

Módulo 3 — Scripts, testing automatizado y Postman vs. Bruno

Fundamentos de Postman y Bruno · Laboratorio Novus Technology

1. Pre-request Scripts — dónde escribirlos y para qué sirven

Código JavaScript que se ejecuta **antes** de enviar el request. Sirve para preparar datos: generar un timestamp, calcular una firma, o guardar un valor en una variable.

Paso a paso en Postman

1. 1 Abre el request donde quieras agregar el script.
2. 2 Ve a la pestaña **"Scripts"** → sub-pestaña **"Pre-request"**.
3. 3 Pega el código, por ejemplo:

```
const timestamp = Date.now();
pm.environment.set("timestamp", timestamp);
```

4. 4 Guarda el request (Ctrl/Cmd + S). El script se ejecutará automáticamente cada vez que envíes ese request.

Paso a paso en Bruno

1. 1 Abre el request y ve a la pestaña **"Script"** → sección **"Pre Request"**.
2. 2 Pega el código equivalente:

```
const timestamp = Date.now();
bru.setEnvVar("timestamp", timestamp);
```

3. 3 Guarda con Ctrl/Cmd + S.

2. Test Scripts (assertions) — dónde escribirlos y cómo ver el resultado

Código que se ejecuta **después** de recibir la respuesta, para validar automáticamente que todo salió como se esperaba.

Paso a paso en Postman

1. 1 En el mismo request, ve a la pestaña **"Scripts"** → sub-pestaña **"Post-response"** (aquí también pueden convivir con el código que guarda variables del Módulo 2).
2. 2 Pega:

```
pm.test("El status code es 200", function () {
  pm.response.to.have.status(200);
});

pm.test("La respuesta incluye un token", function () {
```

```
const data = pm.response.json();
pm.expect(data.token).to.exist;
});

pm.test("El tiempo de respuesta es menor a 500ms", function () {
  pm.expect(pm.response.responseTime).to.be.below(500);
});
```

3. **3** Envía el request ("Send").
4. **4** En el panel de respuesta (parte inferior), busca la pestaña **"Test Results"** — ahí verás cada `pm.test` con un check verde (pasó) o una X roja (falló), junto al mensaje de error si falló.

Paso a paso en Bruno

1. **1** Ve a la pestaña **"Tests"** del request (distinta de "Script", que es para pre/post request).
2. **2** Pega el código equivalente (sintaxis muy similar, basada en el mismo motor Chai):

```
test("El status code es 200", function () {
  expect(res.getStatus()).to.equal(200);
});

test("La respuesta incluye un token", function () {
  expect(res.getBody().token).to.exist;
});
```

3. **3** Envía el request. Los resultados aparecen en la pestaña **"Tests"** del panel de respuesta, con el mismo formato de check/X que Postman.

Por qué importa: con estos scripts, un "request manual" se convierte en un *caso de prueba* repetible. Guardando varios requests con sus tests dentro de una colección, puedes correr **toda la suite de pruebas de la API** con un solo clic.

3. Ejecutar una colección completa

Desde la interfaz — Postman Runner

1. **1** Click derecho sobre la colección → **"Run collection"** (o el botón "Runner" en la barra superior).
2. **2** Selecciona qué requests incluir (por defecto, todos) y el entorno a usar (ej. "QA").
3. **3** Click en **"Run"**.
4. **4** Verás un resumen con cuántos tests pasaron y cuántos fallaron, request por request.

Desde la interfaz — Bruno Runner

1. **1** Click derecho sobre la colección o una carpeta → **"Run"**.
2. **2** Selecciona el entorno activo.
3. **3** Bruno ejecuta todos los requests en orden y muestra el resultado de cada test, igual que Postman.

Desde línea de comandos — Newman (Postman)

1. **1** Instala Newman globalmente: `npm install -g newman`
2. **2** Exporta tu colección: click derecho en la colección → "Export" → guarda el archivo `coleccion.json`.

3. **3** Exporta también el entorno: en "Environments", click en los tres puntos → "Export" → `entorno.json`.
4. **4** Corre: `newman run coleccion.json -e entorno.json`
5. **5** Newman imprime en la terminal el resultado de cada test — ideal para integrarlo a un pipeline de CI/CD.

Desde línea de comandos — Bruno CLI

1. **1** Instala el CLI: `npm install -g @usebruno/cli`
2. **2** Desde la carpeta de tu colección (la carpeta real en disco), corre: `bru run --env qa`
3. **3** El resultado se imprime en la terminal, con el mismo propósito que Newman: integrarlo a CI/CD.

4. Postman vs. Bruno — ¿cuándo usar cada uno?

Criterio	Postman	Bruno
Cuenta requerida	Sí, para sincronizar en la nube	No
Dónde se guardan las colecciones	Nube de Postman (o export JSON manual)	Archivos <code>.bru</code> en tu carpeta de proyecto
Control de versiones (Git)	Posible, pero no es el flujo natural	Flujo natural — cada request es un archivo versionable
Colaboración en equipo	Workspaces compartidos en la nube	Compartiendo el repositorio Git del proyecto
Costo	Gratis con límites; planes pagos para equipos grandes	100% gratuito y de código abierto
Ideal para	Equipos grandes, QA dedicado, necesitas mocking/monitoring	Equipos que ya versionan todo en Git, prefieren herramientas livianas y offline-first

Resumen del módulo

- Pre-request Scripts van en la pestaña "Scripts → Pre-request" (Postman) o "Script → Pre Request" (Bruno).
- Test Scripts van en "Scripts → Post-response" (Postman) o "Tests" (Bruno), y los resultados se ven en la pestaña "Test Results"/"Tests" del panel de respuesta.
- Una colección completa se corre desde la UI (Runner) o por línea de comandos (Newman / Bruno CLI), lo que permite integrarla a CI/CD.
- Postman brilla en equipos grandes con necesidad de nube/integraciones; Bruno brilla en flujos ligados a Git y sin dependencia de cuentas externas.