Documentación, diseño y desarrollo del Sistema Educativo “Enedkiu”

Universidad Nacional de Colombia

Sede Manizales

Administración de Sistemas Informáticos

Ingeniería de Software II

Estudiante

David Reyes Ospina

Docente

Jose Albeiro Montes Gil

10/10/2025

Índice

Introducción

Presentación

El presente documento contiene la información acerca del sistema educativo Enedkiu, una aplicación creada para fines académicos por David Reyes Ospina, como proyecto para el curso ingeniería de software II.

Enedkiu es un sistema educativo similar a Google Classroom y Moodle, cuya finalidad es permitir a los estudiantes y profesores compartir contenido e interactuar a través de la plataforma, mejorando así la comunicación profesor-estudiante, tanto dentro como fuera del aula.

El sistema fue creado bajo una arquitectura basada en microservicios, los cuales se describirán con detalle más adelante, y para el desarrollo se han utilizado diferentes frameworks y lenguajes de programación con el fin de mejorar el aprendizaje en el desarrollo. Así mismo, se han utilizado diferentes motores de bases de datos, tanto relacionales como no relaciones.

Finalmente, para el desarrollo del sistema se ha utilizado git y github para el manejo de versiones y mantener el proyecto centralizado en un repositorio para facilitar el acceso.

Definición de requerimientos

Los requerimientos pensados para el desarrollo (tanto funcionales como no funcionales) del sistema son los siguiente:

Requerimientos funcionales

- **RF1:** El sistema debe permitir el registro de nuevos usuarios y su rol (estudiante, profesor o administrador).
- **RF2:** El sistema debe permitir a los usuarios iniciar y cerrar sesión fácilmente.
- **RF3:** El sistema debe permitir a los usuarios activos cambiar su contraseña.
- **RF4:** El sistema debe asegurar que cada usuario solo pueda realizar acciones especificas de acuerdo a su rol.

- **RF5:** El sistema debe permitir a los administradores crear y gestionar los cursos.
- **RF6:** El sistema debe permitir al administrador asignar cursos a los profesores y estudiantes según su rol.
- **RF7:** El sistema debe permitir a los profesores administrar (crear, eliminar, actualizar, etc.) la información de los cursos que se les han asignado.
- **RF8:** El sistema debe permitir a los profesores asignar tareas a los estudiantes, así como poder calificarlas y entregar una retroalimentación.
- **RF9:** El sistema debe permitir a los estudiantes ver y descargar la información que haya en los cursos que estén inscritos.
- **RF10:** El sistema debe permitir a los estudiantes entregar tareas y consultar la calificación y retroalimentación hecha por el profesor.
- **RF11:** El sistema debe generar reportes de los estudiantes y cursos para los profesores y administradores.
- **RF12:** El sistema debe permitir a los profesores enviar mensajes por correo a los estudiantes.
- **RF13:** El sistema debe permitir a los administradores enviar mensajes por correo electrónico a estudiantes y profesores.

Requerimientos no funcionales

- **RNF1:** Las contraseñas deben ser encriptadas con algoritmos de hash seguros antes de guardarse en las bases de datos.
- **RNF2:** La interfaz debe ser intuitiva y accesible para niños, adolescentes y adultos.
- **RNF3:** La interfaz debe ser responsiva y adaptarse a pantallas pequeñas como las de dispositivos móviles.
- **RNF4:** Cada microservicio debe poder modificarse sin afectar a los otros.
- **RNF5:** El sistema debe soportar al menos 5 usuarios concurrentes sin perder mucho rendimiento.
- **RNF6:** El acceso a recursos debe ser controlado con tokens.
- **RNF7:** Las solicitudes del cliente a las APIs debe tardar menos de 5 segundos en condiciones normales.

Diseño

Arquitectura

El sistema fue construido bajo una arquitectura de microservicios, cada uno de estos funcionando de la manera más independiente posible y solo comunicándose

mediante APIs. Se intento darle a cada uno de estos microservicios solo un tipo de tarea, para así no mezclar funciones y mantener buenas prácticas de la ingeniería de software.

Microservicios

Como se mencionó anteriormente, el sistema fue desarrollado bajo la arquitectura de microservicios los cuales son: seguridad, asignaturas, reportes, mensajes, vistas.

- **Seguridad:** Microservicio encargado del acceso y gestión de los usuarios y sus roles. Desde aquí se envían los tokens de acceso y se encriptar la información personal sensible de los usuarios como contraseñas. Este microservicio fue desarrollado en php – laravel y su base de datos en MySql.
- **Asignaturas:** Microservicio encargado de gestionar las asignaturas y las relaciones de cada usuario con estas. Este microservicio fue desarrollado en java – spring boot y su base de datos en MySql.
- **Reportes:** Microservicio encargado de generar reportes de los estudiantes y cursos del sistema. Este microservicio fue desarrollado en Python – flask, no posee base de datos propia.
- **Mensaje:** Microservicio con las funciones para enviar mensajes de correo electrónico a otros usuarios según los roles del usuario. Este microservicio fue desarrollado en Python – flask y su base de datos en MongoDB.
- **Vistas:** Este microservicio es el “frontend” del sistema, aquí se desarrolló todo el código las interfaces gráficas y llamados a las APIs de los microservicios según la necesidad. Este microservicio fue desarrolla mayormente en javascript – react.

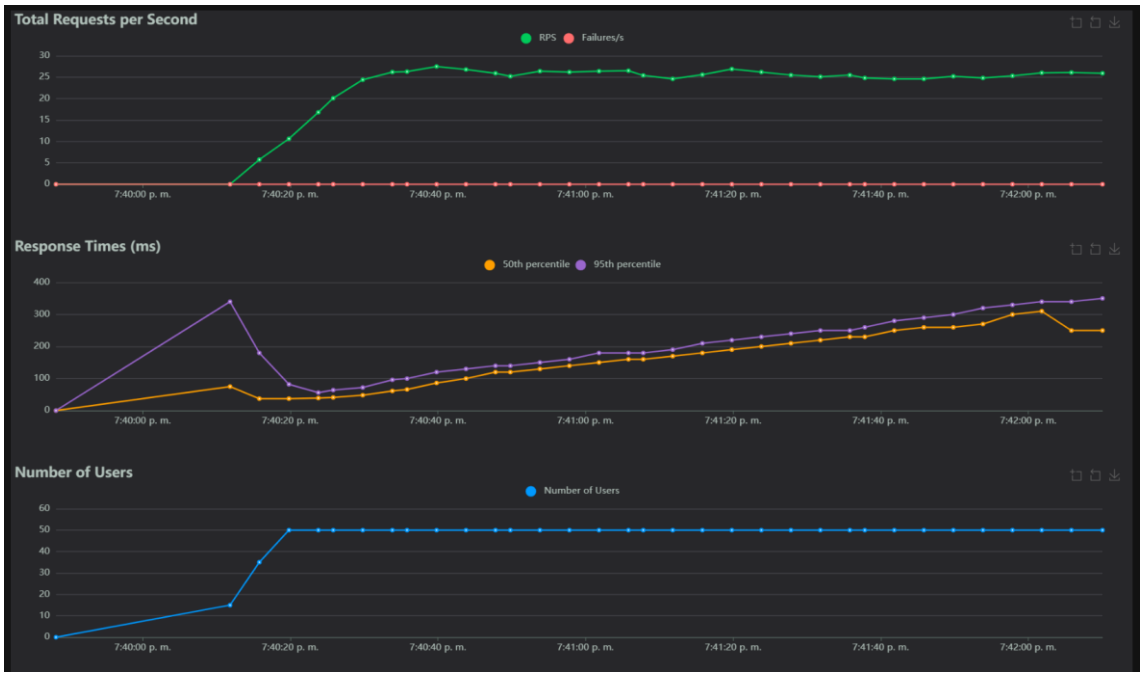
Pruebas

Pruebas de rendimiento

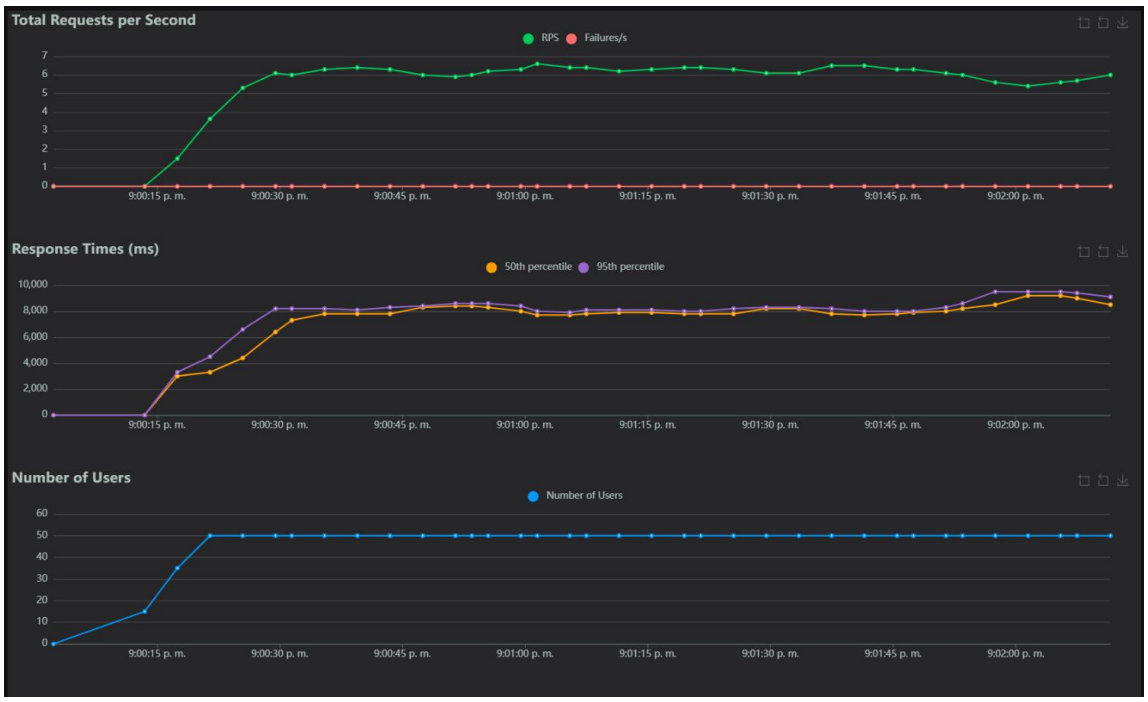
Pruebas de carga

Se espera que el sistema tenga 50 usuarios concurrentes, es por es que las pruebas de carga simulan esta cantidad de usuarios con un Ramp up de 5 usuarios/segundo. Cada prueba se realizó durante 2 minutos.

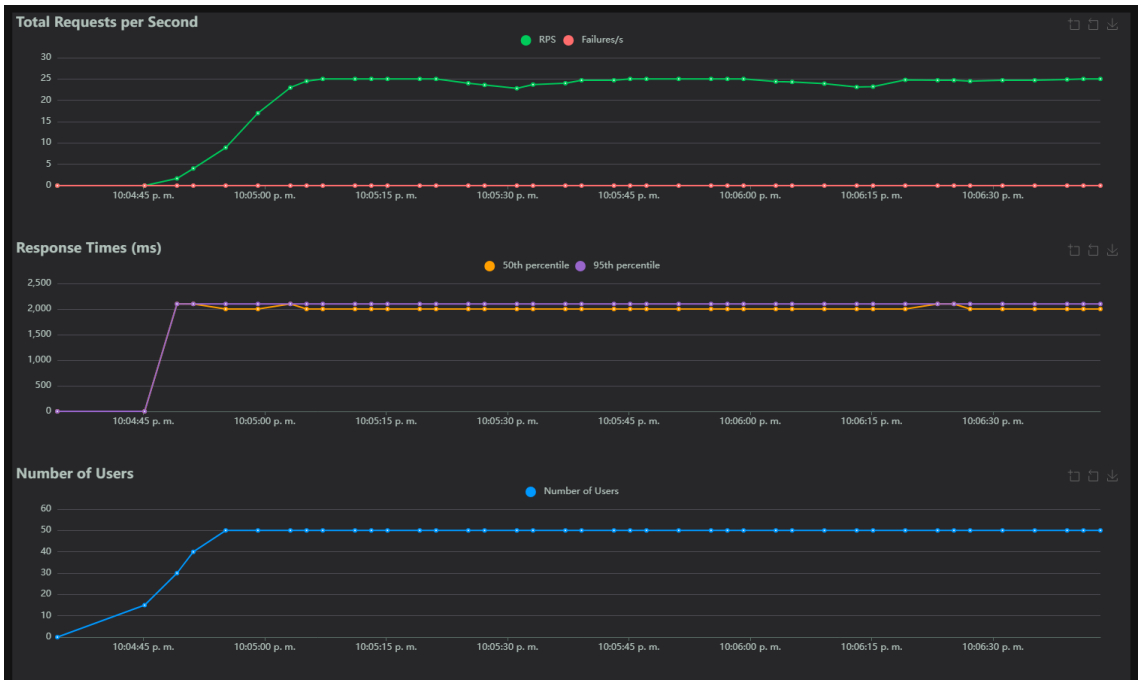
- Microservicio de cursos:



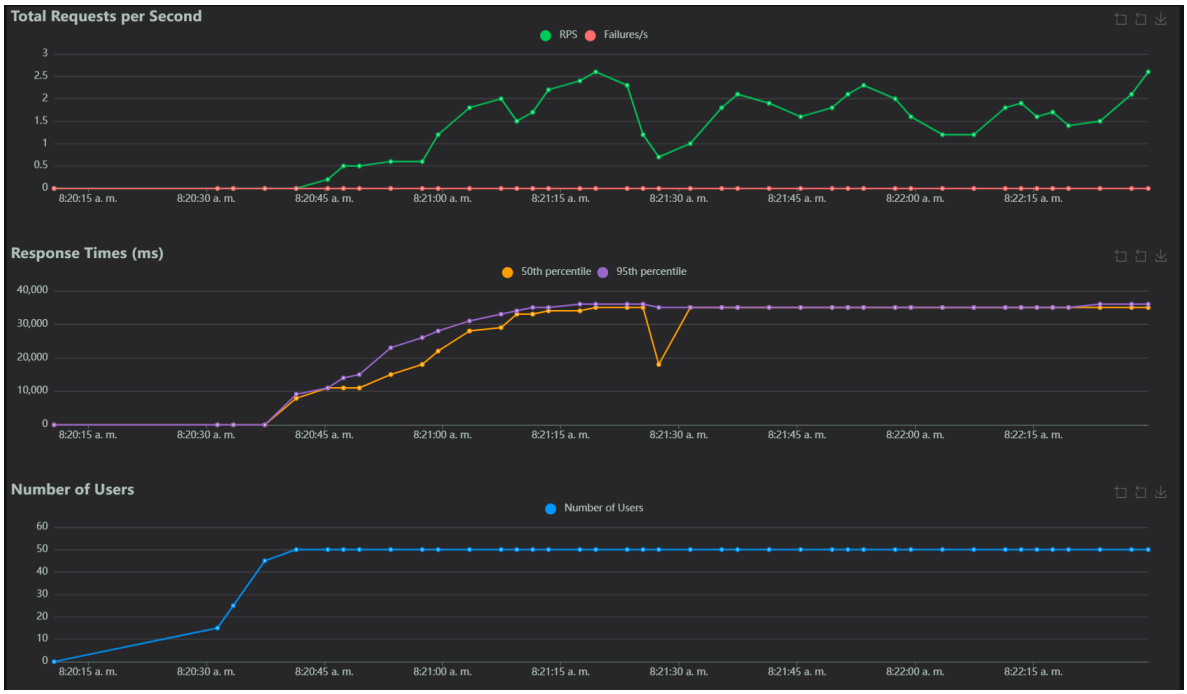
- Microservicio de seguridad:



- Microservicio de mensajes:



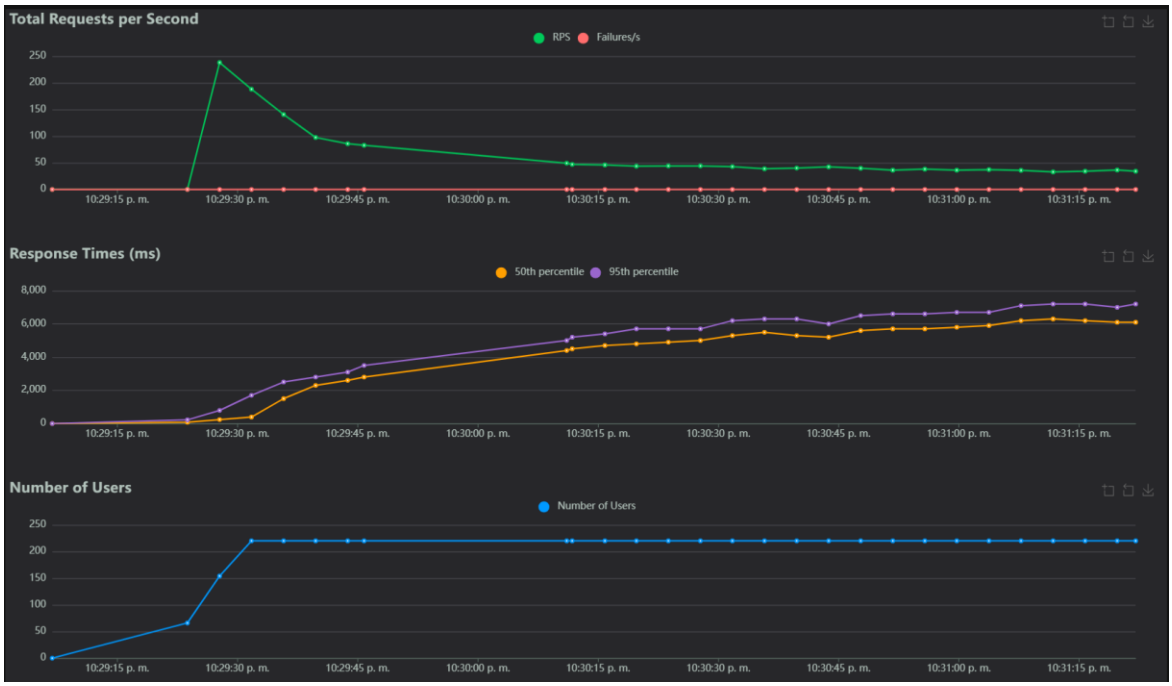
- Microservicio de reportes:



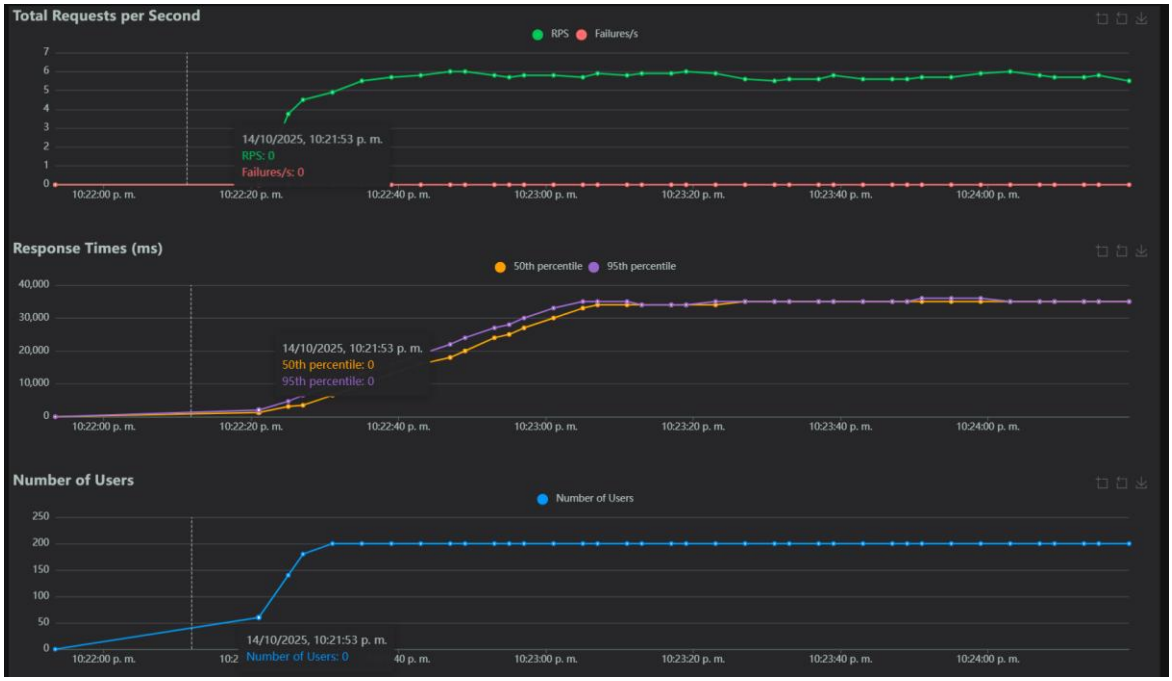
Pruebas de capacidad

Para estas pruebas se fue subiendo poco a poco la cantidad final de usuarios y se dejó la prueba que tenía la máxima cantidad de usuarios y que no hubieron errores, o estos fueron por poco tiempo.

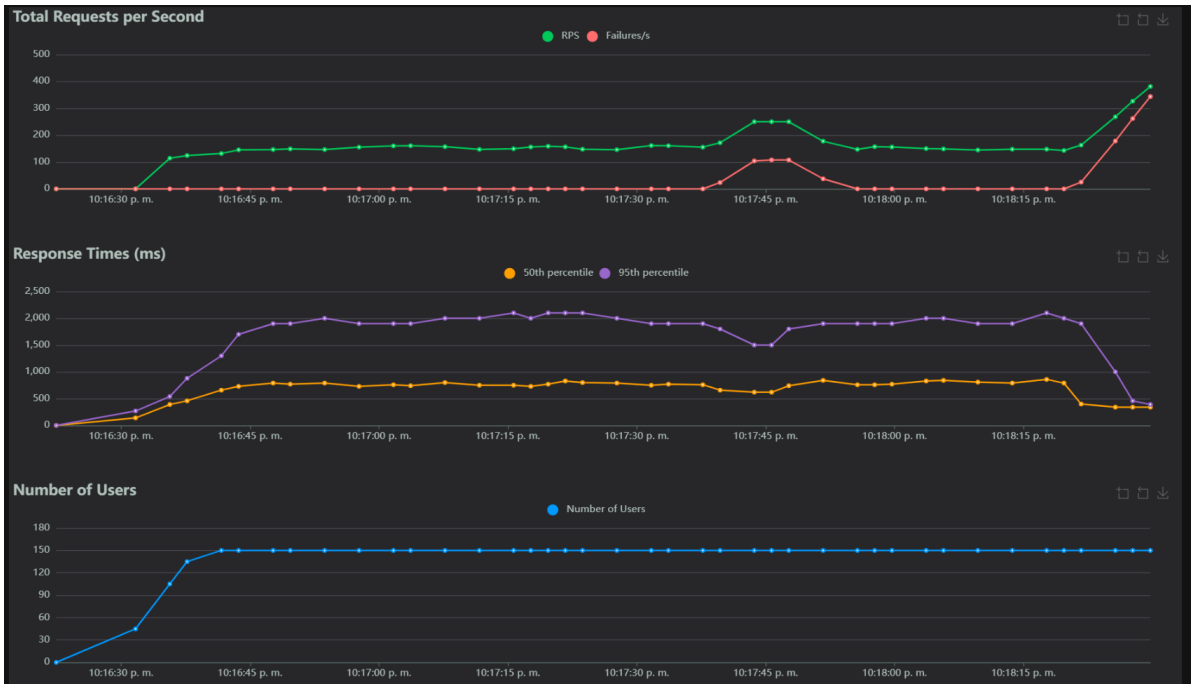
- Microservicio de cursos:



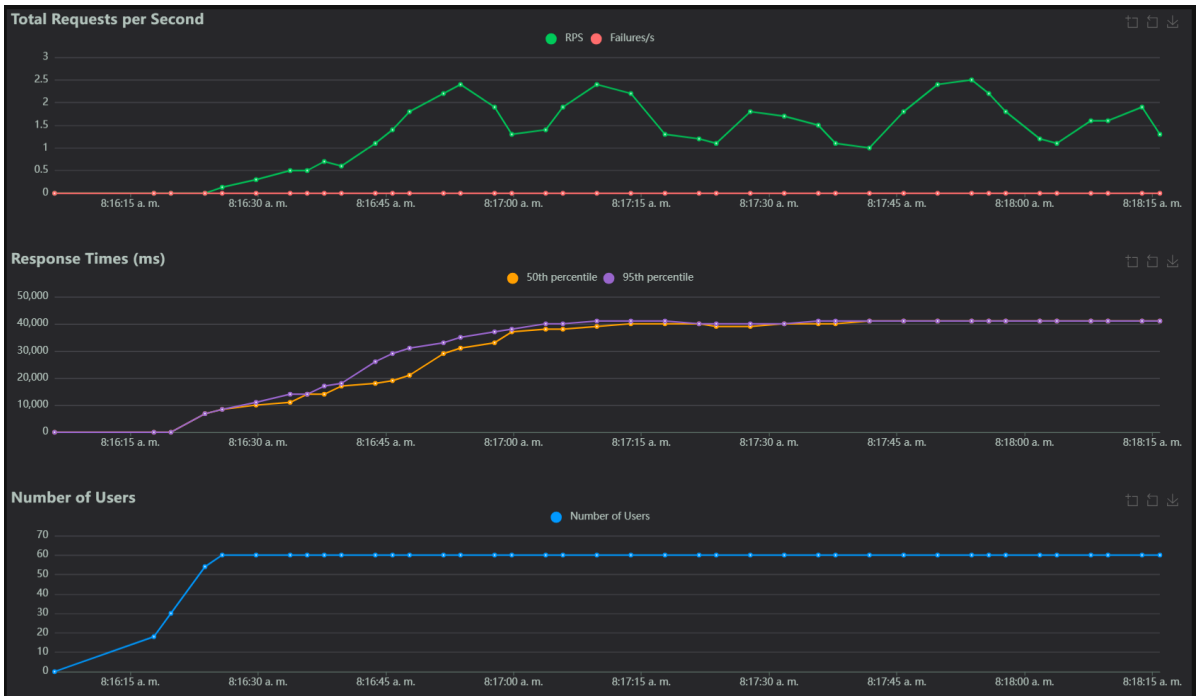
- Microservicio de seguridad:



- Microservicio de mensajes:

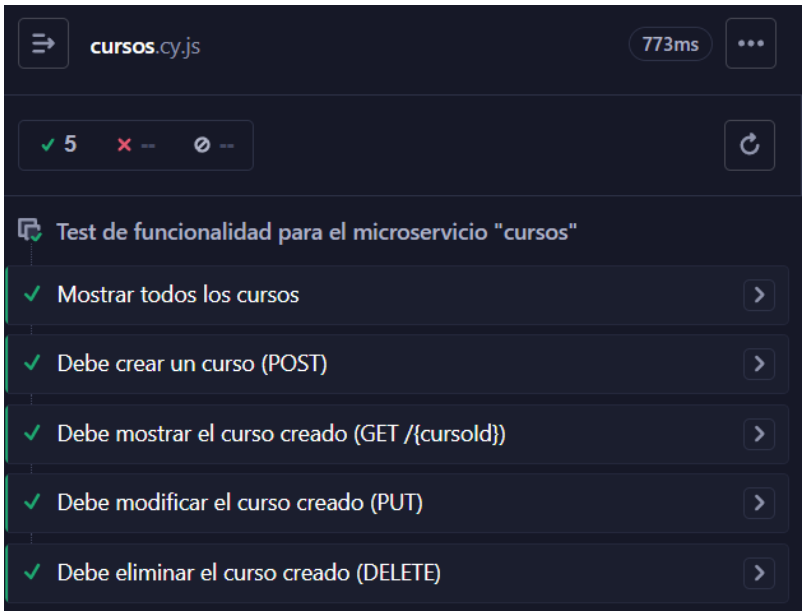


- Microservicio de reportes:

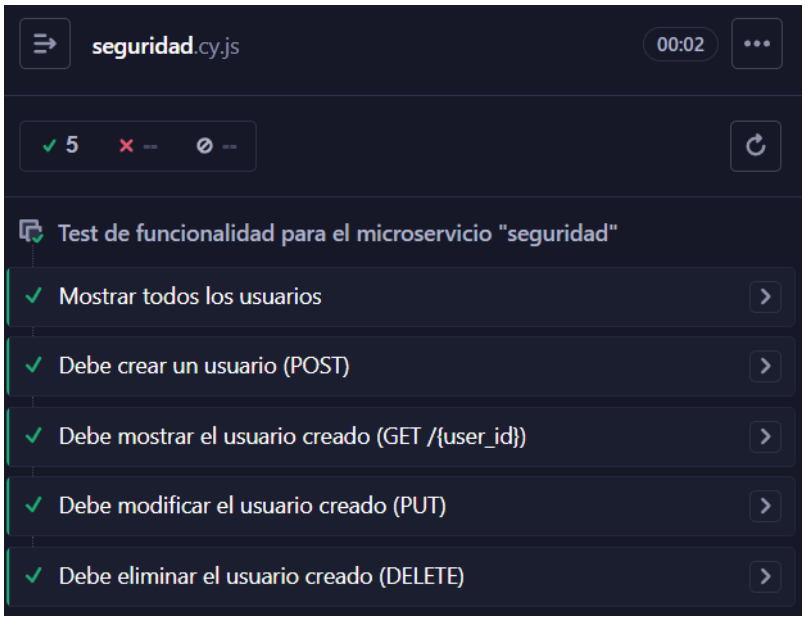


Pruebas de funcionalidad

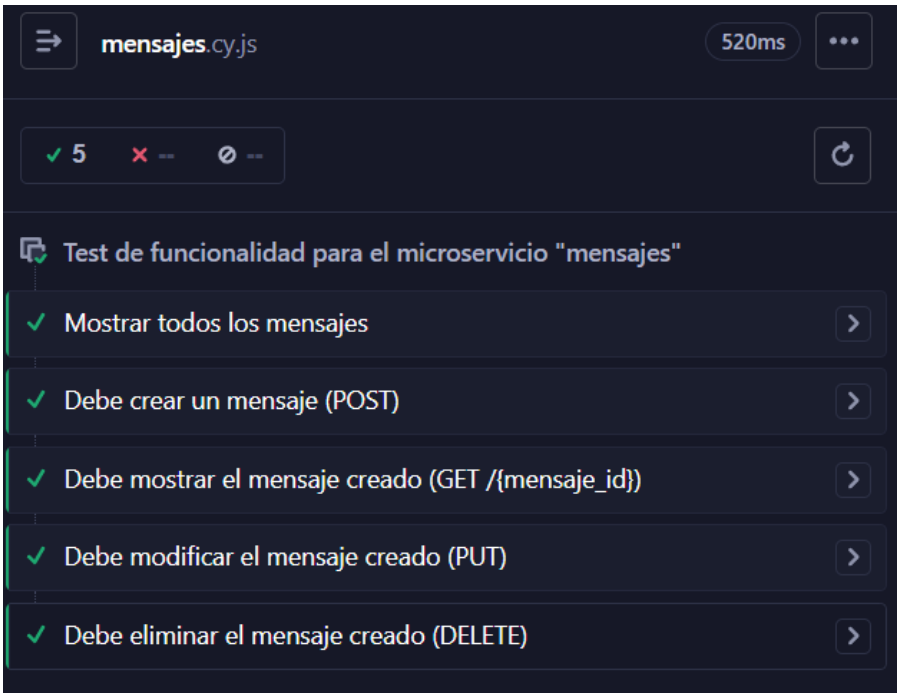
- Microservicio de cursos:



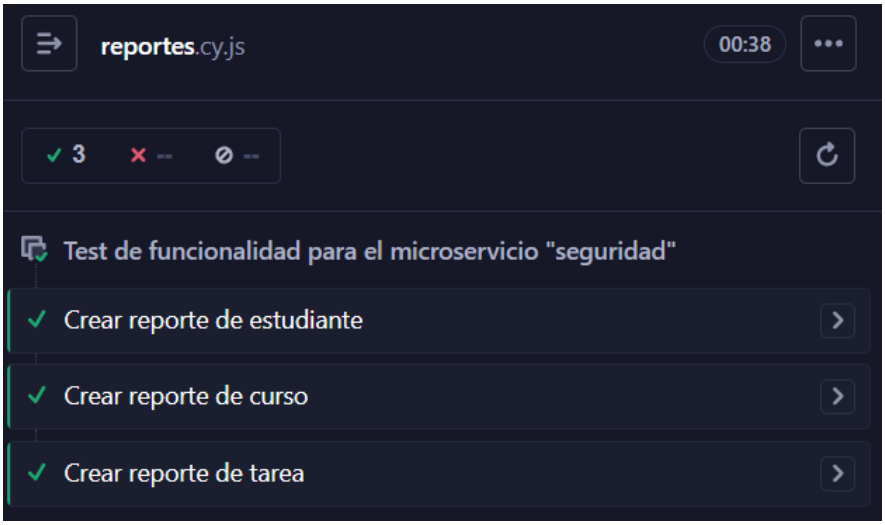
- Microservicio de seguridad:



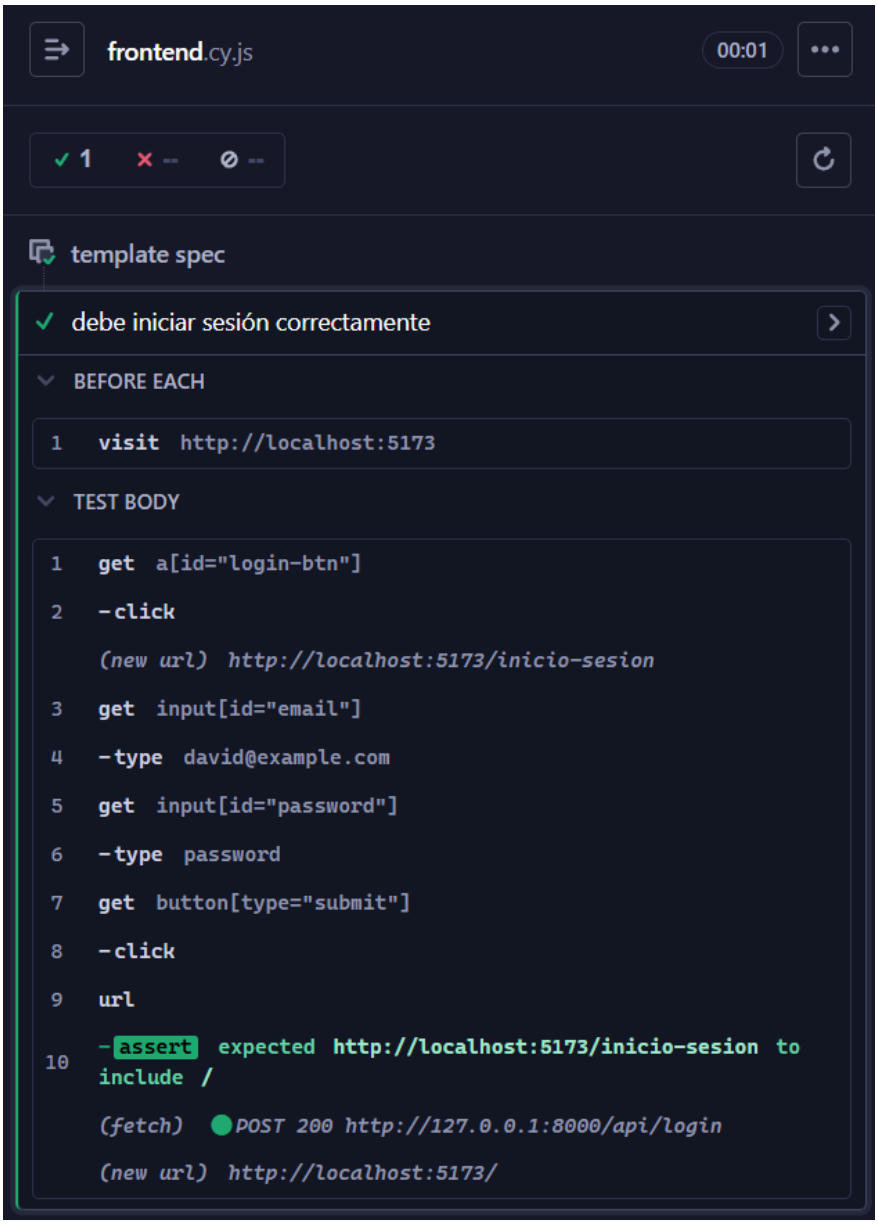
- Microservicio de mensajes:



- Microservicio de reportes:



- Microservicio de frontend:


frontend.cy.js
00:01
✓ 1
× --
⊘ --
template spec
✓ debe iniciar sesión correctamente
BEFORE EACH
1 visit http://localhost:5173
TEST BODY
1 get a[id="login-btn"]
2 -click
(new url) http://localhost:5173/inicio-sesion
3 get input[id="email"]
4 -type david@example.com
5 get input[id="password"]
6 -type password
7 get button[type="submit"]
8 -click
9 url
10 -assert expected http://localhost:5173/inicio-sesion to include /
(fetch) POST 200 http://127.0.0.1:8000/api/login
(new url) http://localhost:5173/