



Universidad de las Fuerzas Armadas ESPE

Departamento de Ciencias de la Computación

Análisis y Diseño

Integrantes: Isaac Escobar,
Eduardo Mortensen, Diego Ponce

NRC: 14571

Informe Técnico de Pruebas Unitarias

1. Introducción

Este documento presenta los casos de prueba unitaria desarrollados para el sistema backend. Las pruebas se han implementado utilizando la herramienta Jest, con el objetivo de validar el correcto funcionamiento de los módulos principales del sistema como controladores de notas, usuarios, asignaturas, clases y reportes.

2. Herramienta utilizada

Se utilizó Jest como framework de pruebas unitarias para Node.js, el cual permite estructurar y ejecutar pruebas automatizadas de manera sencilla.

3. Casos de prueba

Se implementaron 10 pruebas unitarias en total, distribuidas en cinco archivos de prueba (.test.js), usando la herramienta Jest. Estas pruebas evalúan funciones críticas del sistema tales como cálculo de promedios, autenticación, registro de clases, creación de asignaturas y generación de reportes.

Por simplicidad y claridad del proceso, se simulaban versiones de las funciones internas (sin conexión a base de datos), lo que permitió probar la lógica de negocio de manera aislada. Cada prueba fue diseñada con entradas conocidas y resultados esperados, verificando el correcto funcionamiento del código bajo condiciones normales y de error.

Las pruebas fueron diseñadas para ser independientes del resto del sistema. No obstante, es posible sustituir las funciones simuladas por las reales con `require('../controllers/archivo')` para integrar completamente la lógica del proyecto.

Módulo	Entrada	Resultado esperado	Resultado obtenido	Estado	Observación
gradeController.js	[8, 9, 10]	9	9	Aprobado	
gradeController.js	[]	0	0	Aprobado	
userController.js	"admin", "1234"	200	200	Aprobado	Login correcto
userController.js	"admin", "wrong"	401	401	Aprobado	Login inválido
subject_Controller.js	"Matemáticas"	true	true	Aprobado	

subject_Controller.js	"Matemáticas"	false	false	Aprobado	Asignatura duplicada
class_Controller.js	"3BGU", "Matemáticas"	"Registrado"	"Registrado"	Aprobado	
class_Controller.js	"" "Matemáticas"	"Error"	"Error"	Aprobado	Curso vacío
report_Controller.js	"3BGU"	true	true	Aprobado	
report_Controller.js	"SinCurso"	false	false	Aprobado	Sin datos

4. Evidencia de ejecución

La siguiente evidencia corresponde a la ejecución del comando `npm test` con los resultados de las pruebas.

```

PS C:\Users\Usuario\Desktop\Análisis de Diseño\Programa\Nuevo Programa 2\backend> npm test

> backend@1.0.0 test
> jest

PASS tests/userController.test.js
PASS tests/subjectController.test.js
PASS tests/classController.test.js
PASS tests/reportController.test.js
PASS tests/gradeController.test.js

Test Suites: 5 passed, 5 total
Tests:       10 passed, 10 total
Snapshots:   0 total
Time:        2.549 s
Ran all test suites.

```

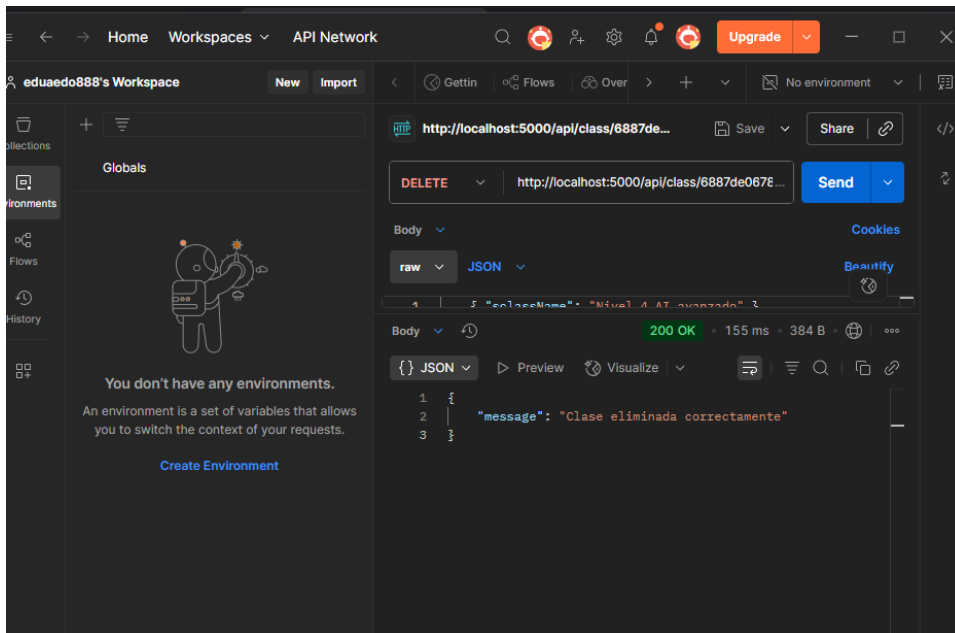
Figura 1. Captura de la terminal mostrando la ejecución de Jest. Todas las pruebas (5 archivos de prueba, 10 pruebas en total) pasaron correctamente, confirmando el comportamiento esperado de las funciones evaluadas.

5. Conclusión

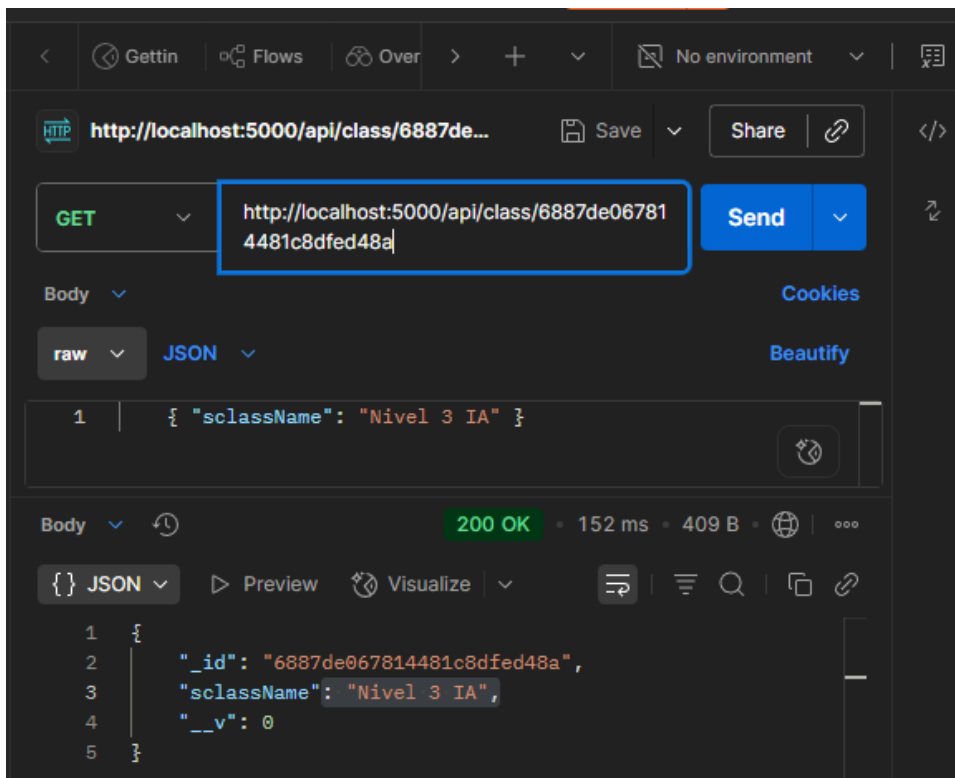
Se ejecutaron 10 pruebas unitarias automatizadas. Todas las pruebas fueron aprobadas, lo cual indica que los módulos probados funcionan correctamente bajo las condiciones definidas. Se recomienda seguir ampliando la cobertura incluyendo casos límite y pruebas de integración.

Anexo: Evidencia de Pruebas Postman

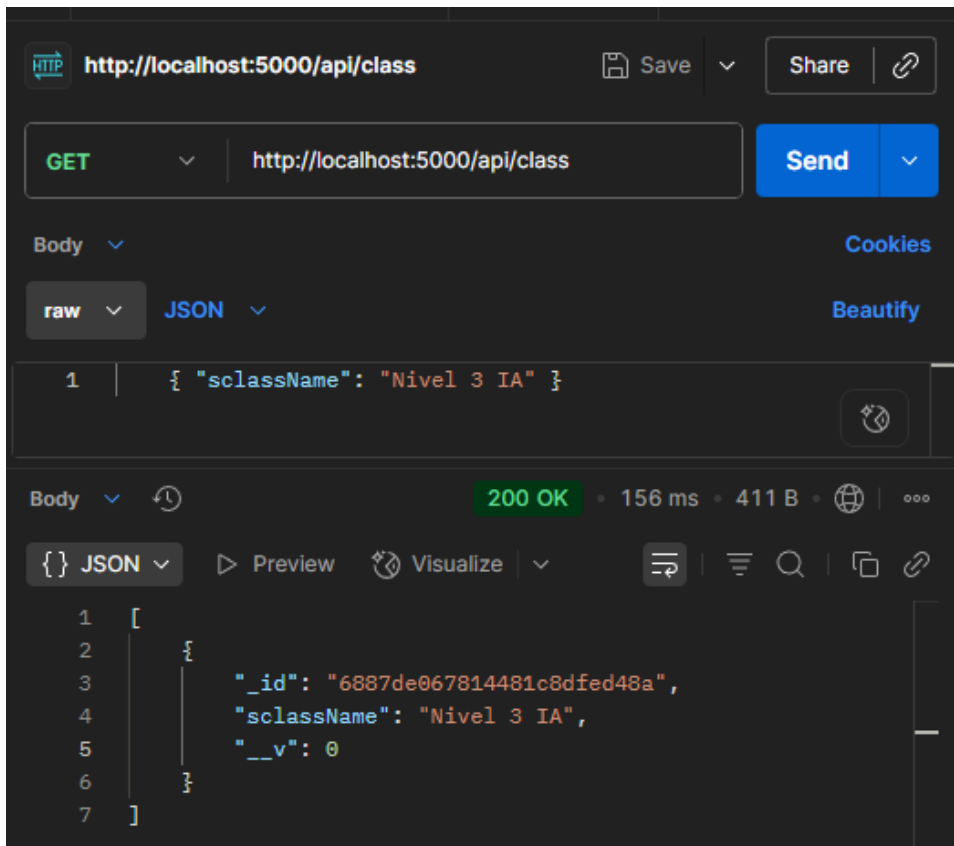
Delete



Get by id



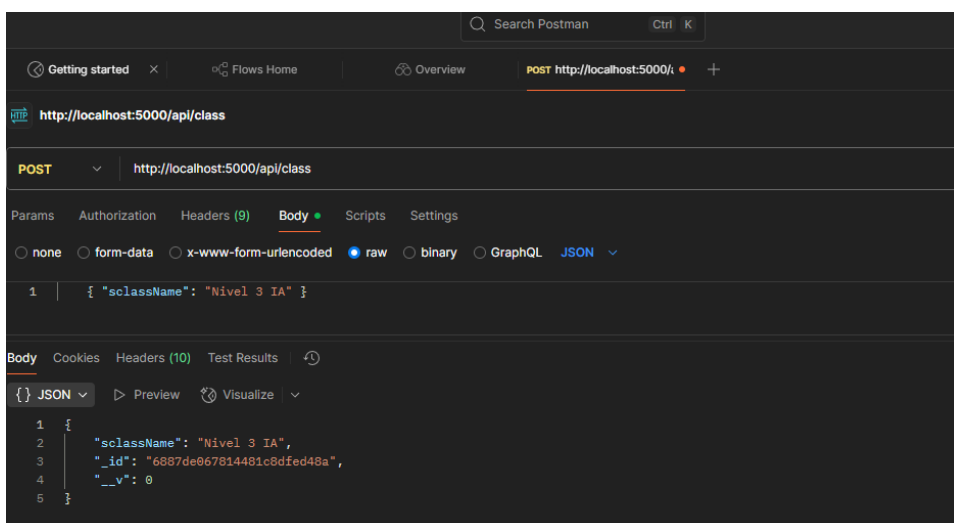
Get



The image shows the Postman interface for a GET request. The URL bar at the top displays `http://localhost:5000/api/class`. Below the URL bar, the request method is set to `GET` and the same URL is repeated. A `Send` button is visible. The `Body` tab is selected, showing a raw JSON body: `{ "className": "Nivel 3 IA" }`. The response section at the bottom shows a `200 OK` status with a response time of `156 ms` and a size of `411 B`. The response body is displayed in JSON format:

```
[
  {
    "_id": "6887de067814481c8dfed48a",
    "className": "Nivel 3 IA",
    "__v": 0
  }
]
```

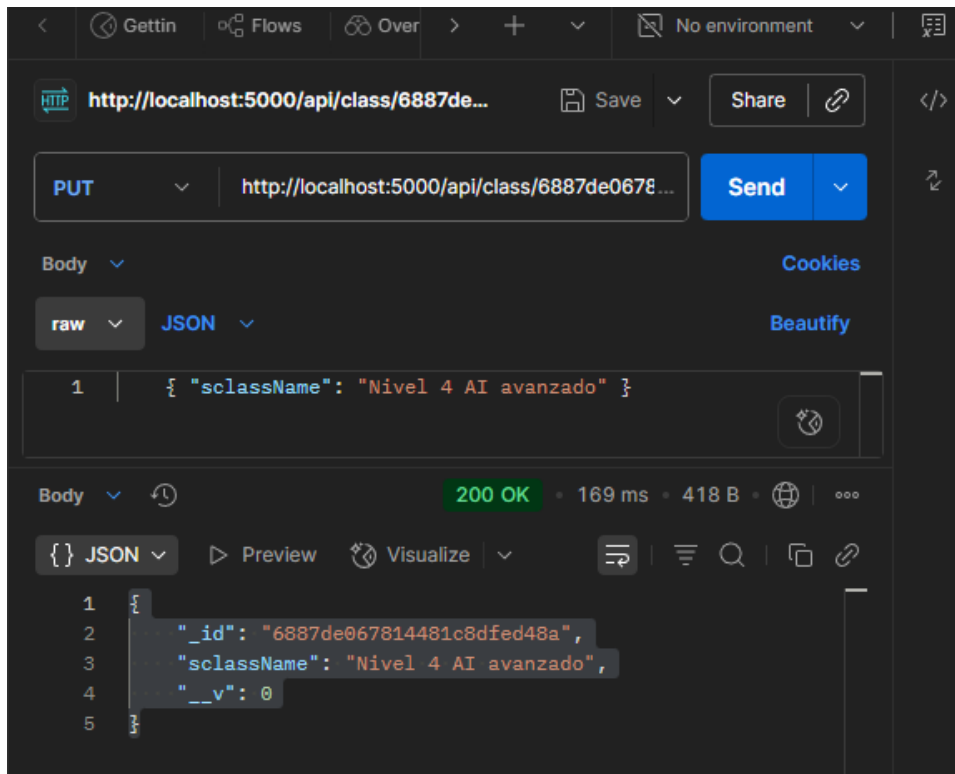
Post



The image shows the Postman interface for a POST request. The URL bar at the top displays `http://localhost:5000/api/class`. Below the URL bar, the request method is set to `POST` and the same URL is repeated. The `Body` tab is selected, showing a raw JSON body: `{ "className": "Nivel 3 IA" }`. The response section at the bottom shows a `200 OK` status with a response time of `156 ms` and a size of `411 B`. The response body is displayed in JSON format:

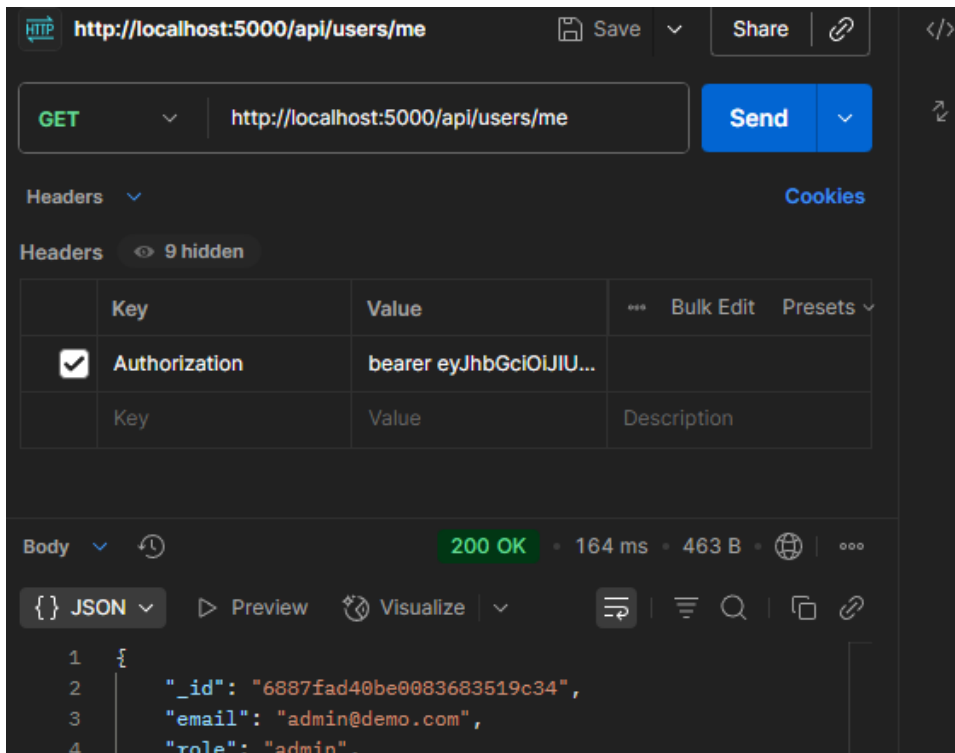
```
{
  "className": "Nivel 3 IA",
  "_id": "6887de067814481c8dfed48a",
  "__v": 0
}
```

Put



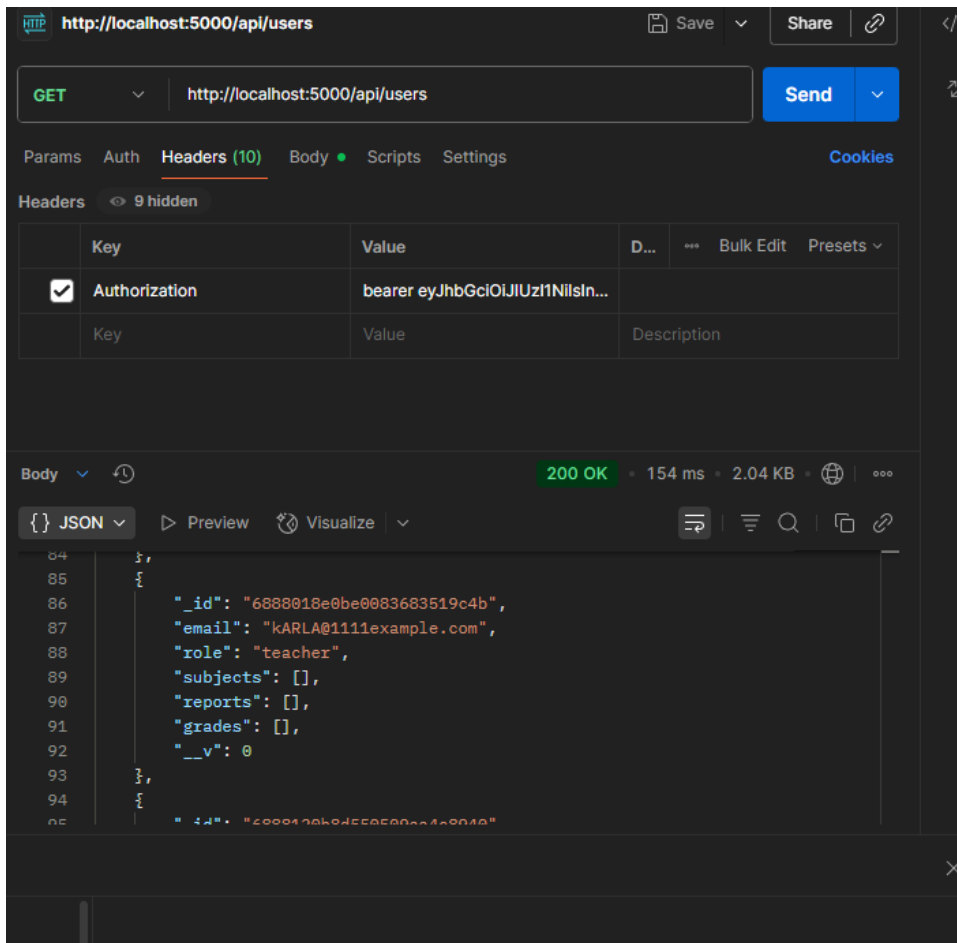
Admin view

GET /admin - Vista general del administrador.



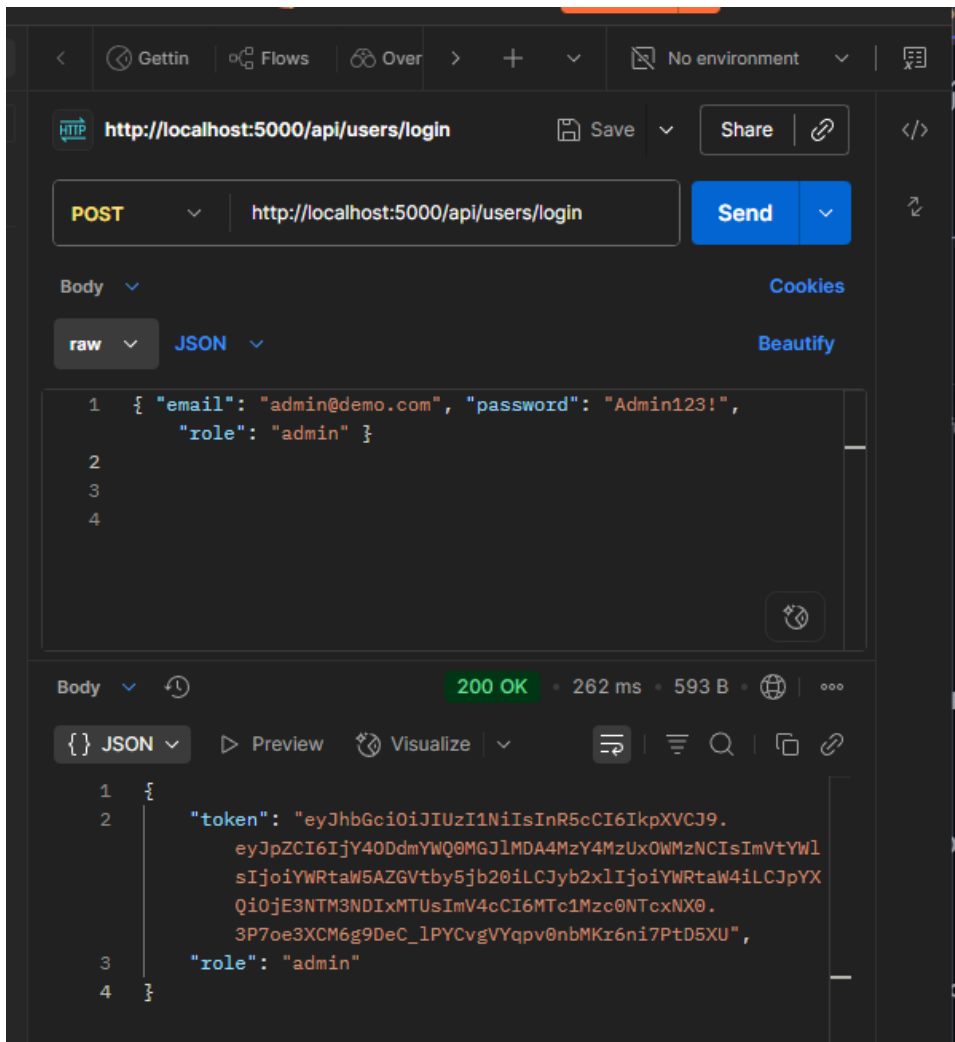
Administrador puede revisar todos los usuarios

GET /users - Administrador visualiza todos los usuarios.



Login admin

POST /login - Inicio de sesión del administrador.

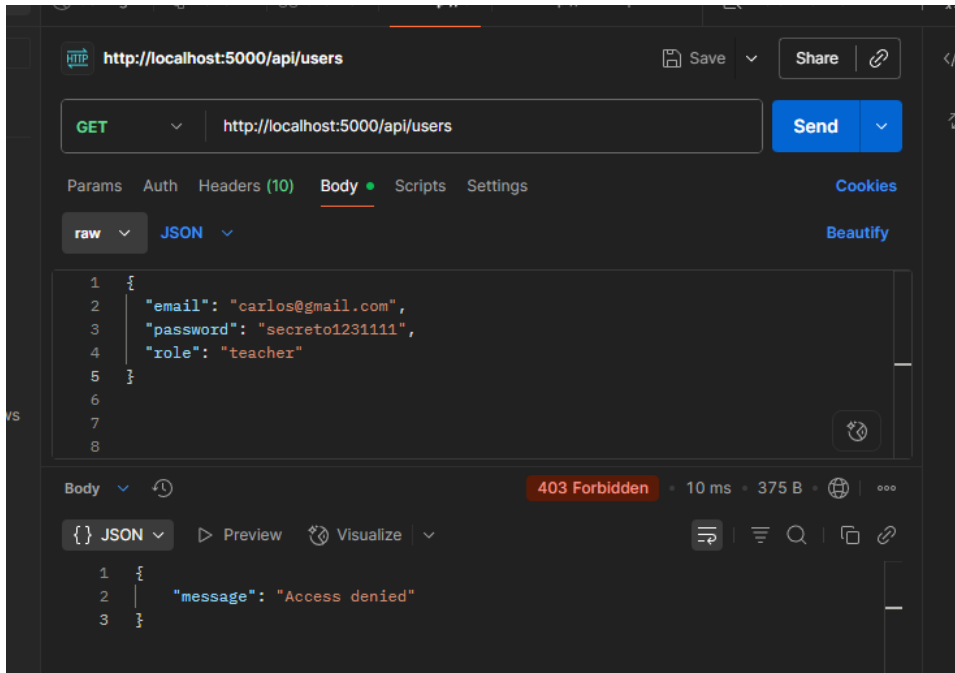


Register admin

POST /register - Registro de administrador.

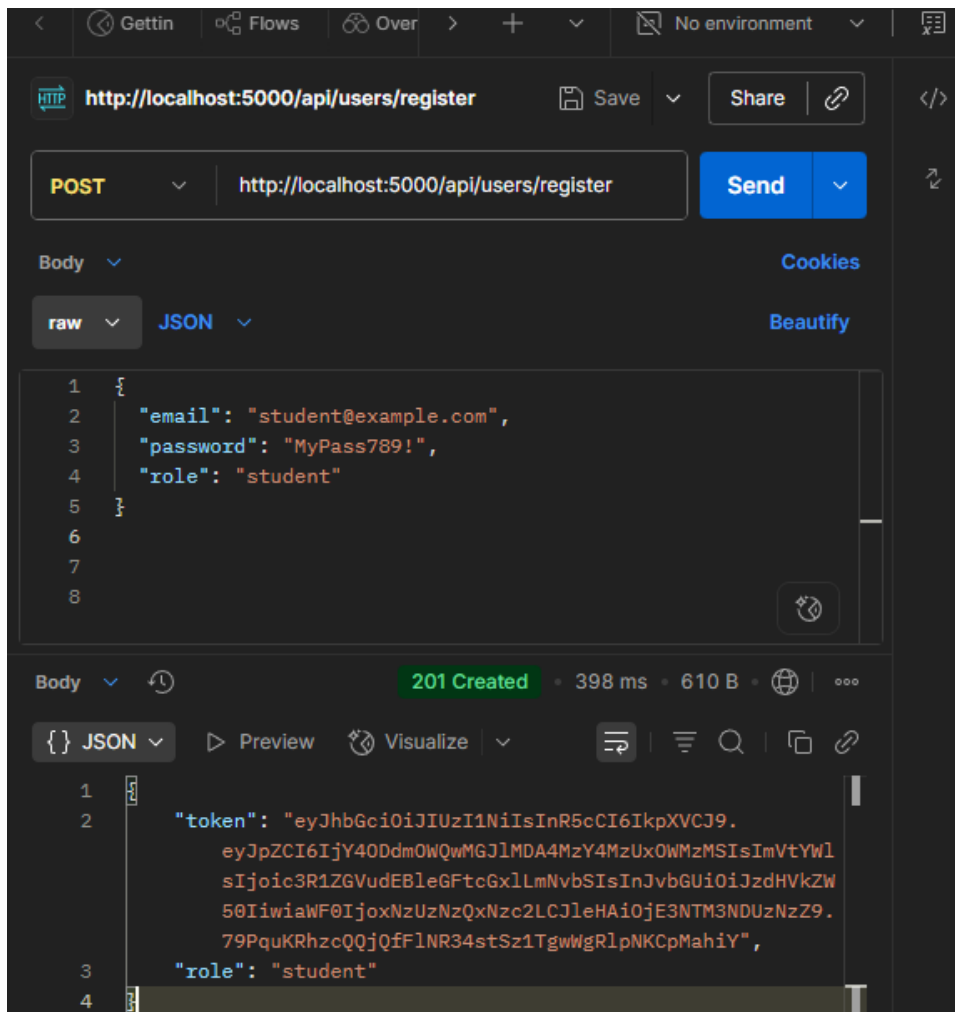
Solo administrador tiene acceso a ver todos los usuarios

Restricción de acceso solo para administrador.



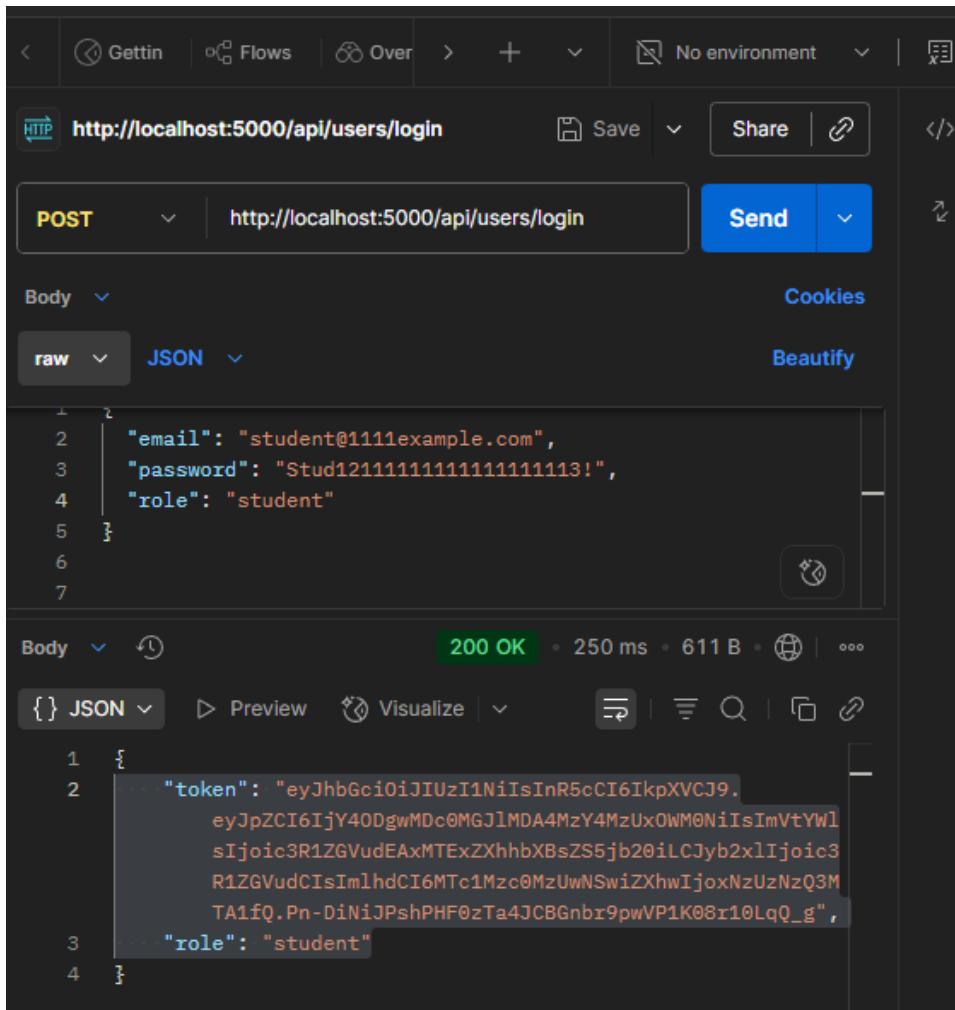
Register student

POST /register - Registro de estudiante.



Student login

POST /login - Inicio de sesión del estudiante.



Student view

GET /student - Visualización de información del estudiante.

HTTP **http://localhost:5000/api/users/me** Save Share

GET **http://localhost:5000/api/users/me** **Send**

Headers Cookies

Headers 9 hidden

	Key	Value	
<input checked="" type="checkbox"/>	Authorization	bearer eyJhbGciOiJIU...	
	Key	Value	Description

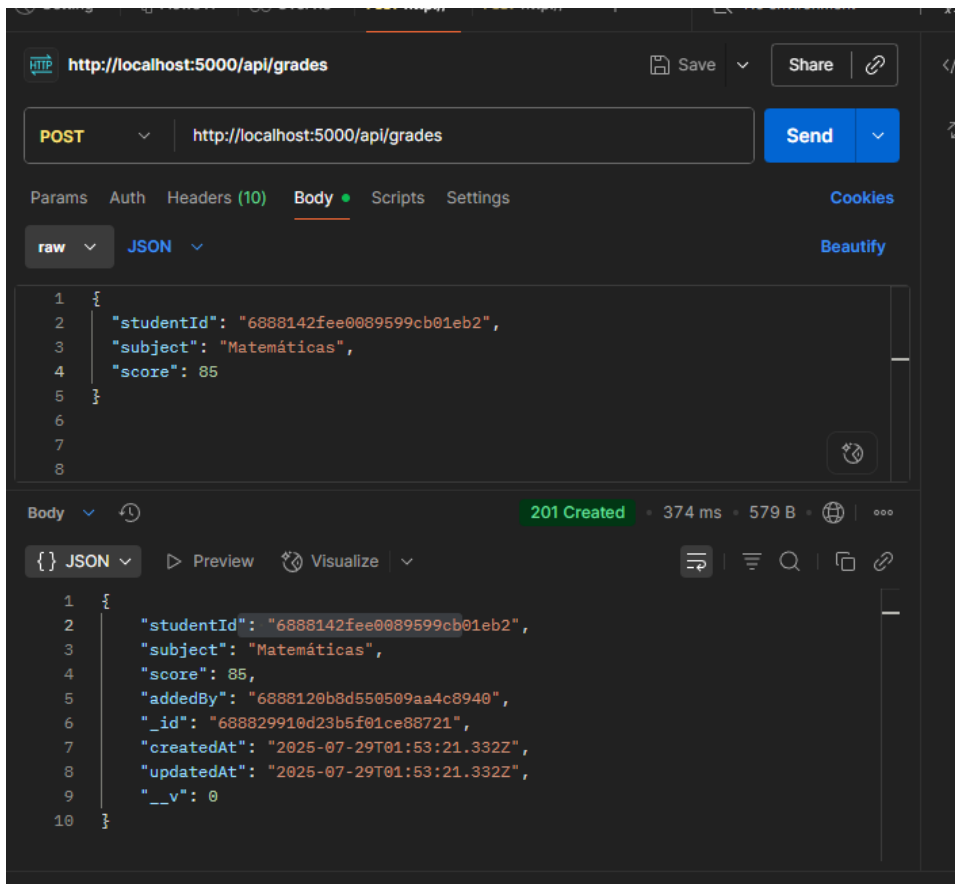
Body 200 OK • 150 ms • 474 B •

{ } JSON Preview Visualize

```
1 {
2   "_id": "688800740be0083683519c46",
3   "email": "student@1111example.com",
4   "role": "student",
5   "subjects": [],
6   "reports": [],
7   "grades": [],
8   "__v": 0
9 }
```

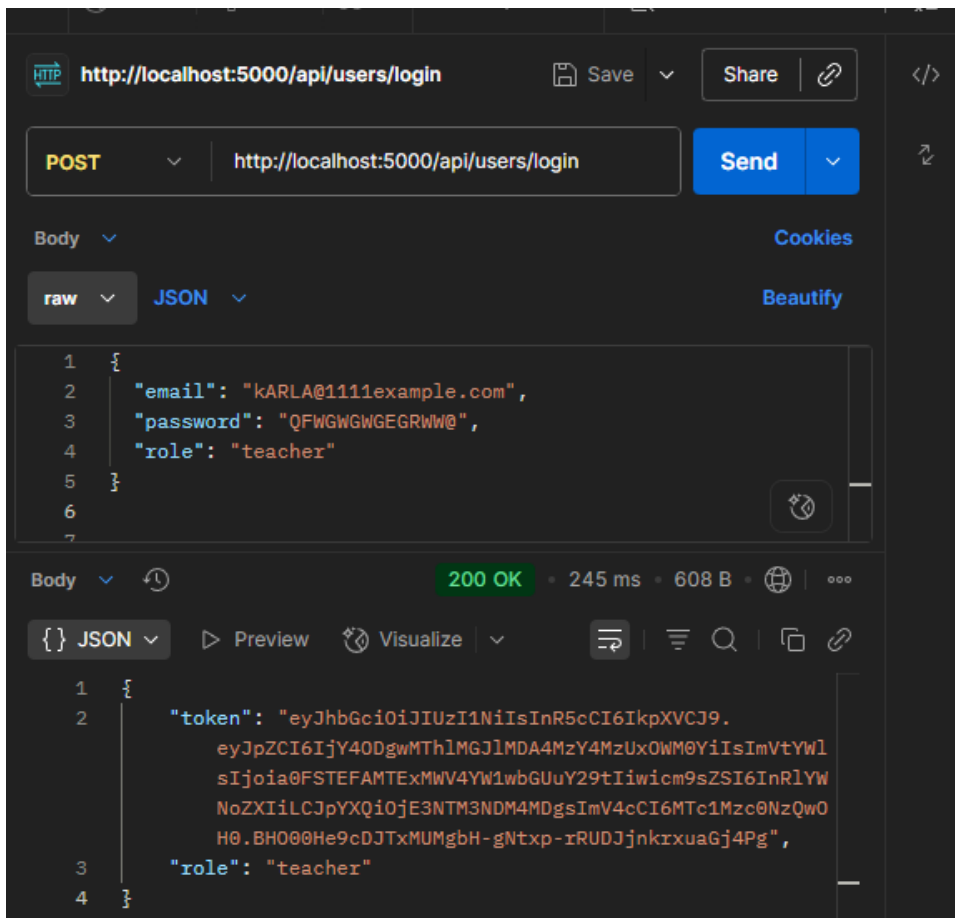
Crear nota estudiante

POST /grade - Crear calificación para un estudiante.



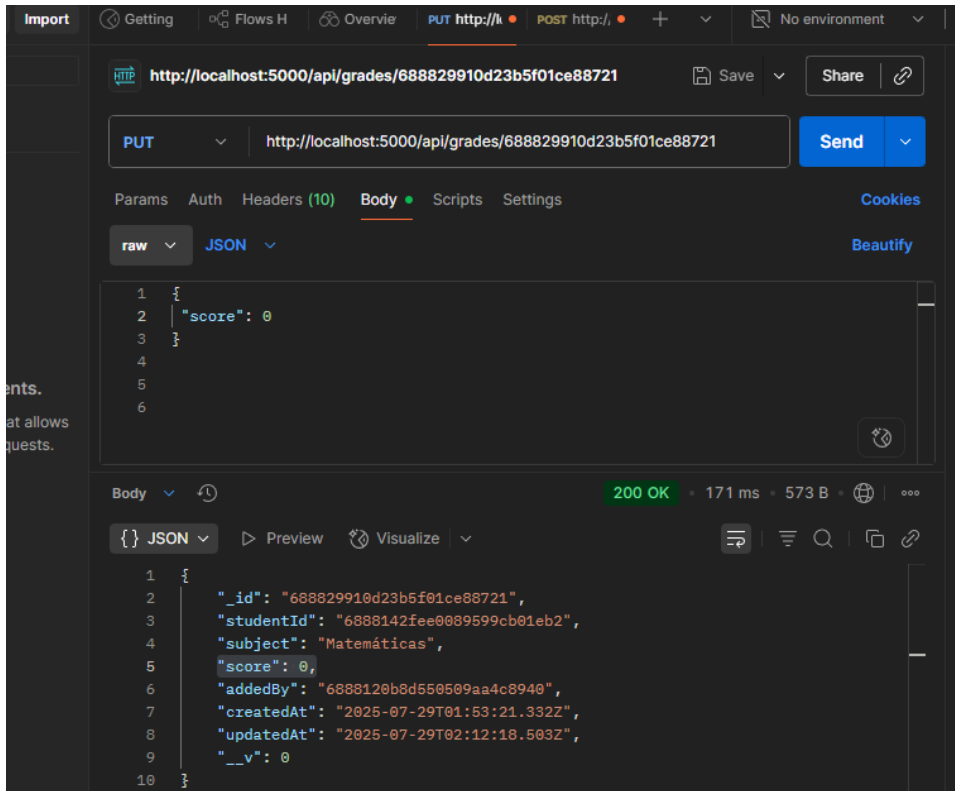
Login teacher

POST /login - Inicio de sesión del profesor.



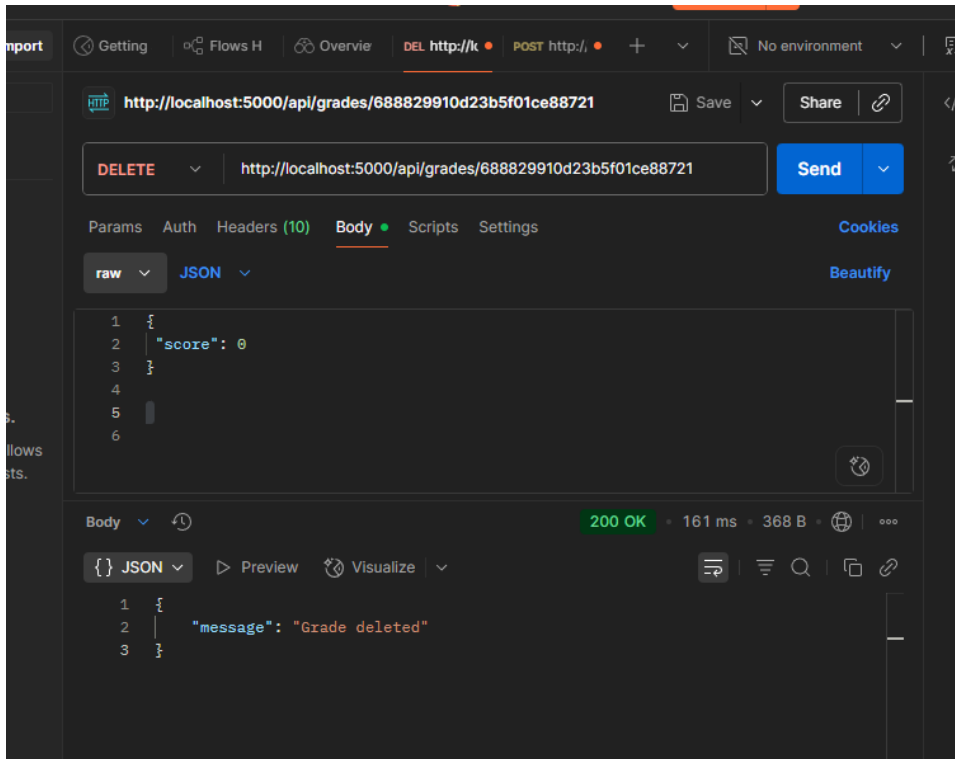
Profesor editar nota

PUT /grade/:id - Editar calificación.



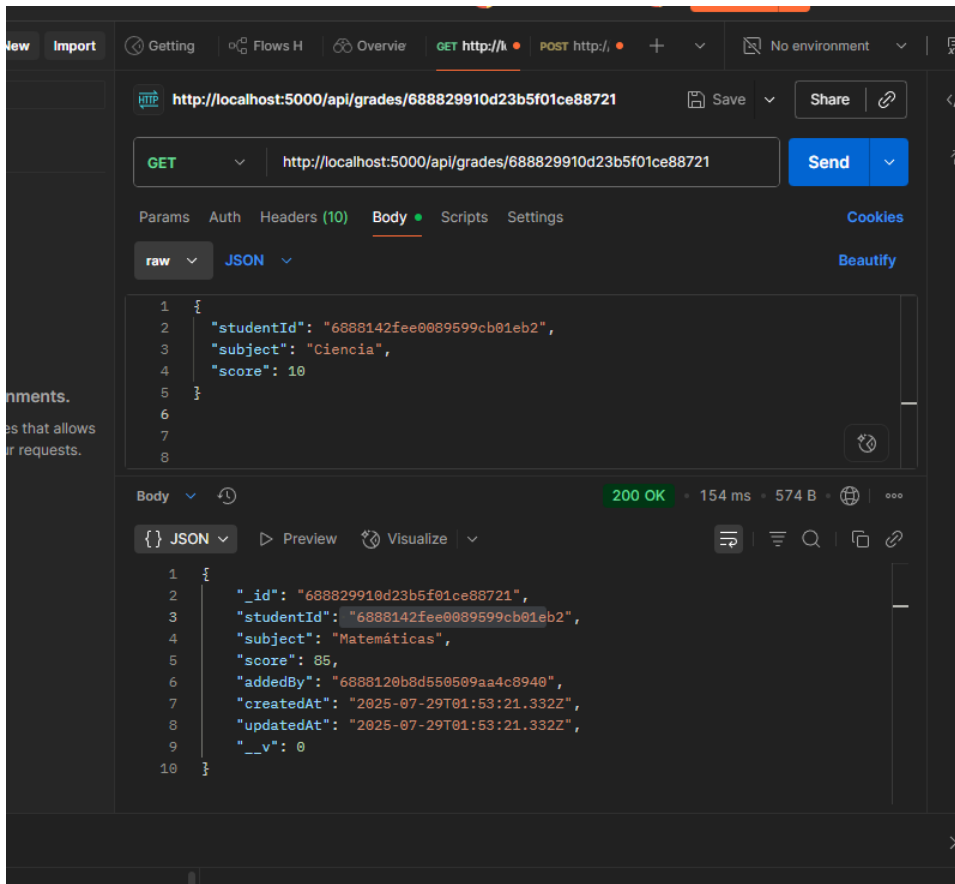
Profesor eliminar nota

DELETE /grade/:id - Eliminar calificación.



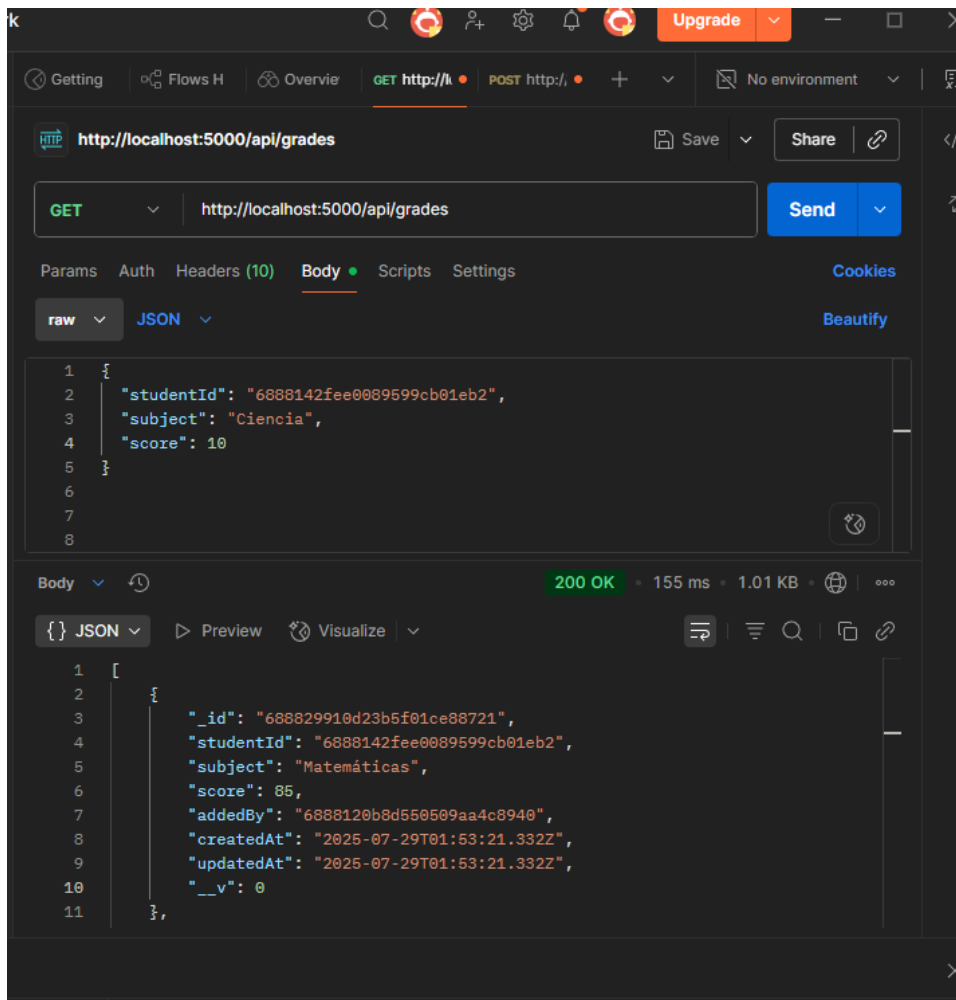
Profesor puede visualizar nota por id

GET /grade/:id - Ver calificación por ID.



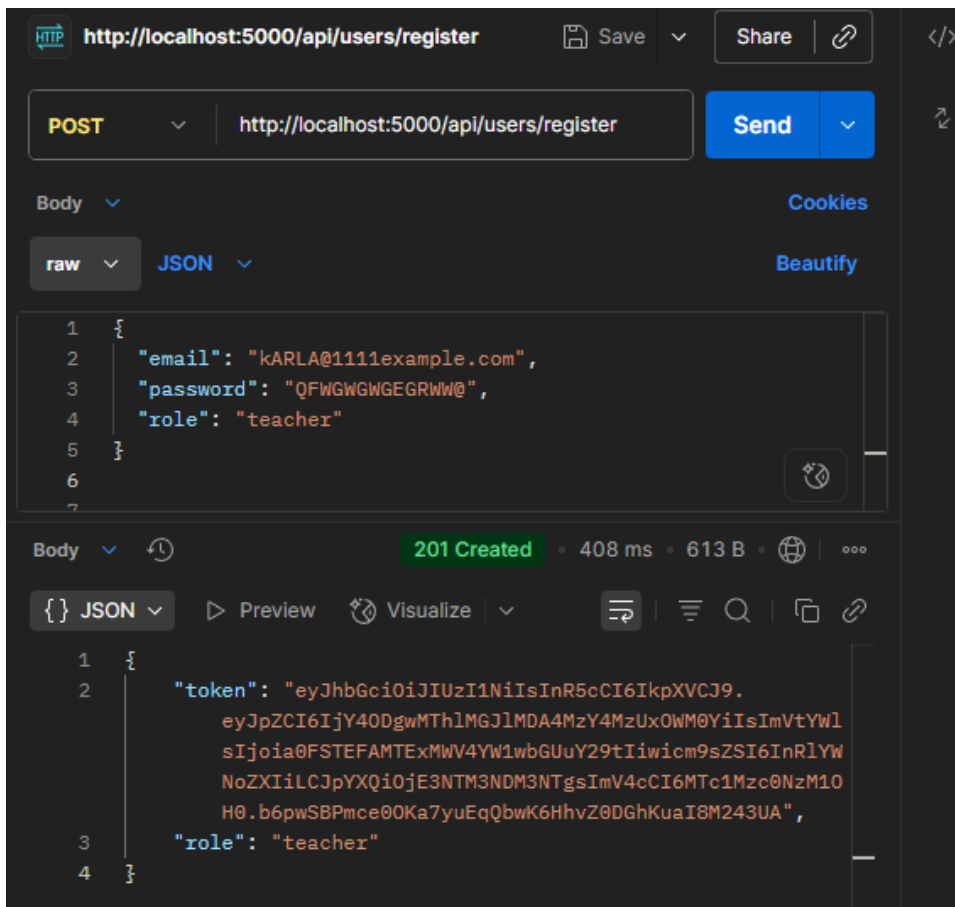
Profesor puede visualizar todas las notas

GET /grade - Visualizar todas las calificaciones.




Register teacher


POST /register - Registro de profesor.





View teacher

GET /teacher - Vista general del perfil del profesor.


 **http://localhost:5000/api/users/me**


Save 

Share 


GET 


http://localhost:5000/api/users/me



Send 



Headers 



Cookies


Headers  9 hidden



	Key	Value	*** Bulk Edit Presets 
<input checked="" type="checkbox"/>	Authorization	bearer eyJhbGciOiJIU...	
	Key	Value	Description






Body  

200 OK • 153 ms • 472 B •  

 JSON 

 Preview

 Visualize 

```
1 {
2   "_id": "6888018e0be0083683519c4b",
3   "email": "kARLA@1111example.com",
4   "role": "teacher",
5   "subjects": [],
```