API testing with Bruno

German perl workshop 2025

https://usebruno.com

What is Bruno?

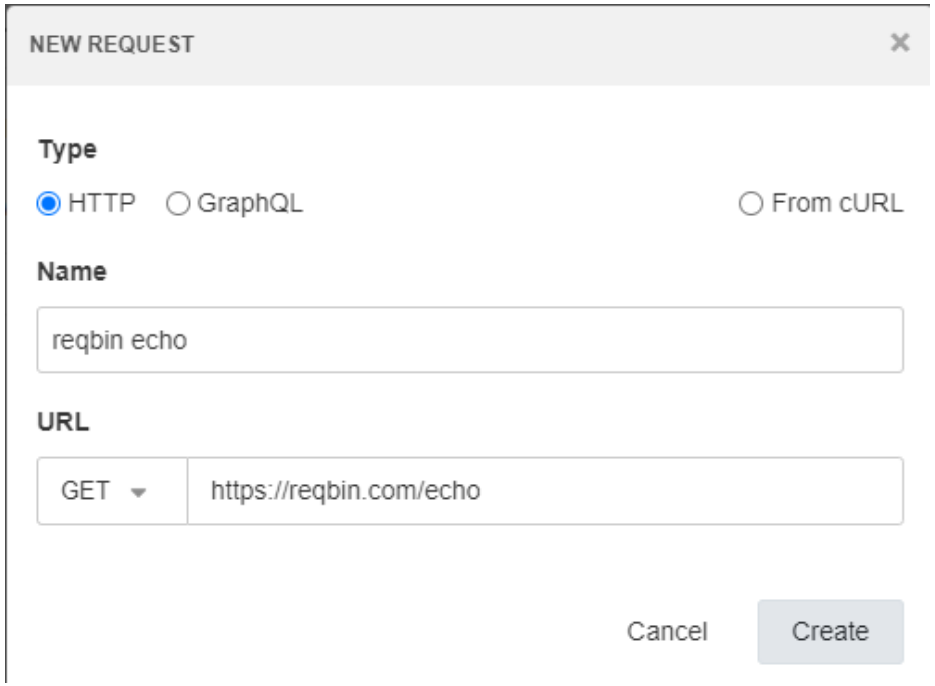
- What postman used to be or should be
- Rest api client gui.
- Text based storage (use git and diff)
- Command line tool for automated testing
- Open source: https://github.com/usebruno/bruno/
- But some paid features

Install Bruno desktop app

- From https://www.usebruno.com/
- Runs local on windows, linux, macOs
- Built using Next.js and React.
  Uses electron to ship a desktop version
- Tipp (for all electron apps): Ctrl-Shift-I for dev-tools (console)
- Demo data:
  - https://dummyjson.com
  - https://restful-api.dev
  - https://reqbin.com

Hello world

- Create collection
- Create request



- Run request

GET https://dummyjson.com/products/:ID?TOKEN={{TOKEN}}

Params 2 Body Headers Auth Vars Script • Assert 1 Tests

Docs

Query

Name	Path	
TOKEN	{{TOKEN}}	☑ 🗑
p	{{PRICE}}	☐ 🗑

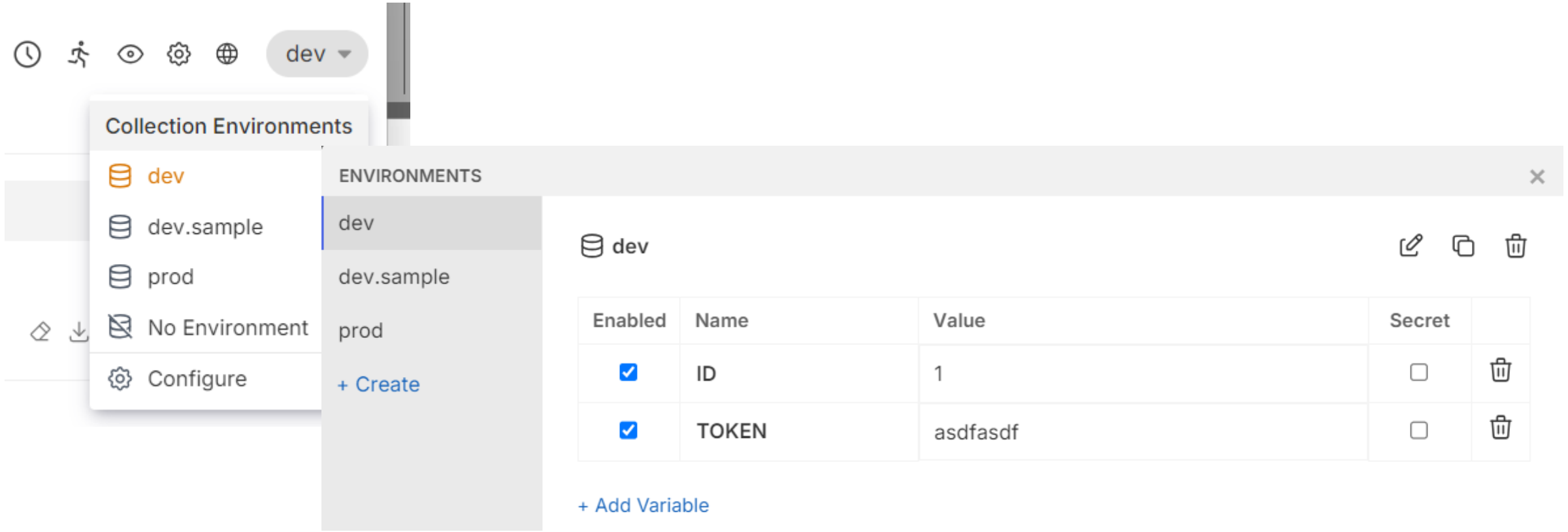
+ Add Param

Path ⓘ

Name	Value
ID	3

GET
POST
PUT
DELETE
PATCH
OPTIONS
HEAD

Environments



Response Headers 22 Timeline Tests 1 200 OK 363ms 1.43KB


```
{
  "id": 3,
  "title": "Powder Canister",
  "description": "The Powder Canister is a finely milled setting powder designed to set makeup and control shine. With a lightweight and translucent formula, it provides a smooth and matte finish.",
  "category": "beauty",
  "price": 14.99,
  "discountPercentage": 9.84,
  "rating": 4.64,
  "stock": 89,
  "tags": [
    "beauty",
    "face powder"
  ],
```

Response Headers 22 Timeline Tests 1 Clear Timeline

200 OK GET [2025-05-03T10:01:13.145Z] 2 minutes ago

https://dummyjson.com/products/3?TOKEN=asdfasdf

Request Response Network Logs

https://dummyjson.com/products/3?TOKEN=asdfasdf

▼ Headers

No Headers found

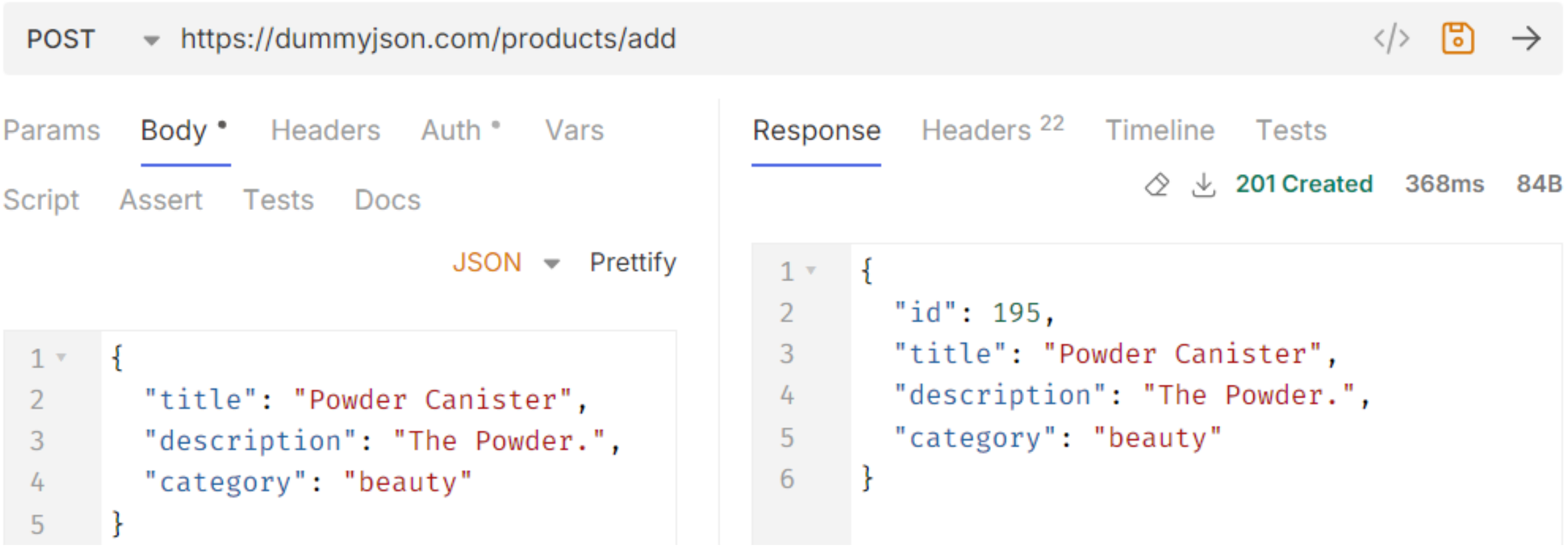
▼ Body

No Body found

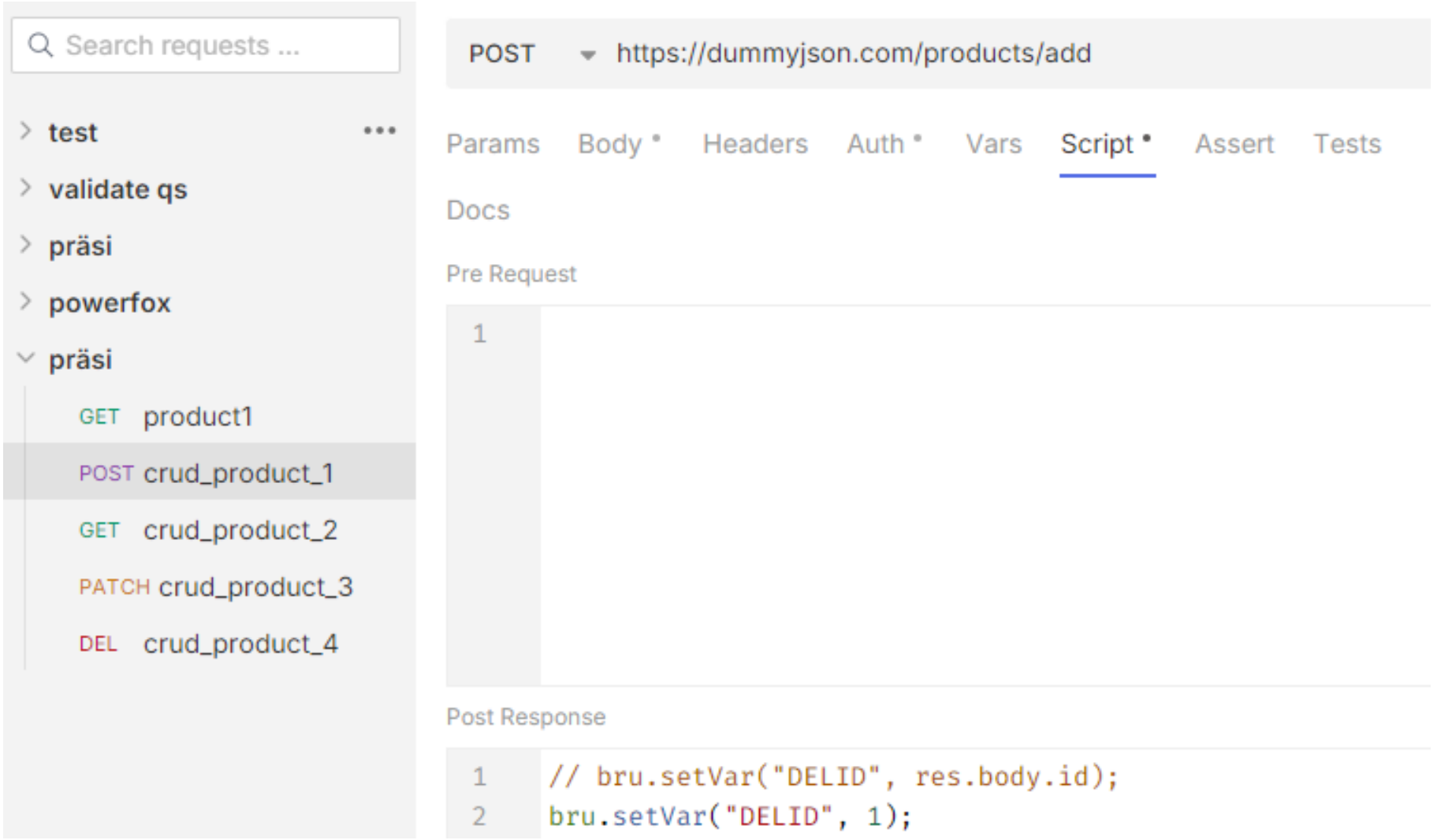
200 OK GET [2025-05-01T13:43:56.649Z] 2 days ago

https://dummyjson.com/products/4?TOKEN=asdfasdf

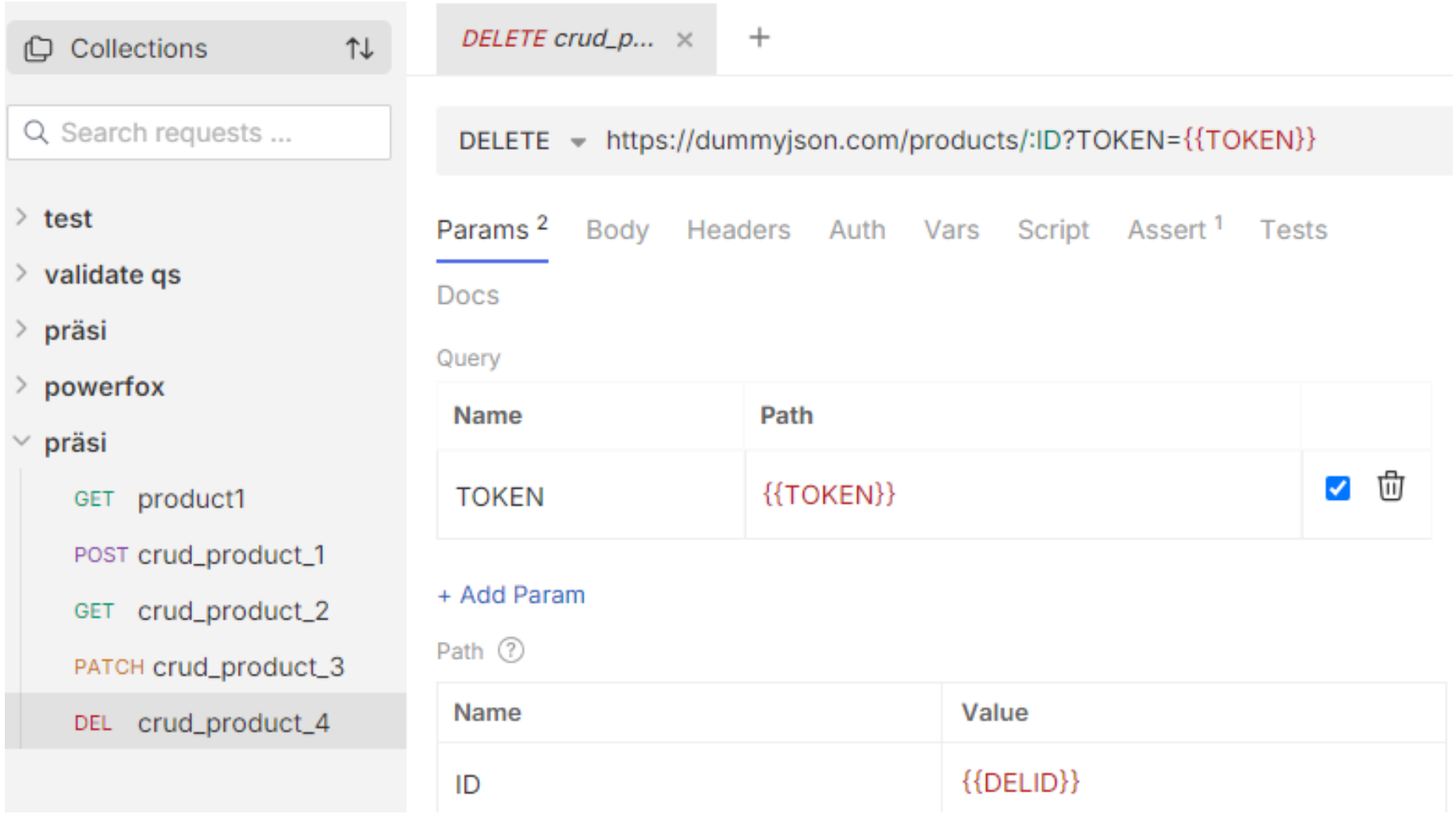
Create, change and delete in testing



Store response data in variable

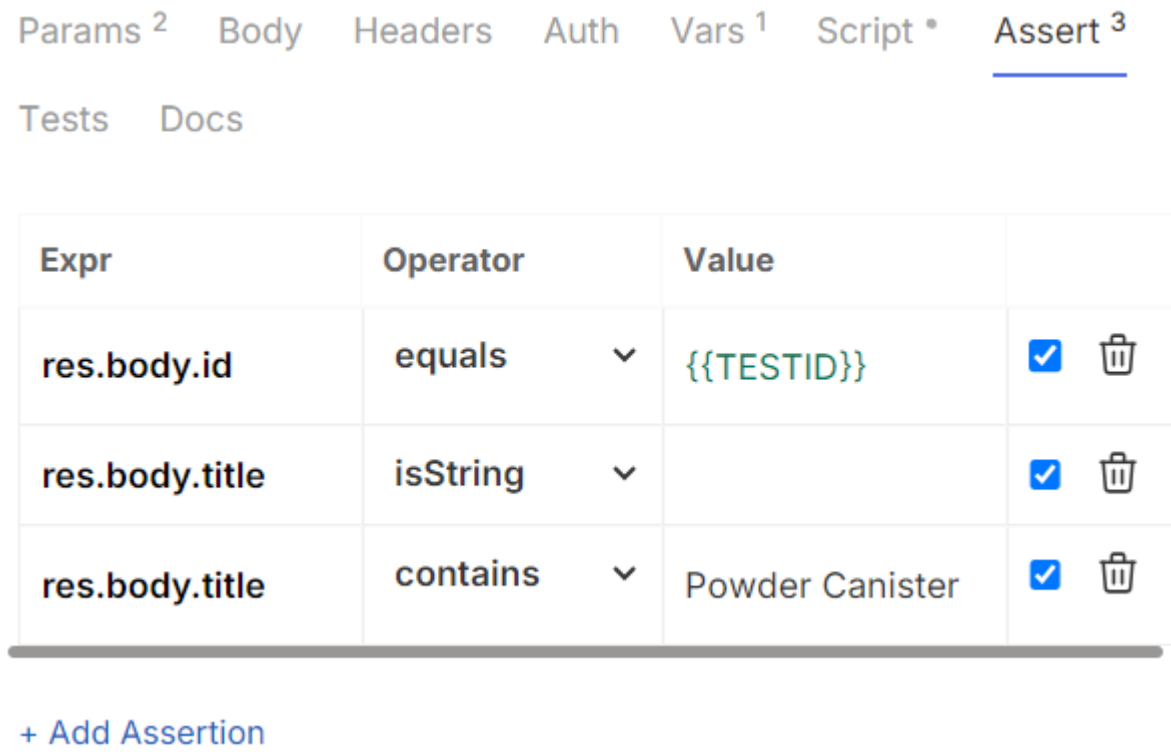


Reuse response data in variable

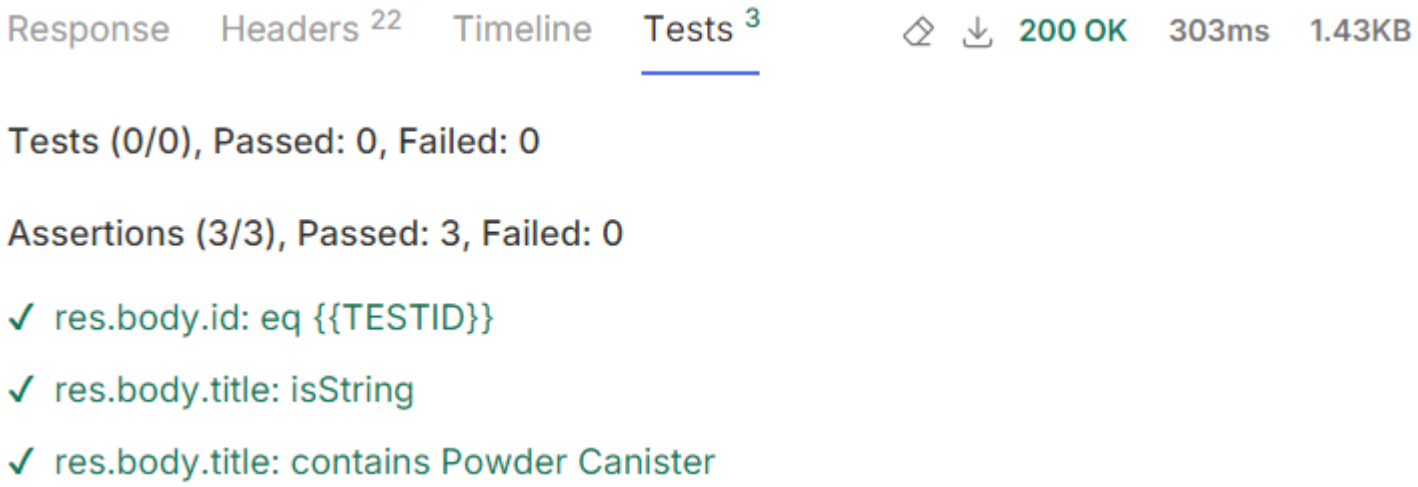


Check result

- Simple assertions



- result



Check result with Javascript (Tests)

Params Body Headers Auth • Vars Script • Assert

Tests • Docs

```
1 test("title should contain 'Mascara'", () => {
2   expect(res.body.title).to.contain('Mascara');
3 })
```

Response Headers 22 Timeline Tests 1 200 OK 307ms 1.47KB

Tests (1/1), Passed: 1, Failed: 0

✓ title should contain 'Mascara'

Assertions (0/0), Passed: 0, Failed: 0

Testing in Javascript - Buildins

- atob - Turn base64-encoded ascii data back to binary.
- btoa - Turn binary data to base64-encoded ascii.
- chai - BDD/TDD assertion library for node.js and the browser `https://www.chaijs.com/`.
- moment - Parse, validate, manipulate, and display dates and times in JavaScript.
- uuid - For the creation of RFC4122 UUIDs.
- nanoid - A tiny, secure, URL-friendly, unique string ID generator for JavaScript.
- crypto-js - JavaScript library of crypto standards.

Debugging tests with console.log()

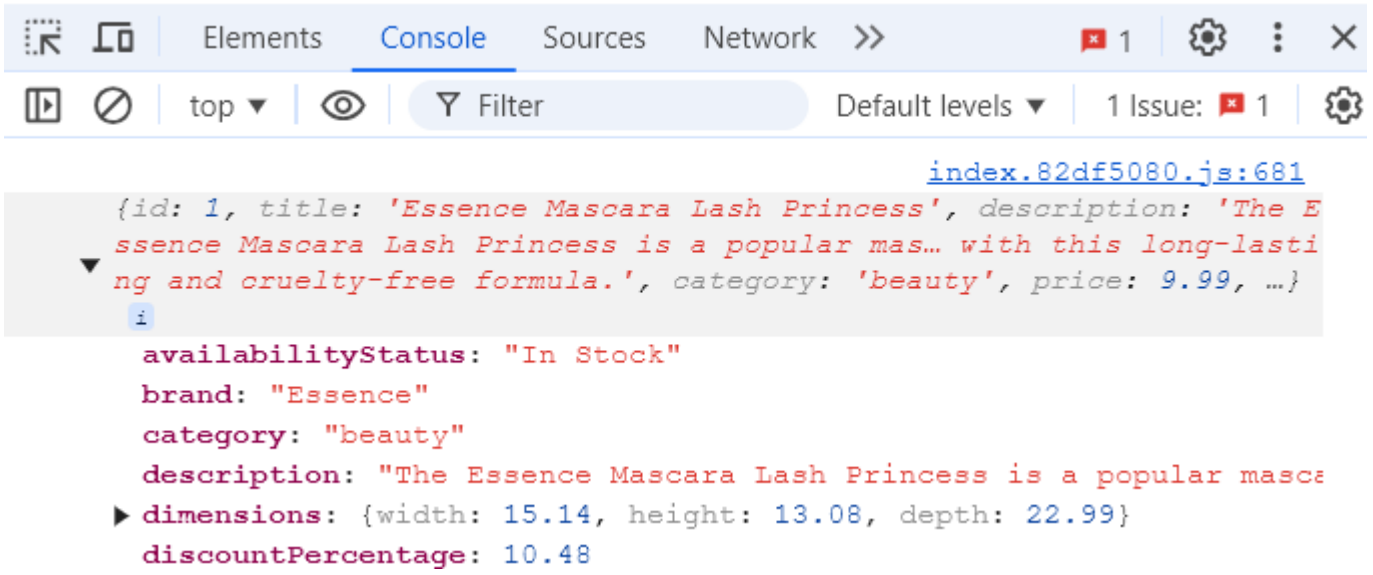
```
1  test("brand should contain 'Mascara'", () => {
2    console.log(res.body);
3    expect(res.body.brand).to.contain('Mascara');
4  })
```

Tests (1/1), Passed: 0, Failed: 1

✗ brand should contain 'Mascara'

expected 'Essence' to include 'Mascara'

Assertions (0/0), Passed: 0, Failed: 0



Bruno CLI

- npm install -g @usebruno/cli
- bru run request.bru
- bru run --env=dev request.bru

- npx @usebruno/cli run --env=dev
- npx @usebruno/cli run --env=dev --output results.html --format html

```
C:\Users\Public\Documents\bruno\praesi>set NODE_OPTIONS="--no-deprecation"

C:\Users\Public\Documents\bruno\praesi>npx @usebruno/cli run --env=dev
Running Folder Recursively

product1 (200 OK) - 401 ms
   ✓ assert: res.body.id: eq {{TESTID}}
   ✓ assert: res.body.title: isString
   ✓ assert: res.body.title: contains Powder Canister
crud_product_1 (201 Created) - 102 ms
crud_product_2 (200 OK) - 104 ms
   ✓ brand should contain 'Essence'
crud_product_3 (200 OK) - 101 ms
   ✓ assert: res.body.id: eq {{ID}}
crud_product_4 (200 OK) - 104 ms
   ✓ assert: res.body.id: eq {{ID}}

Requests:    5 passed, 5 total
Tests:       1 passed, 1 total
Assertions:  5 passed, 5 total
Ran all requests - 812 ms

Requests:    5 passed, 5 total
Tests:       1 passed, 1 total
Assertions:  5 passed, 5 total
```

Bruno run dashboard

Light

Summary | Requests

Iteration 1

Total requests	Total errors	Total Controls	Total Failed Controls
5	0	4	0

Total run duration

0.912 s

SUMMARY ITEM	TOTAL	PASSED	FAILED	SKIPPED	ERROR
Requests	5	5	0	0	0
Assertions	4	4	0	-	-

Bruno run dashboard Light

Summary | Requests

Iteration 1

Show All

product1 - 2/2 Passed

REQUEST INFORMATION

File
product1

Request Method
GET

RESPONSE INFORMATION

Response Code
200

Response time
375 ms

Real live (run in gitlab)

- Gitlab Pipeline schedules
- .gitlab-ci.yml
  stages: - brunoTest
  bruno_test:
  script:
    - pushd tests/Bruno/
    - npm install
    - npx @usebruno/cli run --env=dev
    - popd

tests/Bruno/package.json

package.json 131 B

```
1  {
2      "name": "bruno-validate-api-collection",
3      "version": "1.0.0",
4      "dependencies": {
5          "jsonschema": "^1.4.1"
6      }
7  }
```

- Version 1.5.0 is broken
- Install with „npm i“

Variables in

- Requests
- Collection
- Bruno Environment
- .env-file
- Environment vars

- Use: {{VAR}} or {{process.env.ENVVAR}}

Reusable code in js-module

helper.js 2.37 KiB

```
1  const getElem = (res, name, value) => {
2      return res.body.results[0][name].find((e)=>e.validatorName === value);
3  };
4
5  const testSchema = (res) => {
6      var Validator = require('jsonschema').Validator;
7      var v = new Validator();
8      var schema = {
9          "id": "/ValidateResultV1",
10         "type": "object",
11         "properties": {
12             "requestUUID": {
13                 "type": "string",
14                 "pattern": /^[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12}$/,
15             },
16             "status": {
```

Test with helper module

```
60    const h = require("./lib/helper");
61
62    // schema
63    test("body should be valid schema. check console on failure", () => {
64          expect(h.testSchema(res).errors).to.be.empty;
65    });
66
67    // firstName
68    test("firstName should have validator badword with result false", () => {
69          expect(h.getElem(res, "firstName", "badword").result).to.be.false;
70    });
```

Debugging

- With console.log
- With node debugger, but tricky.
- If you need a debugger, maybe Bruno is not the right tool
- If you program at lot, it is useful. But that‘s not what it‘s for

Can I do the same in Playwright?

Compare Bruno and Playwright

Bruno

+ easy install Gui and runtime
+ simple API testing
+ GUI to develop tests

Playwright

+ debuggable (e.g. vscode plugin)
+ complex API testing (prepare/cleanup)
+ integration with other testing

- big installation