

TALLER 3 - Testing

Presentado por:

Stiven Aguirre Granada - staguirreg@unal.edu.co
Cesar Camilo Velandia Cubillos - cevelandiac@unal.edu.co
Manuel Santiago Mori Ardila - mmori@unal.edu.co

Profesor:


Oscar Eduardo Alvarez Rodriguez
oalvarezr@unal.edu.co

Jueves 27 de Febrero



Universidad Nacional de Colombia
Facultad de Ingeniería
Departamento de Ingeniería Sistemas e Industrial
2024

Anexo de Documentos Relacionados:

-  Proyecto_Final

Introducción

Este documento presenta un resumen de las pruebas realizadas sobre la aplicación de gestión de recetas, una plataforma diseñada para permitir a los usuarios crear, editar, calificar y compartir recetas. La aplicación también busca incluir funcionalidades avanzadas como filtros de búsqueda, seguimiento de usuarios, y un sistema de notificaciones. A continuación se muestran las pruebas de integración realizadas para la parte de los perfiles, esto para asegurar que los componentes de la base de datos y el Backend funcionan correctamente y sin fallos.

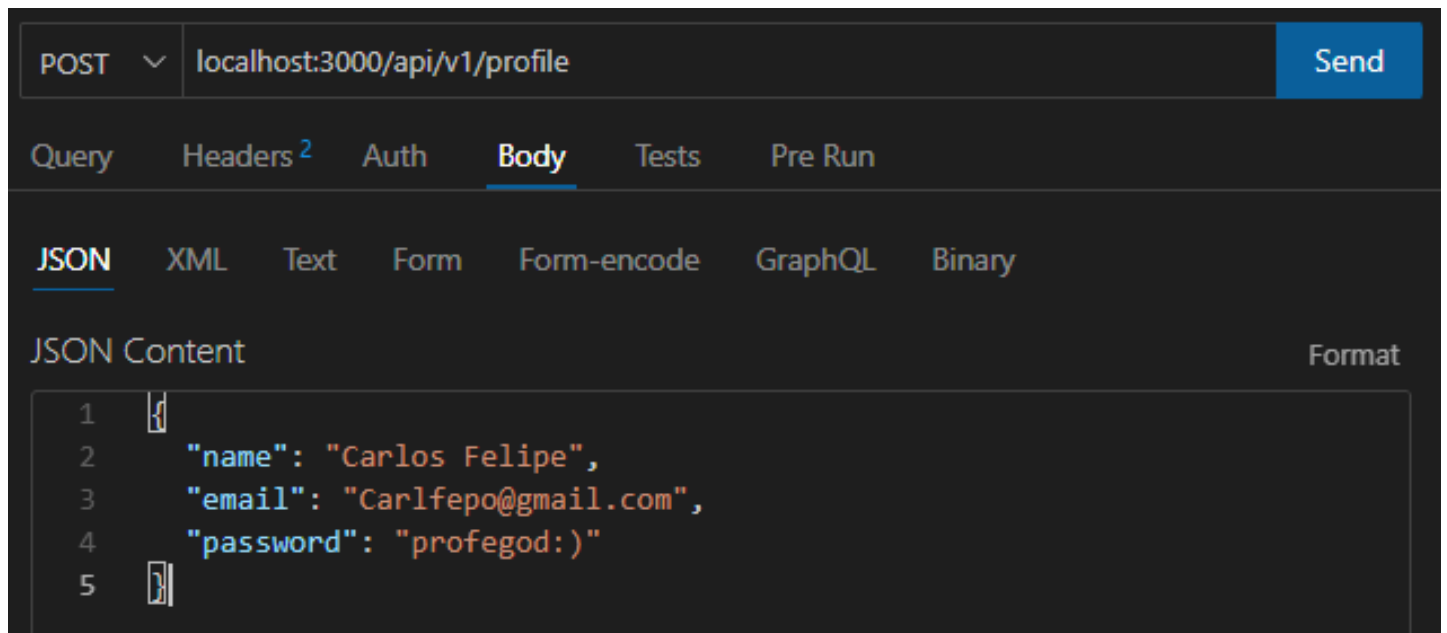
Resumen de los Tests Realizados

A continuación, se presenta una tabla con los detalles de los tests realizados por cada integrante del equipo.

Nombre del Integrante	Tipo de Prueba Realizada	Descripción Breve del Componente Probado	Herramienta o Framework Usado
Stiven Aguirre Granada	Prueba de integración	Creación de usuario en la plataforma.	Thunder Client
Stiven Aguirre Granada	Prueba de integración	Búsqueda de todos los usuarios en la plataforma.	Thunder Client
Cesar Camilo Velandia Cubillos	Prueba de integración	Búsqueda de usuario en la plataforma por id.	Thunder Client
Cesar Camilo Velandia Cubillos	Prueba de integración	Eliminación de usuario en la plataforma por id.	Thunder Client
Manuel Santiago Mori Ardila	Prueba de integración	Creación de recetas: Recibe y da una respuesta adecuada	Postman Post-response tests
Manuel Santiago Mori Ardila	Prueba de integración	Eliminación de receta en la plataforma por id	Postman Post-response tests

Nombre del Integrante	Tipo de Prueba Realizada	Descripción Breve del Componente Probado	Herramienta o Framework Usado
Stiven Aguirre Granada	Prueba de integración	Creación de usuario en la plataforma.	Thunder Client

Screenshot del Código del Test



Resultado de la Ejecución (Screenshot)

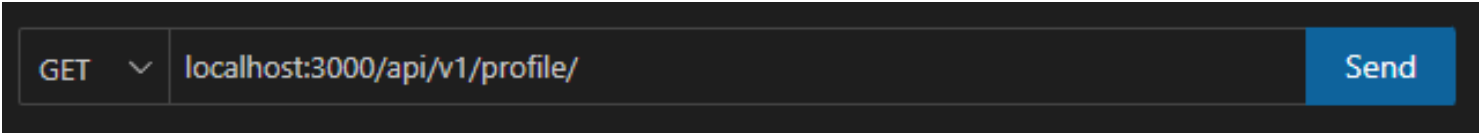
Status: 201 Created Size: 121 Bytes Time: 156 ms

Response Headers⁶ Cookies Results Docs

```
1  {
2    "id": "1a615791-ac2e-41a7-90d6-d857ac15b69b",
3    "name": "Carlos Felipe",
4    "email": "Carlfepo@gmail.com",
5    "password": "profegod:)"
6  }
```

Nombre del Integrante	Tipo de Prueba Realizada	Descripción Breve del Componente Probado	Herramienta o Framework Usado
Stiven Aguirre Granada	Prueba de integración	Búsqueda de todos los usuarios en la plataforma.	Thunder Client

Screenshot del Código del Test



Resultado de la Ejecución (Screenshot)

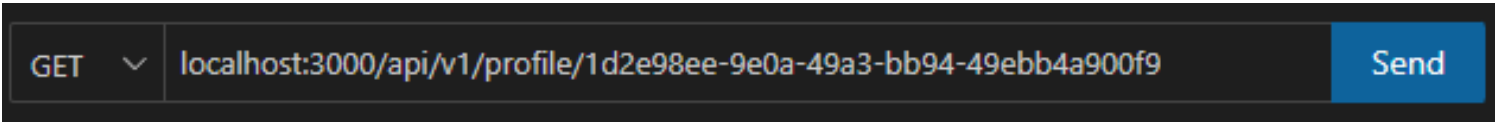
Status: 200 OK Size: 749 Bytes Time: 18 ms

Response Headers ⁶ Cookies Results Docs

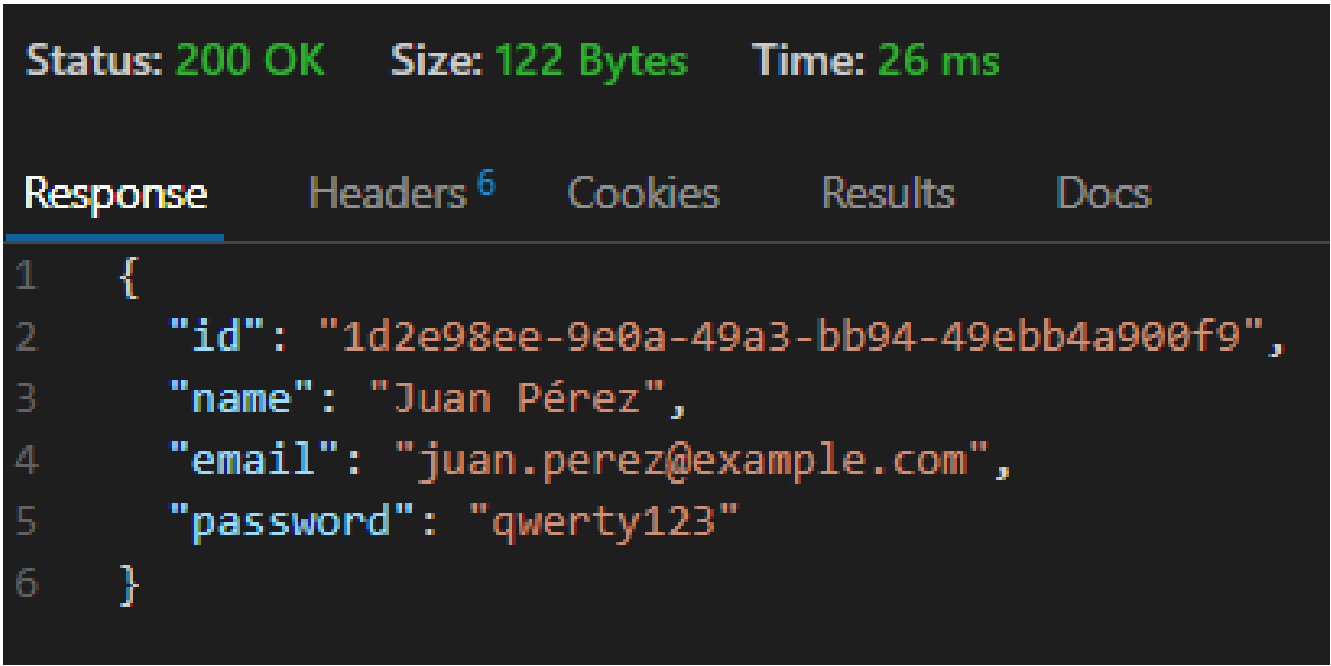
```
1  [
2    {
3      "id": "1d2e98ee-9e0a-49a3-bb94-49ebb4a900f9",
4      "name": "Juan Pérez",
5      "email": "juan.perez@example.com",
6      "password": "qwerty123"
7    },
8    {
9      "id": "570f3422-fb6d-4cfe-942d-eb4081457b56",
10     "name": "Carlos Felipe",
11     "email": "Carlfepo@gmail.com",
12     "password": "profegod:)"
13   },
14   {
15     "id": "761efa36-35ae-40e1-b543-26c9ec780d51",
16     "name": "Carlos Mendoza",
17     "email": "carlos.mendoza@example.com",
18     "password": "password123"
19   },
20   {
21     "id": "82a6b80d-350e-40c2-a408-c6e3b2235d86",
22     "name": "Laura Fernández",
23     "email": "laura.fernandez@example.com",
24     "password": "securepassword"
25   },
26   {
27     "id": "91f67331-4f8e-4014-83c5-a39afa8b2ec5",
28     "name": "Aize",
29     "email": "Aize@gmail.com",
30     "password": "1234"
31   },
32   {
33     "id": "dfbd07e3-c89e-4100-afcc-47e8f1bec9f3",
34     "name": "Pedro Sánchez",
35     "email": "pedro.sanchez@example.com",
36     "password": "1234abcd56"
37   }
38 ]
```

Nombre del Integrante	Tipo de Prueba Realizada	Descripción Breve del Componente Probado	Herramienta o Framework Usado
Cesar Camilo Velandia Cubillos	Prueba de integración	Búsqueda de usuario en la plataforma por id.	Thunder Client

Screenshot del Código del Test

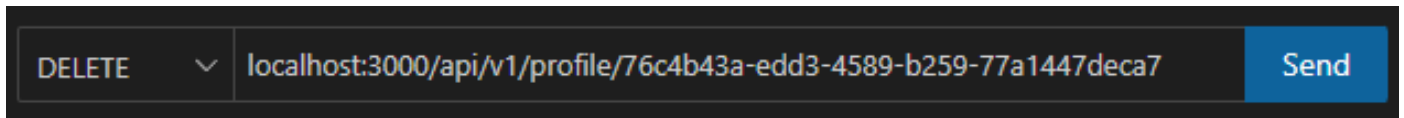


Resultado de la Ejecución (Screenshot)

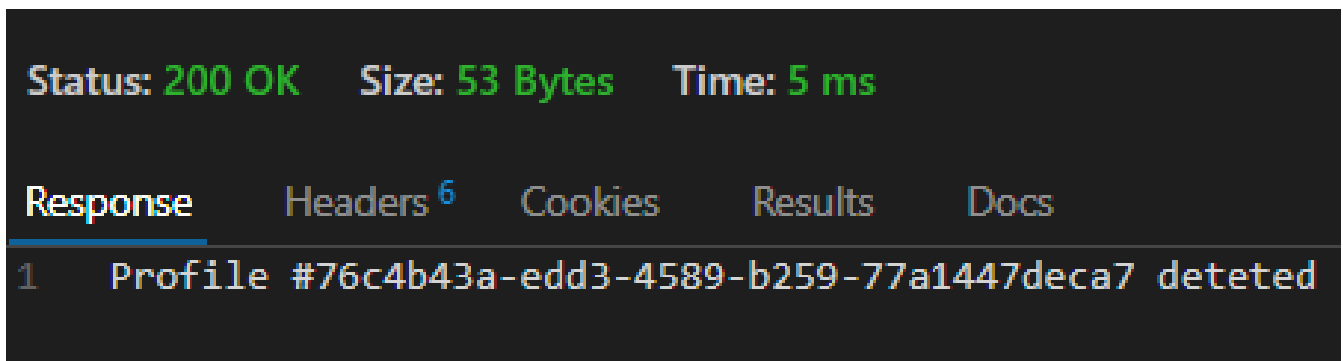


Nombre del Integrante	Tipo de Prueba Realizada	Descripción Breve del Componente Probado	Herramienta o Framework Usado
Cesar Camilo Velandia Cubillos	Prueba de integración	Eliminación de usuario en la plataforma por id.	Thunder Client

Screenshot del Código del Test



Resultado de la Ejecución (Screenshot)



Nombre del Integrante	Tipo de Prueba Realizada	Descripción Breve del Componente Probado	Herramienta o Framework Usado
Manuel Santiago Mori Ardila	Prueba de integración	Creación de recetas: Recibe y da una respuesta adecuada	Postman Post-response tests

Screenshot del Código del Test

POST

http://localhost:3000/api/v1/recipes

ParamsAuthorizationHeaders (9)Body ●Scripts ●Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

12

1

2

3

4

5

6

7

"title": "Tarta de Chocolate",

"description": "Una deliciosa tarta de chocolate",

"tags": "{\\"categoria\\": \\"postre\\", \\"nivel\\": \\"fácil\\"}",

"ingredients": "[{\\"name\\": \\"Harina\\", \\"amount\\": \\"200g\\"}, {\\"name\\": \\"Chocolate\\", \\"amount\\": \\"100g\\"}]",

"profileId": "e7b5f5a0-6f5b-4a3c-b99a-1d52c5f9c654"

ParamsAuthorizationHeaders (9)Body ●Scripts ●Settings

Pre-request

Post-response ●

1

2

3

4

5

6

7

8

9

10

11

12

13

pm.test("La receta fue publicada correctamente", function () {

var jsonData = pm.response.json();

pm.expect(jsonData).to.have.property("id");

pm.expect(jsonData).to.have.property("tags");

pm.expect(jsonData).to.have.property("ingredients");

pm.expect(jsonData.title).to.eql("Tarta de Chocolate");

pm.expect(jsonData.description).to.eql("Una deliciosa tarta de chocolate");

});

pm.test("La respuesta fue rápida (menos de 200 ms)", function () {

pm.expect(pm.response.responseTime).to.be.below(200);

});

Resultado de la Ejecución (Screenshot)

localhost:3000/api/v1/recipes

Dar formato al texto

```
[
  {
    "id": "ff51dbe6-96f6-4e2b-a859-9cda5d79face",
    "title": "Tarta de Chocolate",
    "description": "Una deliciosa tarta de chocolate",
    "creation_date": "2025-03-03T22:26:44.657Z",
    "tags": {
      "nivel": "fácil",
      "categoria": "postre"
    },
    "ingredients": [
      {
        "name": "Harina",
        "amount": "200g"
      },
      {
        "name": "Chocolate",
        "amount": "100g"
      }
    ],
    "profileId": "e7b5f5a0-6f5b-4a3c-b99a-1d52c5f9c654",
    "profile": {
      "id": "e7b5f5a0-6f5b-4a3c-b99a-1d52c5f9c654",
      "name": "pepito",
      "email": "pepito@gmail.com",
      "password": "123"
    },
    "ratings": [],
    "multimedia": []
  }
]
```

Body Cookies Headers (7) Test Results (2/2)

Filter Results

201 Created • 26 ms

PASSED La receta fue publicada correctamente

PASSED La respuesta fue rápida (menos de 200 ms)

*También se incluyó un test de rendimiento para evaluar la velocidad de la respuesta

Nombre del Integrante	Tipo de Prueba Realizada	Descripción Breve del Componente Probado	Herramienta o Framework Usado
Manuel Santiago Mori Ardila	Prueba de integración	Eliminación de receta en la plataforma por id	Postman Post-response tests

Screenshot del Código del Test

The screenshot displays the Postman interface. At the top, the URL bar shows `localhost:3000/api/v1/recipes`. Below it, a JSON body is visible, representing a recipe object with fields like `id`, `title`, `description`, `creation_date`, `tags`, `ingredients`, `profileId`, `profile`, `ratings`, and `multimedia`.

The main part of the screenshot shows a REST client request configured with the method **DELETE** and the URL `http://localhost:3000/api/v1/recipes/439a3bc8-7372-401d-97f1-941e15dc4ffc`. Below the request, the **Post-response** test script is shown, which contains the following code:

```

1 pm.test("La receta se elimina correctamente SI EXISTE", function (done) {
2   let recipeId = pm.request.url.variables[0]; // Id de la url
3
4   pm.sendRequest({
5     url: 'http://localhost:3000/recipes/${recipeId}', // Mediante get, verificar si existe
6     method: "GET"
7   }, function (err, res) {
8     if (res.status === 200) {
9       pm.test("La receta existe y se puede eliminar", function () {
10         pm.expect(pm.response.code).to.be.oneOf([200, 204]);
11       });
12     }
13     done();
14   });
15 });

```

Resultado de la Ejecución (Screenshot)

200 OK • 9 ms

BodyCookiesHeaders (7)Test Results (1/1)⌚

Filter Results ▾

PASSED

La receta se elimina correctamente SI EXISTE

⬅➡🔄ⓘ

localhost:3000/api/v1/recipes

☐ | ★ star+

Dar formato al texto ☐

[]

Lecciones Aprendidas y Dificultades

Durante el proceso de pruebas, adquirimos una valiosa experiencia en el uso de herramientas y técnicas para validar las funcionalidades de la aplicación, tales como **Thunder Client**, **Prisma**, **Docker** y **Postman**. A continuación se detallan algunas de las lecciones aprendidas y dificultades encontradas:

- **Lecciones Aprendidas:**
 - La implementación de pruebas de integración fue esencial para validar que el backend y la base de datos interactúan de manera eficiente.
 - Usar **Thunder Client** resultó ser una herramienta eficiente para realizar pruebas de solicitudes HTTP y obtener resultados claros y detallados.
 - Herramientas incluso como Postman, que uno podría pensar que no están tan enfocadas para el testing, ofrecen [documentación](#) y usos detallados para pruebas, haciendo posible hacer testing en cualquier lugar de la aplicación.
- **Dificultades:**
 - Uno de los principales retos fue asegurar que las rutas del **backend** estuvieran correctamente configuradas y accesibles para las pruebas.
 - A veces los datos en la base de datos se restablecían entre pruebas, lo que ocasionó algunos errores intermitentes. Para solucionar esto, revisamos el código hasta encontrar unos problemas en la conexión con la base de datos mediante la herramienta de **prisma**.
 - También, al ejecutar ciertas pruebas, el contenido de la base de datos cambia y puede hacer que la siguiente prueba diera un falso positivo o falso negativo. Un ejemplo sencillo para visualizar este problema es al hacer 2 deletes seguidos, el primero puede funcionar correctamente, mientras que el segundo intenta borrar un elemento que no existe y dar un error. Para esto hay que considerar muy bien los casos de prueba dentro del test y el estado de la base de datos antes y después.