

TEST LA VIE

De Selenium a Vibium
Cuando los agentes escriben el código, ¿quién lo comprueba?

Boni García
bonigarcia.dev

#VLCTESTING



**CENTRO TECNOLÓGICO
ESPECIALIZADO EN TIC**

De Selenium a Vibium



<https://x.com/hugs>

“*Started Selenium and Appium a long, long time ago. Now Vibium...*”

— Jason Huggins



Selenium

(2004)

Automatización de navegadores



appium

(2012)

Automatización móvil



Vibium

(2025)

Automatización de navegadores para agentes de IA y humanos

Sobre mí

WebDriverManager 
(2015)

Gestión automática de drivers para
Selenium en Java

 Selenium
(2022)

Committer y desarrollador de Selenium
Manager

Disclaimer:

- No formo parte del proyecto Vibium
- Es una revisión independiente y práctica
- Vibium es todavía experimental



“ *The verification layer for coding agents.
Agents make it.
We check it.*

— Vibium

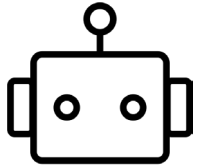


<https://github.com/VibiumDev/vibium>



<https://vibium.com/>

El agente dice: “Done. All tests pass”



- Added the EARLYBIRD banner and promo code
- Updated the HTML, CSS and JS files
- npm test: 16 passed

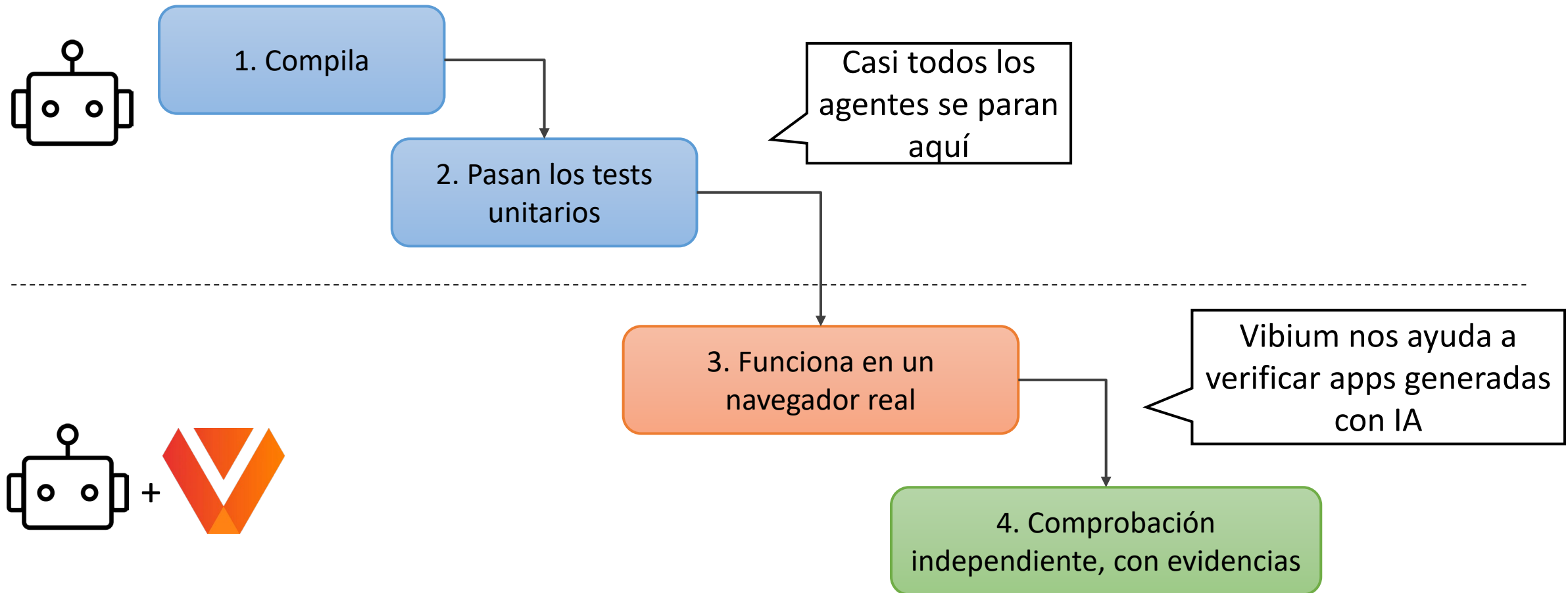
Done. All tests pass.

¿Quién lo verificó, y cómo?

¿Estaría de acuerdo un usuario?

Tests unitarios en verde ≠ algo que funciona para el usuario

De “hecho” a verificado



¿Qué es Vibium?

Vibium es una herramienta de automatización y verificación de navegadores, nativa para IA

Un único binario

~10 MB, escrito en Go, sin dependencias en tiempo de ejecución

Basado en estándares

W3C WebDriver BiDi
(Chrome y Firefox)

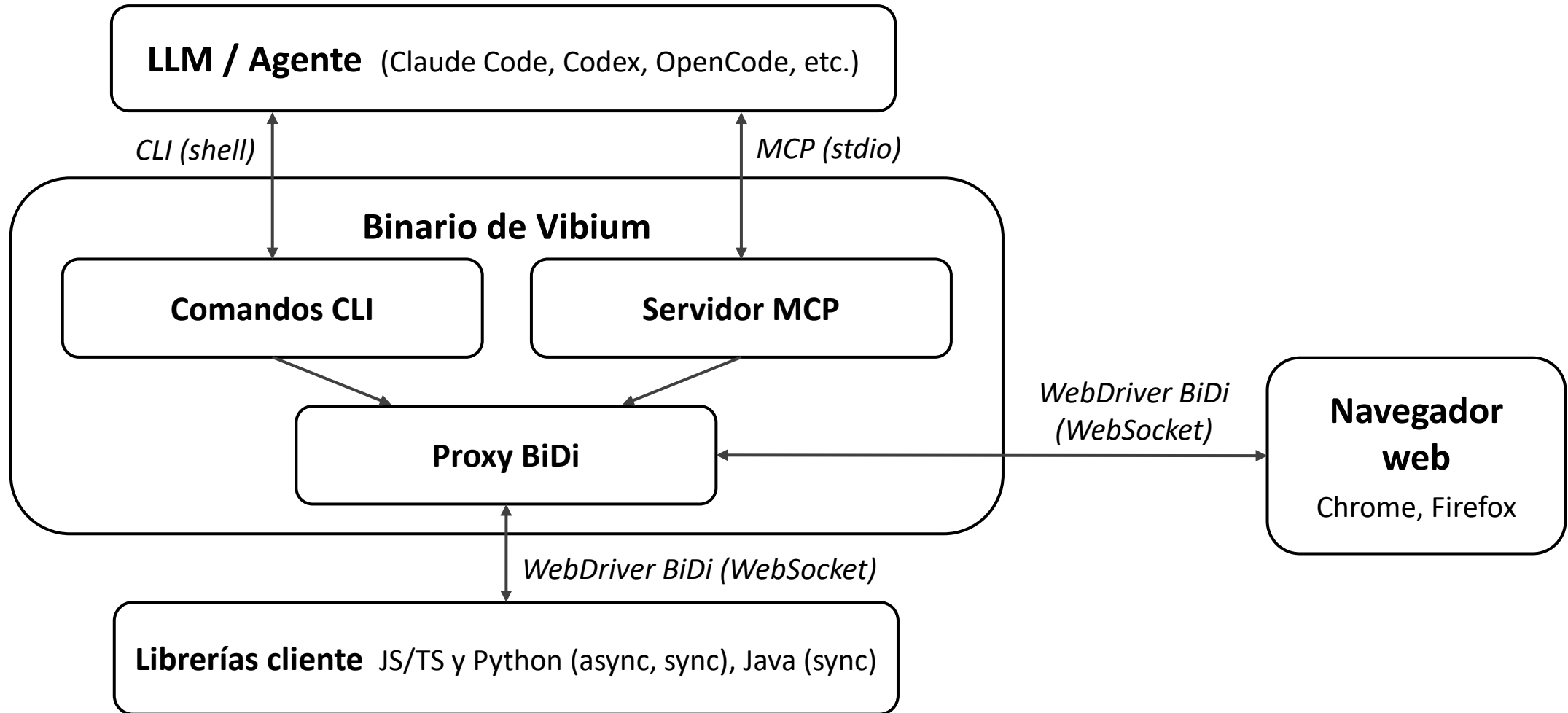
Tres interfaces

CLI · servidor MCP ·
API (JS/TS, Python, Java)

Código abierto

Apache-2.0, en desarrollo activo

Arquitectura



WebDriver BiDi en pocas palabras

WebDriver clásico

- HTTP, petición/respuesta
- El cliente pregunta qué ha pasado
- Recomendación W3C



WebDriver BiDi

- WebSocket, bidireccional
- El navegador envía eventos: red, consola, navegación...
- Estándar W3C en desarrollo

Novedad en la especificación:

`browsingContext.startScreencast` → grabación nativa de vídeo (Firefox 154+)

<https://w3c.github.io/webdriver-bidi/>

Instalación y configuración de Vibium

Versión estable: CLI, MCP y APIs

```
npm install -g vibium
```

Las nightly añaden run y check (IA)

github.com/VibiumDev/vibium/releases (nightly-*)

```
vibium setup    # navegador, proveedor IA, skills
```

```
vibium ready   # comprueba la configuración
```

Vibium no llama a ningún modelo

go · click · fill · screenshot...

Tú o tu agente decidís cada paso

Vibium llama a un modelo

run · check

Un modelo decide los pasos por ti

Ambas, desde la CLI, el servidor MCP y las APIs

Proveedores de modelos: OpenAI · Anthropic · Google · xAI · compatibles con OpenAI (p. ej. OpenRouter) · local

Se guarda en ~/.config/vibium/ai.env

Interfaces de Vibium – CLI

```
vibium go http://localhost:5173
vibium map
vibium fill @e1 "Ada Example"
vibium click @e6 # ref de map
vibium screenshot -o page.png
vibium stop
```

```
@e1 [input type="text"] name="name"
@e2 [input type="email"] name="email"
@e3 [select] "General, €120
      Student, €60"
@e4 [input type="text"] name="promo"
@e5 [input type="checkbox"] name="privacy"
@e6 [button type="submit"] "Register"
```

map lista los elementos interactivos con referencias cortas (@e1, @e2...)

Un daemon mantiene el navegador abierto entre comandos

Los mismos comandos que un agente puede ejecutar desde una shell

Interfaces de Vibium – API

```
import { browser } from 'vibium'
const bro = await browser.start()
const page = await bro.page()
await page.go('http://localhost:5173')
await page.find('#promo').fill('FRIENDS10')
await page.find('button[type=submit]').click()
await bro.stop()
```

JS/TS (async)

```
import com.vibium.Vibium;
var bro = Vibium.start();
var page = bro.page();
page.go("http://localhost:5173");
page.find("#promo").fill("FRIENDS10");
page.find("button[type=submit]").click();
bro.stop();
```

Java (sync)

```
from vibium import browser
bro = browser.start()
page = bro.page()
page.go("http://localhost:5173")
page.find("#promo").fill("FRIENDS10")
page.find("button[type=submit]").click()
bro.stop()
```

Python (sync)

Async y sync: JS/TS, Python

Sync: Java

Scripts ejecutables en examples/ (en el repo de GitHub)

Interfaces de Vibium – servidor MCP

```
{  
  "mcpServers": {  
    "vibium": {  
      "command": "vibium",  
      "args": ["mcp"]  
    }  
  }  
}
```

.mcp.json

90 herramientas de navegador, por ejemplo:

browser_navigate	browser_map
browser_click	browser_fill
browser_screenshot	
browser_record_start	
vibium_run	vibium_check

Definition of done for changes the user can see

Unit tests are necessary but not enough.

1. Run `npm test`. All tests must pass.
2. Start the dev server and open the page in a real browser with the Vibium tools.
3. Go through the complete user flow as a user would, including filling in and submitting the form.
4. Take a screenshot as evidence and report what you checked in the browser.

CLAUDE.md

El contexto del agente constructor:
herramientas (.mcp.json) + instrucciones
(CLAUDE.md)

Operaciones de IA: run y check

vibium run "objetivo"

→ completed / not_completed + evidencias

vibium check "afirmación"

→ PASS / FAIL / INCONCLUSIVE + evidencias

CLI · MCP · API

Runtime de Vibium
↔ modelo

Herramientas de
navegador

WebDriver BiDi ↔
navegador

Nuevo contexto: el verificador no hereda la conversación del agente que implementó la app

Manos a la obra: app de demo creada con IA

Fork me on GitHub

Acme Testing Conference 2026

Date: 12 November 2026

Venue: Acme Convention Center, Springfield

Program

1. 09:30 · The Pyramid Is a Lie (and Other Testing Shapes) · Marta Quillfeather
2. 10:30 · Flaky Tests: A Field Guide · Tobias Wrenfield
3. 11:30 · Unit Tests You Will Not Hate in Six Months · Ingrid Solvang
4. 14:00 · Browser Automation Without Tears · Leandro Pizarro
5. 15:00 · Testing in Production, On Purpose · Naoko Hartwell
6. 16:00 · When the AI Writes the Tests · Desmond Okafor-Lind

Registration

Name

Email

Ticket type

Promo code (optional)

☐ I accept the privacy policy

Register

App: HTML, CSS, JS · Vite · Vitest

Constructor: Claude Code + Vibium MCP

Verificador: vibium check, otro modelo

Ticket

Add a fixed banner at the bottom of the screen announcing the EARLYBIRD promo code (20% off until 31 October 2026), and make the code work.

App generada a partir de un prompt: [prompts/create-app.md](#)

<https://github.com/bonigarcia/vibium-demo>

Parte 1 – Un agente con “ojos”

Implementa banner + EARLYBIRD



npm test → todo en verde



Navegador: el clic en Register falla;
el banner está encima



Añade margen, vuelve a
comprobar → el flujo funciona

Registration

Name

Email

Ticket type
General (€120) ▼

Promo code (optional)

☐ I accept the privacy policy

Use promo code **EARLYBIRD** for 20% off until 31 October 2026

- Code and tests are in (16 passing); now checking the real page in the browser.
Read 1 file, called vibium 9 times
- Found that the banner covers the Register button during auto-scroll since it lands right behind it—bottom padding helps with manual scrolling, but I'll fix the auto-scroll issue too and rerun the flow from scratch.
- `Update(src\style.css)`

Parte 2 – Verificación independiente

```
vibium check "A banner at the bottom of the screen announces the promo code EARLYBIRD with 20% off until 31 October 2026"
```

PASS The banner visibly reads: "Use code EARLYBIRD for 20% off until 31 October 2026".

```
vibium check "Registering with name Ada Example, email ada@example.com, a General ticket, promo code EARLYBIRD and the privacy policy accepted shows Registration confirmed with a final price of 96 euros"
```

PASS After submission, the page displayed: "Registration confirmed: Ada Example, €96".

```
vibium check "The user can dismiss the promo banner"
```

FAIL The accessibility tree and interactive-element map expose no dismiss/close control for the promo banner.

Control: una afirmación que sabemos falsa (no hay botón de cerrar).
Si no sabe decir FAIL, su PASS no vale nada

demo/verify.sh

Verificador: openai/gpt-5.6-sol (vía OpenRouter), distinto del modelo constructor

Parte 2 – Verificación independiente

Scenario: Early-bird discount

Given the registration page

When I register as Ada Example
(ada@example.com), General ticket,
promo code EARLYBIRD,
privacy accepted

Then I see Registration confirmed
with a final price of **€96**

```
vibium check "Given the registration page, When I
register as Ada Example, ..., promo code EARLYBIRD,
privacy accepted, Then I see Registration confirmed
with a final price of €96"
```

PASS The page displayed "Registration confirmed: Ada Example, €96".

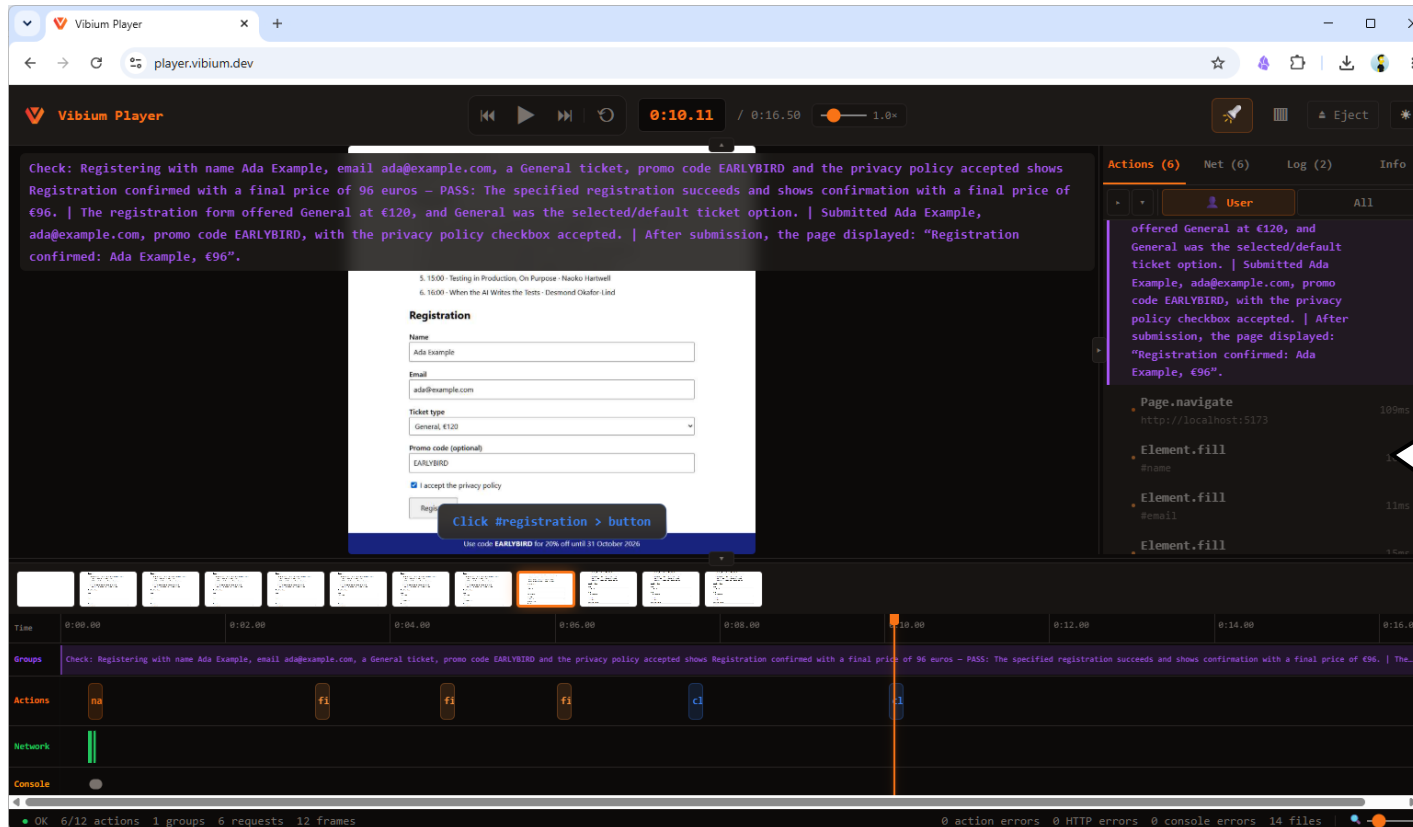
```
vibium check "Given the registration page, When I
register as Ada Example, ..., promo code EARLYBIRD,
privacy accepted, Then I see Registration confirmed
with a final price of €100"
```

FAIL Registration was confirmed, but the final price was €96, not the claimed €100.

El mismo escenario, pasado como una afirmación. Solo cambia el precio

Los escenarios Gherkin funcionan como afirmaciones: sin step definitions ni selectores

Parte 3 – Evidencias, no confianza



Lo que hizo de verdad el verificador:
abrió la página, rellenó el formulario, pulsó Register y leyó el resultado. Una captura por paso.

<https://player.vibium.dev/>

Un PASS también es una afirmación: la grabación permite comprobar cómo se llegó a él

Parte 3 – Evidencias, no confianza

```
vibium check -i evidence/claim-2-<label>.zip ^
```

"After submitting, no part of the Register button is covered by the banner"



The screenshot shows a registration form with the following elements:

- Ticket type:** A dropdown menu showing "General, €120".
- Promo code (optional):** A text input field containing "EARLYBIRD".
- Privacy policy:** A checkbox labeled "I accept the privacy policy" which is checked.
- Registration confirmed:** A green text message stating "Ada Example, €96".
- Register button:** A button labeled "Register".
- Banner:** A dark blue banner at the bottom of the form area with white text: "Use code **EARLYBIRD** for 20% off until 31 October 2026".

Último fotograma de la afirmación 2: tras la confirmación, Register toca el banner

Pregunta a la grabación, no a la app
Con -i solo se juzga la evidencia guardada: sin app, sin navegador y sin volver a ejecutar

También lee trazas de Playwright

--report verdict.json guarda el veredicto para CI

El veredicto dice lo que hay; si importa o no, lo decidimos nosotros

Vibium: resumen

*¿Está listo para
producción?*

Todavía no

Solo en builds nightly, y la
API todavía en desarrollo

*¿Me puedo fiar del
veredicto?*

En parte

Vale lo que vale la
afirmación, y el veredicto
cambia con el modelo

*¿Puede sustituir a mis
tests?*

No debería

Añade latencia y no es
determinista: los hallazgos
deben acabar en tests

Úsalo para explorar y verificar; fija lo que encuentre con tests deterministas

Basado en las builds nightly de Vibium, octubre de 2026

De hallazgos a tests deterministas

```
import { afterAll, beforeAll, expect, test } from 'vitest'
import { browser } from 'vibium'
import { PROMO_CODES, finalPrice } from '../src/registration.js'

const APP_URL = process.env.APP_URL ?? 'http://localhost:5173'
let bro
beforeAll(async () => {
  bro = await browser.start({ headless: true })
}, 60_000)
afterAll(async () => { await bro?.stop() })

for (const code of Object.keys(PROMO_CODES)) {
  test(`register with ${code}`, async () => {
    const page = await bro.page()
    await page.go(APP_URL)
    await page.find('#name').fill('Ada Example')
    // ... email, promo code, privacy ...
    await page.find('button[type=submit]').click()
    expect(await page.find('#confirmation').text())
      .toContain(`€${finalPrice('general', code)}`)
  }, 60_000)
}
```

e2e/registration.e2e.test.js

```
npm run test:e2e
```

“For stable assertions you run repeatedly, deterministic tests are usually faster, cheaper, and more repeatable.”

— Documentación de Vibium

`click()` espera hasta que nada tape el botón: si está tapado, el test falla

Conclusiones

1

El “done” de un agente es una afirmación, no una evidencia

2

Un navegador real, evidencias y una comprobación independiente convierten afirmaciones en resultados verificados

3

Es recomendable mantener al desarrollador en el bucle y seguir usando tests deterministas

De Selenium a Vibium

Las slides, en:



<https://bonigarcia.dev/>

¡Muchas gracias!

Boni García

boni.garcia@uc3m.es

La historia, en:



<https://medium.com/@boni.gg>

TEST LA VIE

#VLCTESTING



CENTRO TECNOLÓGICO
ESPECIALIZADO EN **TIC**