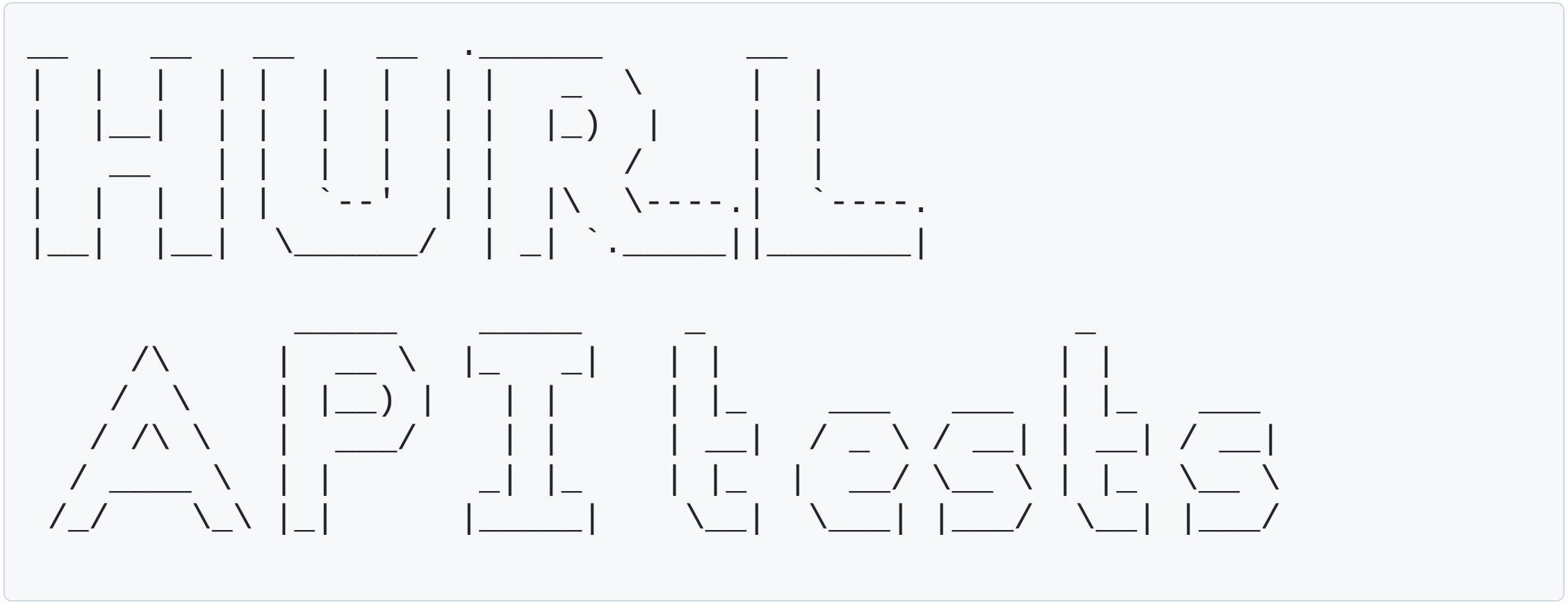
Hurl, API testing in plain text



HTTP APIs

About me

Silen Locatelli

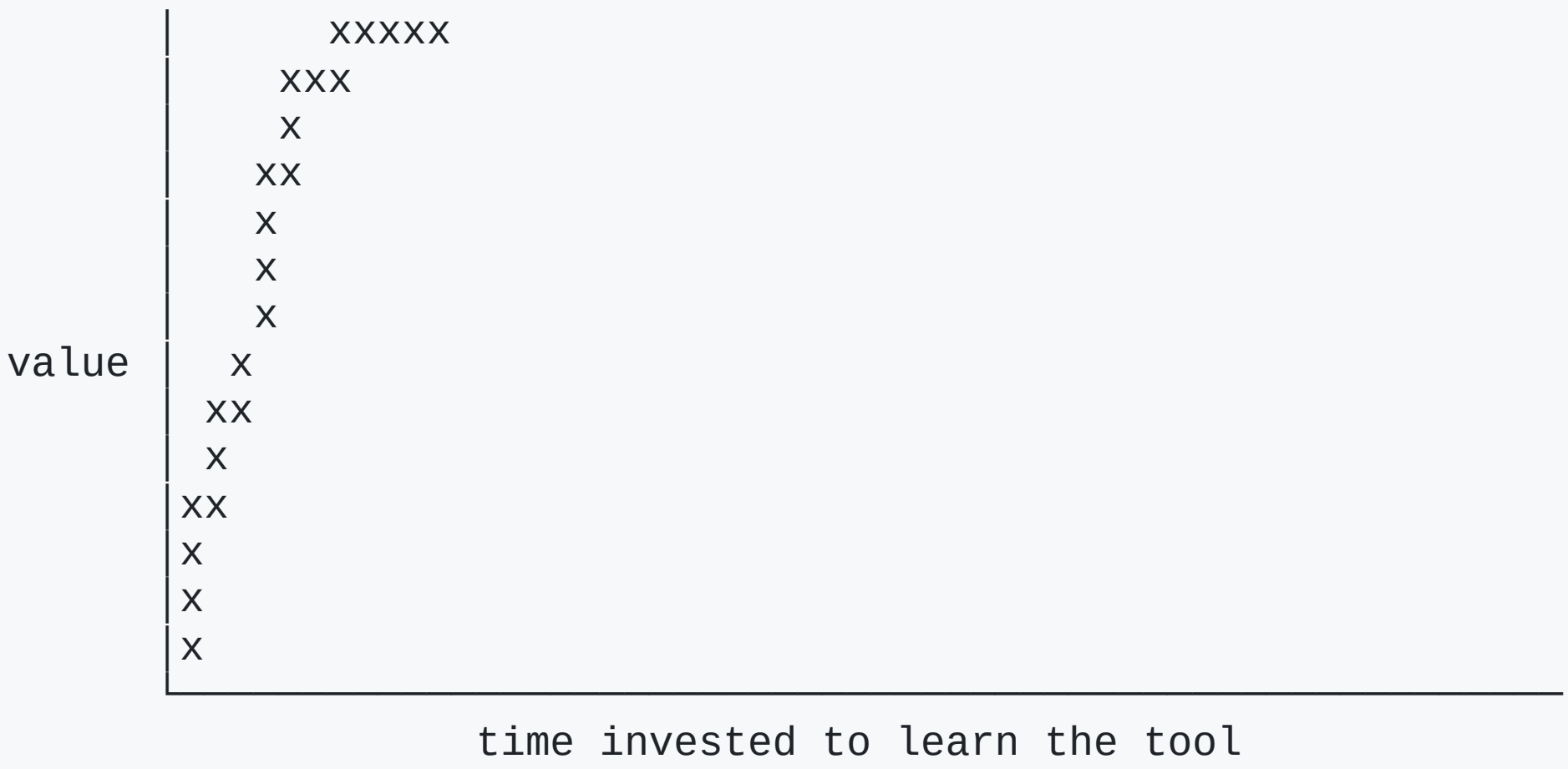
- Married
- I code, read and will start playing Handball next week
- Living in Baselland

About me

Silen Locatelli

- Optravis LLC
- Rust Basel
- Fullstack, lately with Rust + HTMX

What I search in tools



Who made Hurl?

- Owned by Orange - Open Source
- Apache License - Version 2.0

Main Maintainers:

- https://github.com/lepapareil
- https://github.com/fabricereix
- https://github.com/jcamiel

Written in Rust, depends on and lives off libcurl

What is Hurl?

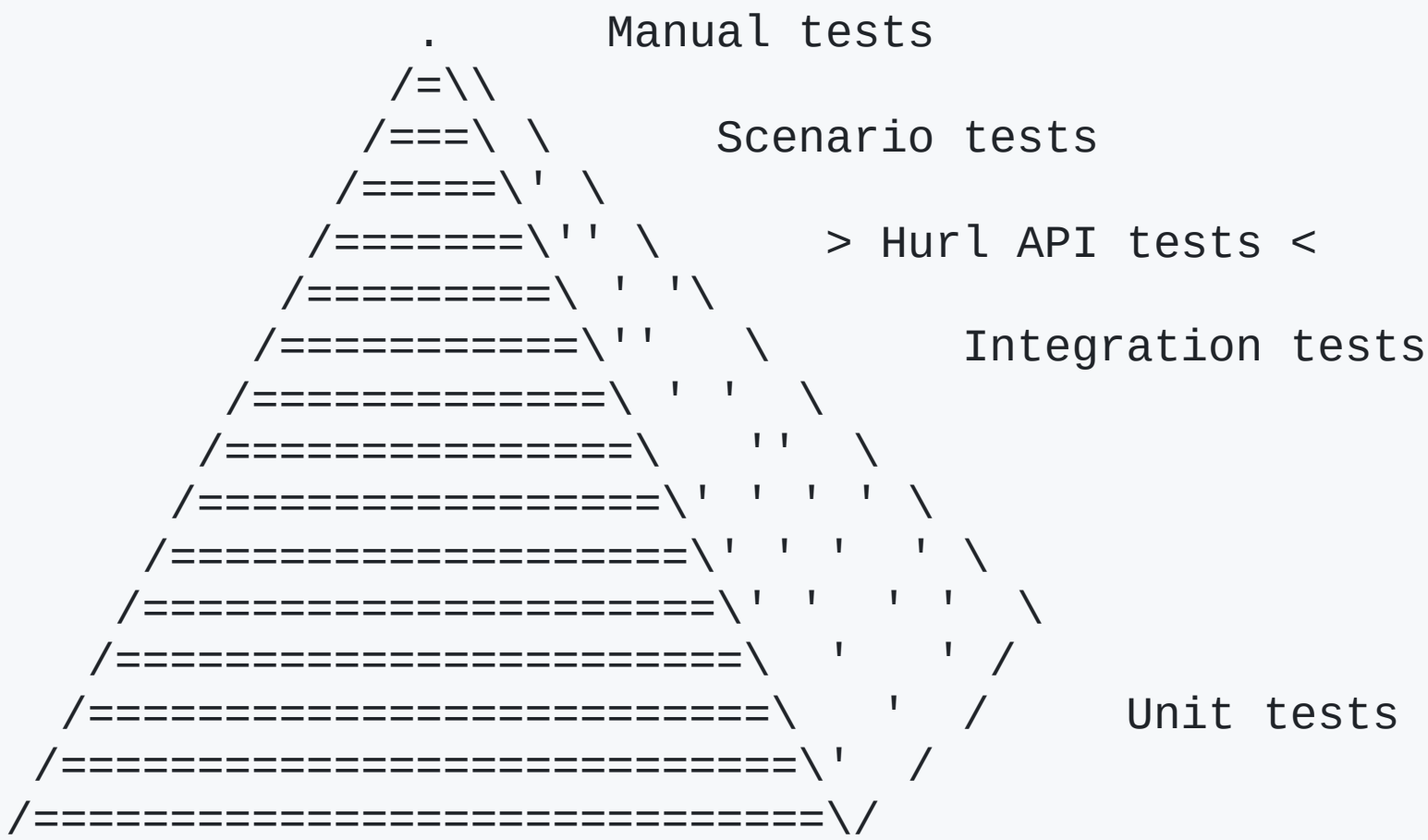
- Think: Postman - Selenium - Karate
- But:
  - Defined in a simple plain text format
  - Run with a CLI

File format

CLI (environment)

Output

In the testing pyramid



The Hurl file

example.hurl

```
# We can write comments and describe what we are doing in short
# Test if api/cats is available
GET {{target}}/api/cats
HTTP 200

# This is a second entry in the hurl file
GET {{target}}/api/cats
HTTP 200
[Asserts]
# We can assert headers
header "Content-Type" contains "application/json"
# We can assert the body with jsonpath over json
jsonpath "$.cats" count == 49
jsonpath "$.cats[0].name" == "Felix"
jsonpath "$.cats[0].lives" == 9
```

The Hurl file

example_2.hurl

```
# There is more
GET {{target}}/bar
x-any-header-i-want: some value
HTTP *


[Asserts]
# We can assert the body with filters over regex
regex /^(\d{4}-\d{2}-\d{2})$/ == "2018-12-31"
# We can assert send + response time
duration < 1000
# We can assert statuses with predicates
status < 300
```

The Hurl file

example_3.hurl

```
# There is more
POST {{target}}/bar

[FormParams]
name: John Doe
key1: {{some_variable}}

HTTP 204
```

The Hurl file

example_4.hurl

```
# There is more
GET {{target}}/api/cats

[Captures]
# We can capture a value
cat_name: jsonpath "$.cats[0].name"

# And reuse the value
POST {{target}}/api/pet/{{cat_name}}
HTTP 200

[Asserts]
body contains "Miau!"
```

Showcase

Install:

npm

npm install --save-dev @orangeopensource/hurl

unix

curl / sh
brew install hurl

mac

brew install hurl
port install hurl

Showcase

Install:

windows

scoop install hurl
choco install hurl
winget install hurl
installer

via rust

cargo install hurl

Showcase

docker

docker pull ghcr.io/orange-opensource/hurl:latest

- https://hurl.dev/docs/installation.html

Showcase

```
cd showcases/rocket
hurl --version
```

```
cat api_tests/variables
cat api_tests/implemented/healthz.hurl
cat api_tests/implemented/protected.hurl
cat api_tests/implemented/create_pokemons.hurl
cat api_tests/implemented/pokemon_html.hurl
```

Showcase

manual

```
lsof -t -i:8090 | xargs -r kill
cargo run & 2>&1
hurl api_tests/implemented/healthz.hurl --retry 4 --delay 1000 --variables-file api_tests/variables --test
hurl api_tests/implemented/*.hurl --variables-file api_tests/variables --test
lsof -t -i:8090 | xargs -r kill
echo "Verify done"
```

just task runner

```
just --dry-run api_tests
just api_tests
```

sh

```
cat api_tests.sh
sh api_tests.sh
```

Adjustments

```
lsof -t -i:8090 | xargs -r kill
cargo run & 2>&1
hurl api_tests/implemented/healthz.hurl --retry 4 --delay 1000 --variables-file api_tests/variables --test
hurl api_tests/implemented/*.hurl --variables-file api_tests/variables --test   --report-html .
lsof -t -i:8090 | xargs -r kill
echo "Verify done"
```

IMO

- on the API
  - meaningful siblings (post/get)
  - a good feeling about the actual complexity
  - precise and close to production tests
  - a good understanding of what state means in your application

IMO

- on your local environment
  - a tested run command
  - tested environment variables
  - a tested docker-compose file

And very important: A good night sleep

When to stop

- There is no "best setup"
- Hurl is as slow (fast) as your application
- Hurl will reflect the complexity of your API, all of it.

Hurl is the right choice

- if you prefer CLI
- if you prefer plain text
- if you like to test with curl
- if you like small overhead

Hurl is the wrong choice

- if you prefer GUIs
- if you want to script
- if you do not care about it

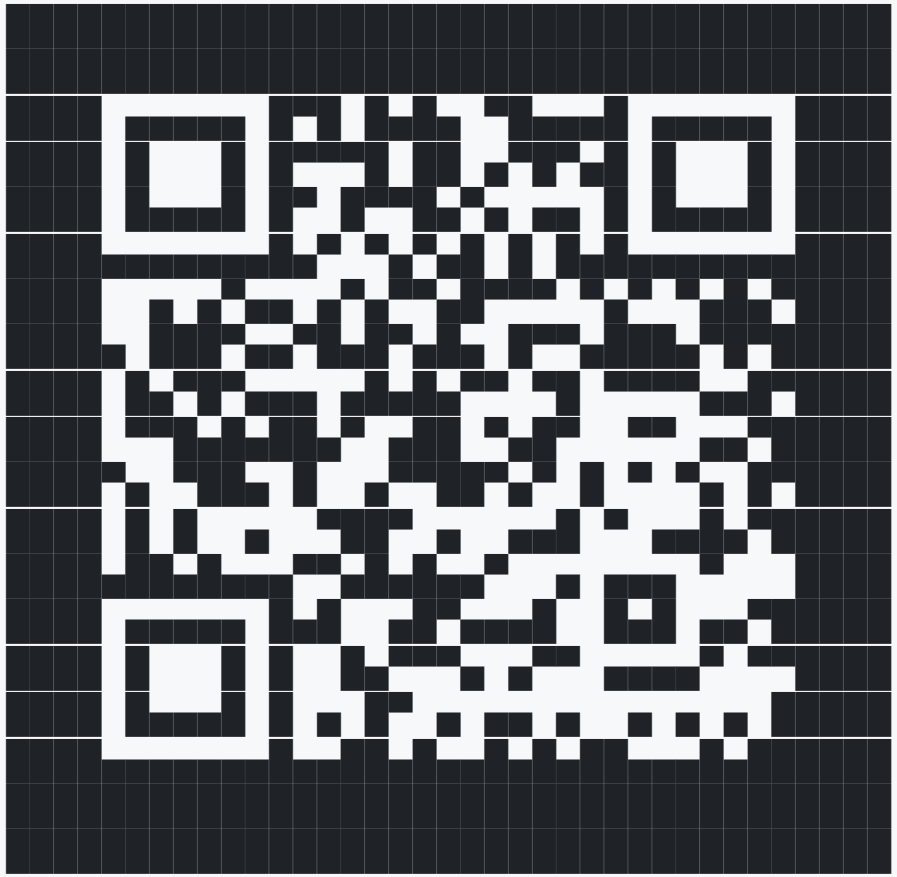
Things left to say

> The Hurl project is maintainer and usage oriented

> Hurl is not the whole story

> -> Variable injection is the way to get data into Hurl
> -> if you think you need something else
> -> use http

The talk and more links at github



Points to https://github.com/SilenLoc/baselOne2024