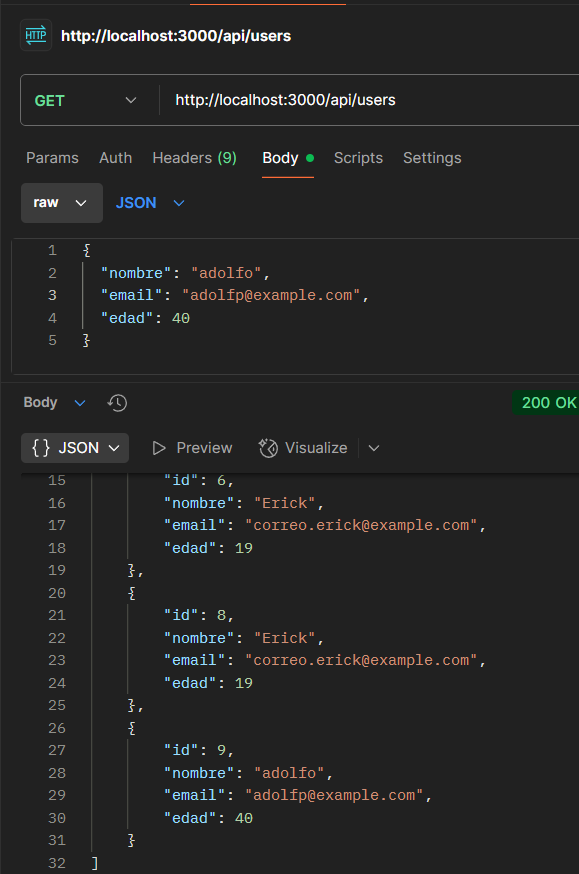
***POST:***

***GET:***

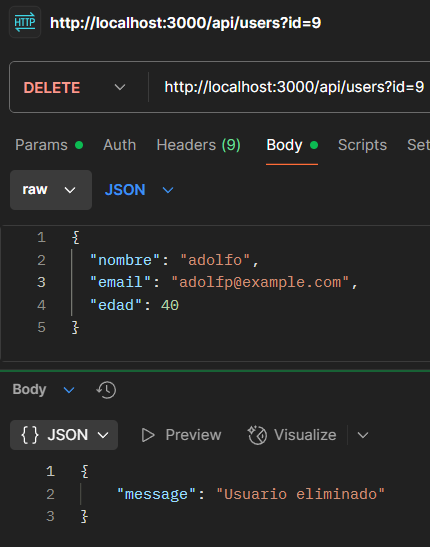
***GET POR ID:***

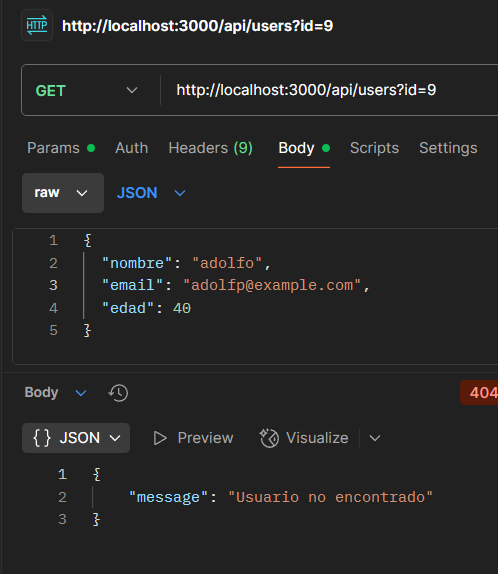
******

***PUT:***

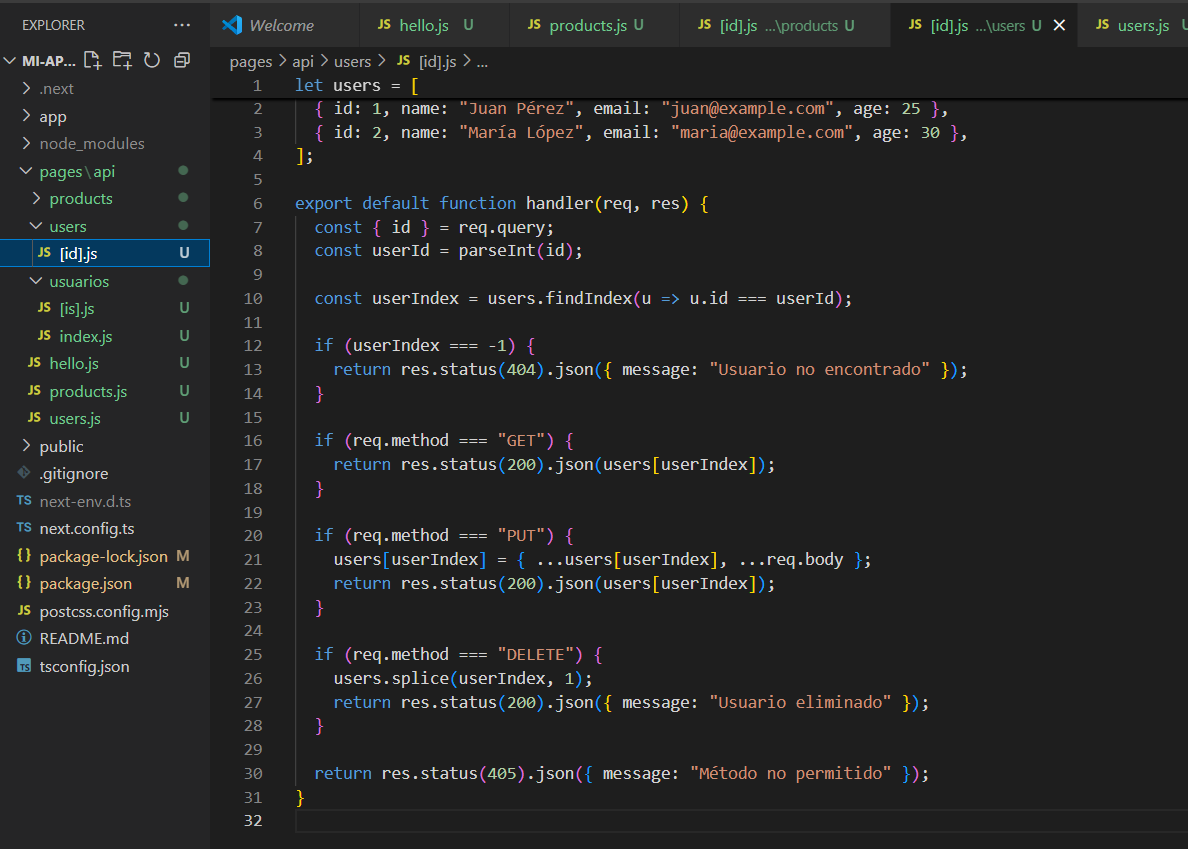
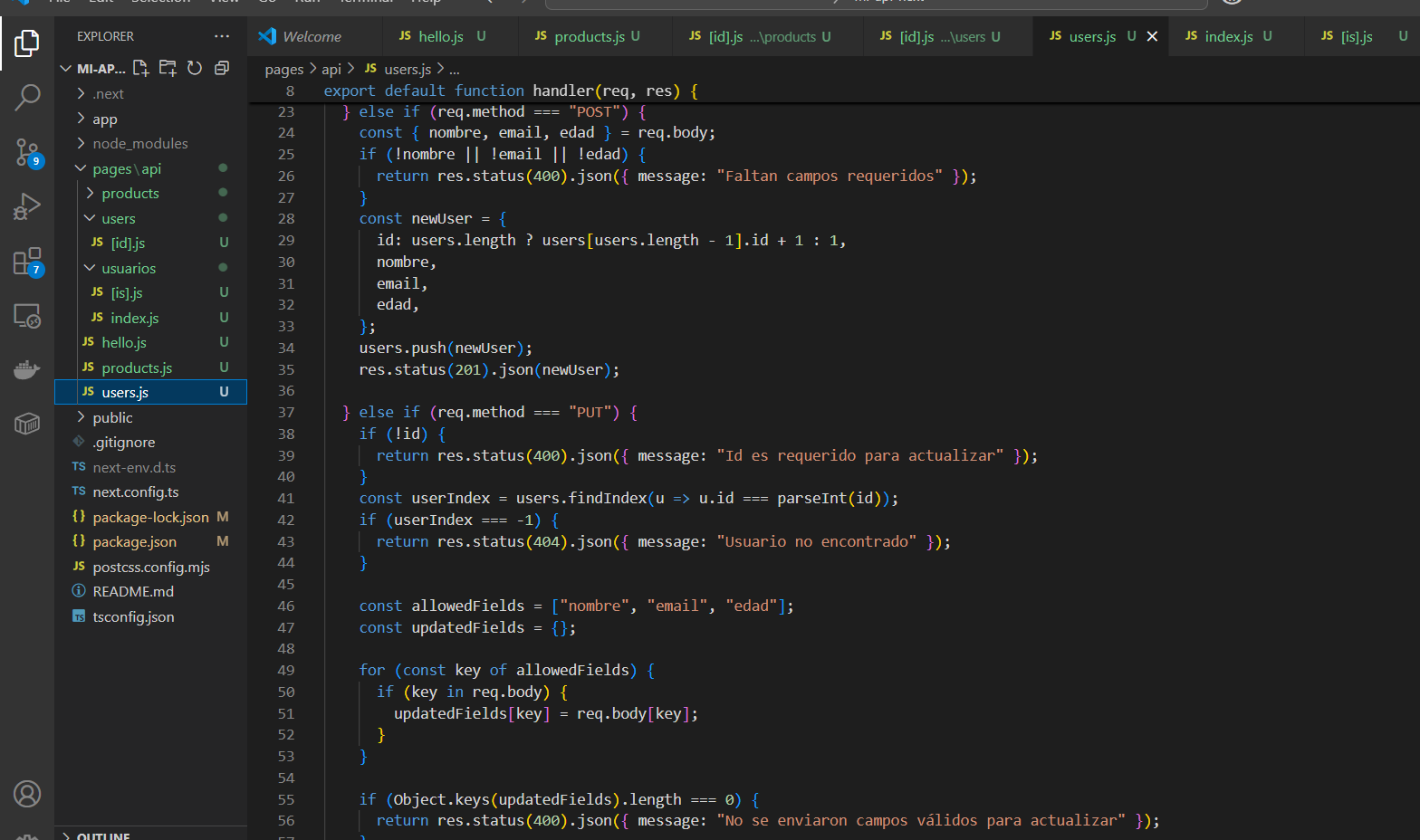
******

***DELETE:***

******

******

***CODIGO:***

******

**OBSERVACIONES:**

-Postman facilita las pruebas manuales y automáticas de APIs REST, permitiendo validar rápidamente el comportamiento esperado de las operaciones CRUD mediante interfaces intuitivas.

-La API jsonplaceholder.typicode.com es una excelente herramienta para pruebas, ya que permite simular peticiones reales sin afectar datos sensibles o reales

-Los scripts de test y pre-request en JavaScript son muy poderosos: permiten automatizar la validación de respuestas y la generación dinámica de datos, reduciendo errores humanos.

-El uso del Collection Runner optimiza las pruebas secuenciales, especialmente al pasar datos entre peticiones mediante variables de entorno (por ejemplo, el postId).

-Al probar otra API externa, se observó que cada API tiene sus propias particularidades, como autorización, formatos requeridos o limitaciones, lo que resalta la importancia de leer bien la documentación.

**CONCLUSIONES:**

-Dominar Postman es clave para cualquier desarrollador o tester backend, ya que permite probar APIs de forma rápida, clara y automatizada.

-Comprender las operaciones CRUD es esencial para interactuar con cualquier servicio web RESTful y garantizar que los datos puedan gestionarse correctamente

-La validación automática de respuestas aumenta la calidad del software, al detectar errores o inconsistencias de forma inmediata.

-El uso de variables de entorno y scripts dinámicos permite realizar pruebas más realistas y flexibles, acercándose a escenarios del mundo real.

-Este laboratorio permitió afianzar conceptos tanto técnicos (HTTP, JSON, status codes) como prácticos (automatización de pruebas), siendo una base sólida para proyectos más complejos que involucren APIs.