

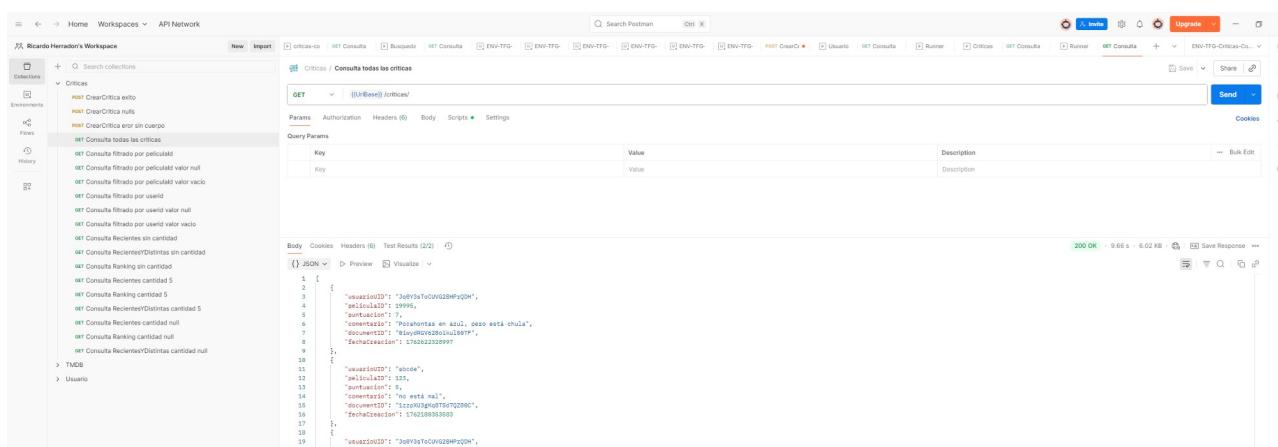
## Herramientas utilizadas:

- 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:

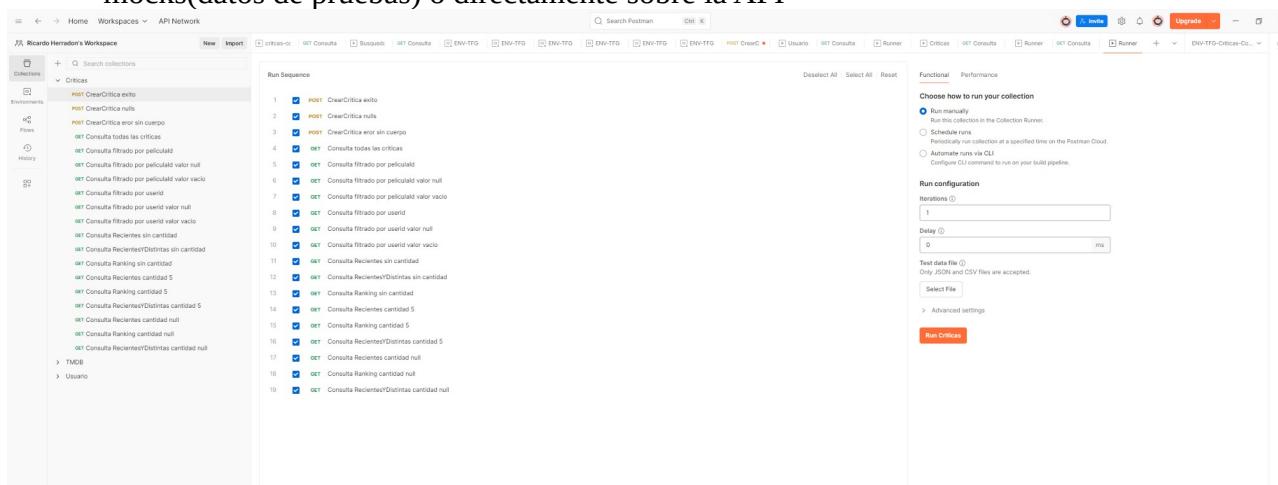
- Realización de pruebas de las API realizando solicitudes http a los endpoint y visualizar las respuestas.



The screenshot shows the Postman interface with a collection named 'Criticas'. A GET request is selected with the URL `/criticas/`. The response body is a JSON array containing two review objects:

```
[{"id": 1, "usuarioID": "3409397eC0VQ2BHPzQ0H", "secciónID": 19996, "fechaCreación": "2023-08-22T10:00:00Z", "comentari": "Pocahontas en azul, pero est\u00f3 chula", "documentoID": "1d1e0000000000000000000000000000", "fechaActualizaci\u00f3n": "2023-08-22T10:00:00Z"}, {"id": 2, "usuarioID": "abob", "secciónID": 123, "fechaCreaci\u00f3n": "2023-08-22T10:00:00Z", "comentari": "no est\u00f3 mal", "documentoID": "1d1e0000000000000000000000000000", "fechaActualizaci\u00f3n": "2023-08-22T10:00:00Z"}]
```

- 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



The screenshot shows the Postman interface with the 'Run Sequence' tab selected. A sequence of 19 requests is listed, all of which are checked for execution. The requests include various POST and GET operations from the 'Criticas' collection.

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

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Shows 'Ricardo Herradon's Workspace' with collections like 'Criticas', 'TMDB', and 'Usuario'.
- Center Panel:** Displays a 'Run Sequence' of 18 API requests under the 'Criticas' collection.
- Right Panel:** Contains sections for 'Functional' and 'Performance' testing, and a 'Run configuration' section with iterations set to 1.
- Bottom Center:** An 'Export collection' dialog is open, prompting to share with a teammate via email. It also includes options to 'Invite teammates' (with an input field for comma-separated emails), 'Export as' (Collection v2 or Collection v2.1), and 'Export JSON'.

- Gestión de colecciones de solicitudes relacionadas.

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** Shows 'Ricardo Herradon's Workspace' with collections like 'Criticas', 'TMDB', and 'Usuario'.
- Center Panel:** Displays the contents of the 'TMDB' collection, which includes several 'GET' requests for movie names and IDs.
- Bottom Center:** A list of API requests from other collections:
  - 'Criticas' collection: 'POST CrearCritica exito', 'POST CrearCritica null', 'POST CrearCritica error en el cuerpo', 'GET Consulta todas las criticas', 'GET Consulta filtrado por peliculaid', 'GET Consulta filtrado por peliculaid valor null', 'GET Consulta filtrado por peliculaid valor vacio', 'GET Consulta filtrado por userid', 'GET Consulta filtrado por userid valor null', 'GET Consulta filtrado por userid valor vacio', 'GET Consulta Recientes sin cantidad', 'GET Consulta Recientes/Distintas sin cantidad', 'GET Consulta Recientes cantidad 5', 'GET Consulta Recientes cantidad 5', 'GET Consulta Recientes/Distintas cantidad 5', 'GET Consulta Recientes cantidad null', 'GET Consulta Ranking cantidad null', 'GET Consulta Recientes/Distintas cantidad null'.
  - 'Usuario' collection: 'POST CrearUsuario correcto', 'POST CrearUsuario nick ya esta en uso', 'POST CrearUsuario correo no valido', 'GET Consulta por ID usuario encontrado exitosamente', 'GET Consulta por ID usuario no encontrado', 'PUT Actualizar Usuario actualizado exitosamente', 'PUT Actualizar Correo no valido', 'PUT Actualizar Usuario no valido', 'PUT Actualizar nick duplicado', 'GET Consulta por Nick exitosa', 'GET Consulta todos los usuarios', 'DEL Eliminacion exitosa', 'DEL Eliminacion no existe usuario'.

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

The screenshot shows the Postman interface for a POST request to `/{{UrlBase}}/criticas/`. The 'Scripts' tab is selected. The 'Post-response' section contains the following JavaScript code:

```
1 pm.test("Codigo de estado 200, critica creada", function () {
2 | pm.response.to.have.status(200);
3 });
4
5
6 var json = pm.response.json()
7
8 pm.environment.set("documentID", json.documentID)
9
```

Filter Results 

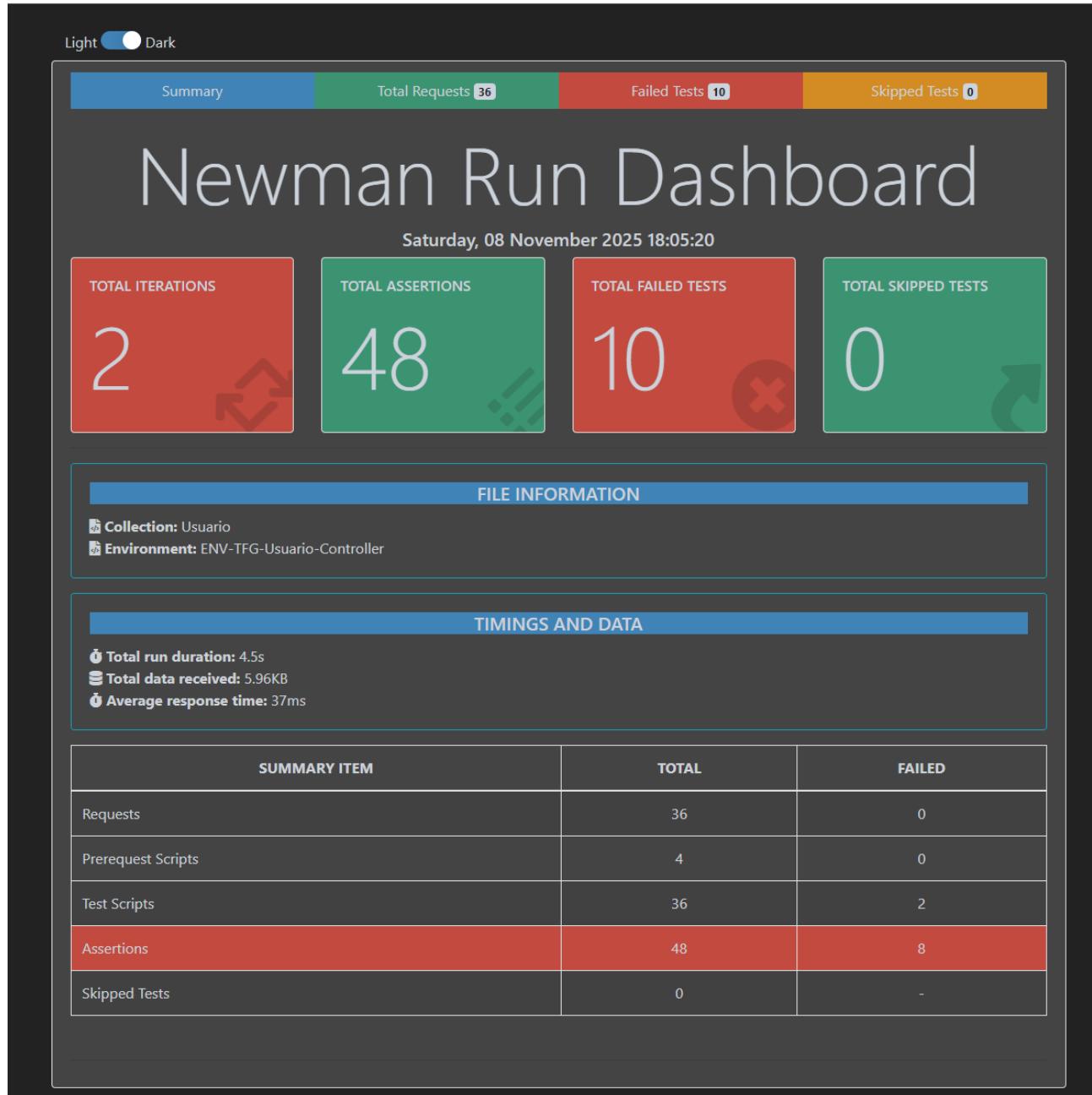
**PASSED** Código de respuesta 200, consulta correcta.

**PASSED** Se obtiene almenos un elemento id de críticas, id: 0iwydRGV628oikul50TF

**Node con Newman:** Node es un entorno de ejecución de código abierto y multiplataforma que permite ejecutar código en JavaScript. Newman es una herramienta de línea de comandos para la realización 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": "xtj33ICXMUhfRIoVAA",
  "correo": "Ricardoricardo.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					
Name	Passed	Failed	Skipped		
Codigo de estado 201, usuario creado	1	0	0		
Usuario creado - documentID: xtj33HcXNWUhfR1oMAZA - nick: Ricardo-Test	1	0	0		
<b>Total</b>	<b>2</b>	<b>0</b>	<b>0</b>		

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**

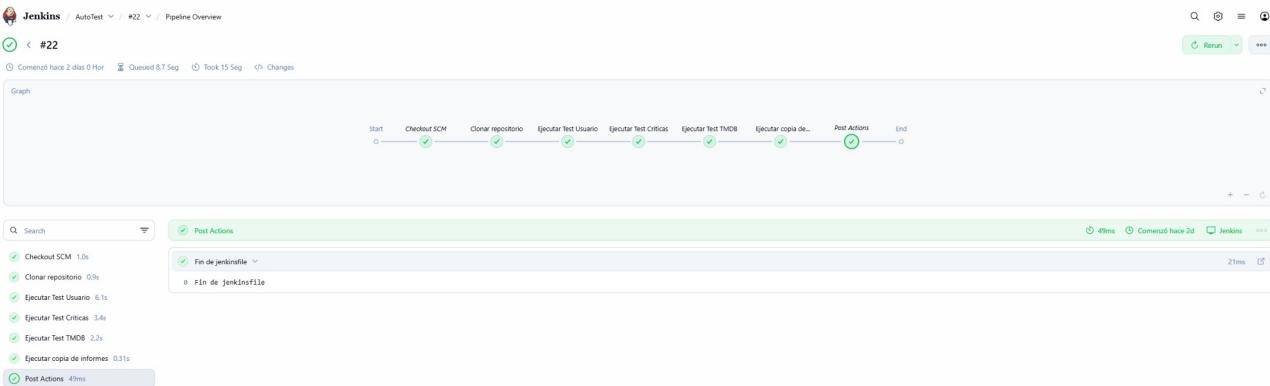
```
expected response to have status code 500 but got 404
```

**Jenkins:** Es un servicio de automatización de código abierto escrito en Java. Se utiliza para la integración 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.

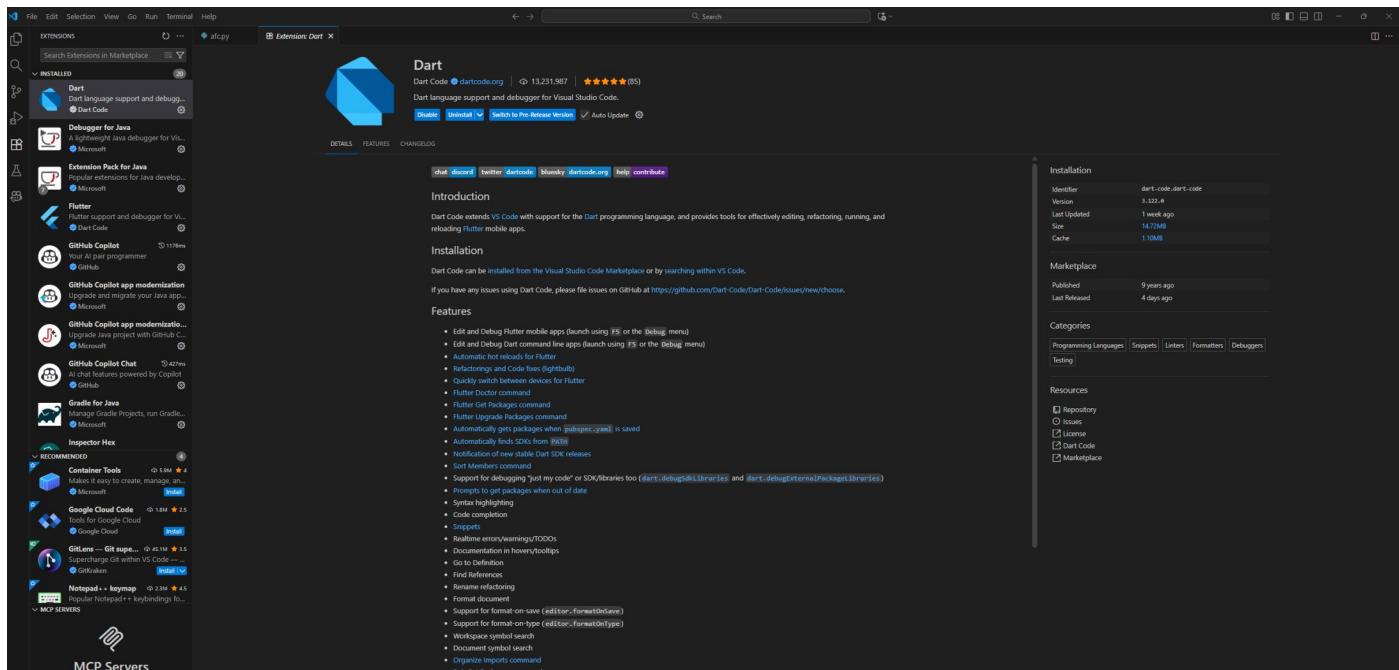
```
Started by GitHub push by RicardoHB996
Obtained Testing/jenkinsfile from git https://github.com/TFG-FlixScore/proyectoTFG.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\AutoTest
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: 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 b539048ca64278afdfa0241cdbc2973d25051f0a (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 b539048ca64278afdfa0241cdbc2973d25051f0a # 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 característica 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 lanzaría 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 grabación y reproducción de pruebas a través del driver del navegador correspondiente y la posibilidad de la exportación del código en múltiples lenguajes.

The screenshot shows the Selenium IDE interface. On the left, the 'TESTS' tab is selected, displaying a list of recorded steps: 1. Open https://gorgeted-lesly-aerodynamically.ngrok-free.dev/jobs/AutoTest, 2. Click css=.inline-flex, 3. Click id=j\_username, 4. Type id=j\_username admin, 5. Type id=j\_password admin, 6. Click css=label:nth-child(2), 7. Click css=label:nth-child(2), 8. Click name=Submit, 9. Click css=app-jenkins-logo > jenkins-mobile-hide, 10. Click css=#job\_AutoTest .jenkins-table\_link > span. The 'URL' field at the top is set to https://gorgeted-lesly-aerodynamically.ngrok-free.dev/jobs/AutoTest. To the right, a Jenkins integration panel is open, showing the 'Status' of the 'AutoTest' job. It lists several builds: #22 (18:18), #21 (18:05), #20 (18:03), #19 (18:02), #18 (18:01), #17 (17:36), #16 (17:33), #15 (17:31), #14 (17:26), and #13 (17:25). The Jenkins URL is also visible in the browser's address bar.

Generando con una exportación en python el siguiente código, agilizando el proceso de creación de pruebas en Python.

```
C: > Users > RICK > Desktop > new-test.py > ...
1  # Generated by Selenium IDE
2  import pytest
3  import time
4  import json
5  from selenium import webdriver
6  from selenium.webdriver.common.by import By
7  from selenium.webdriver.common.action_chains import ActionChains
8  from selenium.webdriver.support import expected_conditions
9  from selenium.webdriver.support.wait import WebDriverWait
10 from selenium.webdriver.common.keys import Keys
11 from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
12
13
14 class TestNewTest():
15
16     def setup_method(self, method):
17         self.driver = webdriver.Chrome()
18         self.vars = {}
19
20     def teardown_method(self, method):
21         self.driver.quit()
22
23     def test_newTest(self):
24         self.driver.get("https://gorgeted-lesly-aerodynamically.ngrok-free.dev ")
25         self.driver.find_element(By.CSS_SELECTOR, ".inline-flex").click()
26         self.driver.find_element(By.ID, "j_username").click()
27         self.driver.find_element(By.ID, "j_username").send_keys("admin")
28         self.driver.find_element(By.ID, "j_password").send_keys("admin")
29         self.driver.find_element(By.CSS_SELECTOR, "label:nth-child(2)").click()
30         self.driver.find_element(By.CSS_SELECTOR, "label:nth-child(2)").click()
31         self.driver.find_element(By.NAME, "Submit").click()
32         self.driver.find_element(By.CSS_SELECTOR, ".app-jenkins-logo > .jenkins-mobile-hide").click()
33         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.