|  |  | Kibernum IT Academy  Programa Especialización en Automatización de Pruebas |
| --- | --- | --- |
|  | BOTIC-SOFOF- 24-28-05-0024 |
|  |  |
|  |  |
|  |  |
|  |  |

**Pruebas de Servicio Web**

**Módulo 6 – Evaluación**

|  |  | **Equipo 4:** |
| --- | --- | --- |
|  | Felipe Lobos |
|  | Fabiola Díaz |
|  | Eduardo Arellano  Carlos Vasquez |
|  |  |
|  |  |

**Descripción:**

La empresa ficticia “TechMarket” es un Marketplace digital que permite a pequeños y medianos emprendedores publicar y vender sus productos de manera online. Este sistema dispone de una API REST que permite gestionar los productos, usuarios y perdidos. Actualmente, la empresa se encuentra en un proceso de mejora continua y ha solicitado al equipo de calidad implementar un plan de pruebas automatizadas que permita validar el correcto funcionamiento y seguridad de sus servicios web.

Como parte del equipo de QA, se te ha encomendado realizar una suite de pruebas que integre buenas prácticas de diseño, seguridad, validaciones mediante aserciones y automatización de pruebas utilizando las herramientas vistas en el módulo**.**

**1. API REST utilizada:** [**https://jsonplaceholder.typicode.com**](https://jsonplaceholder.typicode.com)

**2.**

* **¿Que es una API REST?**

Una API (Application Programming Interface o Interfaz de Programación de Aplicaciones) es la interfaz que nos permitirá conectar diferentes sistemas y aplicaciones para compartir información. Las API REST son una forma estándar de la industria para que los servicios web envíen y reciban datos. Utilizan métodos de solicitud HTTP para facilitar el ciclo de solicitud-respuesta y, por lo general, transfieren datos mediante JSON y, más raramente, HTML, XML u otros formatos. Las principales consultas usadas en una API REST son: **GET** (para obtener datos), **POST** (para crear datos), **PUT** (para actualizar datos), **PATCH** (para actualizar parcialmente datos) y **DELETE** (para eliminar datos)

* **Componentes principales de una API REST**

Las APIs REST siguen una serie de características o principios clave que aseguran simplicidad, escalabilidad y eficiencia en la comunicación entre sistemas

| Recursos | Representan los datos o servicios que la API expone. Por ejemplo, /usuarios o /productos. |
| --- | --- |
| Puntos finales (Endpoints) | Son las URL que identifican y permiten acceder a los recursos. Actúan como la dirección del recurso en el servidor. |
| Métodos HTTP | Indican la acción que el cliente desea realizar sobre el recurso. Los más comunes son:   * GET: Para solicitar información o un recurso. * POST: Para crear un nuevo recurso. * PUT: Para actualizar un recurso existente o crearlo si no existe. * DELETE: Para eliminar un recurso * PATCH: Actualizar un determinado campo de un recursos existente. |
| Cabeceras (Headers) | Son metadatos que proporcionan información adicional sobre la solicitud y la respuesta. Incluyen detalles como el formato de los datos (JSON, XML) o información de autenticación |
| Cuerpo de la solicitud | Contiene los datos que se envían al servidor, especialmente para las operaciones de creación y actualización (POST, PUT |

* **Buenas prácticas de diseño de API REST**

| URIs limpias y orientadas a recursos | Las URIs deben representar sustantivos (recursos), no verbos (acciones). Usar nombres en plural para colecciones: /users, /products. |
| --- | --- |
| Uso correcto de métodos HTTP | Semántica clara: GET para obtener, POST para crear, PUT para reemplazar, PATCH para modificar parcialmente y DELETE para eliminar. |
| Estructura jerárquica de recursos | Reflejar relaciones entre recursos para mantener una estructura lógica y navegable. |
| Utilizar códigos de estado HTTP apropiados | Evitar respuestas genéricas como 200 para todo. Usar códigos específicos para cada situación. |
| Hacer uso del principio de "stateless" | Cada solicitud debe ser independiente, conteniendo toda la información necesaria para ser procesada. |
| Utilizar headers correctamente | Ejemplo: Content-Type, Accept, Authorization, etc. para enriquecer la comunicación. |
| Documentar la API | Preferiblemente con Swagger/OpenAPI, para facilitar el consumo por parte de otros equipos o aplicaciones |

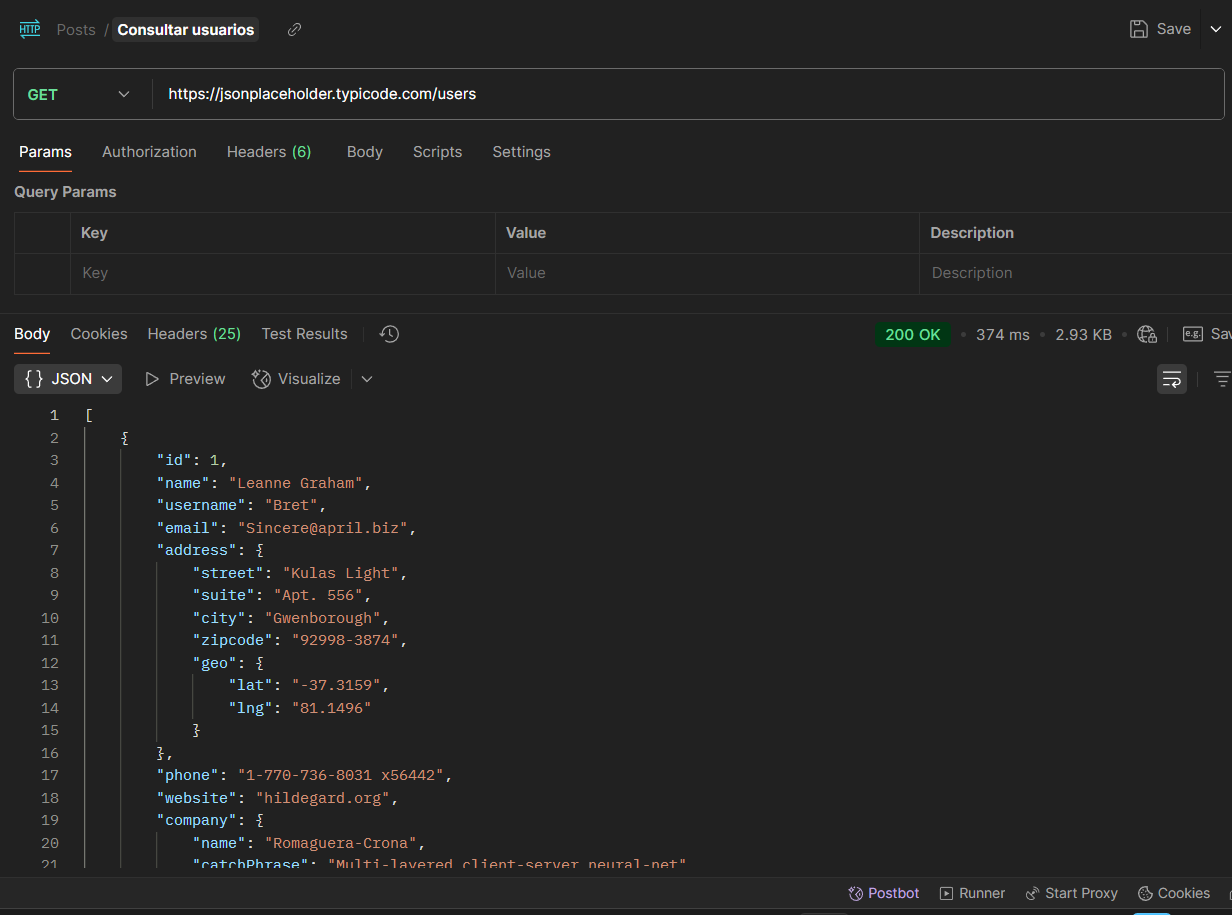
**3. Identificar y describir las amenazas de seguridad comunes en APIs y las buenas prácticas para mitigar estos riesgos. Debes implementar al menos un mecanismo de autenticación en las pruebas (API Key, Token, JWT, o Autenticación Básica).**

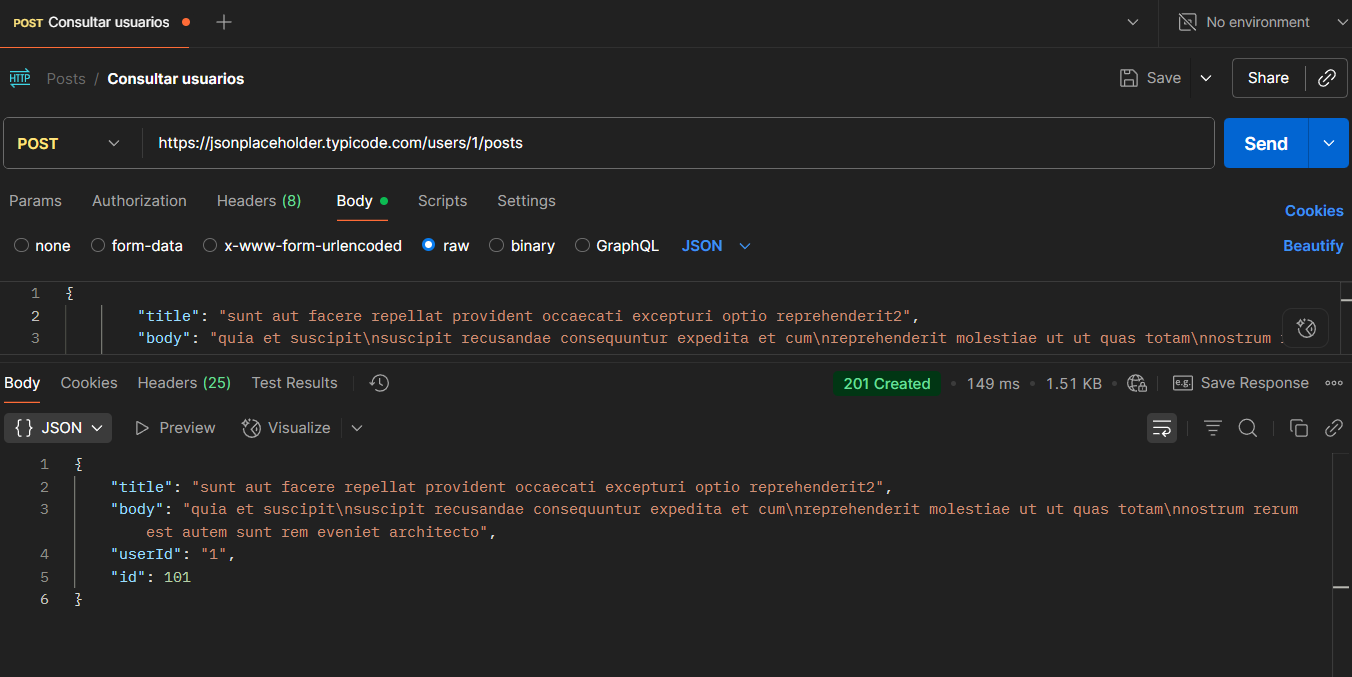
| **Amenaza** | **Descripción** | **Mitigación** |
| --- | --- | --- |
| Inyecciones de código | Cuando un atacante "engaña" a la API enviando código malicioso disfrazado de datos normales, con el fin de robar o modificar información. | - Validar y filtrar los datos de entrada  - Usar consultas seguras (prepared statements)  - Nunca confiar en lo que escribe el usuario |
| Robo de credenciales o tokens | Cuando un atacante intercepta o adivina la clave o token que se usa para autenticar una sesión. | - Usar HTTPS siempre  - Expirar los tokens después de un tiempo corto  - Evitar almacenar claves/token en lugares inseguros |
| Replay Attack | El atacante graba una solicitud válida y la repite más tarde para obtener los mismos beneficios. | - Usar tokens de un solo uso (nonces)  - Agregar marca de tiempo y verificar su validez  - Firmar digitalmente las solicitudes |
| Falsificación de solicitudes (CSRF) | Un sitio malicioso engaña al usuario para que envíe sin querer una solicitud a otra API donde está autenticado. | - Usar tokens CSRF únicos por sesión  - Verificar origen de la solicitud  - Implementar políticas de CORS correctamente |
| Suplantación de identidad (Spoofing) | Hacerse pasar por otro usuario o sistema para engañar a la API. | - Múltiples factores de verificación de identidad  - Validación del origen de la solicitud  - Registrar y analizar comportamientos sospechosos (Logs) |
| Ataques de denegación de servicio (DoS o DDoS) | Un atacante envía miles de solicitudes en poco tiempo para saturar la API y hacer que deje de funcionar. | - Limitar la cantidad de solicitudes por IP  - Usar servicios de protección como Cloudflare  - Monitorear y bloquear comportamientos sospechosos |

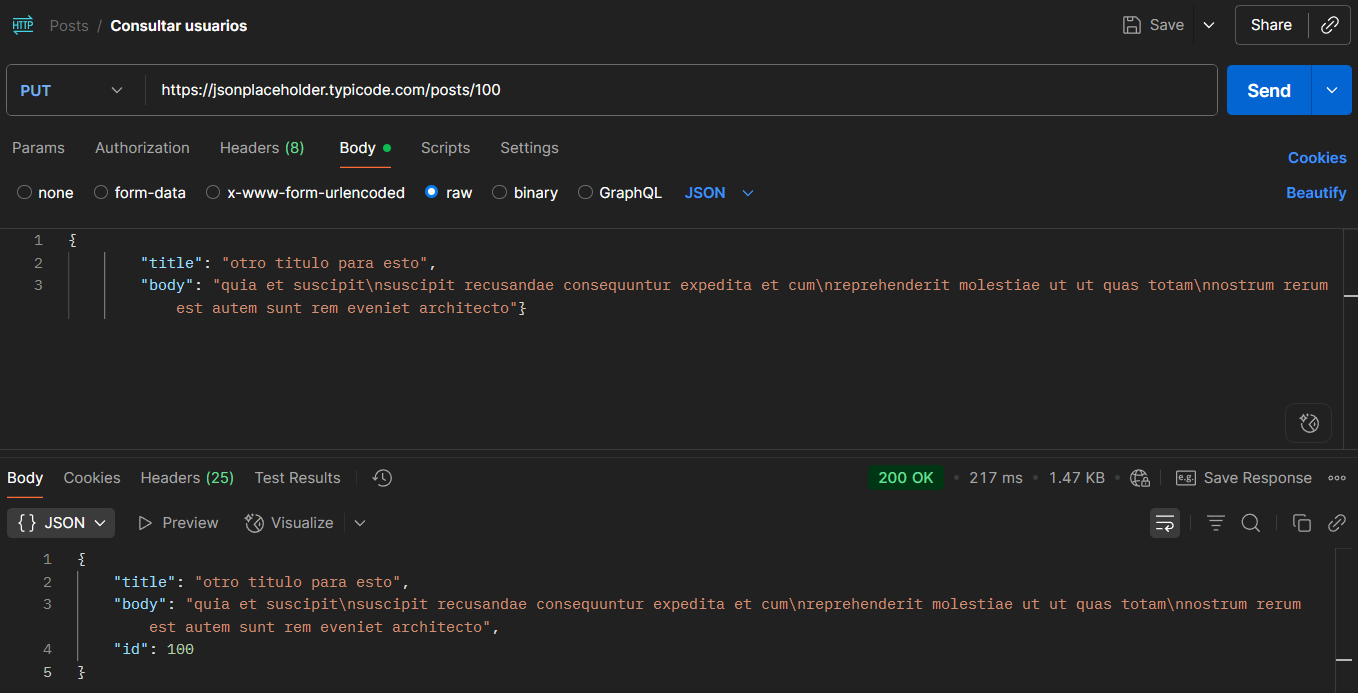
**4. Desarrollar una colección de pruebas manuales utilizando Postman que contemple: o Operaciones básicas (GET, POST, PUT, DELETE).**

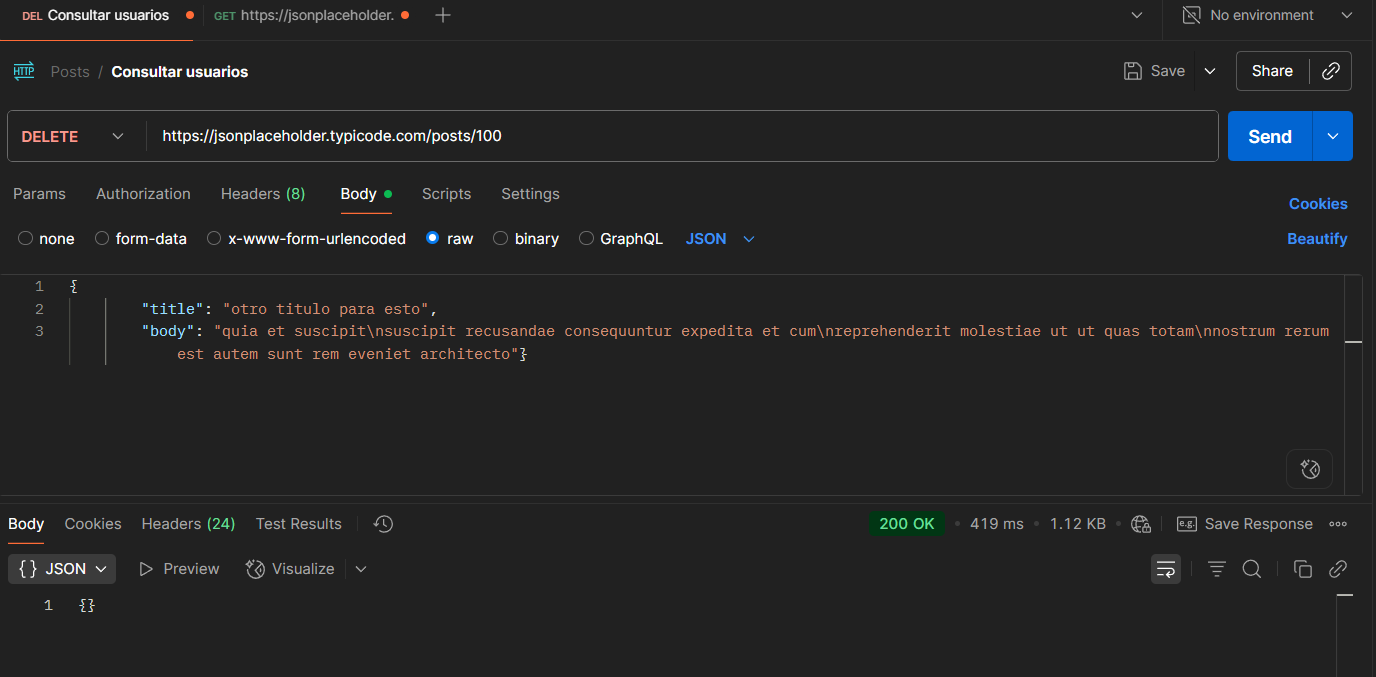
**Operaciones básicas (GET, POST, PUT, DELETE).**

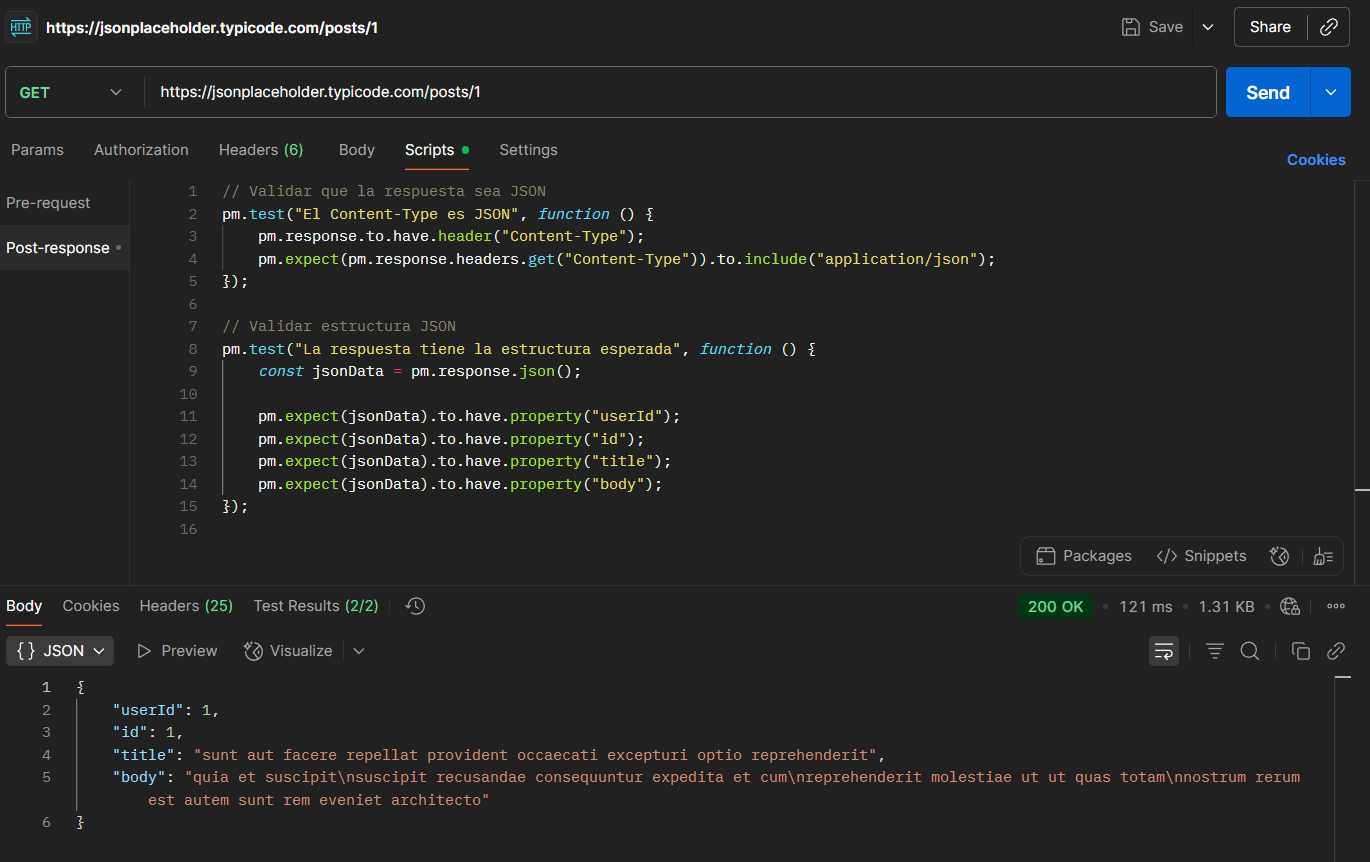
**GET**

****

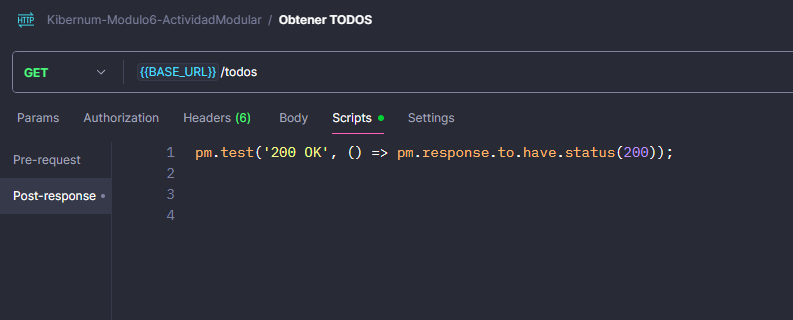
**POST  
**

**PUT  
**

**DELETE  
**

**Validación de estructura del cuerpo de respuesta (JSON).   
**

**Validación de códigos de estado HTTP.**

****

**Aserciones de contenido específicas.**

****