

GUÍA PASO A PASO

Prueba tu tienda desde Claude

La skill webapp-testing de Anthropic para explorar el recorrido de compra con Playwright, y cómo convertir esa exploración en un test que corre en tu CI. Con una tienda de prueba sin pagos y su test, probado en local.

PARA	Claude Code y proyectos con Node 18 o más reciente
PROBADO CON	@playwright/test 1.62 · Node 22 · Windows 11 · la skill no se ejecutó
ACTUALIZADA	26 de octubre de 2026
DOCUMENTACIÓN	github.com/anthropics/skills · playwright.dev
ARTÍCULO	ecomlabs.dev/g/test
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de
empezar

Qué vas a montar

Paso 01

Arranca la tienda y su test

Paso 02

Instala la skill

Paso 03

Explora el recorrido

Paso 04

Pásalo a un test

Paso 05

Comprueba que el test atrapa fallos

Paso 06

Llévalo a tu CI

Límites

Datos, pagos y producción

Si algo
falla

Solución de problemas

Al
terminar

Checklist

Qué vas a montar

La skill webapp-testing le da a Claude Code un método para probar webs locales: escribes scripts de Playwright en Python, arranca el servidor con `scripts/with_server.py`, esperas a `networkidle`, haces capturas y eliges selectores antes de actuar. Esta guía la usa para explorar el recorrido producto, carrito y checkout, y lleva ese recorrido a un test de `@playwright/test` que corre en CI.

CUIDADO

El test de la tienda está probado en local. La skill no la ejecutamos: necesita Playwright para Python, que no instalamos en el equipo de prueba. El workflow está validado, pero no lo corrimos en GitHub.

Lo que trae el ZIP

ARCHIVO	PARA QUÉ SIRVE
<code>server.js</code>	Sirve la tienda en el puerto 4173, sin dependencias.
<code>public/</code>	Inicio, producto, carrito y checkout sin pagos.
<code>tests/compra.spec.js</code>	El recorrido de compra con selectores por rol y texto.
<code>playwright.config.js</code>	Arranca la tienda antes de los tests.
<code>.github/workflows/e2e.yml</code>	Corre los tests en cada PR.

Requisitos

- Node 18 o más reciente (probado con Node 22).
- Para la skill: Python y Playwright para Python.

Arranca la tienda y su test

```
npm install
npx playwright install chromium
npm test
```

Deben pasar los dos tests. `npm start` sirve la tienda en `http://localhost:4173` para verla en el navegador. El servidor:

`server.js`

```
// Servidor mínimo de la tienda de prueba: sirve public/ con Node, sin dependencias.
import { createServer } from "node:http";
import { readFile } from "node:fs/promises";
import { extname, join, normalize, sep } from "node:path";
```

```
const raiz = join(import.meta.dirname, "public");
const tipos = { ".html": "text/html; charset=utf-8", ".js": "text/javascript; charset=utf-8" };
const puerto = Number(process.env.PORT ?? 4173);

createServer(async (req, res) => {
  const ruta = new URL(req.url, "http://localhost").pathname;
  const archivo = normalize(join(raiz, ruta === "/" ? "index.html" : ruta));
  if (!archivo.startsWith(raiz + sep)) {
    res.writeHead(403).end();
    return;
  }
  try {
    const cuerpo = await readFile(archivo);
    res.writeHead(200, { "content-type": tipos[extname(archivo)] ?? "application/octet-stream" }).end(cuerpo);
  } catch {
    res.writeHead(404, { "content-type": "text/plain; charset=utf-8" }).end("No encontrado");
  }
}).listen(puerto, () => console.log(`Tienda de prueba en http://localhost:${puerto}`));
```

Comprueba que la ruta pedida queda dentro de `public/` antes de leer el archivo, así una URL con `../` no sirve nada de fuera.

PASO 02

Instala la skill

```
pip install playwright
playwright install chromium
```

Y dentro de Claude Code:

```
/plugin marketplace add anthropics/skills
/plugin install example-skills@anthropic-agent-skills
```

El plugin `example-skills` trae doce skills, webapp-testing entre ellas. El repositorio avisa de que son ejemplos con fines de demostración y aprendizaje: lee el `SKILL.md` antes de usarla.

PASO 03

Explora el recorrido

Pide la exploración con la tienda parada; según su `SKILL.md`, la skill la arranca con `with_server.py`:

```
Arranca la tienda con "node server.js" en el puerto 4173. Recorre: abre la Camiseta básica,
añádela al carrito, ve al checkout y confirma con el email prueba@example.com. Haz una captura
en cada paso y dime qué selectores usarías para un test.
```

Revisa las capturas y la lista de selectores. Pide roles y textos visibles («el botón Añadir al carrito») y descarta clases de CSS o posiciones. Los scripts de la exploración no se guardan: sirven para mirar.

PASO 04

Pásalo a un test

Pasa ese recorrido a un test de `@playwright/test` en `tests/`, con selectores por rol y texto.

El de la tienda de prueba queda así:

`tests/compra.spec.js`

```
// Recorrido de compra en la tienda de prueba: producto, carrito y checkout.
// Usa roles y textos visibles, como los que encuentra la exploración.
import { expect, test } from "@playwright/test";

test("producto, carrito y checkout de prueba", async ({ page }) => {
  await page.goto("/");
  await page.getByRole("link", { name: "Camiseta básica" }).click();

  await page.getByRole("button", { name: "Añadir al carrito" }).click();
  await expect(page.getByRole("status")).toHaveText("Añadido al carrito");
  await expect(page.getByRole("link", { name: "Carrito (1)" })).toBeVisible();

  await page.getByRole("link", { name: "Carrito (1)" }).click();
  await expect(page.getByRole("cell", { name: "Camiseta básica" })).toBeVisible();
  await expect(page.getByRole("cell", { name: "19,99 €" })).toBeVisible();

  await page.getByRole("link", { name: "Ir al checkout" }).click();
  await page.getByLabel("Email").fill("prueba@example.com");
  await page.getByRole("button", { name: "Confirmar pedido de prueba" }).click();

  await expect(page.getByRole("heading", { name: "Pedido de prueba confirmado" })).toBeVisible();
  await expect(page.getByRole("link", { name: "Carrito (0)" })).toBeVisible();
});

test("el carrito vacío no deja ir al checkout", async ({ page }) => {
  await page.goto("/carrito.html");
  await expect(page.getByText("Tu carrito está vacío.")).toBeVisible();
  await expect(page.getByRole("link", { name: "Ir al checkout" })).toBeHidden();
});
```

Y la configuración, que arranca la tienda antes de los tests:

`playwright.config.js`

```
import { defineConfig, devices } from "@playwright/test";

export default defineConfig({
  testDir: "tests",
```

```

retries: process.env.CI ? 1 : 0,
use: { baseURL: "http://localhost:4173", trace: "on-first-retry" },
// Playwright arranca la tienda antes de los tests y la para al terminar.
webServer: { command: "node server.js", url: "http://localhost:4173", reuseExistingServer: !process.env.CI
},
projects: [{ name: "chromium", use: { ...devices["Desktop Chrome"] } }],
});

```

PASO 05

Comprueba que el test atrapa fallos

Un test que nunca falla no prueba nada. En `public/tienda.js`, borra la línea `guardar([]);` del final, que vacía el carrito tras el pedido:

`public/tienda.js`

```

// El carrito vive en localStorage: basta para una tienda de prueba sin backend.
const leer = () => JSON.parse(localStorage.getItem("carrito") ?? "[]");
const guardar = (carrito) => localStorage.setItem("carrito", JSON.stringify(carrito));
const euros = (centimos) => (centimos / 100).toFixed(2).replace(".", ",") + " €";

const contador = document.getElementById("carrito");
const pintarContador = () => (contador.textContent = `Carrito (${leer().length})`);
pintarContador();

document.getElementById("anadir")?.addEventListener("click", () => {
  guardar([...leer(), { nombre: "Camiseta básica", precio: 1999 }]);
  pintarContador();
  document.getElementById("aviso").textContent = "Añadido al carrito";
});

const filas = document.getElementById("filas");
if (filas) {
  const carrito = leer();
  for (const linea of carrito) {
    const fila = filas.insertRow();
    fila.insertCell().textContent = linea.nombre;
    fila.insertCell().textContent = euros(linea.precio);
  }
  document.getElementById("vacio").hidden = carrito.length > 0;
  document.getElementById("ir-checkout").hidden = carrito.length === 0;
}

document.getElementById("checkout")?.addEventListener("submit", (evento) => {
  evento.preventDefault();
  guardar([]);
  pintarContador();
  document.querySelector("main").innerHTML =

```

```
"<h1>Pedido de prueba confirmado</h1><p>No se cobró nada: esta tienda no tiene pasarela de pago.</p>";
});
```

Corre `npm test`. En nuestra prueba, el test de compra falló así:

```
Error: expect(locator).toBeVisible() failed
Locator: getByRole('link', { name: 'Carrito (0)' })
Expected: visible
Error: element(s) not found
```

Restaura la línea y vuelve a correrlo: los dos tests deben pasar.

PASO 06

Llévalo a tu CI

`.github/workflows/e2e.yml`

```
# Corre el recorrido de compra con Playwright en cada PR y en cada push a main.
name: E2E

on:
  pull_request:
  push:
    branches: [main]

jobs:
  e2e:
    runs-on: ubuntu-latest
    timeout-minutes: 15
    permissions:
      contents: read
    steps:
      - uses: actions/checkout@v6
      - uses: actions/setup-node@v7
        with:
          node-version: 22
      - run: npm ci
      - run: npx playwright install --with-deps chromium
      - run: npx playwright test
```

`npm ci` necesita el `package-lock.json` que crea `npm install`: súbelo al repositorio. En CI, `reuseExistingServer` es falso y Playwright arranca siempre su propia tienda.

LÍMITES

Datos, pagos y producción

- **Datos inventados:** `prueba@example.com`, nunca datos de clientes.

- **Sin pagos reales:** modo de prueba de tu pasarela; en Shopify, una tienda de desarrollo con pedidos de prueba.
- **Nunca contra producción:** crea pedidos, descuenta inventario, manda emails y ensucia la analítica.
- **Contraseña:** las tiendas de desarrollo de Shopify suelen tenerla; guárdala en un secreto del CI.

SI ALGO FALLA

Solución de problemas

SÍNTOMA	CAUSA PROBABLE	ARREGLO
«Executable doesn't exist»	Falta el navegador	<code>npx playwright install chromium</code>
El test falla por conexión	La tienda no arrancó	Revisa <code>webServer</code> y el puerto 4173
La skill no se usa	Plugin no instalado	Instala <code>example-skills</code> y reinicia
Los scripts de la skill fallan	Falta Playwright para Python	<code>pip install playwright</code>
<code>npm ci</code> falla en CI	No hay <code>package-lock.json</code>	Corre <code>npm install</code> y súbelo

AL TERMINAR

Checklist

- ☐ `npm test` pasa con la tienda intacta.
- ☐ El test falla si quitas el vaciado del carrito.
- ☐ Los selectores son por rol y texto visible.
- ☐ Los datos son inventados y no hay pagos reales.
- ☐ `e2e.yml` corre en cada PR.