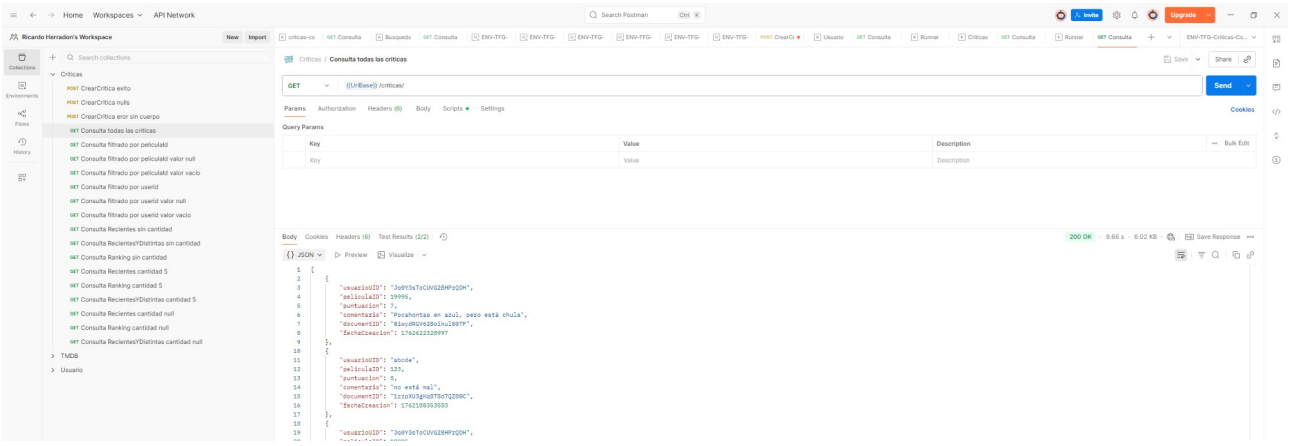
Herramientas utilizadas:

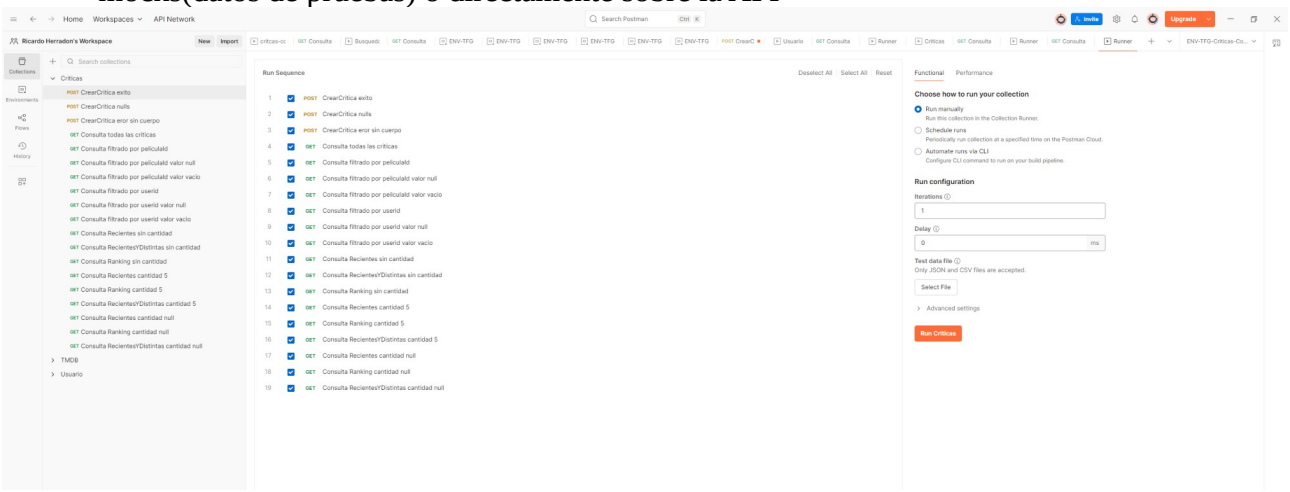
-Postman
-Node con Newman
-Jenkins
-Visual Studio Code
-Selenium IDE
-Python con Selenium y Pytest

Postman: Es una plataforma integral que se utiliza en el proceso de desarrollo y testing para crear, probar, documentar y colaborar en el desarrollo de las APIs.
Sus funciones principales son:

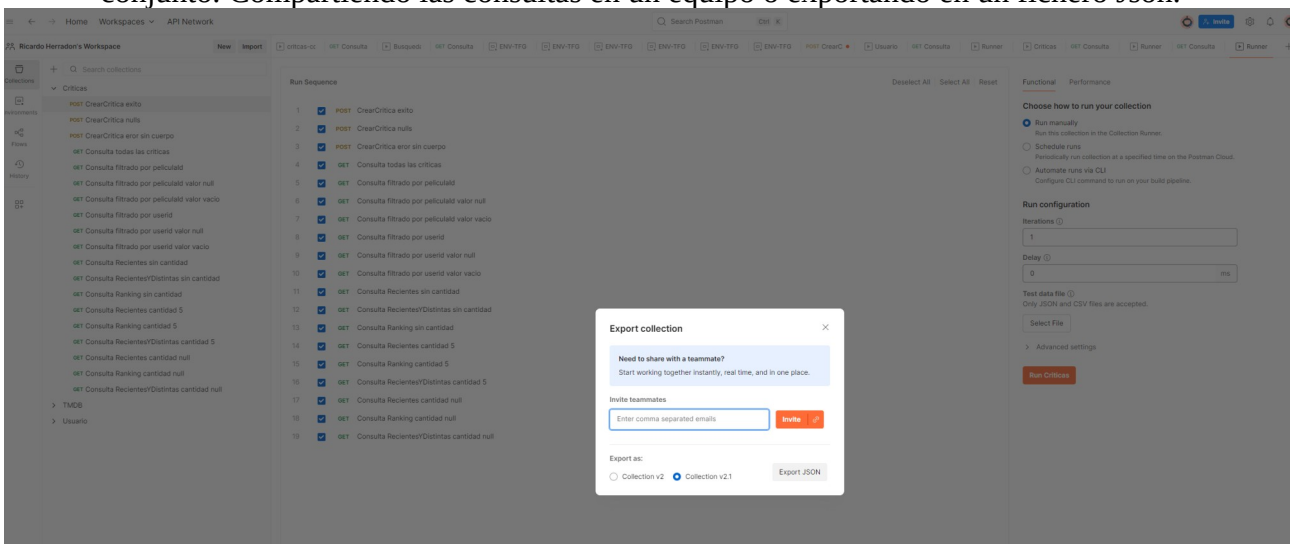
- Realización de pruebas de las API realizando solicitudes http a los endpoint y visualizar las respuestas.



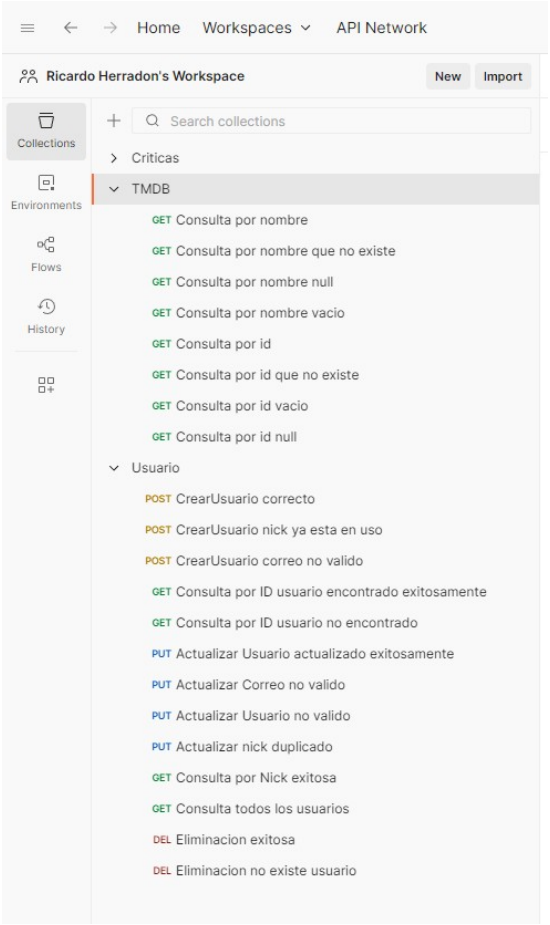
- Automatización de pruebas debido a que tiene la capacidad de crear y ejecutar pruebas(Runner Herramienta de ejecución) según un comportamiento esperado usando mocks(datos de pruebas) o directamente sobre la API



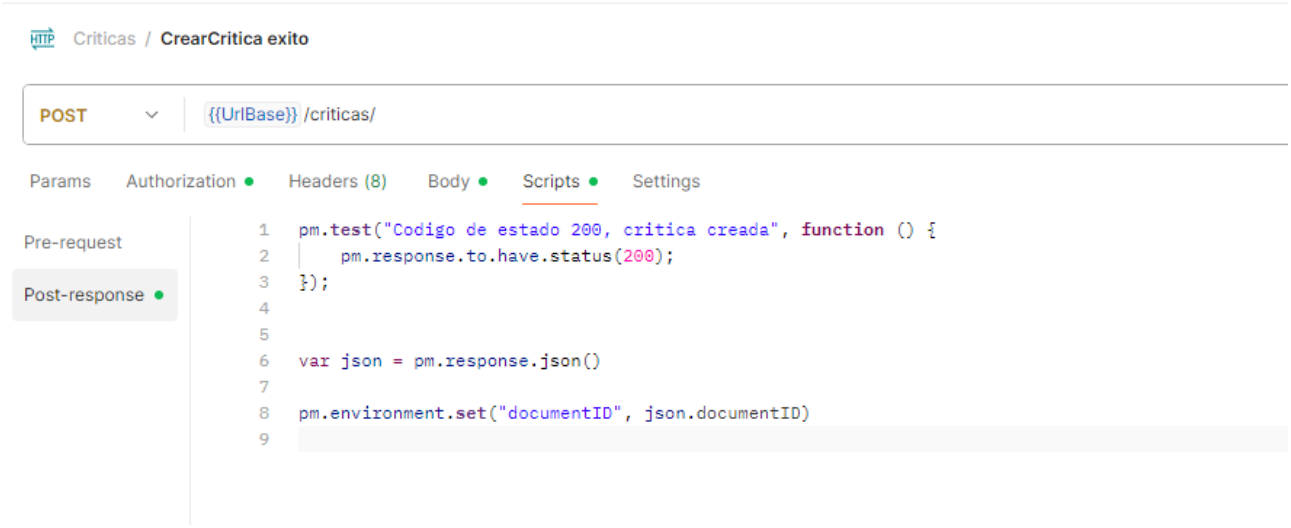
- Permite la colaboración desarrollador/tester para la posibilidad de documentación en conjunto. Compartiendo las consultas en un equipo o exportando en un fichero Json.

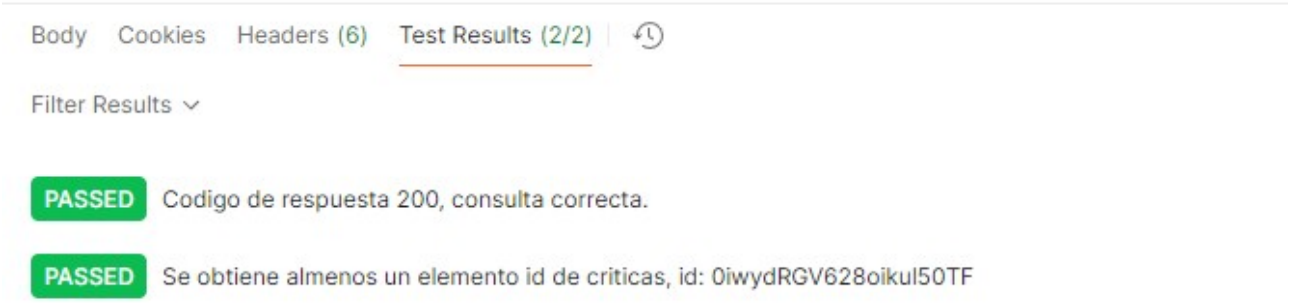


- Gestión de colecciones de solicitudes relacionadas.



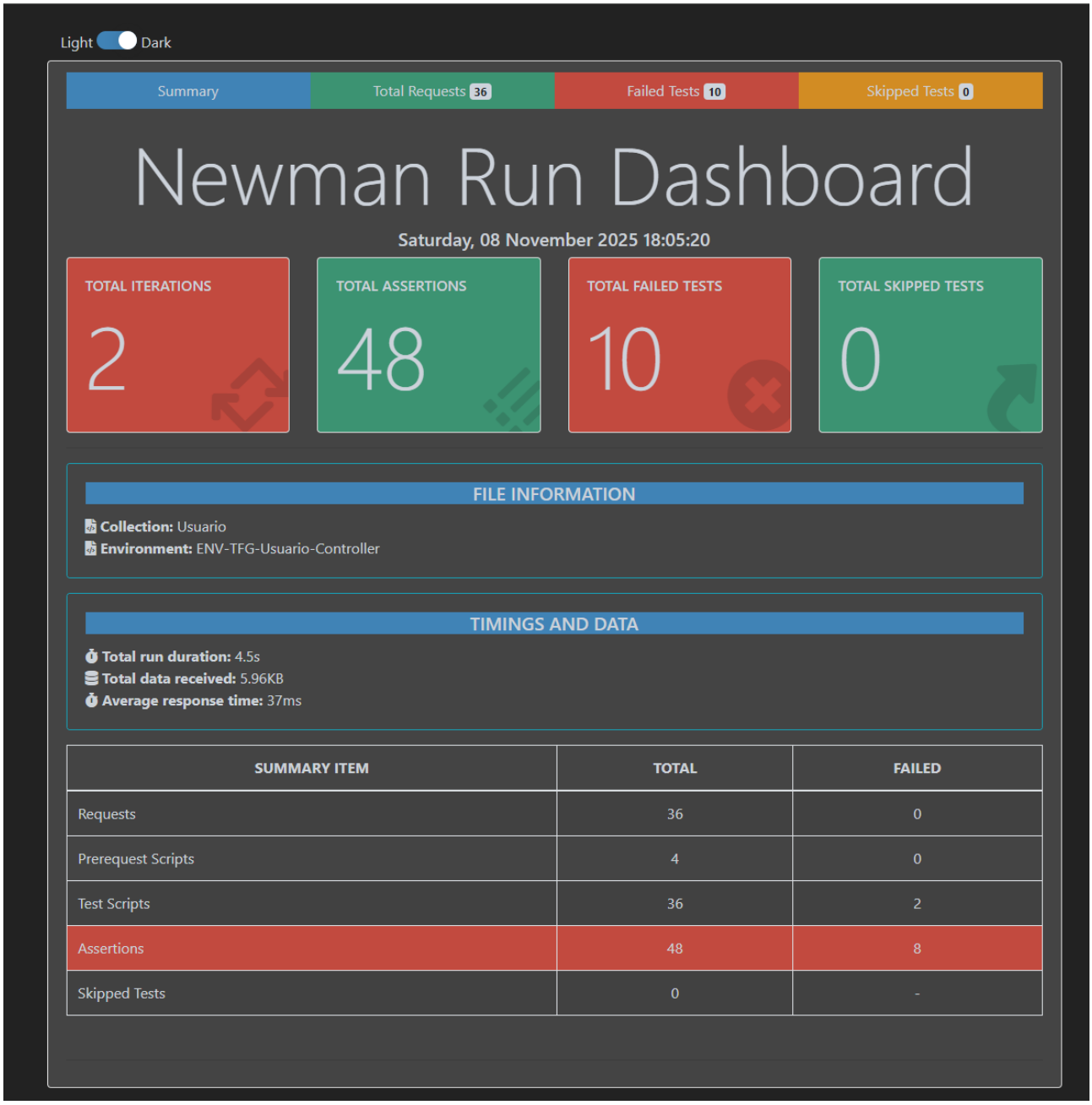
- Herramienta de ejecución con test en Javascript para un feedback de los resultados.





Node con Newman: Node es un entorno de ejecucion de cofigo abierto y multiplataforma que permite ejecutar codigo en JavaScript. Newman es una herramienta de linea de comandos para la realizacion de pruebas exportadas desde Postman directamente en una consola de comandos.

Ademas contiene funciones para la generacion de documentacion de las pruebas y Test generados en postman. Un ejemplo de comando seria el siguiente “**newman run Usuarios.postman_collection.json -e ENV-TFG-Usuario.postman_environment.json -d CSV-Iterador-Test-usuario.csv -r htmlExtra --insecure --folder newman/Usuario'**
bat 'xcopy newman/Usuario /TFG/proyectoTFG/Testing/Usuario /E /H”
Este comando generaría un informe en HTML para su posterior lectura.



Este informe indica detalladamente los datos que se han utilizado en la consulta de postman por ejemplo el cuerpo del mensaje enviado y el recibido.

Parte enviada.

REQUEST HEADERS

Header Name	Header Value
Content-Type	application/json
User-Agent	PostmanRuntime/7.39.1
Accept	*/*
Cache-Control	no-cache
Postman-Token	86566180-4a2b-4ca3-be56-f808cf9f5e75
Host	backend-proyectotfg-600260085391.europe-southwest1.run.app
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Content-Length	281

REQUEST BODY


```
{
  "correo": "Ricardo@ricardo.es",
  "imagen_perfil": "Ricardo",
  "nick": "Ricardo-Test",
  "amigos_id": [
    "Ricardo",
    "Ricardo",
    "Ricardo"
  ],
  "peliculas_criticadas": [
    0,
    1,
    2,
    3
  ],
  "peliculas_favoritas": [
```


Copy to Clipboard

Parte Recibida.

RESPONSE HEADERS

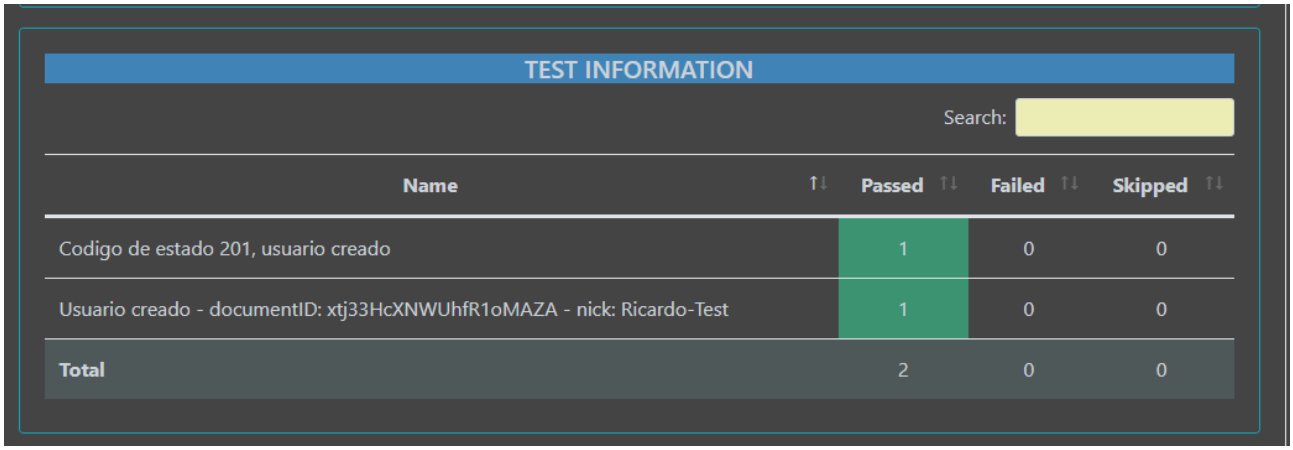
Header Name	Header Value
vary	Origin,Access-Control-Request-Method,Access-Control-Request-Headers
content-type	application/json
x-cloud-trace-context	8630143c7d134584724204d5388fc148;o=1
date	Sat, 08 Nov 2025 17:05:16 GMT
server	Google Frontend
Content-Length	249
Alt-Svc	h3=":443"; ma=2592000,h3-29=":443"; ma=2592000

RESPONSE BODY


```
{
  "documentID": "xtj33HcXNUUhfRIoMAZA",
  "correo": "Ricardo@ricardo.es",
  "imagen_perfil": "Ricardo",
  "nick": "Ricardo-Test",
  "amigos_id": [
    "Ricardo",
    "Ricardo",
    "Ricardo"
  ],
  "peliculas_criticadas": [
    0,
    1,
    2,
    3
  ],
```


Copy to Clipboard

Batería de pruebas realizadas en esta consulta.

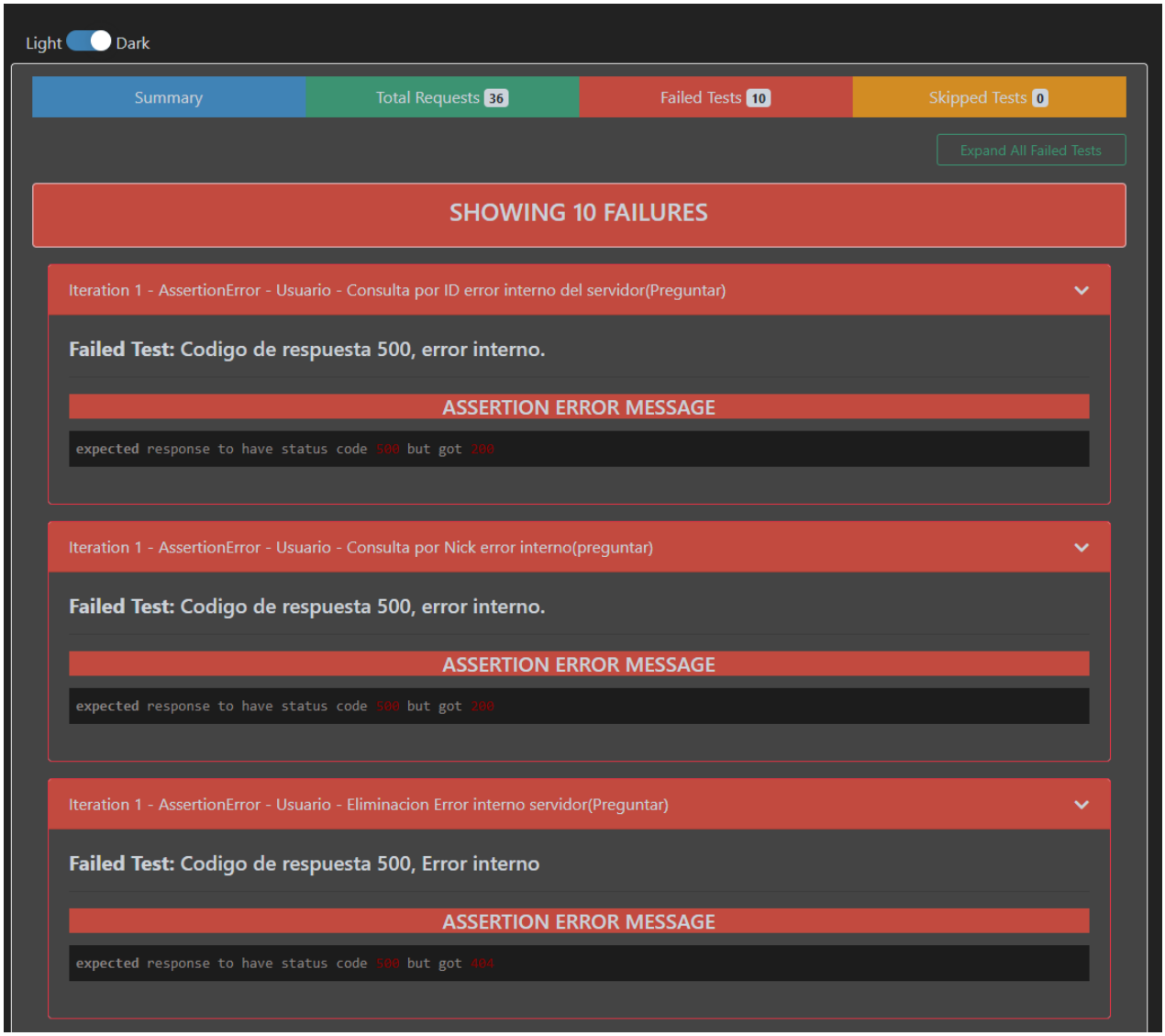


TEST INFORMATION

Search:

Name	Passed	Failed	Skipped
Codigo de estado 201, usuario creado	1	0	0
Usuario creado - documentID: xtj33HcXNWUhfR1oMAZA - nick: Ricardo-Test	1	0	0
Total	2	0	0

En caso de reportes erroneos hay un resumen con todos los campos y el motivo que se puede consultar desde el dashboard.



Light Dark

Summary | Total Requests 36 | Failed Tests 10 | Skipped Tests 0

Expand All Failed Tests

SHOWING 10 FAILURES

Iteration 1 - AssertionError - Usuario - Consulta por ID error interno del servidor(Preguntar)

Failed Test: Codigo de respuesta 500, error interno.

ASSERTION ERROR MESSAGE

```
expected response to have status code 500 but got 200
```

Iteration 1 - AssertionError - Usuario - Consulta por Nick error interno(preguntar)

Failed Test: Codigo de respuesta 500, error interno.

ASSERTION ERROR MESSAGE

```
expected response to have status code 500 but got 200
```

Iteration 1 - AssertionError - Usuario - Eliminacion Error interno servidor(Preguntar)

Failed Test: Codigo de respuesta 500, Error interno

ASSERTION ERROR MESSAGE

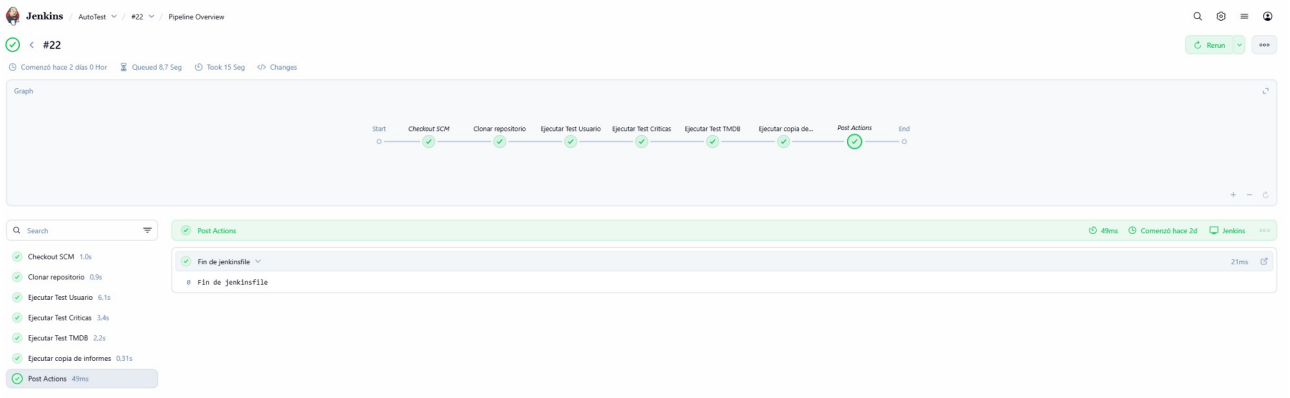
```
expected response to have status code 500 but got 404
```

Jenkins: Es un servido de automatización de código abierto escrito en Java. Se utiliza para la integracion continua(CI) y entrega continua(CD) en el desarrollo de software y su mantenimiento. Sus principales características son:

- Automatización: Permite automatizar partes del desarrollo de software que son repetitivas y propensas a errores, como la compilación, las pruebas y el despliegue. Por ejemplo la clonación del repositorio en un servidor propio.

```
Started by GitHub push by RicardoHB996
Obtained Testing/jenkinsfile from git https://github.com/TFG-FlixScore/proyectoTFG.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins  in C:\ProgramData\Jenkins\.jenkins\workspace\AutoTest
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
 > C:\Program Files\Git\cmd\git.exe rev-parse --resolve-git-dir
C:\ProgramData\Jenkins\.jenkins\workspace\AutoTest\.git # timeout=10
Fetching changes from the remote Git repository
 > C:\Program Files\Git\cmd\git.exe config remote.origin.url https://github.com/TFG-
FlixScore/proyectoTFG.git # timeout=10
Fetching upstream changes from https://github.com/TFG-FlixScore/proyectoTFG.git
 > C:\Program Files\Git\cmd\git.exe --version # timeout=10
 > git --version # 'git version 2.45.1.windows.1'
 > C:\Program Files\Git\cmd\git.exe fetch --tags --force --progress -- https://github.com/TFG-
FlixScore/proyectoTFG.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > C:\Program Files\Git\cmd\git.exe rev-parse "refs/remotes/origin/Testing^{commit}" # timeout=10
Checking out Revision b539048ca64278afdca0241cdbc2973d25051f0a (refs/remotes/origin/Testing)
 > C:\Program Files\Git\cmd\git.exe config core.sparsecheckout # timeout=10
 > C:\Program Files\Git\cmd\git.exe checkout -f b539048ca64278afdca0241cdbc2973d25051f0a # timeout=10
Commit message: "subida de ficheros actualizado"
 > C:\Program Files\Git\cmd\git.exe rev-list --no-walk b93954111f42146c0f2e872f122fbe56fbe74127 # timeout=10
```

- Integración y Entrega Continua (CI/CD): La principal caracteristica de este punto es implementar flujos de trabajo donde cada cambio en el código se compila, prueba y despliega de manera automática y continua. En nuestro caso aplica que cuando se genera un cambio en Github se lanzaria una serie de pruebas con Newman.



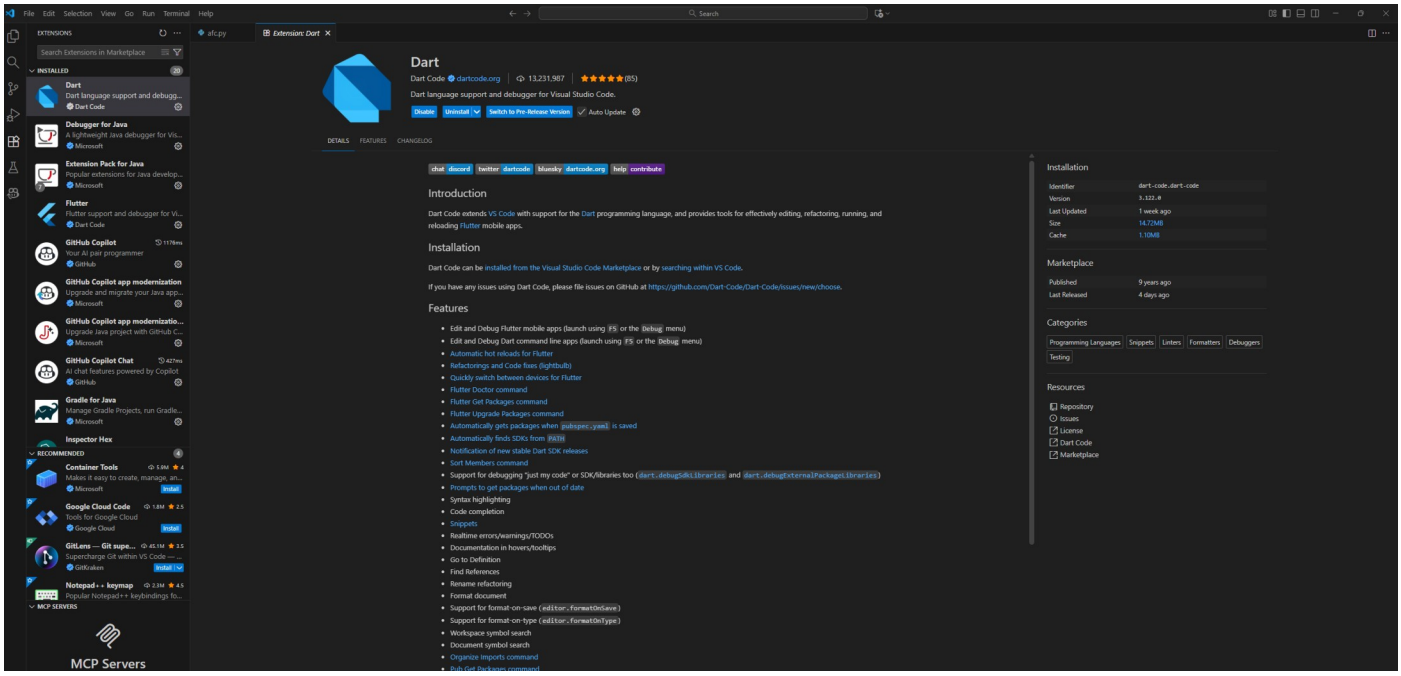
- Expansibilidad: Dispone de una enorme biblioteca de plugin desarrollados por la comunidad y para expandir sus funcionalidades. Por ejemplo conexión con github o con entornos IDE como Visual Studio Code.

- Flexibilidad: Dispone de acceso web adaptado a múltiples dispositivos.

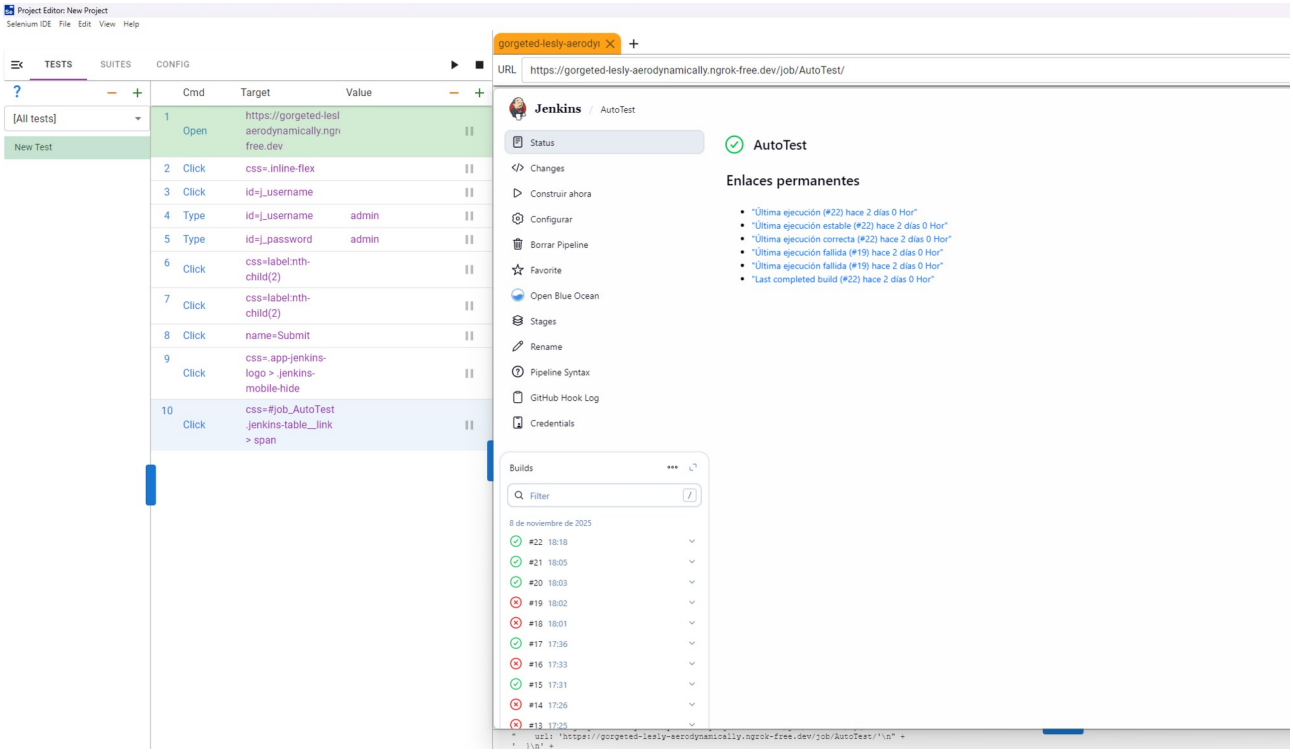
- Detección temprana de errores: Al ejecutar pruebas de forma automatizada en cada cambio de código, los errores se detectan mucho antes en el ciclo de desarrollo.

- Código abierto y comunidad: Dispone de soporte continuo y cuenta con el apoyo de una extensa comunidad.

Visual Studio Code: Editor de código fuente gratuito y multiplataforma de Microsoft para Windows. Compatible con multitud de lenguajes y Adaptable a las necesidades debido a su gran cantidad de extensiones. Por ejemplo Dart, lenguaje utilizado en el FrontEnd o Java con Spring Boot para el Backend.



Selenium IDE: Interfaz gráfica para la automatización de pruebas en aplicaciones Web, funciona con los principales navegadores como Firefox, Chrome o Microsoft Edge. Dispone de funcionalidades como la grabacion y reproduccion de pruebas a traves del driver del navegador correspondiente y la posibilidad de la exportacion del codigo en multiples lenguajes.



Generando con una exportacion en python el siguiente codigo, agilizando el proceso de creacion de pruebas en Python.

C: > Users > RICK > Desktop > new-test.py > ...

```python
# Generated by Selenium IDE
import pytest
import time
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities


class TestNewTest():

def setup_method(self, method):
  self.driver = webdriver.Chrome()
  self.vars = {}

def teardown_method(self, method):
  self.driver.quit()

  def test_newTest(self):
    self.driver.get("https://gorgeted-lesly-aerodynamically.ngrok-free.dev ")
    self.driver.find_element(By.CSS_SELECTOR, ".inline-flex").click()
    self.driver.find_element(By.ID, "j_username").click()
    self.driver.find_element(By.ID, "j_username").send_keys("admin")
    self.driver.find_element(By.ID, "j_password").send_keys("admin")
    self.driver.find_element(By.CSS_SELECTOR, "label:nth-child(2)").click()
    self.driver.find_element(By.CSS_SELECTOR, "label:nth-child(2)").click()
    self.driver.find_element(By.NAME, "Submit").click()
    self.driver.find_element(By.CSS_SELECTOR, ".app-jenkins-logo > .jenkins-mobile-hide").click()
    self.driver.find_element(By.CSS_SELECTOR, "#job_AutoTest .jenkins-table__link > span").click()
```

Python con Selenium y Pytest: Python es lenguaje de programación interpretado, de alto nivel, de código abierto y multiparadigma. La librerías de Selenium proveen de funcionalidades de automatización de pruebas Web. Pytest se utiliza para la administración y ejecución de pruebas respecto a los resultados obtenidos de la automatización de Selenium. Volviendo al ejemplo anterior, se puede verificar múltiples cosas, como que el código autogenerado no esta correctamente tabulado y que solo ha generado un único test al ser un ejemplo sencillo. Con esta base es con lo que se trabajara para la realización de las pruebas del FontEnd con resultados esperados y automatizado desde Jenkins para su ejecución cada vez que se modifique el Github al realizar un push.