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 simularon 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.

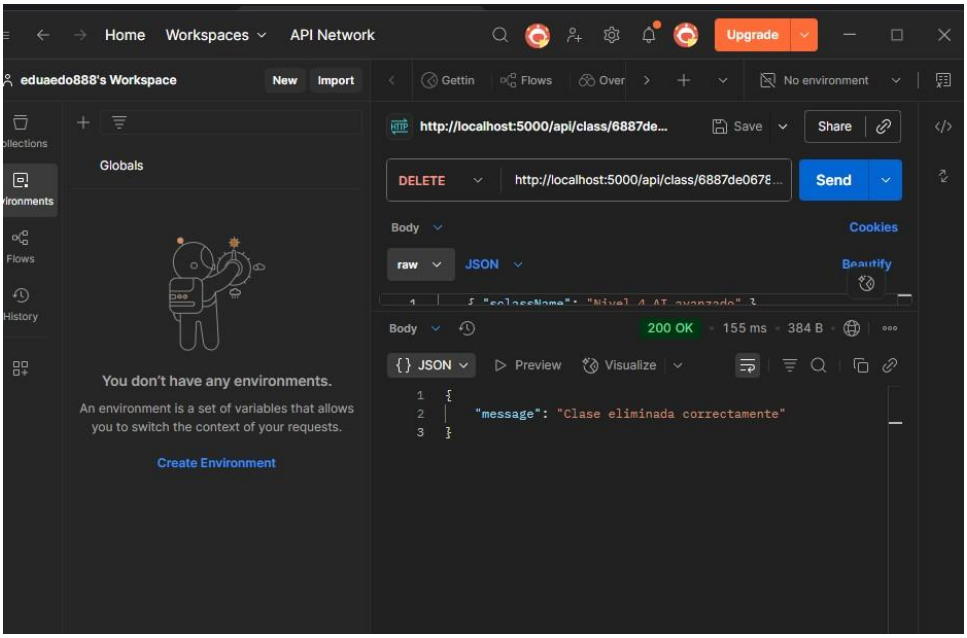
Figura 1. Captura de la terminal mostrando la ejecución de Jest. Todas las pruebas (5 archivos de prueba, 10 pruebas en total) pasaron correctamente, confirmando el comportamiento esperado de las funciones evaluadas.

5. Conclusión

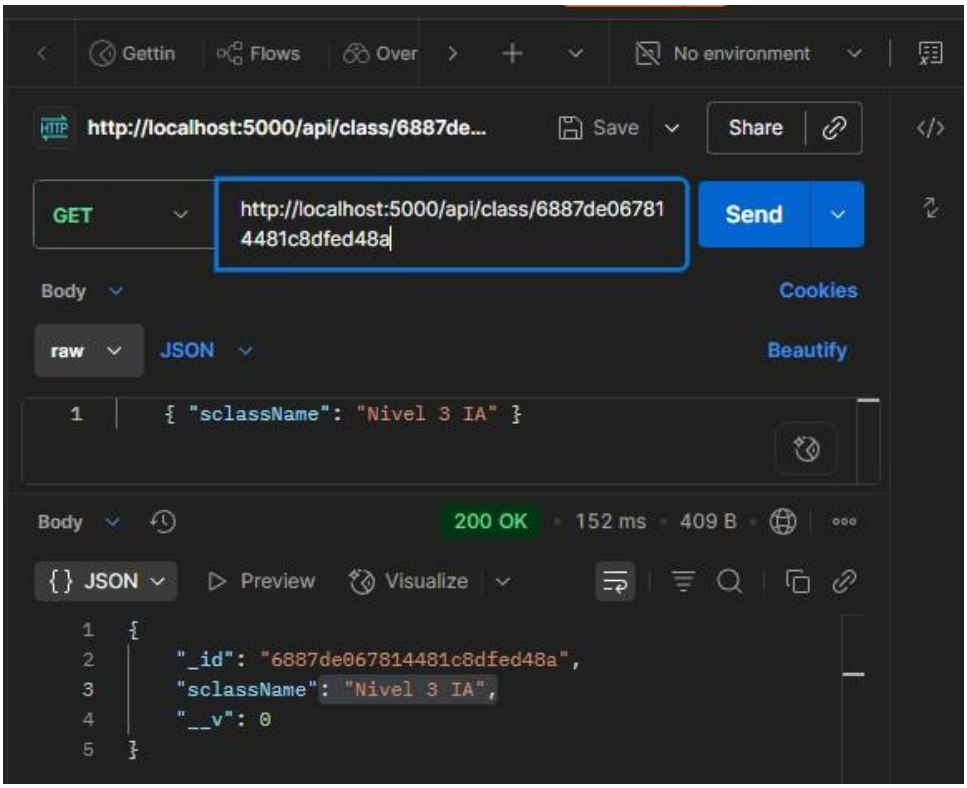
Se ejecutaron 10 pruebas unitarias automatizadas. Todas las pruebas fueron aprobadas, lo cual indica que los módulos probados funcionan correctamente bajo las condiciones definidas. Se recomienda seguir ampliando la cobertura incluyendo casos límite y pruebas de integración.

Anexo: Evidencia de Pruebas Postman

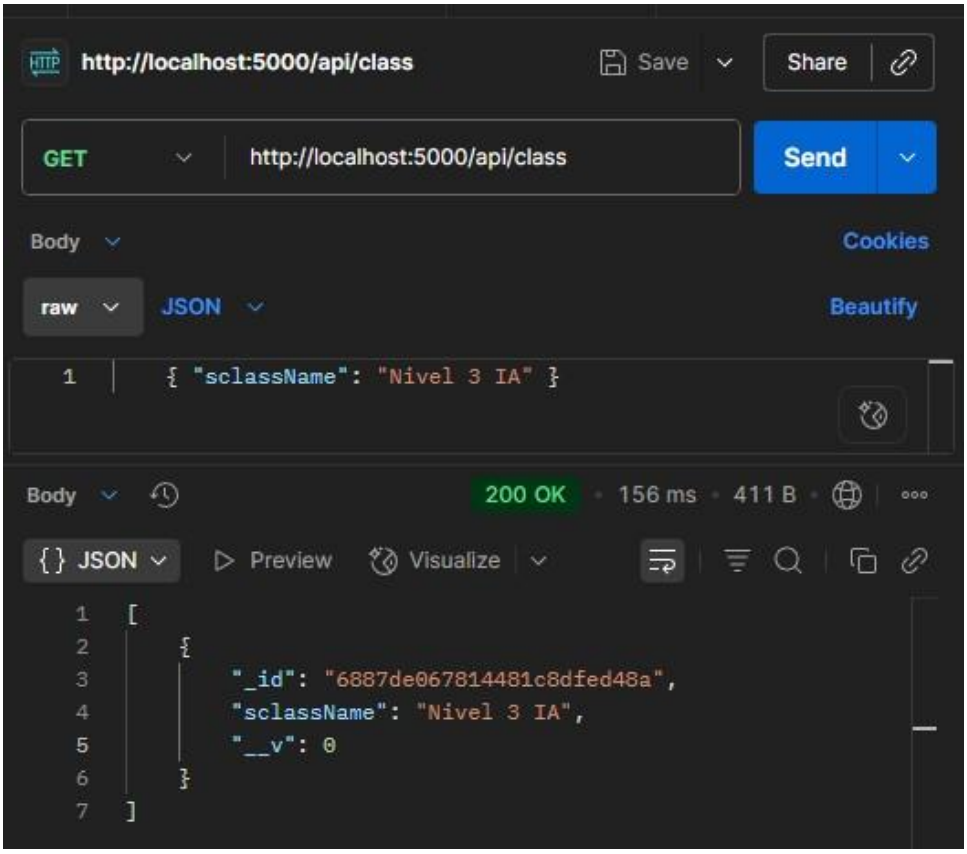
Delete



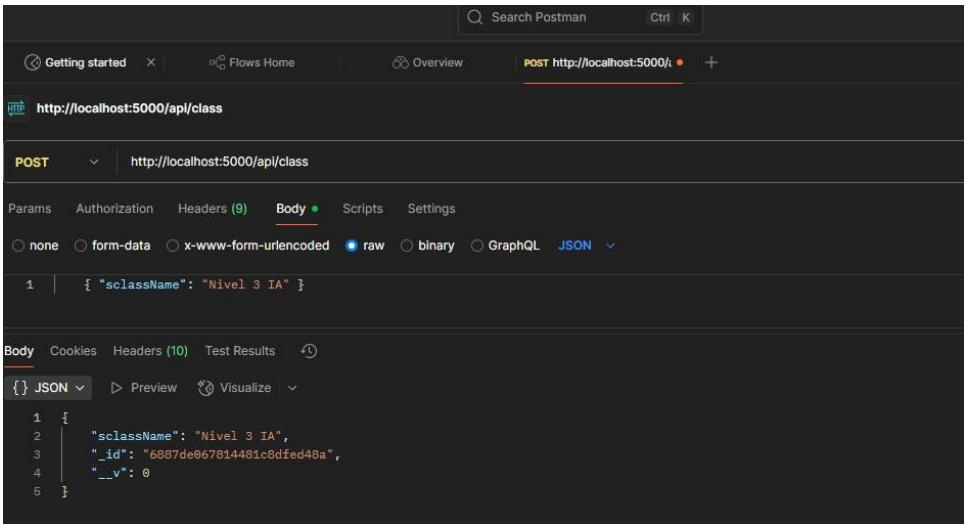
Get by id



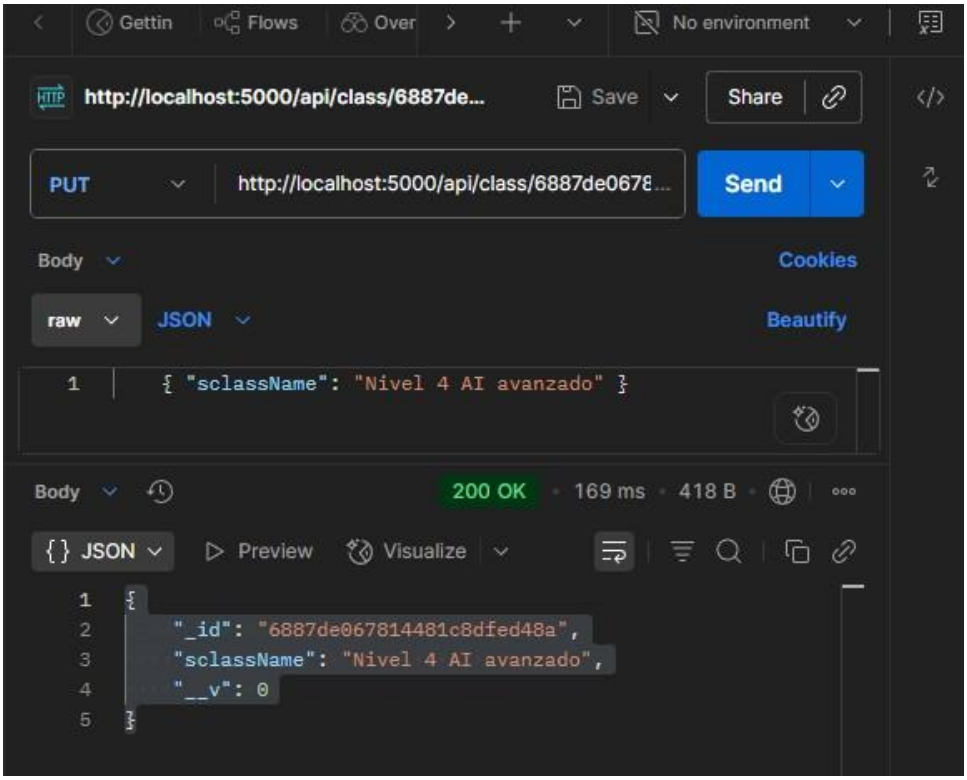
Get


http://localhost:5000/api/class
Save
Share
GET
http://localhost:5000/api/class
Send
Body
Cookies
raw
JSON
Beautify
1 { "sclassName": "Nivel 3 IA" }
Body
200 OK
156 ms
411 B
{ } JSON
Preview
Visualize
1 [
2 {
3 "_id": "6887de067814481c8dfed48a",
4 "sclassName": "Nivel 3 IA",
5 "__v": 0
6 }
7]

Post

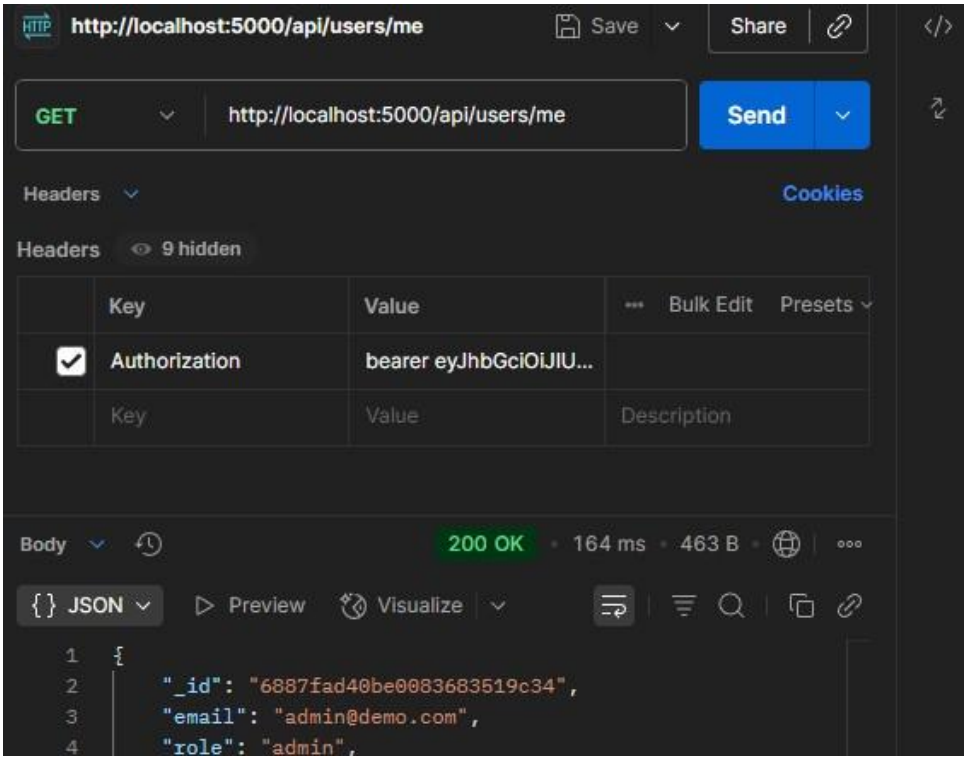

Search Postman
Ctrl K
Getting started
Flows Home
Overview
POST http://localhost:5000/
http://localhost:5000/api/class
POST
http://localhost:5000/api/class
Params
Authorization
Headers (9)
Body
Scripts
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
1 { "sclassName": "Nivel 3 IA" }
Body
Cookies
Headers (10)
Test Results
{ } JSON
Preview
Visualize
1 {
2 "sclassName": "Nivel 3 IA",
3 "_id": "6887de067814481c8dfed48a",
4 "__v": 0
5 }

Put



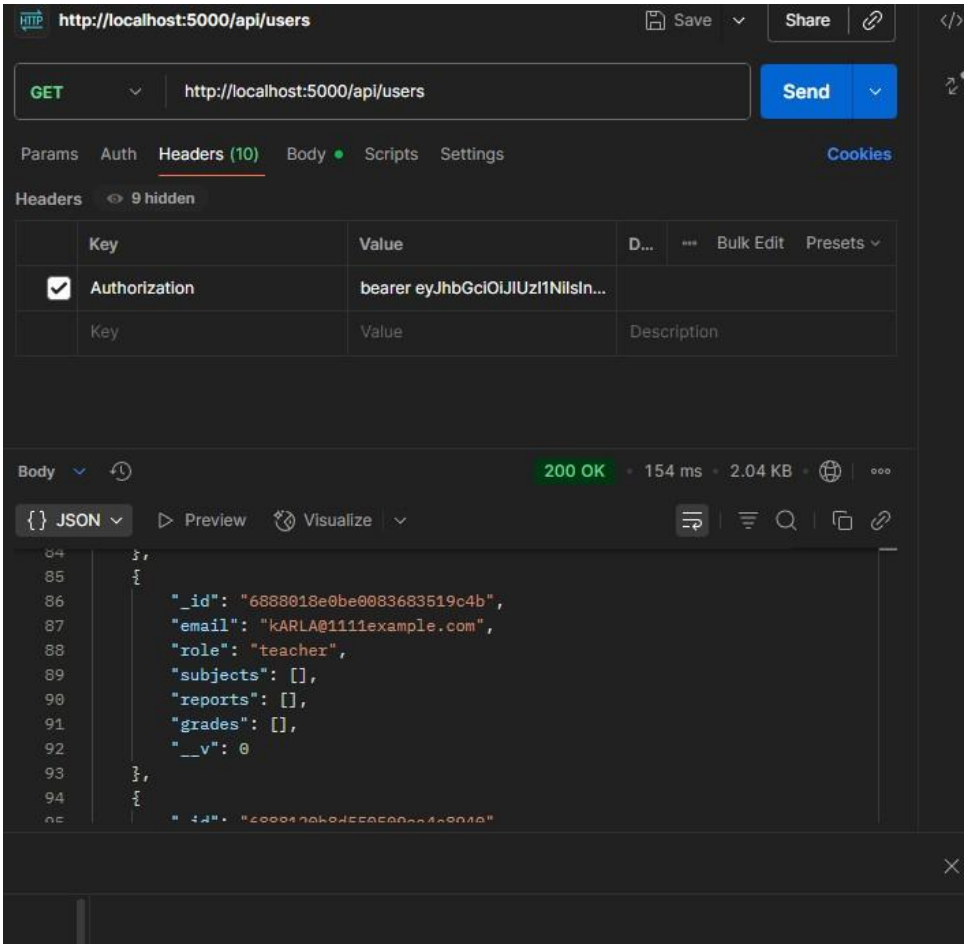
Admin view

GET /admin - Vista general del administrador.



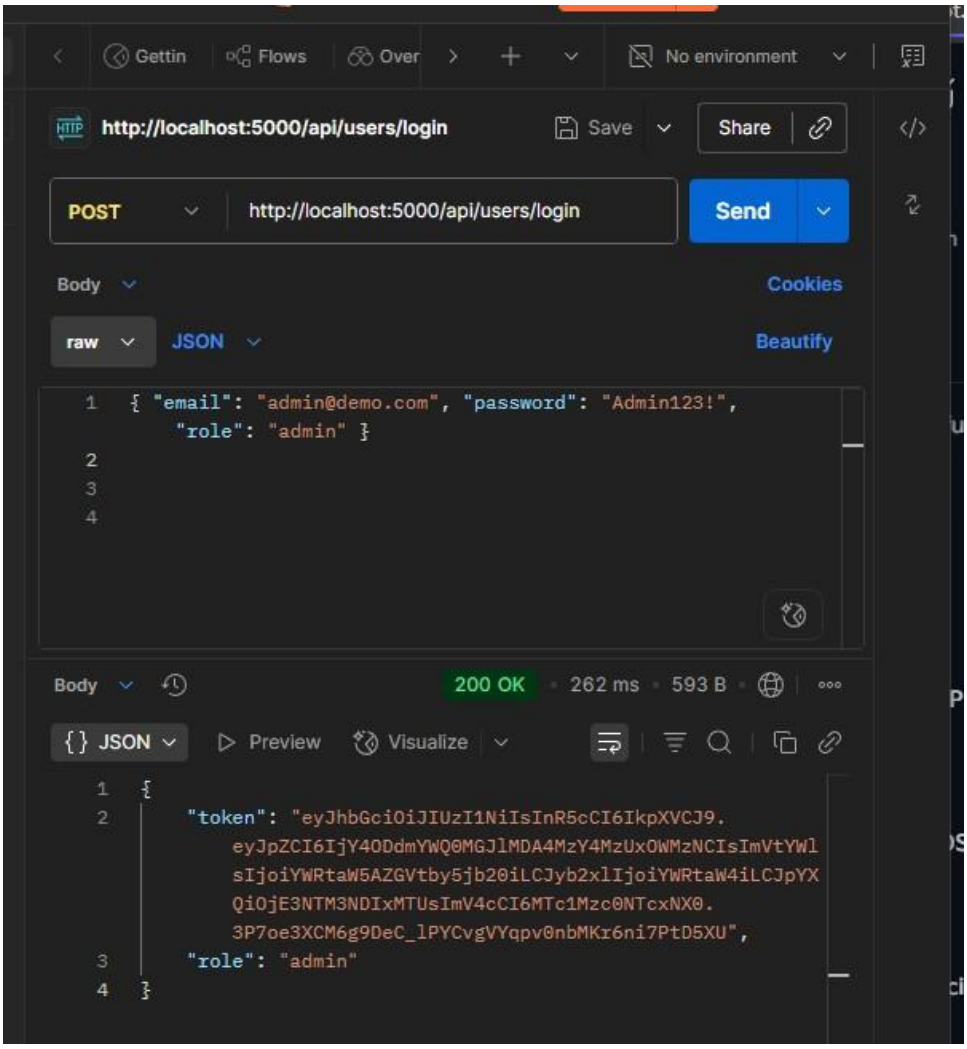
Administrador puede revisar todos los usuarios

GET /users - Administrador visualiza todos los usuarios.



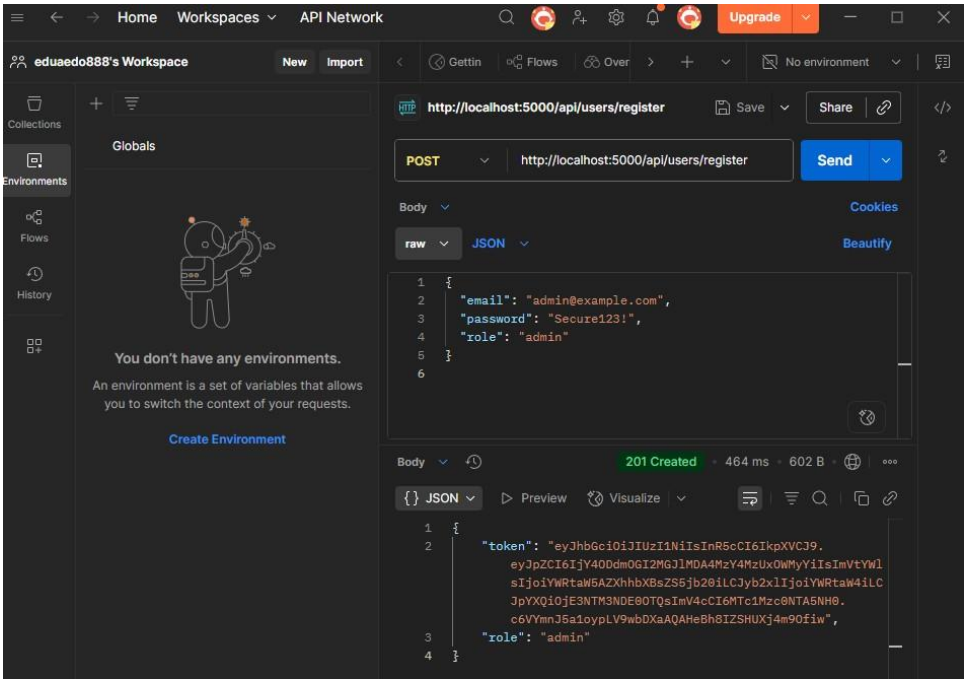
Login admin

POST /login - Inicio de sesión del administrador.



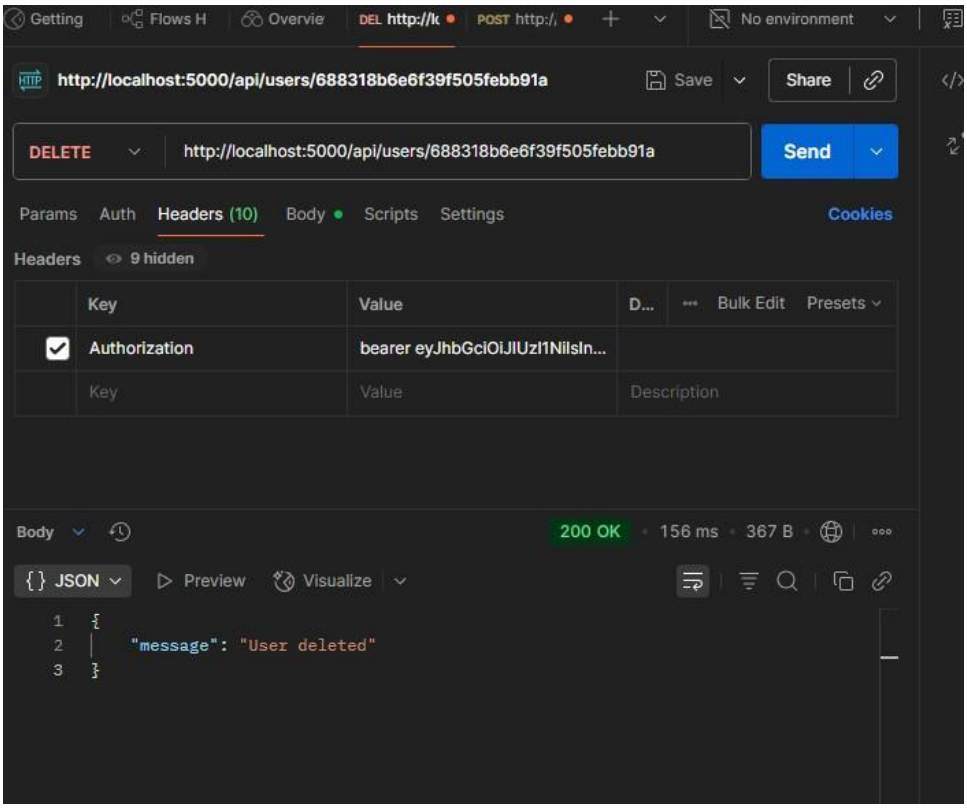
Register admin

POST /register - Registro de administrador.

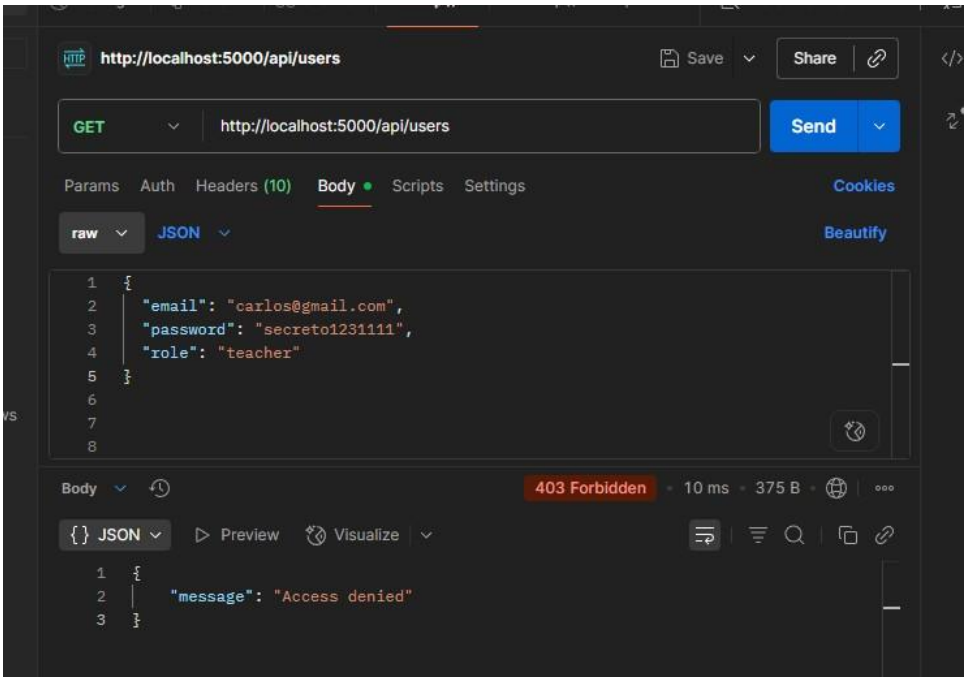


Solo administrador puede eliminar

DELETE /user/:id - Solo el administrador puede eliminar un usuario.

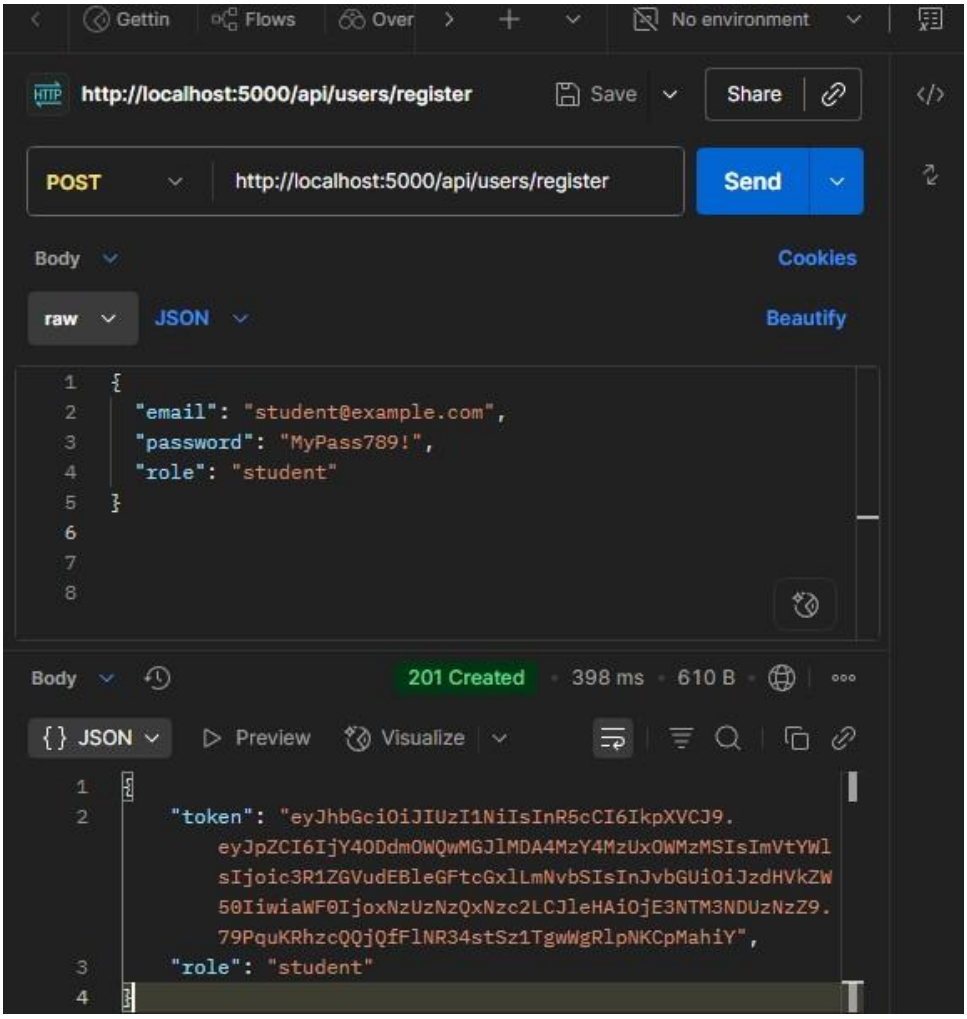


Solo administrador tiene accceso a ver todos los usuarios Restricción de acceso solo para administrador.



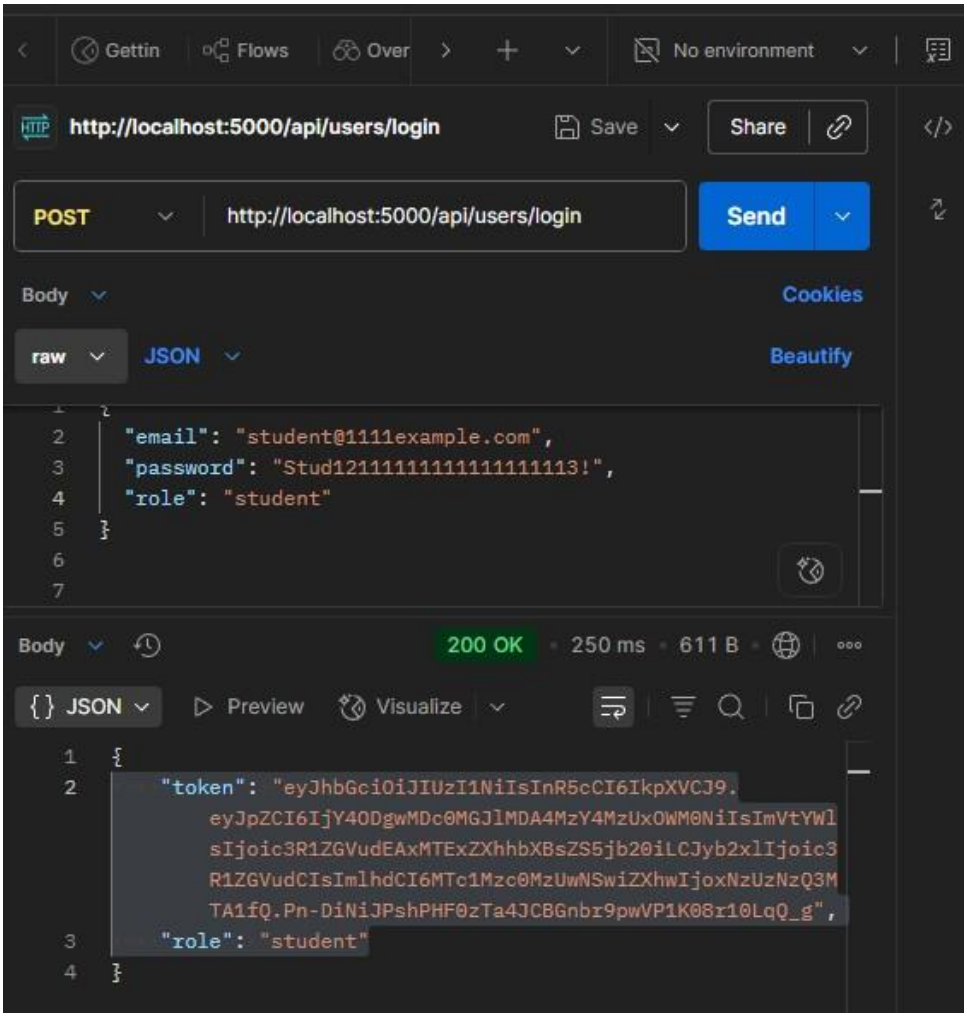
Register student

POST /register - Registro de estudiante.



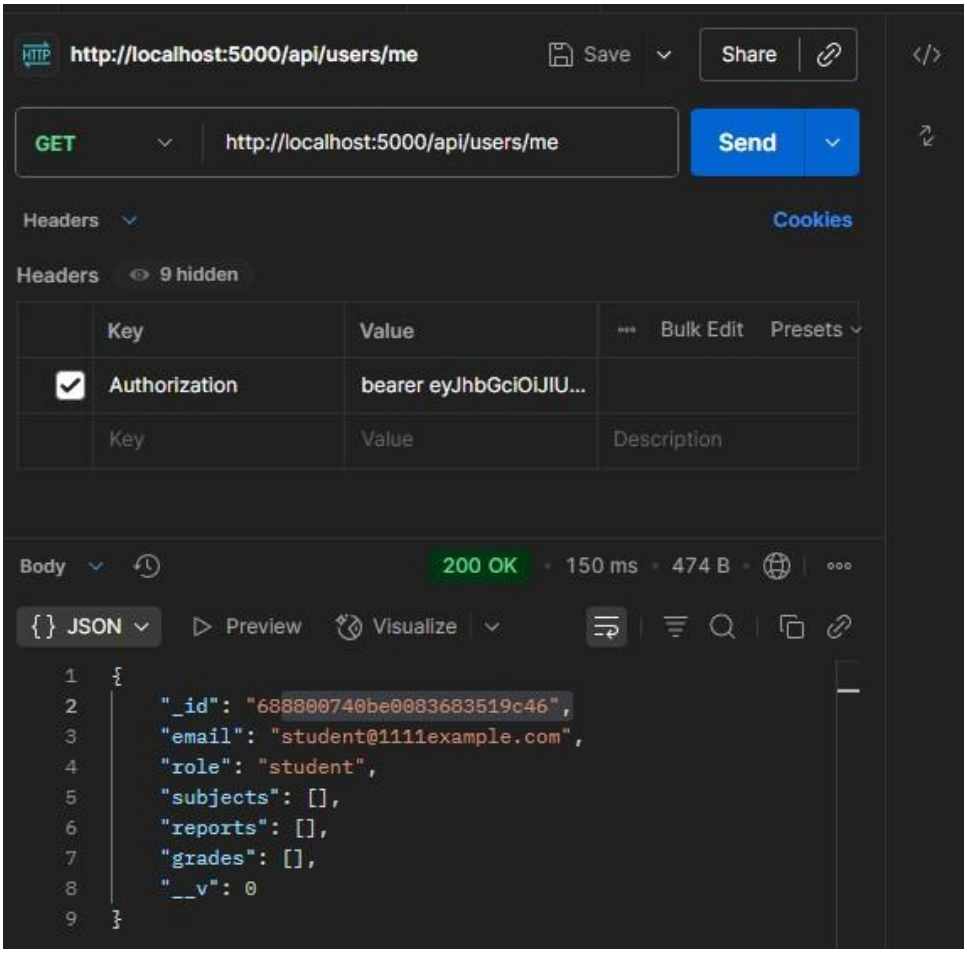
Student login

POST /login - Inicio de sesión del estudiante.



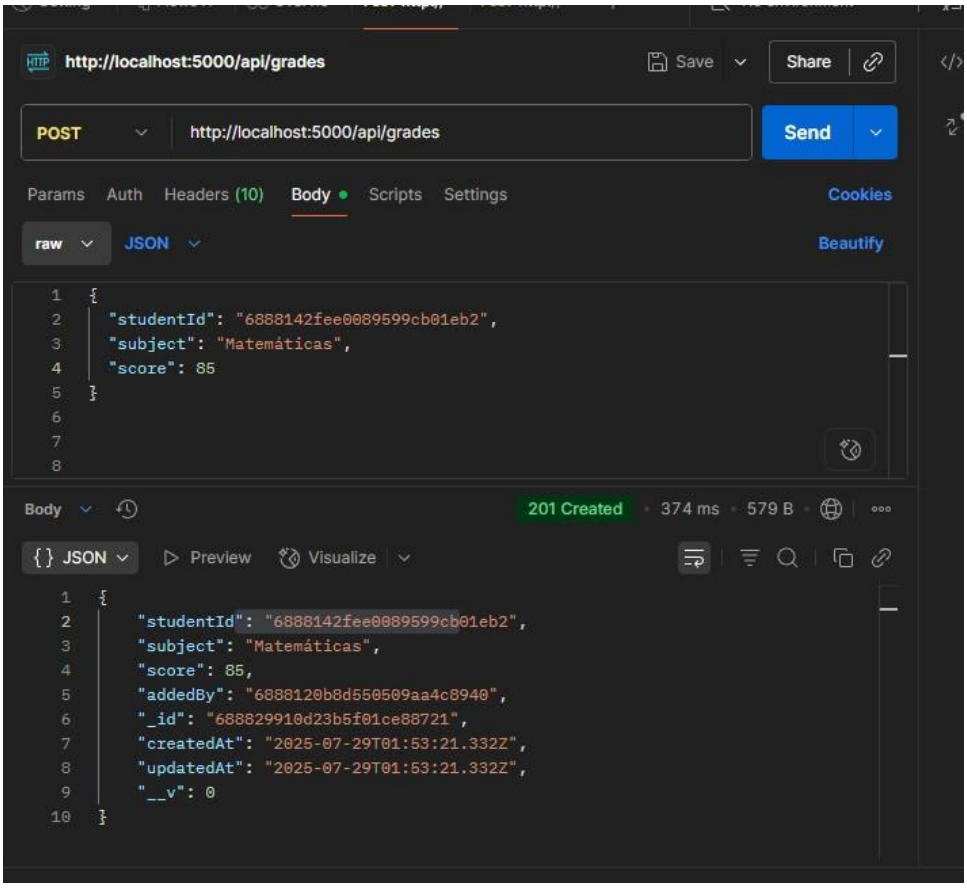
Student view

GET /student - Visualización de información del estudiante.



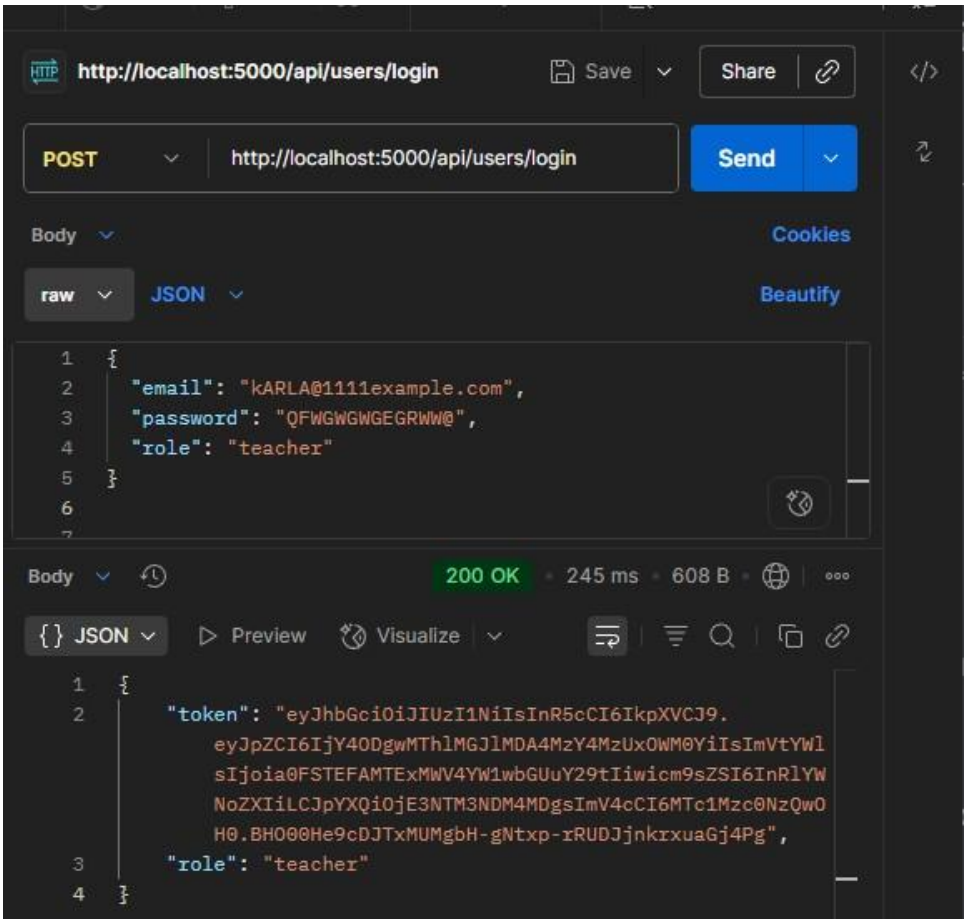
Crear nota estudiante

POST /grade - Crear calificación para un estudiante.



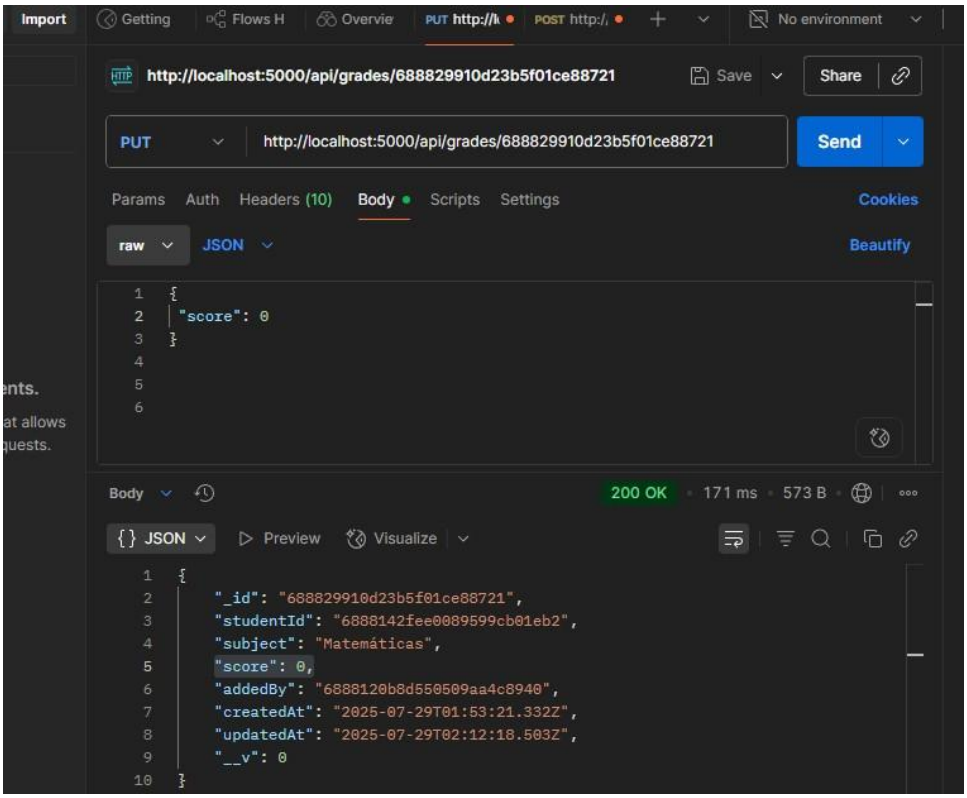
Login teacher

POST /login - Inicio de sesión del profesor.



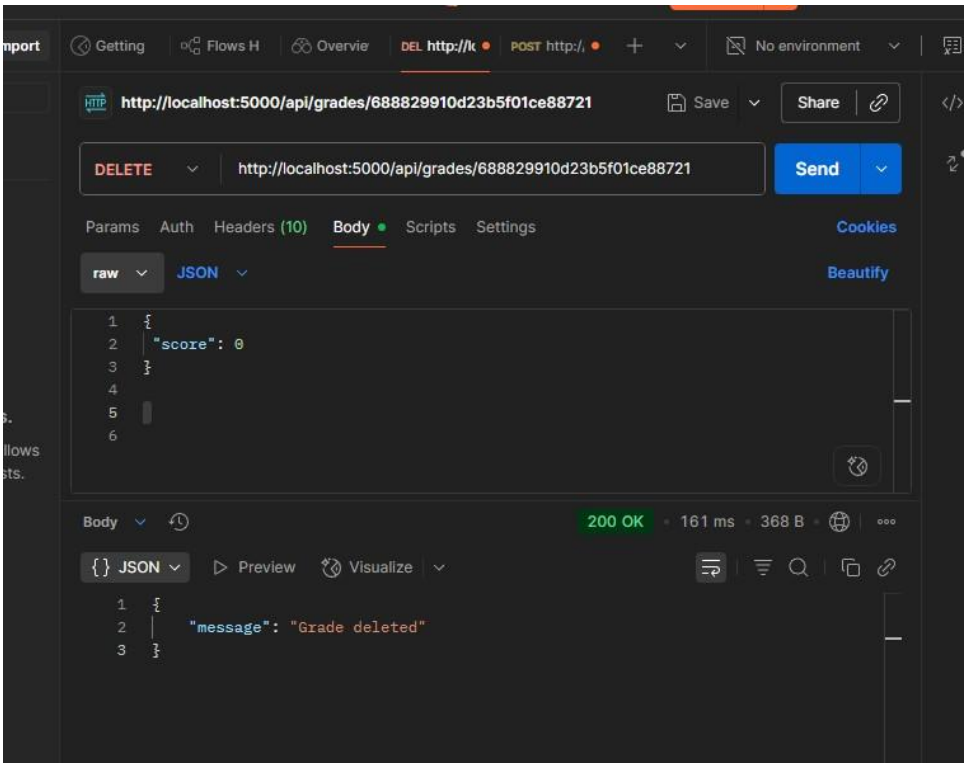
Profesor editar nota

PUT /grade/:id - Editar calificación.



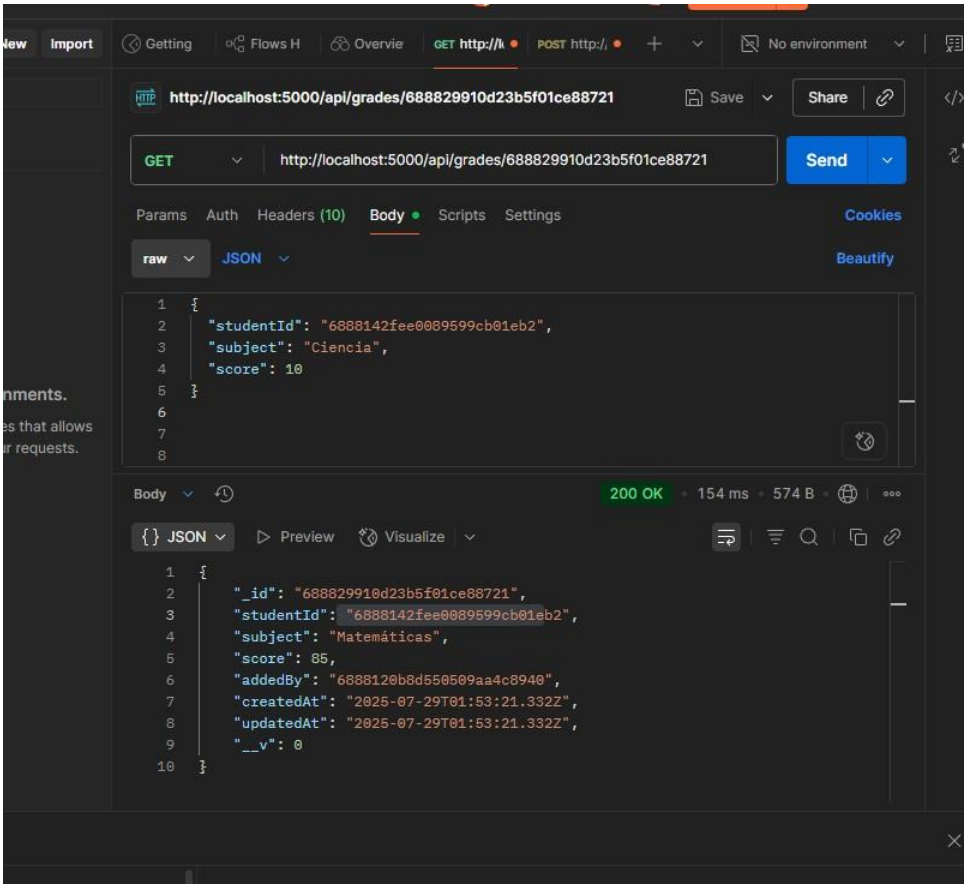
Profesor elimnar nota

DELETE /grade/:id - Eliminar calificación.



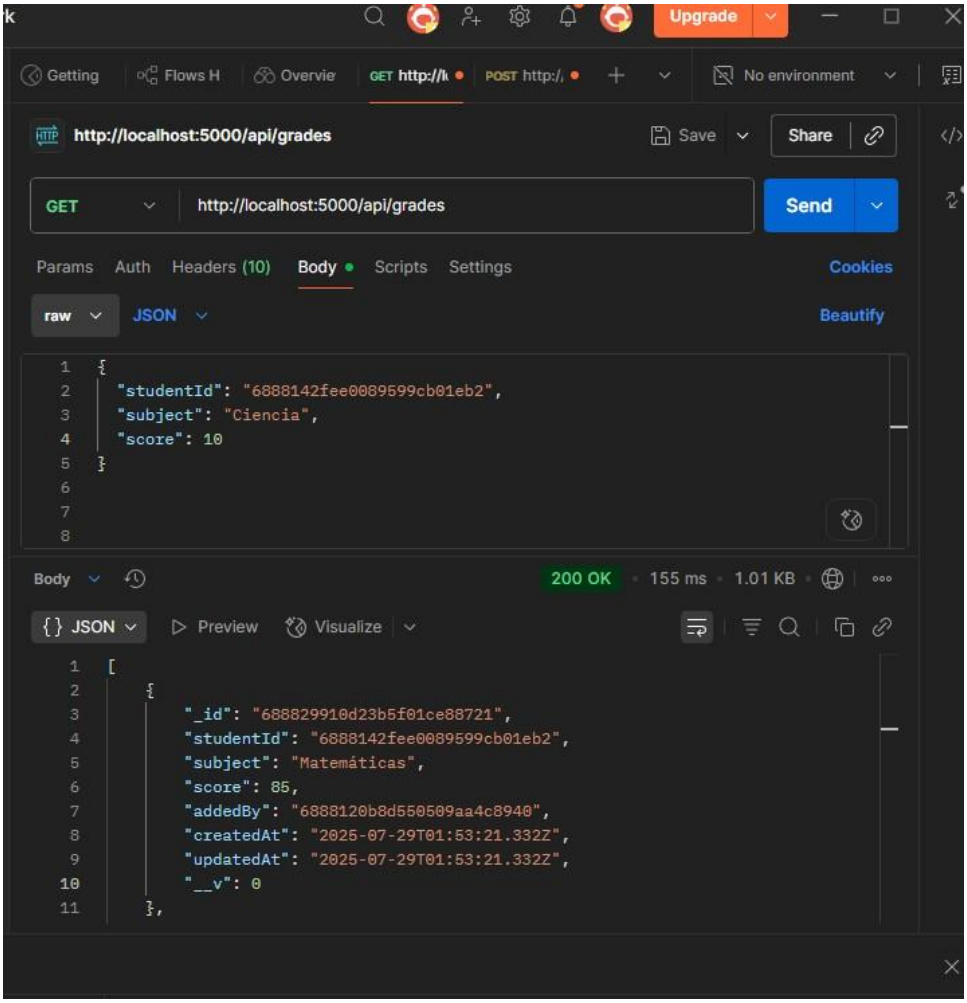
Profesor puede visualizar nota por id GET

/grade/:id - Ver calificación por ID.



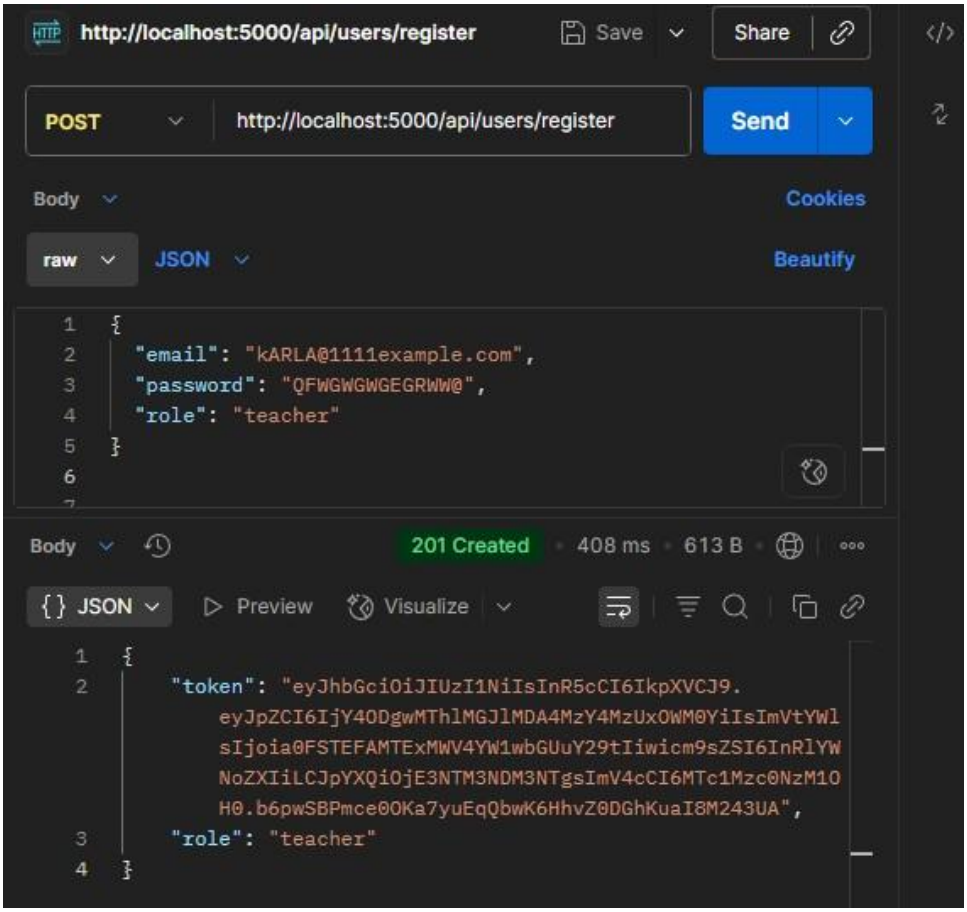
Profesor puede visualizar todas las notas GET

/grade - Visualizar todas las calificaciones.



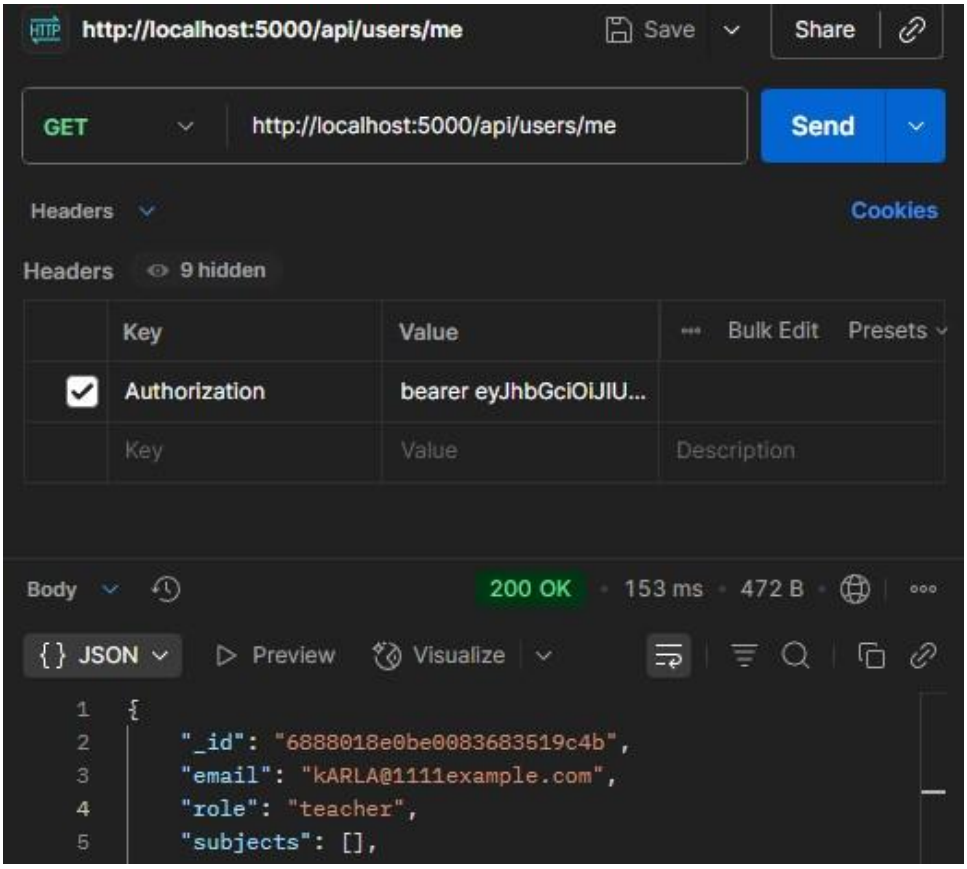
Register teacher

POST /register - Registro de profesor.



View teacher

GET /teacher - Vista general del perfil del profesor.



Conclusiones:

- Se ejecutaron 10 pruebas unitarias automatizadas con Jest, abarcando módulos clave del backend como controladores de notas, usuarios, asignaturas, clases y reportes.
- Todas las pruebas fueron aprobadas, lo que demuestra que la lógica de negocio evaluada funciona correctamente bajo las condiciones de entrada establecidas.

Recomendación:

- Ampliar la cobertura de pruebas incluyendo casos límite (valores extremos, entradas nulas o inválidas más complejas) para reforzar la robustez del sistema.
- Implementar pruebas de integración que utilicen funciones reales y conexión a base de datos para validar el comportamiento del sistema en un entorno completo.