**Feature 05**

* **Título: GP|F05|TC01|** **Test Happy Path**

**Escenario: Crear un nuevo usuario**

* **Método:** POST
* **Endpoint:** /api/Usuario/register
* **Headers:**
  + Content-Type: application/json
* **Body:**

json

{

  "email": "timonrock2015@gmail.com",

  "password": "Prueba1234"

}

**Pasos:**

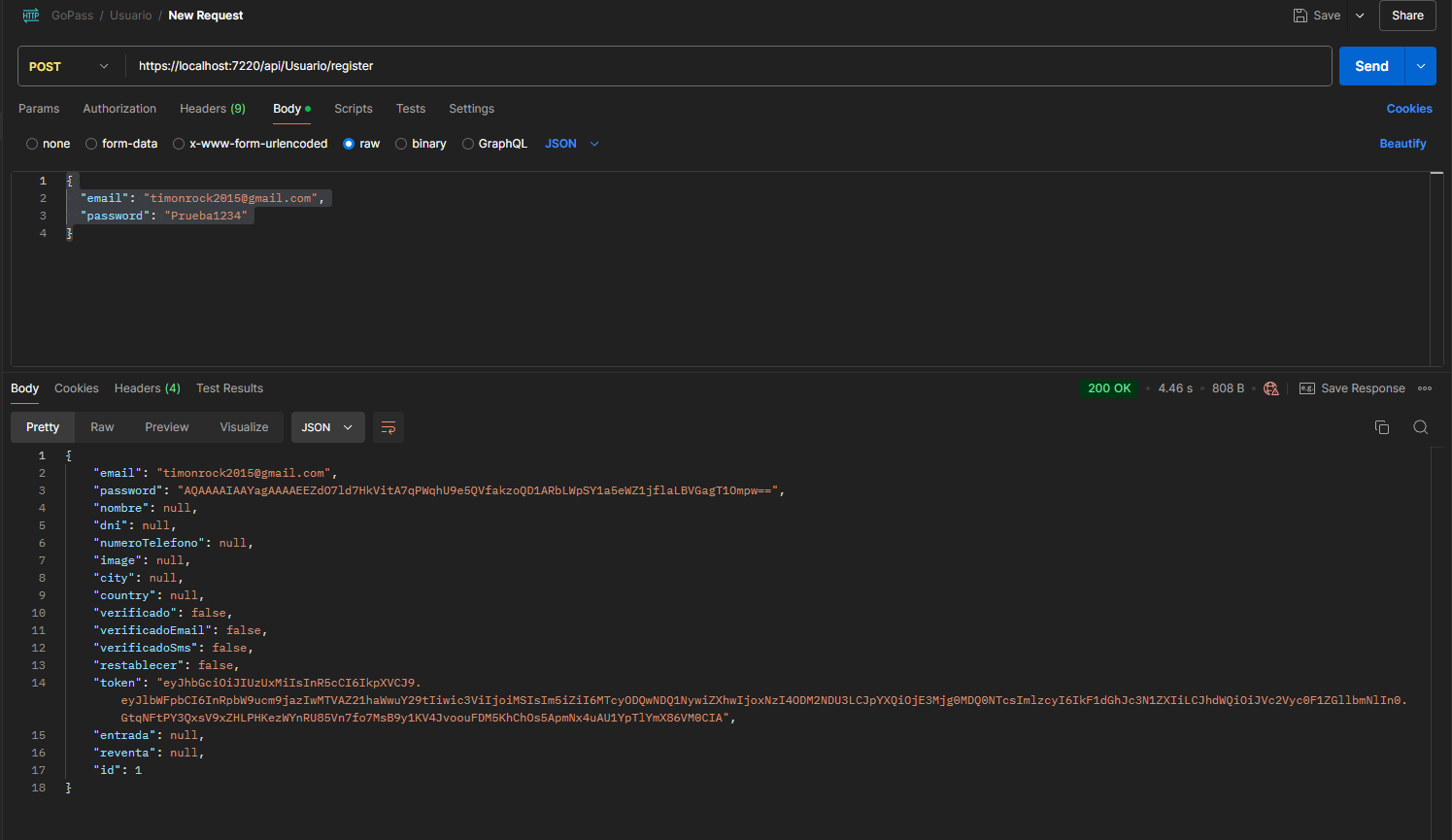
1. Configurar la petición como POST a /api/Usuario/register.
2. En los headers, incluir el Content-Type como application/json.
3. Copia y pega el cuerpo JSON para enviar la petición.
4. Envía la petición y asegúrate de obtener un código de respuesta 200 OK
5. Verifica que la respuesta contenga los detalles del usuario recién creado, sin exponer datos sensibles como la contraseña.

**Validaciones:**

* Verifica que el usuario fue creado exitosamente (código de estado 201).
* Asegúrate de que los datos en la respuesta coincidan con los enviados, excepto la contraseña.
* Valida que no se ha duplicado el usuario si lo intentas crear dos veces.

**Título**: GP|F05|TX01| **Test Happy Path**

* **Descripción**: Verificar que el usuario puede actualizar su información personal.
* **Fecha de Ejecución**: 2024-10-08
* **Ejecutor**: Carlos Daniel Jiménez Cálcena
* Estado: Aprobado



**Título: GP|F05|TC02|** **Prueba de Error Path (Error Handling)**

**Escenario: Crear un nuevo usuario sin proporcionar un correo electrónico**

* **Método:** POST
* **Endpoint:** /api/Usuario/register
* **Headers:**
  + Content-Type: application/json
* **Body:**

Json:

{

  "email": "test",

  "password": "Prueba1234"

}

**Pasos:**

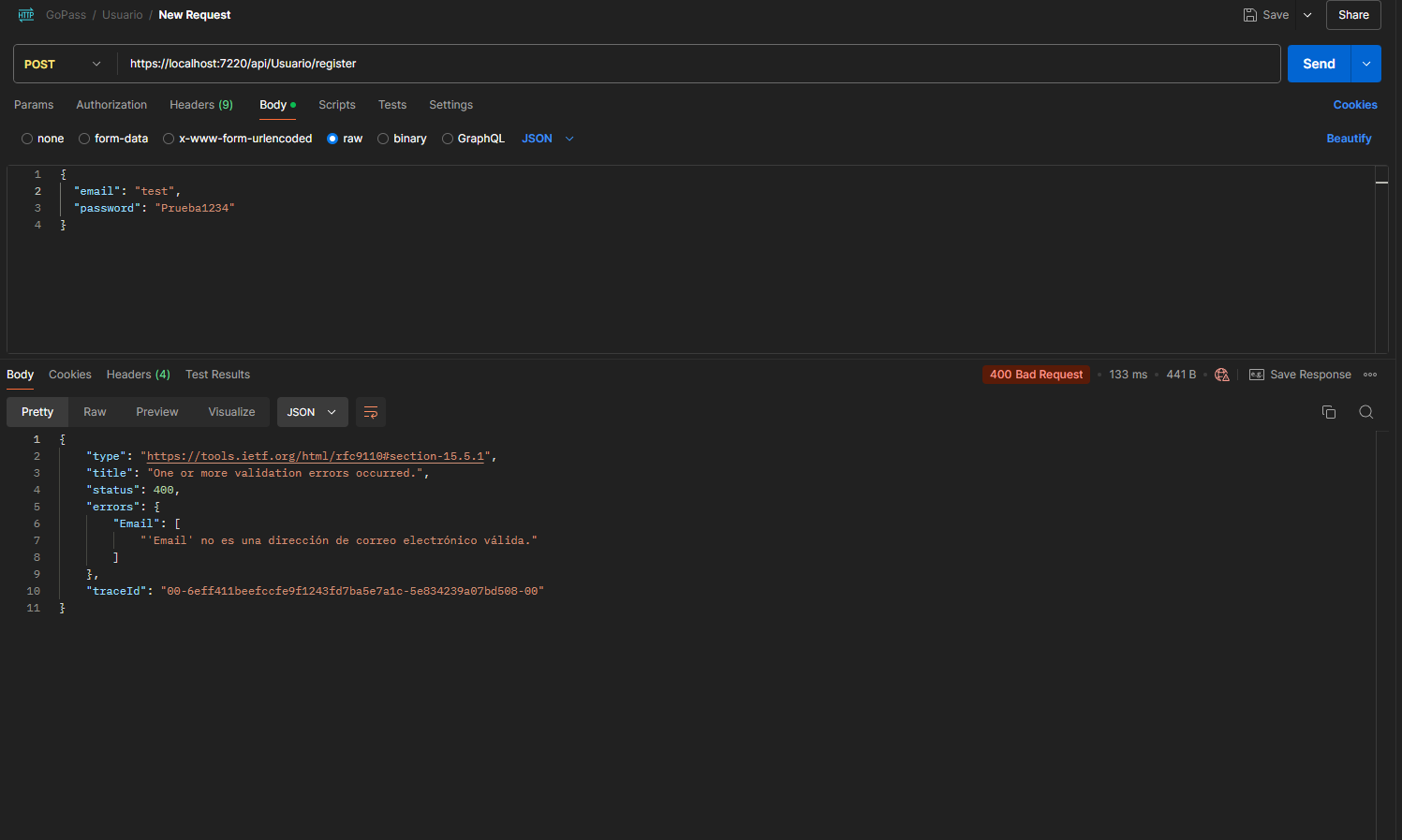
1. Haz una petición POST con el body con un email inválido.
2. Envía la petición.
3. Verifica que recibes un código de respuesta 400 Bad Request.
4. Asegúrate de que el mensaje de error en la respuesta indique que el campo email es obligatorio.

**Validaciones:**

* El código de estado es 400 Bad Request.
* El mensaje de error indica claramente qué campo faltaba.

**Título**: GP|F05|TX02| **Prueba de Error Path (Error Handling)**

* **Descripción**: Verificar que el usuario puede actualizar su información personal.
* **Fecha de Ejecución**: 2024-10-08
* **Ejecutor**: Carlos Daniel Jiménez Cálcena
* Estado: Aprobado



**Feature 06**

**Título: GP|F06|TC01|Prueba de SQL Injection en el Login**

**Escenario: Intentar inyección SQL en el campo de contraseña del login**

* **Método:** POST
* **Endpoint:** /api/Usuario/login
* **Headers:**
  + Content-Type: application/json
* **Body:**

{

"email": "testuser",

"password": "' OR '1'='1"

}

**Pasos:**

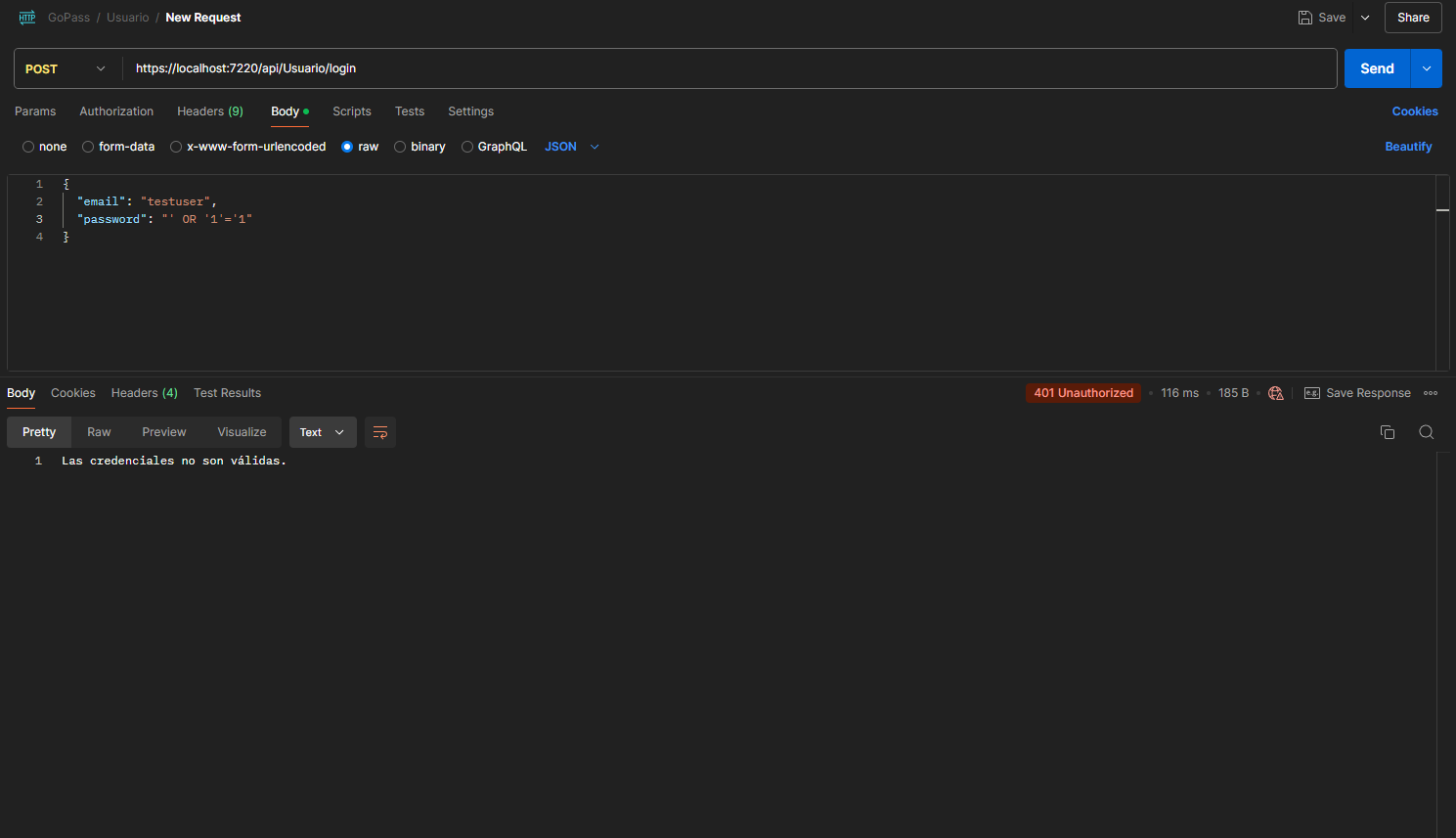
1. Haz una petición POST al endpoint de login con el cuerpo que contiene la inyección SQL en el campo de contraseña.
2. Envía la petición.
3. El sistema no debe permitir el inicio de sesión y que y no debe responder con un código 200 OK.
4. Asegúrate de que la respuesta contenga un código de error 401 Unauthorized o un mensaje de error relacionado con credenciales incorrectas.

**Resultado esperado:**

* El código de estado debe ser 401 Unauthorized.
* El sistema no debe permitir el acceso ni devolver información sensible.
* La base de datos no debe ejecutar la inyección SQL

**Título**: GP|F06|TX01| **Prueba de SQL Injection en el Login**

* **Descripción**: Verificar que el usuario puede actualizar su información personal.
* **Fecha de Ejecución**: 2024-10-08
* **Ejecutor**: Carlos Daniel Jiménez Cálcena
* Estado: Aprobado



**Feature 07**

**Título: GP|F07|TC01|Pruebas de Carga y Rendimiento**

**Escenario: Simular múltiples usuarios creando cuentas simultáneamente**

* **Herramienta:** Postman
* **Método:** POST
* **Endpoint:** /api/Usuario/register
* **Headers:**
  + Content-Type: application/json
* **Pre Request:**

javascript

let uniqueEmail = "testuser" + Math.floor(Math.random() \* 1000000) + "@test.com";

pm.variables.set("uniqueEmail", uniqueEmail);

* **Body:**

json

{

"email": "{{uniqueEmail}}",

"password": "Clave1234"

}

**Pasos:**

1. En Postman, utilizar la funcionalidad de *Runner* para enviar múltiples peticiones a la vez.
2. Crear un test que envíe 150 peticiones simultáneamente.
3. Configura los detalles de la prueba para simular múltiples usuarios creando cuentas.
4. Observa la respuesta del servidor y el tiempo de respuesta.
5. Verifica que no haya errores como 500 Internal Server Error debido a la alta carga.

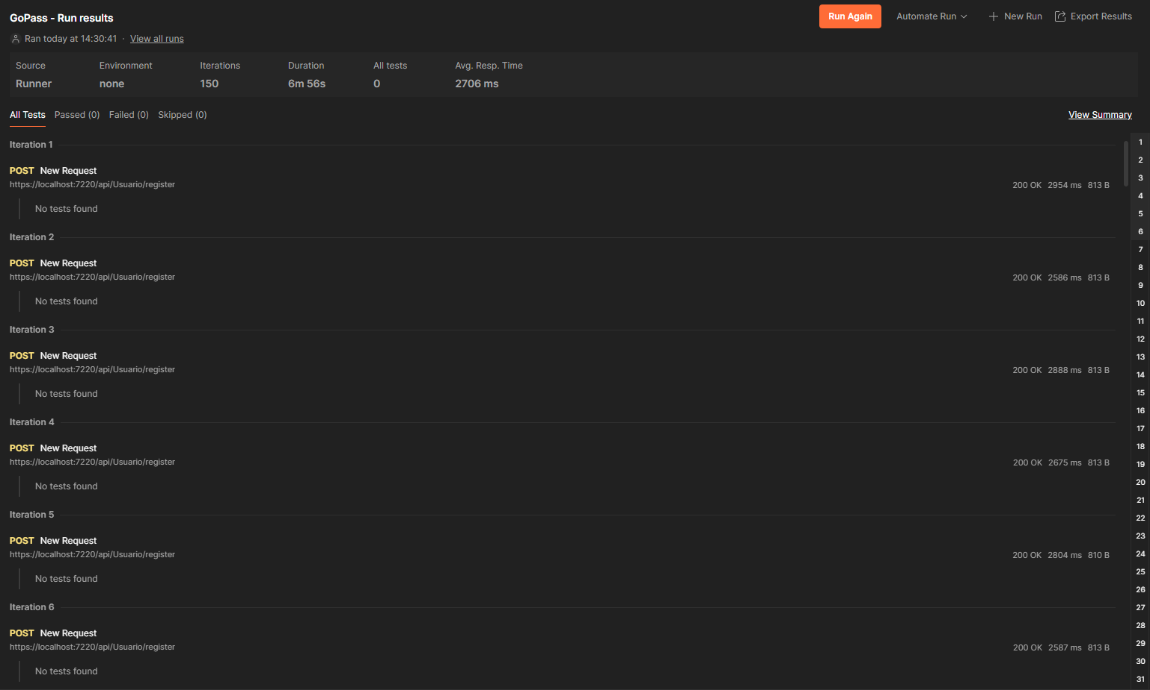
**Resultado esperado:**

* El servidor debe responder consistentemente sin errores.
* Los tiempos de respuesta deben ser aceptables bajo alta carga.
* La base de datos debe registrar correctamente todas las nuevas cuentas.

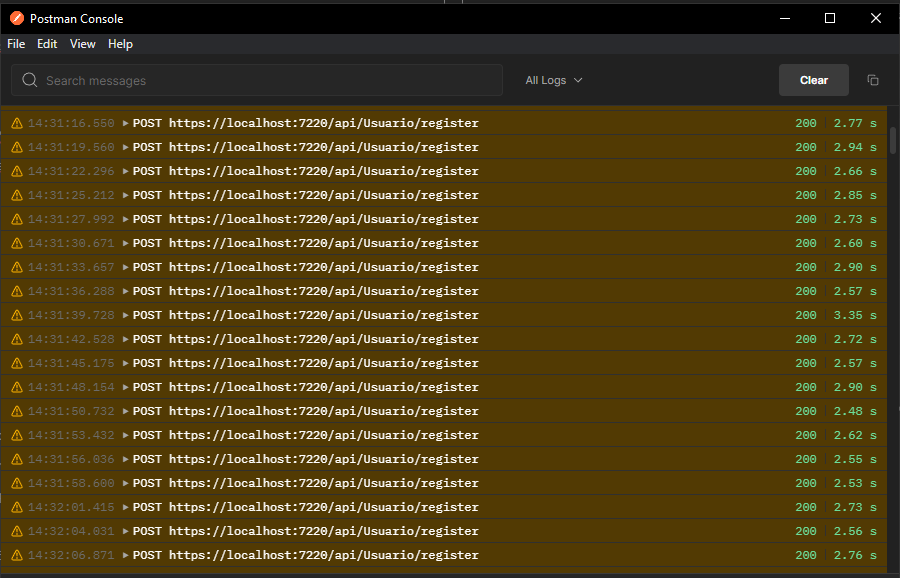
**Título**: GP|F07|TX01| **Prueba de SQL Injection en el Login**

* **Descripción**: Verificar que el usuario puede actualizar su información personal.
* **Fecha de Ejecución**: 2024-10-08
* **Ejecutor**: Carlos Daniel Jiménez Cálcena
* Estado: Aprobado

Se observa que todas las respuestas han sido correctas – (200 Ok), los usuarios han sido creados correctamente.



El tiempo de respuesta de las solicitudes se encuentra dentro del rango aceptable:



Feature 08

**Título: GP|F08|TC01| Verificación de las Respuestas del Servidor**

1. Solicitud de creación de usuario.

Body:

Json:

{

"email": "testuser9@test.com",

"password": "Clave1234"

}

1. Scripts/Post-Request de tu petición, escribe el siguiente código para verificar que no haya errores en la respuesta:

Javascript:

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

    pm.response.to.have.status(201);

});

pm.test("Response time is below 1000ms", function () {

    pm.expect(pm.response.responseTime).to.be.below(1000);

});

Resultado esperado:

Validar que el código de estado sea **200 OK** (que indica que el usuario se creó correctamente)

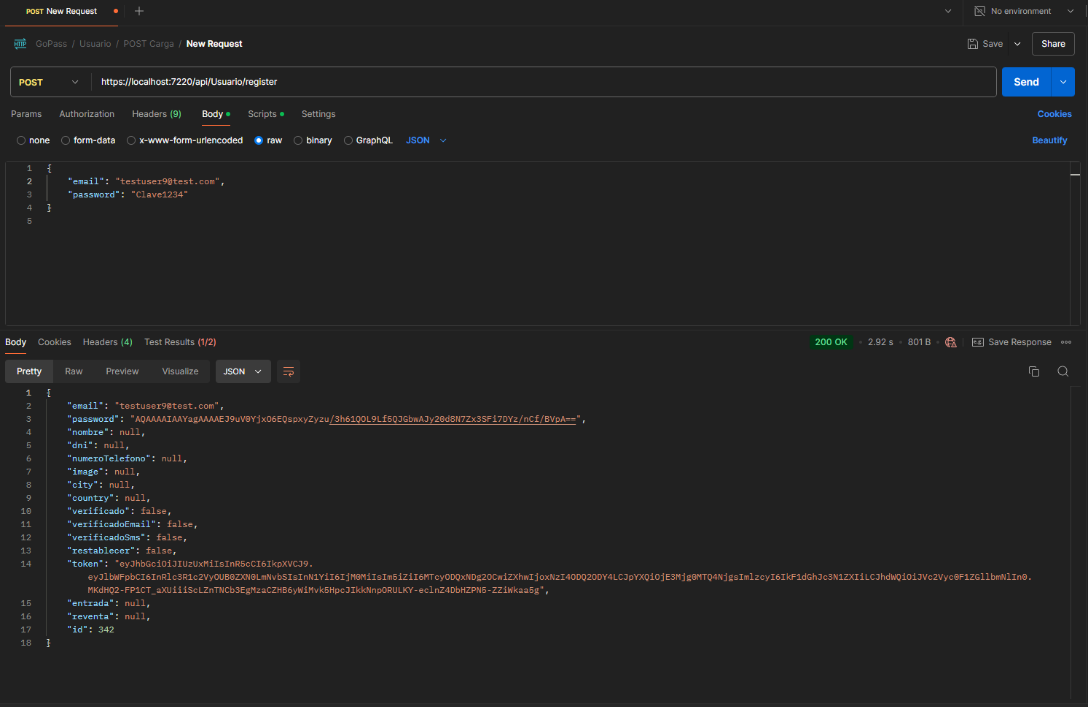
Validar que el tiempo de respuesta sea menor a 1000 ms.

**Título**: GP|F08|TX01| **Verificación de las Respuestas del Servidor**

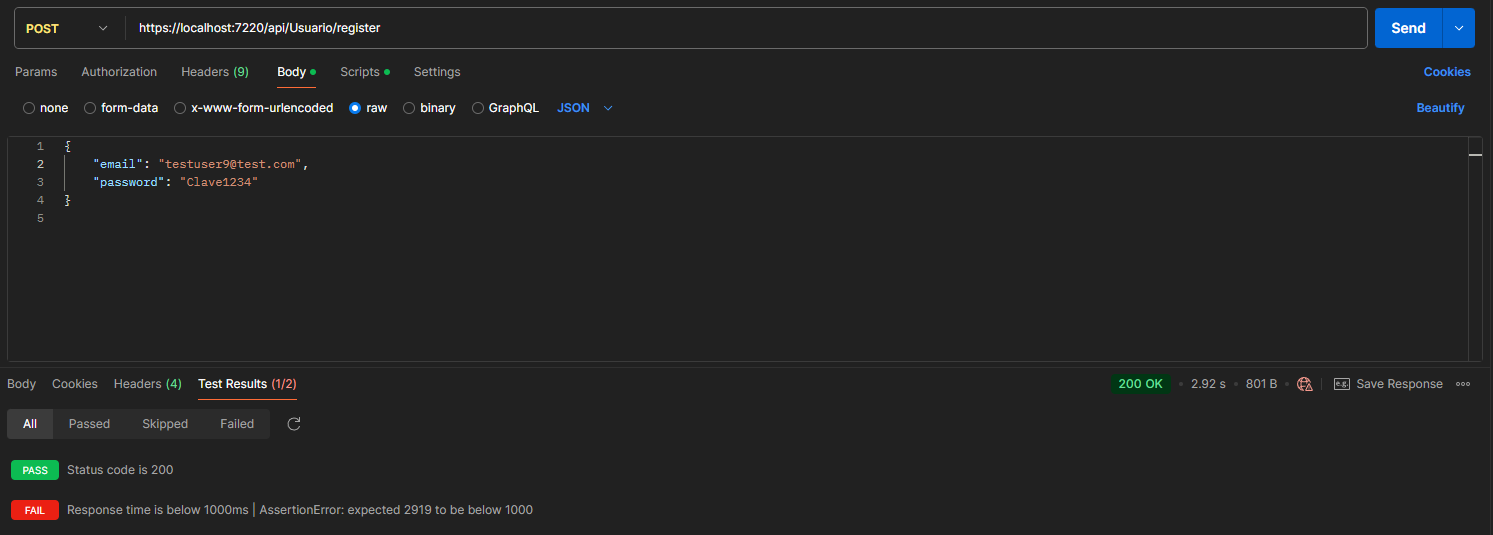
* **Descripción**: Verificar que el usuario puede actualizar su información personal.
* **Fecha de Ejecución**: 2024-10-08
* **Ejecutor**: Carlos Daniel Jiménez Cálcena
* Estado: Aprobado/Fallido

Resultado esperado:

Validar que el código de estado sea **200 OK** (que indica que el usuario se creó correctamente – APROBADO



Validar que el tiempo de respuesta sea menor a 1000 ms. – FALLIDO



Feature 09

**Título: GP|F08|TC01| Prueba de Recuperación de Usuarios**

**Descripción:** Verificar que se pueden recuperar todos los usuarios existentes.

**Precondiciones:** Debe haber al menos un usuario en la base de datos.

**Pasos:**

1. Enviar una solicitud GET a /api/Usuario
2. Esperar la respuesta del servidor.

[

{

"id": 201,

"email": "testuser561066@test.com”

}

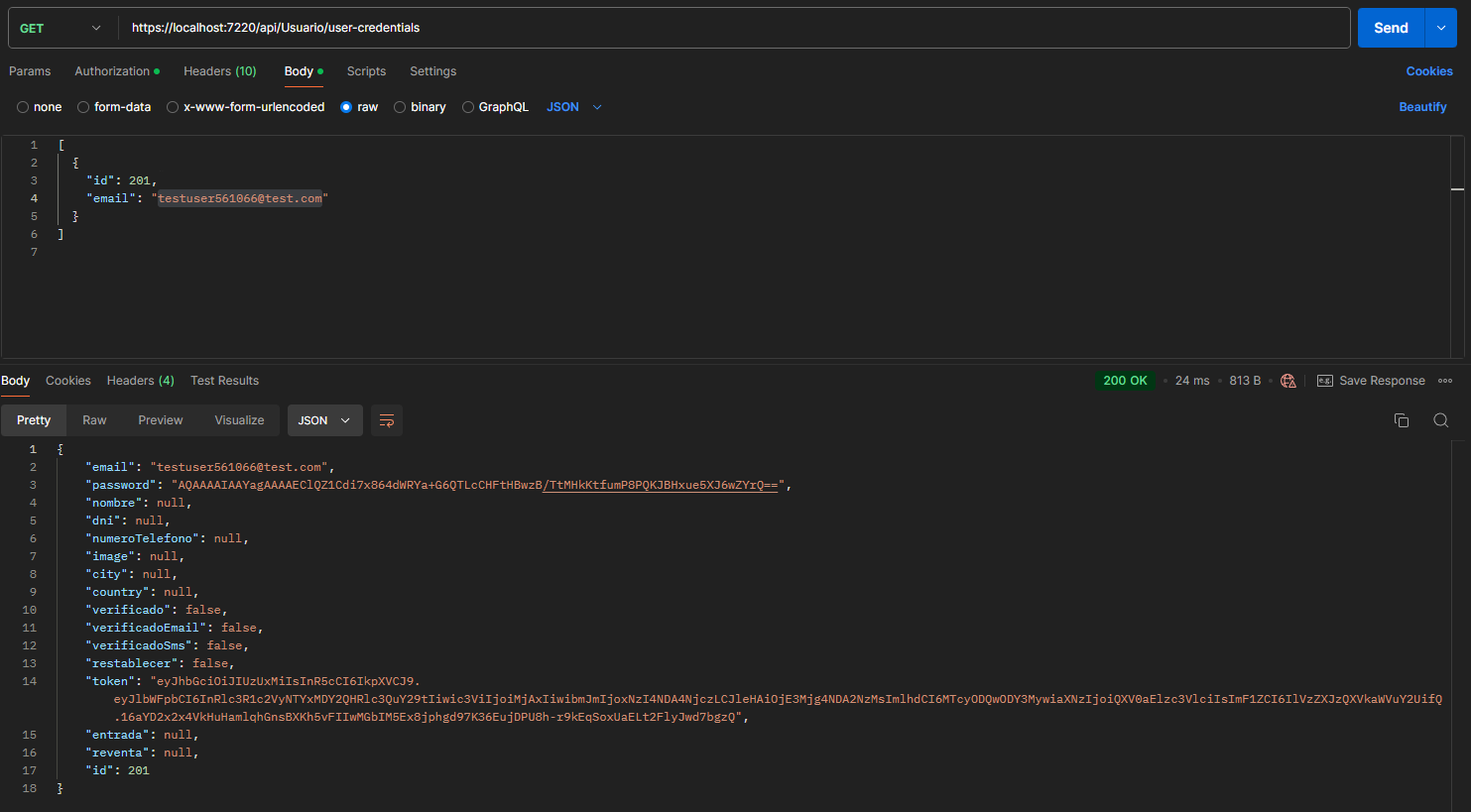
]

**Resultado Esperado:**

* Código de estado HTTP: 200 OK
* Cuerpo de la respuesta debe ser una lista de usuarios:

**Título**: GP|F09|TX01| **Verificación de las Respuestas del Servidor**

* **Descripción**: Verificar que el usuario puede actualizar su información personal.
* **Fecha de Ejecución**: 2024-10-08
* **Ejecutor**: Carlos Daniel Jiménez Cálcena
* Estado: Aprobado



**Resultado Obtenido:**

* Código de estado HTTP: 200 OK