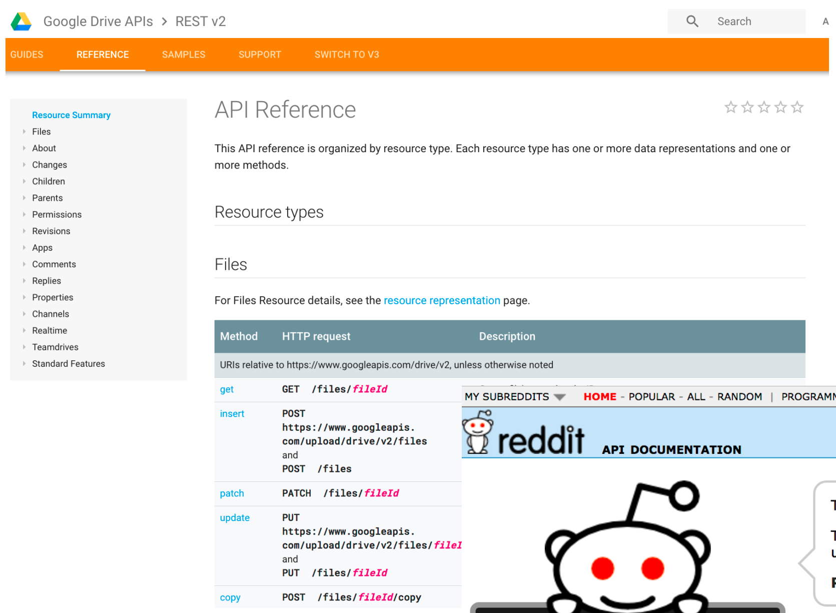
Fuzz Testing Web APIs: Overview of Existing Tools

Prof. Andrea Arcuri

Kristiania University of Applied Sciences
and OsloMet

REST APIs are used everywhere...



API Reference

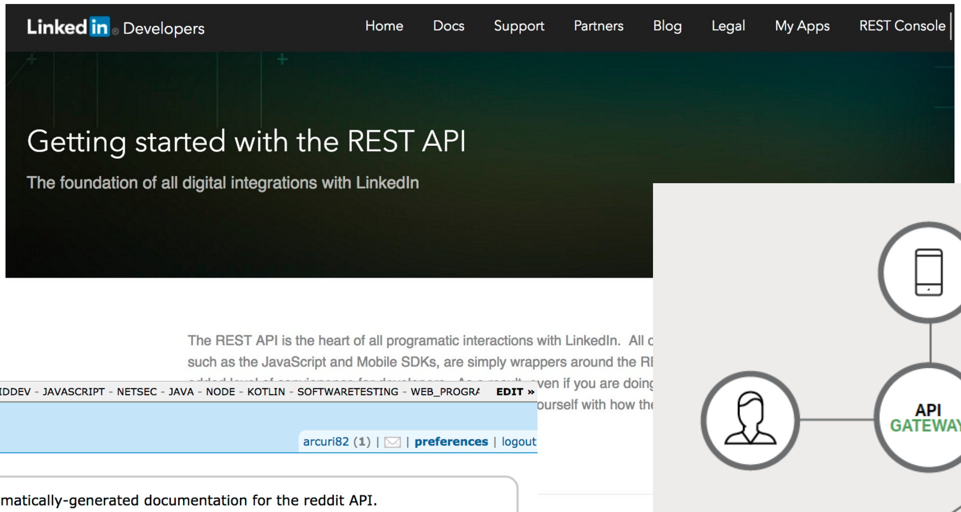
This API reference is organized by resource type. Each resource type has one or more data representations and one or more methods.

Resource types

Files

For Files Resource details, see the resource representation page.

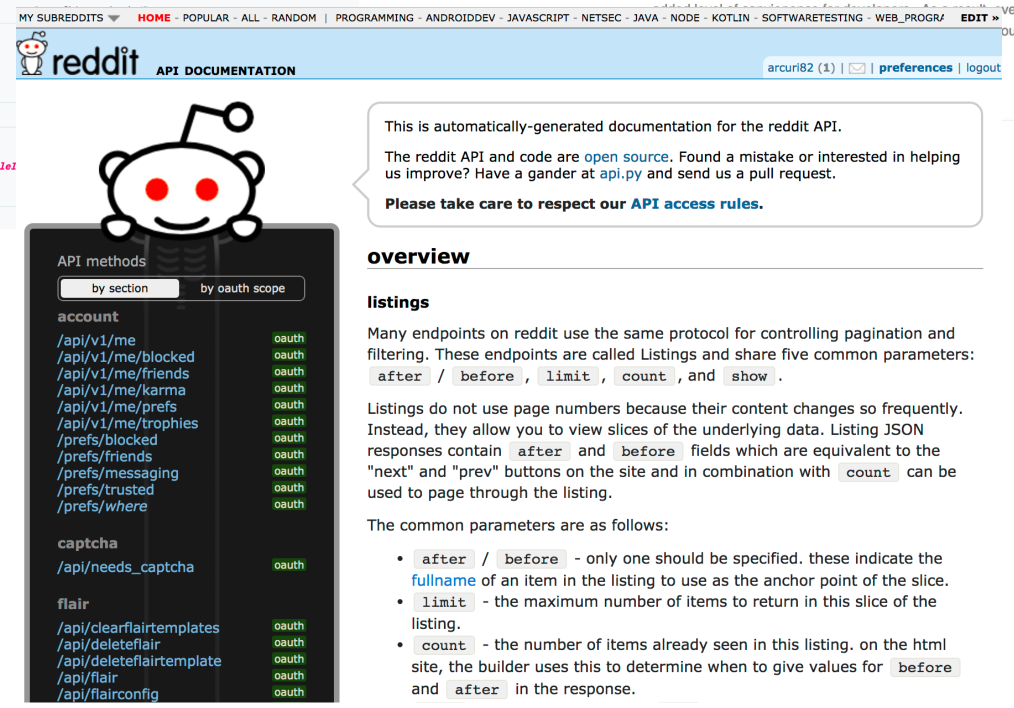
Method	HTTP request	Description
URIs relative to https://www.googleapis.com/drive/v2, unless otherwise noted		
get	GET /files/*fileId*	
insert	POST https://www.googleapis.com/upload/drive/v2/files and POST /files	
patch	PATCH /files/*fileId*	
update	PUT https://www.googleapis.com/upload/drive/v2/files/*fileI* and PUT /files/*fileId*	
copy	POST /files/*fileId*/copy	



Getting started with the REST API

The foundation of all digital integrations with LinkedIn

The REST API is the heart of all programatic interactions with LinkedIn. All c
such as the JavaScript and Mobile SDKs, are simply wrappers around the RE
ven if you are doing
ourself with how the


MY SUBREDDITS
HOME - POPULAR - ALL - RANDOM | PROGRAMMING - ANDROIDDEV - JAVASCRIPT - NETSEC - JAVA - NODE - KOTLIN - SOFTWARETESTING - WEB_PROGR/ EDIT »
reddit API DOCUMENTATION
arcuri82 (1) | | preferences | logout
API methods
by section
by oauth scope

account

/api/v1/me oauth
/api/v1/me/blocked oauth
/api/v1/me/friends oauth
/api/v1/me/karma oauth
/api/v1/me/prefs oauth
/api/v1/me/trophies oauth
/prefs/blocked oauth
/prefs/friends oauth
/prefs/messaging oauth
/prefs/trusted oauth
/prefs/*where* oauth

captcha

/api/needs_captcha oauth

flair

/api/clearflairtemplates oauth
/api/deleteflair oauth
/api/deleteflairtemplate oauth
/api/flair oauth
/api/flairconfig oauth

This is automatically-generated documentation for the reddit API.

The reddit API and code are open source. Found a mistake or interested in helping us improve? Have a gander at api.py and send us a pull request.

Please take care to respect our API access rules.

overview

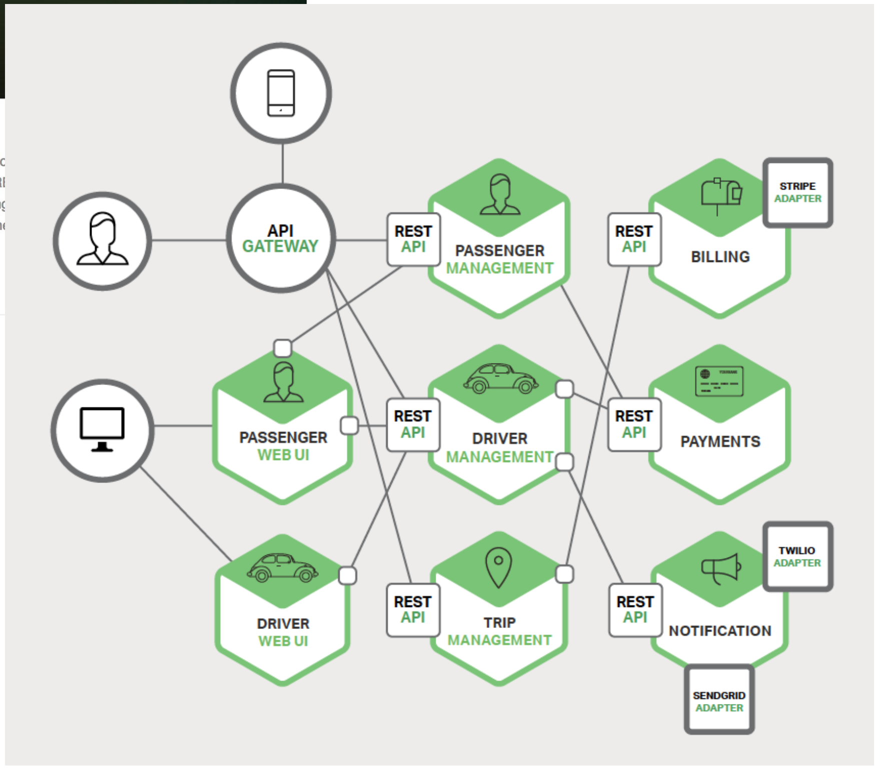
listings

Many endpoints on reddit use the same protocol for controlling pagination and filtering. These endpoints are called Listings and share five common parameters: `after` / `before`, `limit`, `count`, and `show`.

Listings do not use page numbers because their content changes so frequently. Instead, they allow you to view slices of the underlying data. Listing JSON responses contain `after` and `before` fields which are equivalent to the "next" and "prev" buttons on the site and in combination with `count` can be used to page through the listing.

The common parameters are as follows:

- `after` / `before` - only one should be specified. these indicate the fullname of an item in the listing to use as the anchor point of the slice.
- `limit` - the maximum number of items to return in this slice of the listing.
- `count` - the number of items already seen in this listing. on the html site, the builder uses this to determine when to give values for `before` and `after` in the response.



REST API Testing Challenges

- How to choose **query** and **path** parameters?
- How to prepare **body payloads** (e.g. JSON)?
- How to choose data to insert into **SQL** databases?
- Goals:
  - **Finding faults** (eg crashes, security issues)
  - **Maximize schema coverage**
  - **Maximize code coverage**
- Writing high coverage tests *by hand* for every single endpoint is time consuming

What about **Automated Test Generation** for RESTful APIs?

- Automatically write all the test cases
- Not just execution, but choice of all the inputs
- Hard, complex problem

2 Uses of Generated Tests

- If automated oracles: **automatically detect faults**
  - e.g., HTTP response giving 500, schema mismatches, security vulnerability
- No oracles / faults: **regressing testing**
  - Tests can be added to Git, to capture current behavior of system
  - If in future introduce new bug that breaks functionality, regression tests will start to fail

Fuzzers

- Tools that automatically generate test inputs
- Different strategies: from **random** inputs to advanced **AI** techniques
- Can automatically create and evaluate **millions** of test cases
- Used in many different domains
  - eg, parser libraries and unit testing
- REST fuzzing is a more recent development
  - eg, Restler, Schemathesis, RESTest, Fuzz-Lightyear and EvoMaster

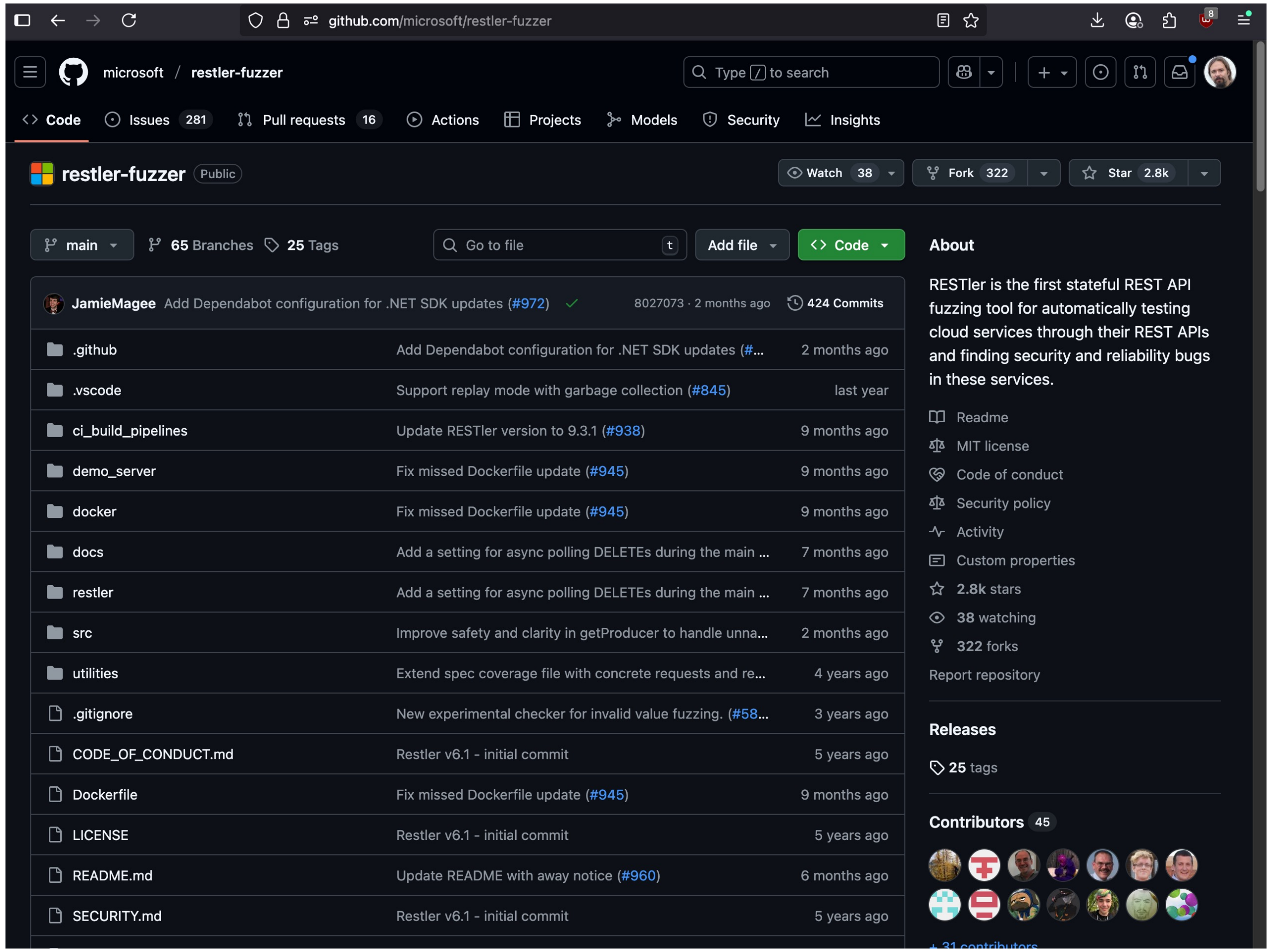
Fuzzers for REST APIs

- There are at least **25** open-source fuzzers out there for REST APIs
  - but many are just academic proof-of-concept
  - few have been discontinued (eg Dredd)

- Top 4 *currently maintained* fuzzers on **GitHub**
  - as of October 2025
- **RESTler** (+2800⭐)
- **Schemathesis** (+2700⭐)
- **CATS** (+1300⭐)
- **EvoMaster** (+600⭐)

RESTler

- Made by Microsoft Research
- Open-source since 2020
- Written in Python
- Requires cloning Git repository to build locally
- +2800⭐



github.com/microsoft/restler-fuzzer

microsoft / restler-fuzzer

Code · Issues 281 · Pull requests 16 · Actions · Projects · Models · Security · Insights

restler-fuzzer Public

Watch 38 · Fork 322 · Star 2.8k

main · 65 Branches · 25 Tags · Go to file · Add file · Code

JamieMagee Add Dependabot configuration for .NET SDK updates (#972) ✓ 8027073 · 2 months ago · 424 Commits

.github	Add Dependabot configuration for .NET SDK updates (#...	2 months ago
.vscode	Support replay mode with garbage collection (#845)	last year
ci_build_pipelines	Update RESTler version to 9.3.1 (#938)	9 months ago
demo_server	Fix missed Dockerfile update (#945)	9 months ago
docker	Fix missed Dockerfile update (#945)	9 months ago
docs	Add a setting for async polling DELETEs during the main ...	7 months ago
restler	Add a setting for async polling DELETEs during the main ...	7 months ago
src	Improve safety and clarity in getProducer to handle unna...	2 months ago
utilities	Extend spec coverage file with concrete requests and re...	4 years ago
.gitignore	New experimental checker for invalid value fuzzing. (#58...	3 years ago
CODE_OF_CONDUCT.md	Restler v6.1 - initial commit	5 years ago
Dockerfile	Fix missed Dockerfile update (#945)	9 months ago
LICENSE	Restler v6.1 - initial commit	5 years ago
README.md	Update README with away notice (#960)	6 months ago
SECURITY.md	Restler v6.1 - initial commit	5 years ago

About

RESTler is the first stateful REST API fuzzing tool for automatically testing cloud services through their REST APIs and finding security and reliability bugs in these services.

Readme
MIT license
Code of conduct
Security policy
Activity
Custom properties
2.8k stars
38 watching
322 forks
Report repository

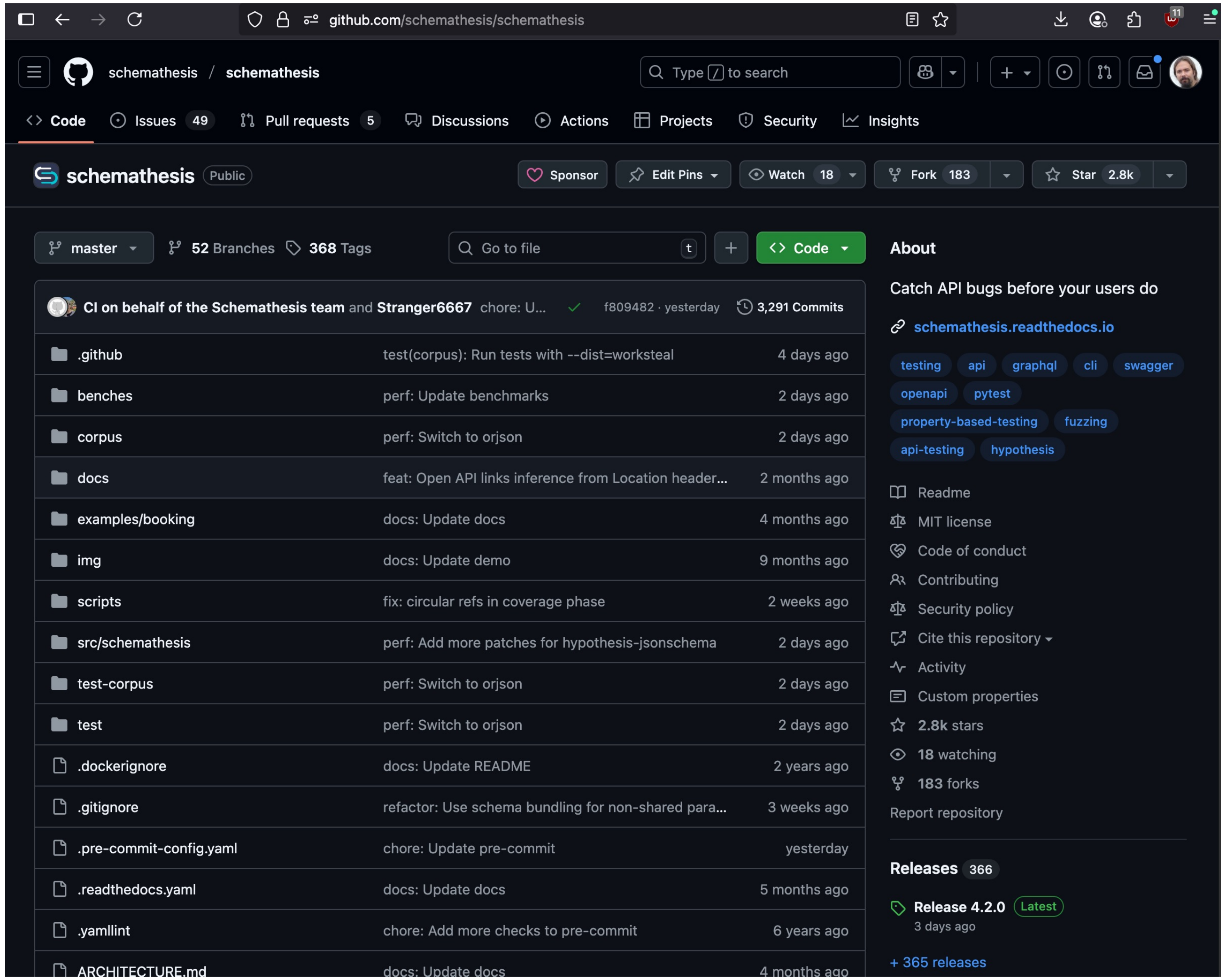
Releases

25 tags

Contributors 45

SchemaThesis

- Made by a Software Engineer: Dmitry Dygalo
- Open-source since 2019
- Written in Python
- Available via **pip**
  - eg, “pip install schemathesis”
- +2700⭐



CI on behalf of the Schemathesis team and Stranger6667 chore: U... f809482 · yesterday 3,291 Commits

.github	test(corpus): Run tests with --dist=worksteal	4 days ago
benches	perf: Update benchmarks	2 days ago
corpus	perf: Switch to orjson	2 days ago
docs	feat: Open API links inference from Location header...	2 months ago
examples/booking	docs: Update docs	4 months ago
img	docs: Update demo	9 months ago
scripts	fix: circular refs in coverage phase	2 weeks ago
src/schemathesis	perf: Add more patches for hypothesis-jsonschema	2 days ago
test-corpus	perf: Switch to orjson	2 days ago
test	perf: Switch to orjson	2 days ago
.dockerignore	docs: Update README	2 years ago
.gitignore	refactor: Use schema bundling for non-shared para...	3 weeks ago
.pre-commit-config.yaml	chore: Update pre-commit	yesterday
.readthedocs.yaml	docs: Update docs	5 months ago
.yamllint	chore: Add more checks to pre-commit	6 years ago
ARCHITECTURE.md	docs: Update docs	4 months ago

About

Catch API bugs before your users do

schemathesis.readthedocs.io

testing api graphql cli swagger openapi pytest property-based-testing fuzzing api-testing hypothesis

Readme
MIT license
Code of conduct
Contributing
Security policy
Cite this repository
Activity
Custom properties
2.8k stars
18 watching
183 forks
Report repository

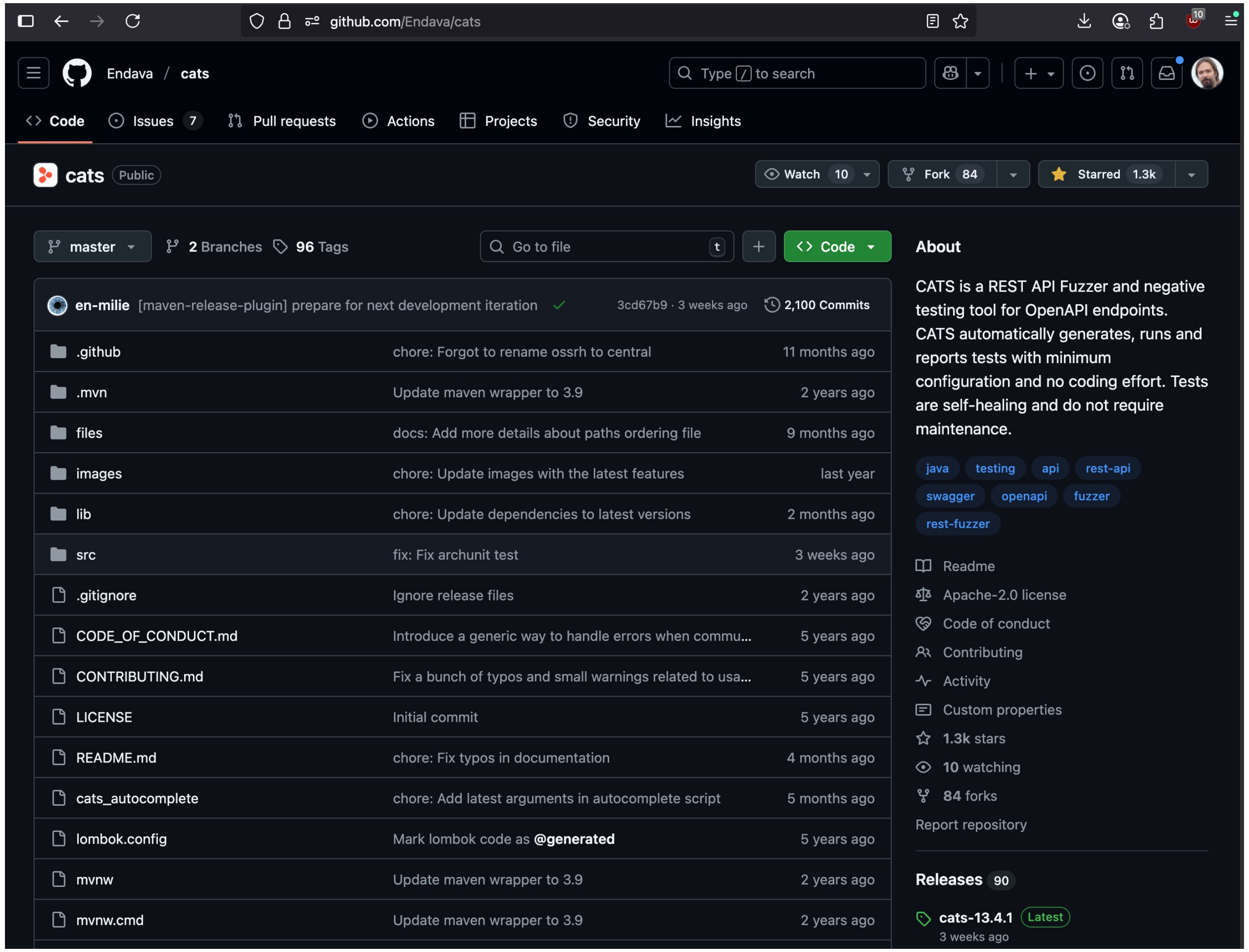
Releases 366

Release 4.2.0 Latest
3 days ago

+ 365 releases

CATS

- Made by Endava.com
- Open-source since 2020
- Written in Java
- Available via installers and **brew**
- +1300⭐


github.com/Endava/cats

Endava / cats

cats Public

master · 2 Branches · 96 Tags · Go to file · Code

en-milie [maven-release-plugin] prepare for next development iteration ✓ 3cd67b9 · 3 weeks ago · 2,100 Commits

.github	chore: Forgot to rename ossrh to central	11 months ago
.mvn	Update maven wrapper to 3.9	2 years ago
files	docs: Add more details about paths ordering file	9 months ago
images	chore: Update images with the latest features	last year
lib	chore: Update dependencies to latest versions	2 months ago
src	fix: Fix archunit test	3 weeks ago
.gitignore	Ignore release files	2 years ago
CODE_OF_CONDUCT.md	Introduce a generic way to handle errors when commu...	5 years ago
CONTRIBUTING.md	Fix a bunch of typos and small warnings related to usa...	5 years ago
LICENSE	Initial commit	5 years ago
README.md	chore: Fix typos in documentation	4 months ago
cats_autocomplete	chore: Add latest arguments in autocomplete script	5 months ago
lombok.config	Mark lombok code as **@generated**	5 years ago
mvnw	Update maven wrapper to 3.9	2 years ago
mvnw.cmd	Update maven wrapper to 3.9	2 years ago

About

CATS is a REST API Fuzzer and negative testing tool for OpenAPI endpoints. CATS automatically generates, runs and reports tests with minimum configuration and no coding effort. Tests are self-healing and do not require maintenance.

java · testing · api · rest-api · swagger · openapi · fuzzer · rest-fuzzer

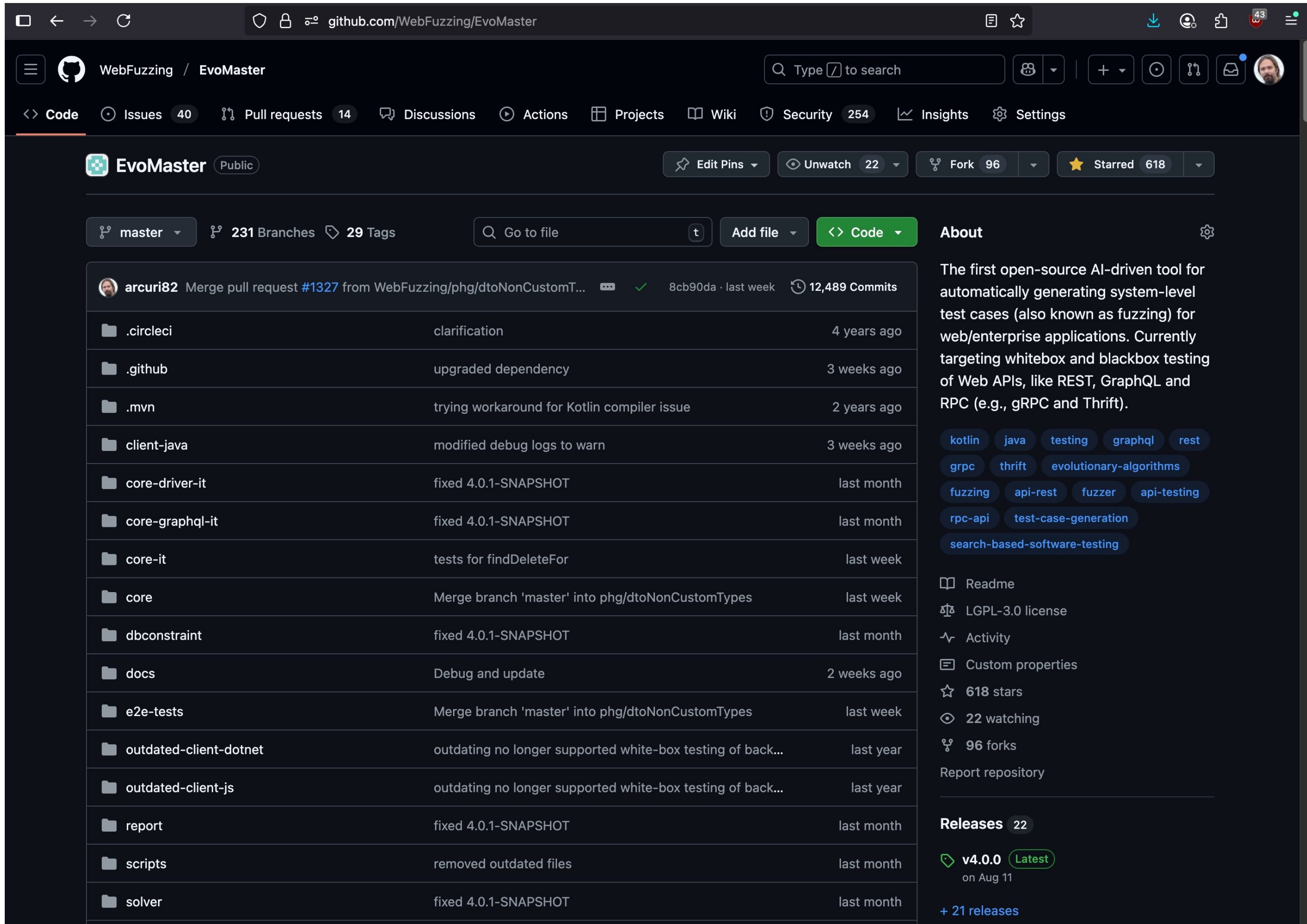
Readme
Apache-2.0 license
Code of conduct
Contributing
Activity
Custom properties
1.3k stars
10 watching
84 forks
Report repository

Releases 90

cats-13.4.1 Latest
3 weeks ago

EvoMaster

- Made by me (and a team of academics)
- Open-source since 2016
- Written in Kotlin
- Available via installers and **Docker**
- +600⭐



github.com/WebFuzzing/EvoMaster

WebFuzzing / EvoMaster

Code | Issues 40 | Pull requests 14 | Discussions | Actions | Projects | Wiki | Security 254 | Insights | Settings

EvoMaster Public

Edit Pins | Unwatch 22 | Fork 96 | Starred 618

master | 231 Branches | 29 Tags | Go to file | Add file | Code

arcuri82 Merge pull request #1327 from WebFuzzing/phg/dtoNonCustomT... 8cb90da · last week 12,489 Commits

.circleci	clarification	4 years ago
.github	upgraded dependency	3 weeks ago
.mvn	trying workaround for Kotlin compiler issue	2 years ago
client-java	modified debug logs to warn	3 weeks ago
core-driver-it	fixed 4.0.1-SNAPSHOT	last month
core-graphql-it	fixed 4.0.1-SNAPSHOT	last month
core-it	tests for findDeleteFor	last week
core	Merge branch 'master' into phg/dtoNonCustomTypes	last week
dbconstraint	fixed 4.0.1-SNAPSHOT	last month
docs	Debug and update	2 weeks ago
e2e-tests	Merge branch 'master' into phg/dtoNonCustomTypes	last week
outdated-client-dotnet	outdating no longer supported white-box testing of back...	last year
outdated-client-js	outdating no longer supported white-box testing of back...	last year
report	fixed 4.0.1-SNAPSHOT	last month
scripts	removed outdated files	last month
solver	fixed 4.0.1-SNAPSHOT	last month

About

The first open-source AI-driven tool for automatically generating system-level test cases (also known as fuzzing) for web/enterprise applications. Currently targeting whitebox and blackbox testing of Web APIs, like REST, GraphQL and RPC (e.g., gRPC and Thrift).

kotlin java testing graphql rest grpc thrift evolutionary-algorithms fuzzing api-rest fuzzer api-testing rpc-api test-case-generation search-based-software-testing

Readme
LGPL-3.0 license
Activity
Custom properties
618 stars
22 watching
96 forks
Report repository

Releases 22

v4.0.0 Latest
on Aug 11

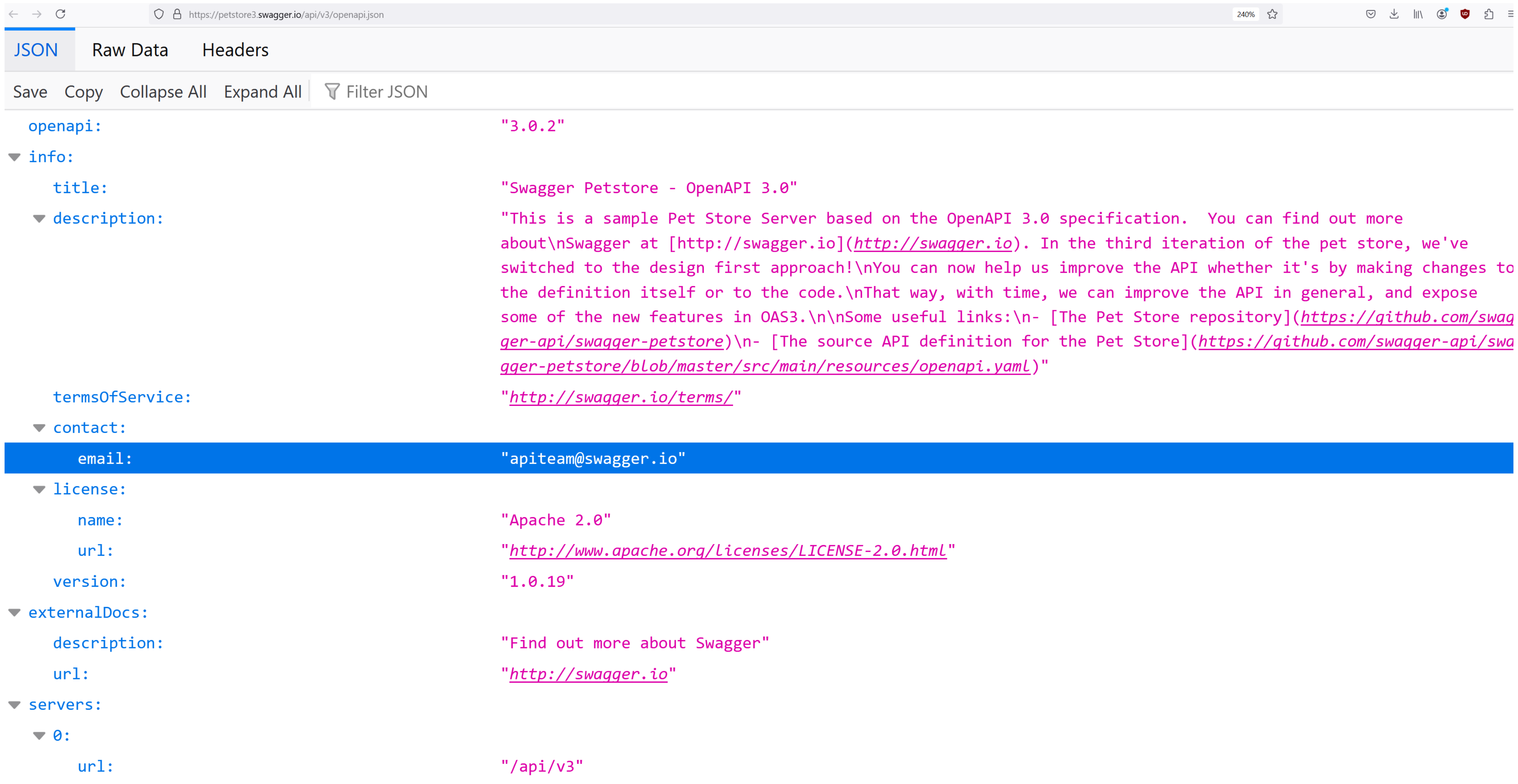
+ 21 releases

Input: OpenAPI/Swagger Schema

- Need to know what endpoints are available, and their parameters
- Schema defining the APIs
- OpenAPI is the most popular one
- Defined as JSON file, or YAML

Example: PetStore

- Online schema at https://petstore3.swagger.io/api/v3/openapi.json

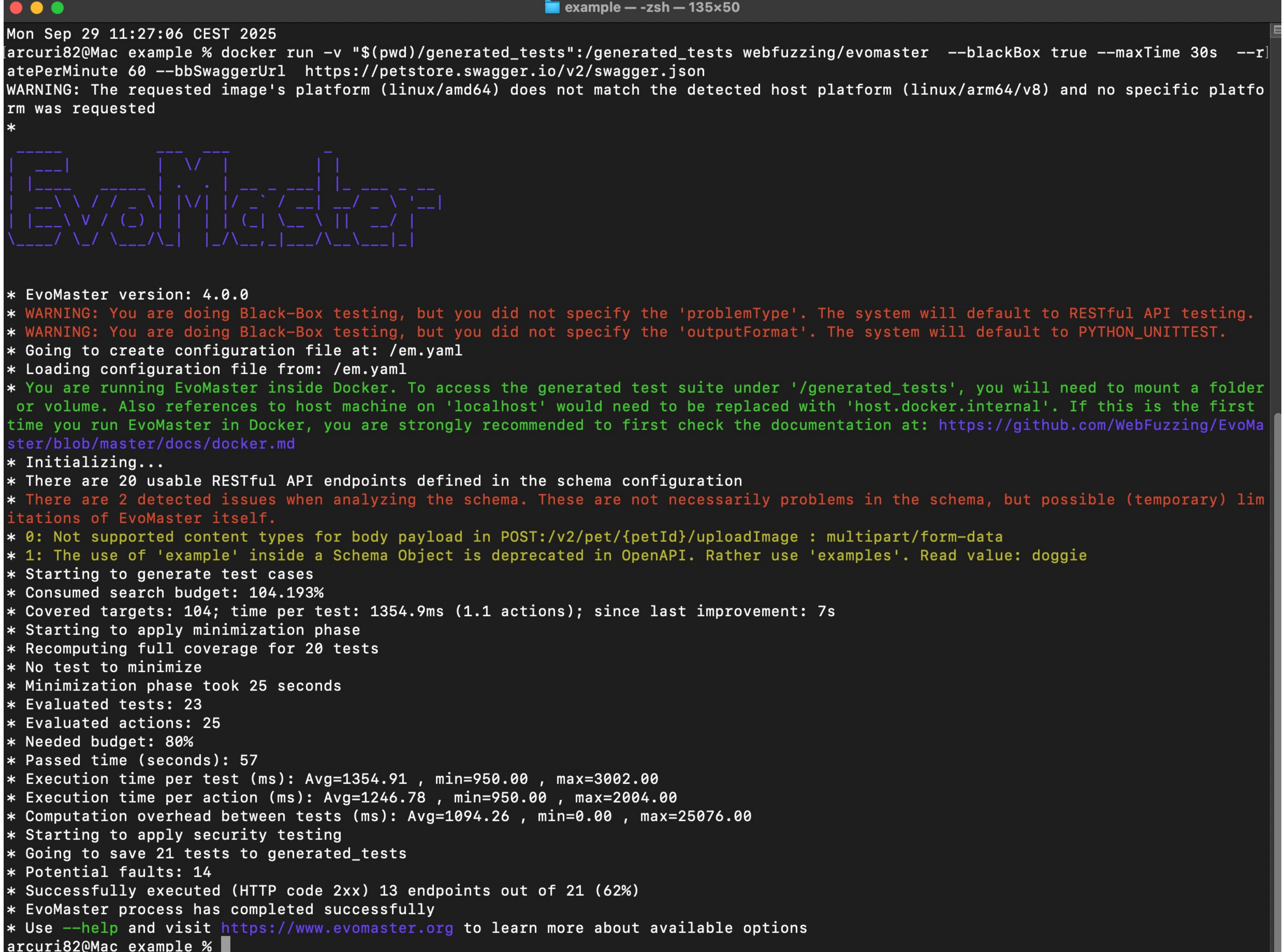


```
openapi:                  "3.0.2"
▼ info:
    title:                "Swagger Petstore - OpenAPI 3.0"
  ▼ description:          "This is a sample Pet Store Server based on the OpenAPI 3.0 specification.  You can find out more
                          about\nSwagger at [http://swagger.io](http://swagger.io). In the third iteration of the pet store, we've
                          switched to the design first approach!\nYou can now help us improve the API whether it's by making changes to
                          the definition itself or to the code.\nThat way, with time, we can improve the API in general, and expose
                          some of the new features in OAS3.\n\nSome useful links:\n- [The Pet Store repository](https://github.com/swag
                          ger-api/swagger-petstore)\n- [The source API definition for the Pet Store](https://github.com/swagger-api/swa
                          gger-petstore/blob/master/src/main/resources/openapi.yaml)"
    termsOfService:       "http://swagger.io/terms/"
  ▼ contact:
      email:              "apiteam@swagger.io"
  ▼ license:
      name:               "Apache 2.0"
      url:                "http://www.apache.org/licenses/LICENSE-2.0.html"
    version:              "1.0.19"
▼ externalDocs:
    description:          "Find out more about Swagger"
    url:                  "http://swagger.io"
▼ servers:
  ▼ 0:
      url:                "/api/v3"
```

What Can Expect?

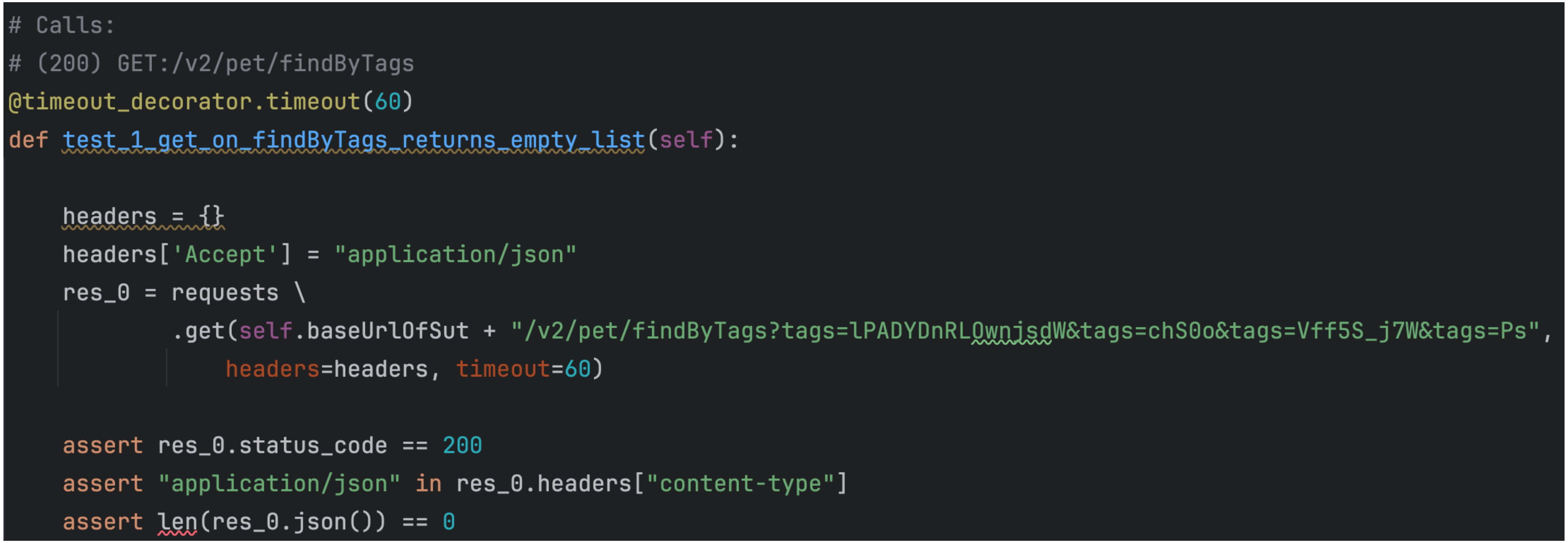
- All these tools will analyze the schema
- Send requests with many different strategies
  - there is lot of research in academia on this
- Check if any error in the API can be identified
- Output executable test cases
  - in different formats, eg Python, Java, Kotlin and JavaScript

```
docker run
    -v "$(pwd)/generated_tests":/generated_tests
    webfuzzing/evomaster
    --blackBox true
    --maxTime 30s
    --ratePerMinute 60
    --bbSwaggerUrl  https://petstore.swagger.io/v2/swagger.json
```



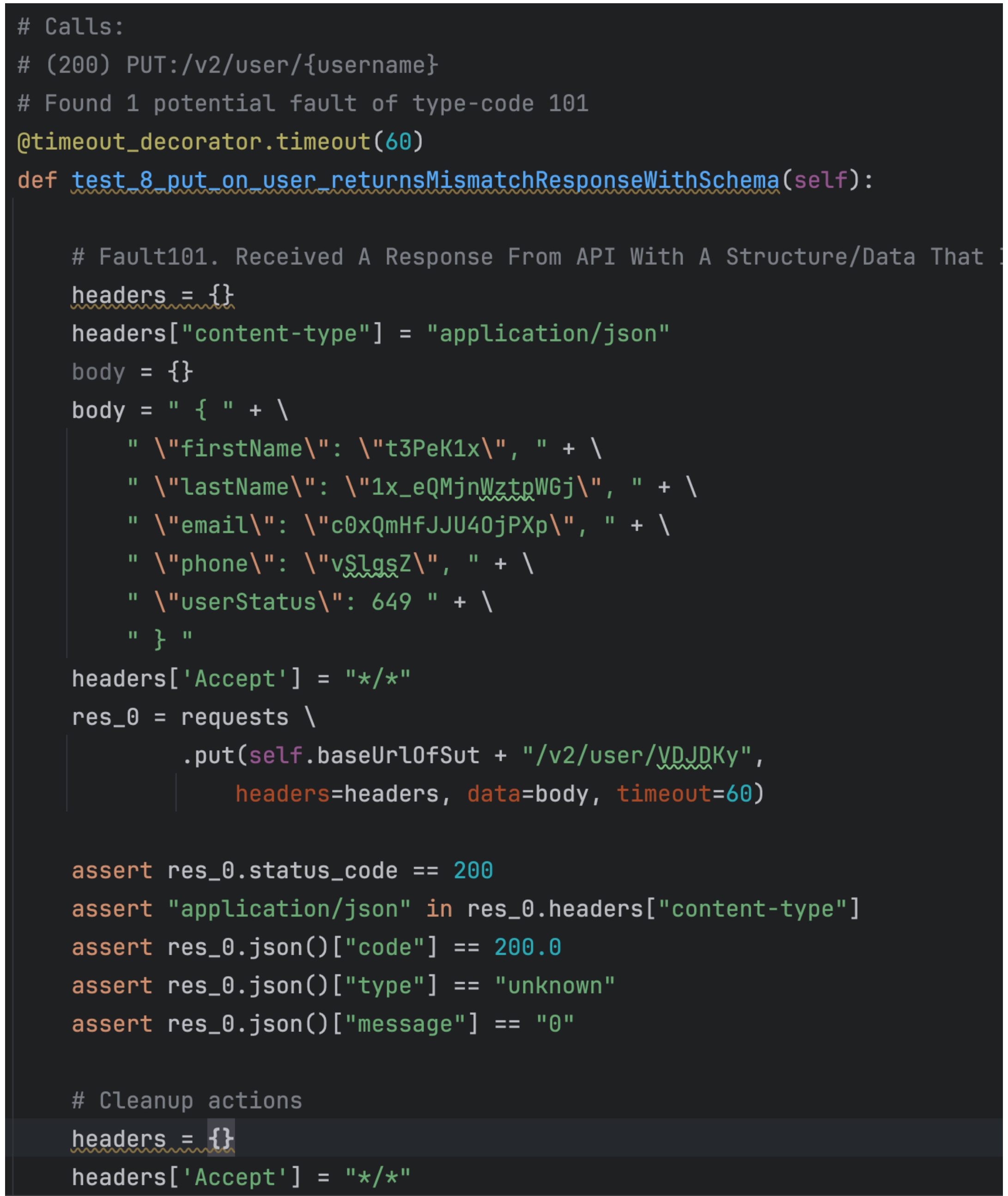
```
Mon Sep 29 11:27:06 CEST 2025
arcuri82@Mac example % docker run -v "$(pwd)/generated_tests":/generated_tests webfuzzing/evomaster  --blackBox true --maxTime 30s  --r
atePerMinute 60 --bbSwaggerUrl  https://petstore.swagger.io/v2/swagger.json
WARNING: The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platfo
rm was requested
*
 _____          ___  ___         _
|  ___|         |  \/  |        | |
| |__   _____  | .  . | __ _ ___| |_ ___ _ __
|  __\ \ / / _ \| |\/| |/ _` / __| __/ _ \ '__|
| |___\ V / (_) | |  | | (_| \__ \ ||  __/ |
\____/ \_/ \___/\_|  |_/\__,_|___/\__\___|_|


* EvoMaster version: 4.0.0
* WARNING: You are doing Black-Box testing, but you did not specify the 'problemType'. The system will default to RESTful API testing.
* WARNING: You are doing Black-Box testing, but you did not specify the 'outputFormat'. The system will default to PYTHON_UNITTEST.
* Going to create configuration file at: /em.yaml
* Loading configuration file from: /em.yaml
* You are running EvoMaster inside Docker. To access the generated test suite under '/generated_tests', you will need to mount a folder
 or volume. Also references to host machine on 'localhost' would need to be replaced with 'host.docker.internal'. If this is the first
time you run EvoMaster in Docker, you are strongly recommended to first check the documentation at: https://github.com/WebFuzzing/EvoMa
ster/blob/master/docs/docker.md
* Initializing...
* There are 20 usable RESTful API endpoints defined in the schema configuration
* There are 2 detected issues when analyzing the schema. These are not necessarily problems in the schema, but possible (temporary) lim
itations of EvoMaster itself.
* 0: Not supported content types for body payload in POST:/v2/pet/{petId}/uploadImage : multipart/form-data
* 1: The use of 'example' inside a Schema Object is deprecated in OpenAPI. Rather use 'examples'. Read value: doggie
* Starting to generate test cases
* Consumed search budget: 104.193%
* Covered targets: 104; time per test: 1354.9ms (1.1 actions); since last improvement: 7s
* Starting to apply minimization phase
* Recomputing full coverage for 20 tests
* No test to minimize
* Minimization phase took 25 seconds
* Evaluated tests: 23
* Evaluated actions: 25
* Needed budget: 80%
* Passed time (seconds): 57
* Execution time per test (ms): Avg=1354.91 , min=950.00 , max=3002.00
* Execution time per action (ms): Avg=1246.78 , min=950.00 , max=2004.00
* Computation overhead between tests (ms): Avg=1094.26 , min=0.00 , max=25076.00
* Starting to apply security testing
* Going to save 21 tests to generated_tests
* Potential faults: 14
* Successfully executed (HTTP code 2xx) 13 endpoints out of 21 (62%)
* EvoMaster process has completed successfully
* Use --help and visit https://www.evomaster.org to learn more about available options
arcuri82@Mac example %
```

Success Calls: Random but Valid Data

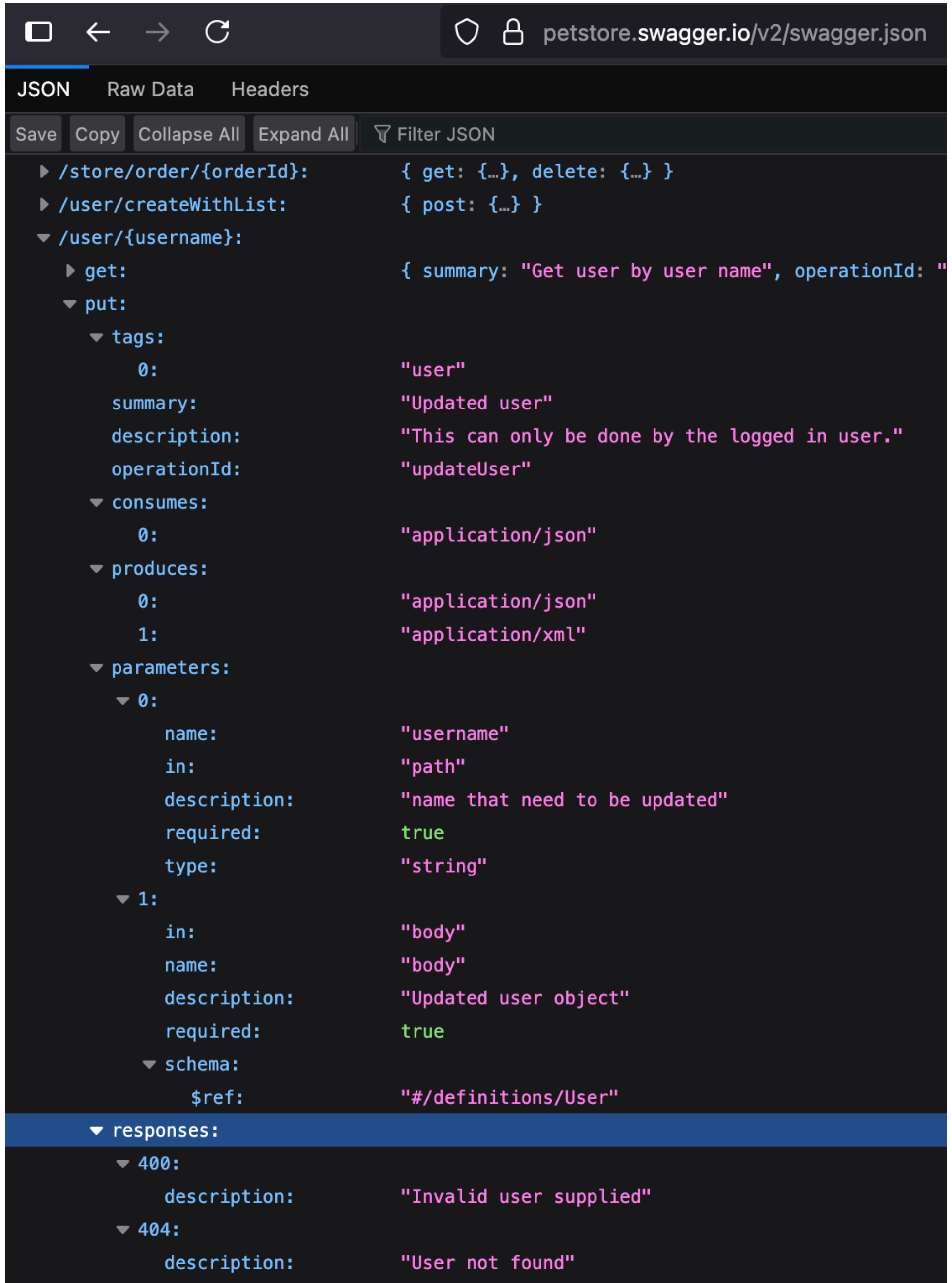


```python
# Calls:
# (200) GET:/v2/pet/findByTags
@timeout_decorator.timeout(60)
def test_1_get_on_findByTags_returns_empty_list(self):

    headers = {}
    headers['Accept'] = "application/json"
    res_0 = requests \
            .get(self.baseUrlOfSut + "/v2/pet/findByTags?tags=lPADYDnRLQwnjsdW&tags=chS0o&tags=Vff5S_j7W&tags=Ps",
                headers=headers, timeout=60)

    assert res_0.status_code == 200
    assert "application/json" in res_0.headers["content-type"]
    assert len(res_0.json()) == 0
```

Schema Mismatch (eg undeclared 200)



```
# Calls:
# (200) PUT:/v2/user/{username}
# Found 1 potential fault of type-code 101
@timeout_decorator.timeout(60)
def test_8_put_on_user_returnsMismatchResponseWithSchema(self):

    # Fault101. Received A Response From API With A Structure/Data That
    headers = {}
    headers["content-type"] = "application/json"
    body = {}
    body = " { " + \
        " \"firstName\": \"t3PeK1x\", " + \
        " \"lastName\": \"1x_eQMjnWztpWGj\", " + \
        " \"email\": \"c0xQmHfJJU40jPXp\", " + \
        " \"phone\": \"vSlgsZ\", " + \
        " \"userStatus\": 649 " + \
        " } "
    headers['Accept'] = "*/*"
    res_0 = requests \
            .put(self.baseUrlOfSut + "/v2/user/VDJDKy",
                headers=headers, data=body, timeout=60)

    assert res_0.status_code == 200
    assert "application/json" in res_0.headers["content-type"]
    assert res_0.json()["code"] == 200.0
    assert res_0.json()["type"] == "unknown"
    assert res_0.json()["message"] == "0"

    # Cleanup actions
    headers = {}
    headers['Accept'] = "*/*"
```



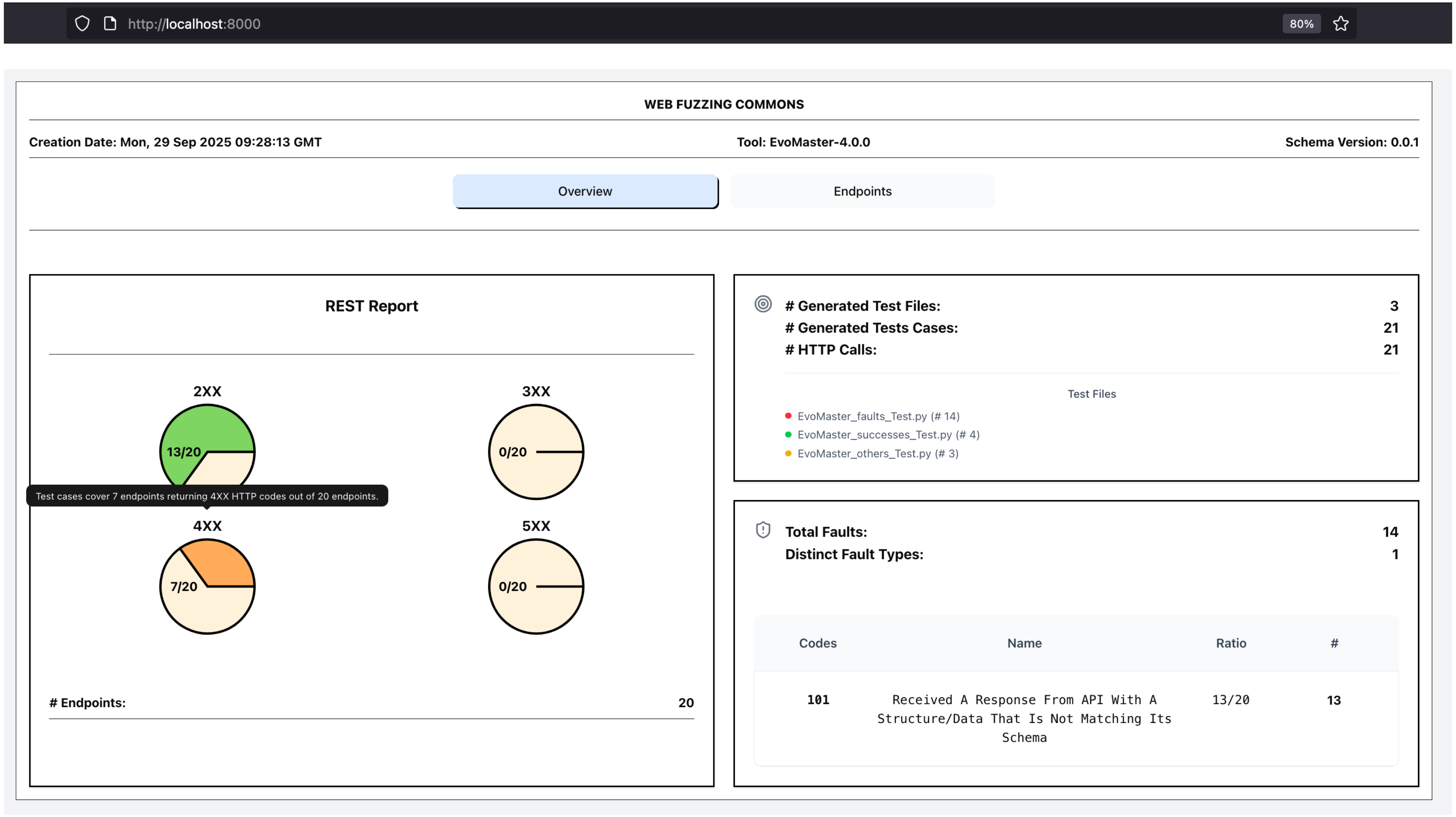
petstore.swagger.io/v2/swagger.json

JSON   Raw Data   Headers

Save   Copy   Collapse All   Expand All   Filter JSON

```
▸ /store/order/{orderId}:          { get: {…}, delete: {…} }
▸ /user/createWithList:            { post: {…} }
▾ /user/{username}:
  ▸ get:                           { summary: "Get user by user name", operationId: "
  ▾ put:
    ▾ tags:
        0:                         "user"
      summary:                     "Updated user"
      description:                 "This can only be done by the logged in user."
      operationId:                 "updateUser"
    ▾ consumes:
        0:                         "application/json"
    ▾ produces:
        0:                         "application/json"
        1:                         "application/xml"
    ▾ parameters:
      ▾ 0:
          name:                    "username"
          in:                      "path"
          description:             "name that need to be updated"
          required:                true
          type:                    "string"
      ▾ 1:
          in:                      "body"
          name:                    "body"
          description:             "Updated user object"
          required:                true
        ▾ schema:
            $ref:                  "#/definitions/User"
    ▾ responses:
      ▾ 400:
          description:             "Invalid user supplied"
      ▾ 404:
          description:             "User not found"
```

Interactive Test Reports



WEB FUZZING COMMONS

Creation Date: Mon, 29 Sep 2025 09:28:13 GMT

Tool: EvoMaster-4.0.0

Schema Version: 0.0.1

REST Report

Test cases cover 7 endpoints returning 4XX HTTP codes out of 20 endpoints.

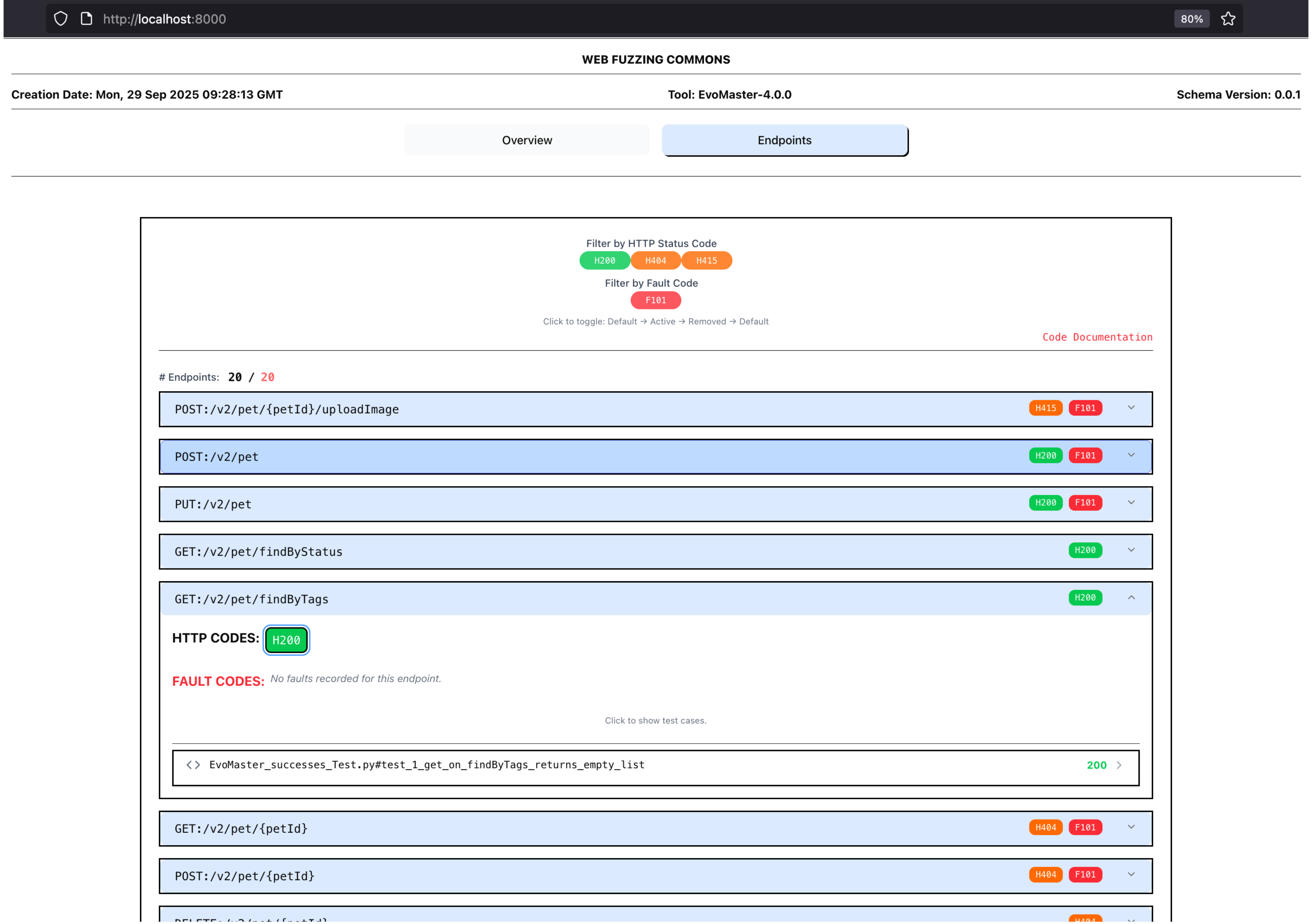
# Endpoints:	20
# Generated Test Files:	3
# Generated Tests Cases:	21
# HTTP Calls:	21

Test Files

- ● EvoMaster_faults_Test.py (# 14)
- ● EvoMaster_successes_Test.py (# 4)
- ● EvoMaster_others_Test.py (# 3)

Total Faults:	14
Distinct Fault Types:	1

Codes	Name	Ratio	#
101	Received A Response From API With A Structure/Data That Is Not Matching Its Schema	13/20	13



http://localhost:8000

WEB FUZZING COMMONS

Creation Date: Mon, 29 Sep 2025 09:28:13 GMT	Tool: EvoMaster-4.0.0	Schema Version: 0.0.1

Overview	Endpoints

Filter by HTTP Status Code
H200 H404 H415
Filter by Fault Code
F101
Click to toggle: Default → Active → Removed → Default

Code Documentation

Endpoints: 20 / 20

POST:/v2/pet/{petId}/uploadImage H415 F101

POST:/v2/pet H200 F101

PUT:/v2/pet H200 F101

GET:/v2/pet/findByStatus H200

GET:/v2/pet/findByTags H200

HTTP CODES: H200

FAULT CODES: *No faults recorded for this endpoint.*

Click to show test cases.

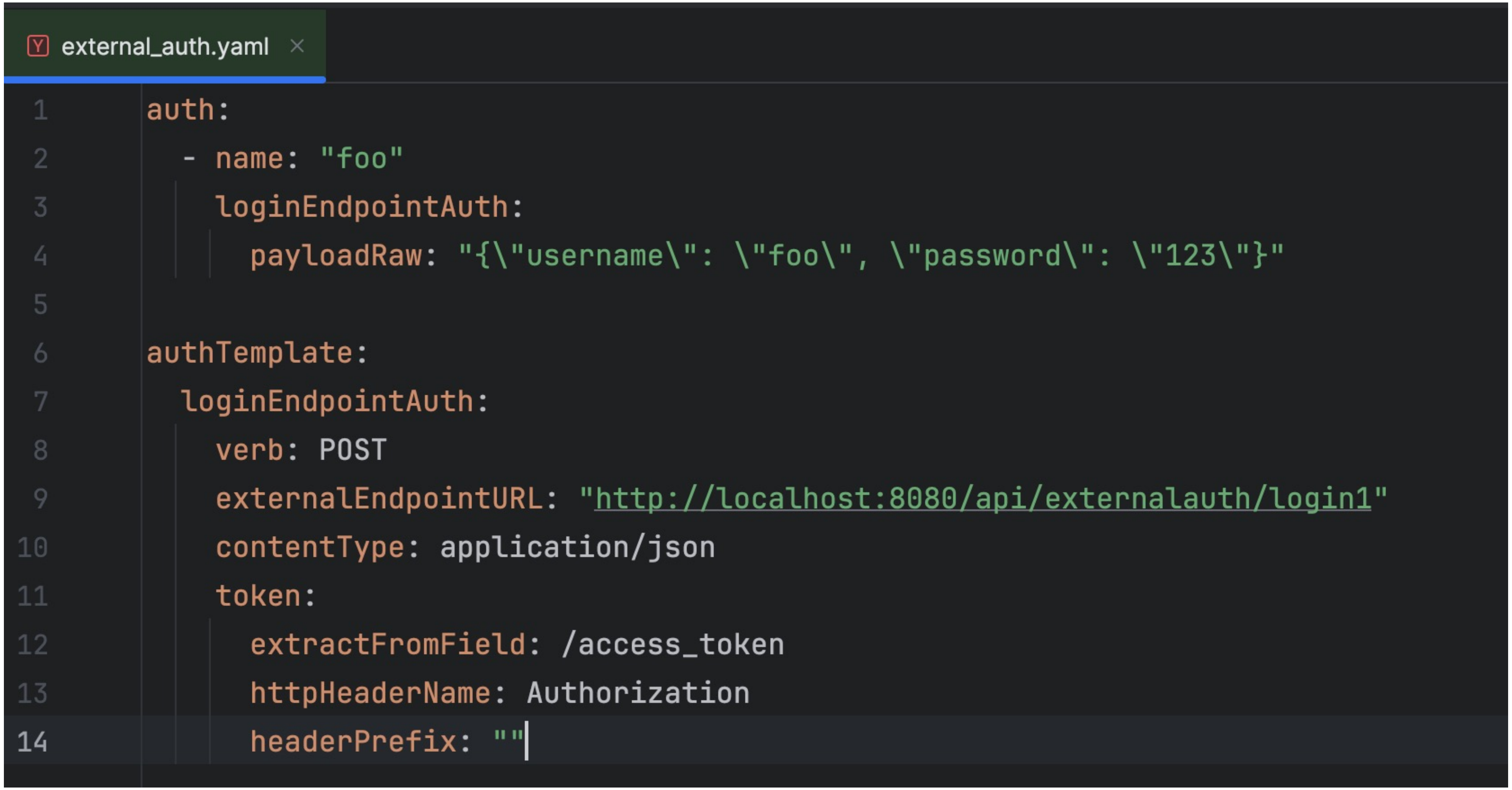
<> EvoMaster_successes_Test.py#test_1_get_on_findByTags_returns_empty_list 200

GET:/v2/pet/{petId} H404 F101

POST:/v2/pet/{petId} H404 F101

What about authentication?

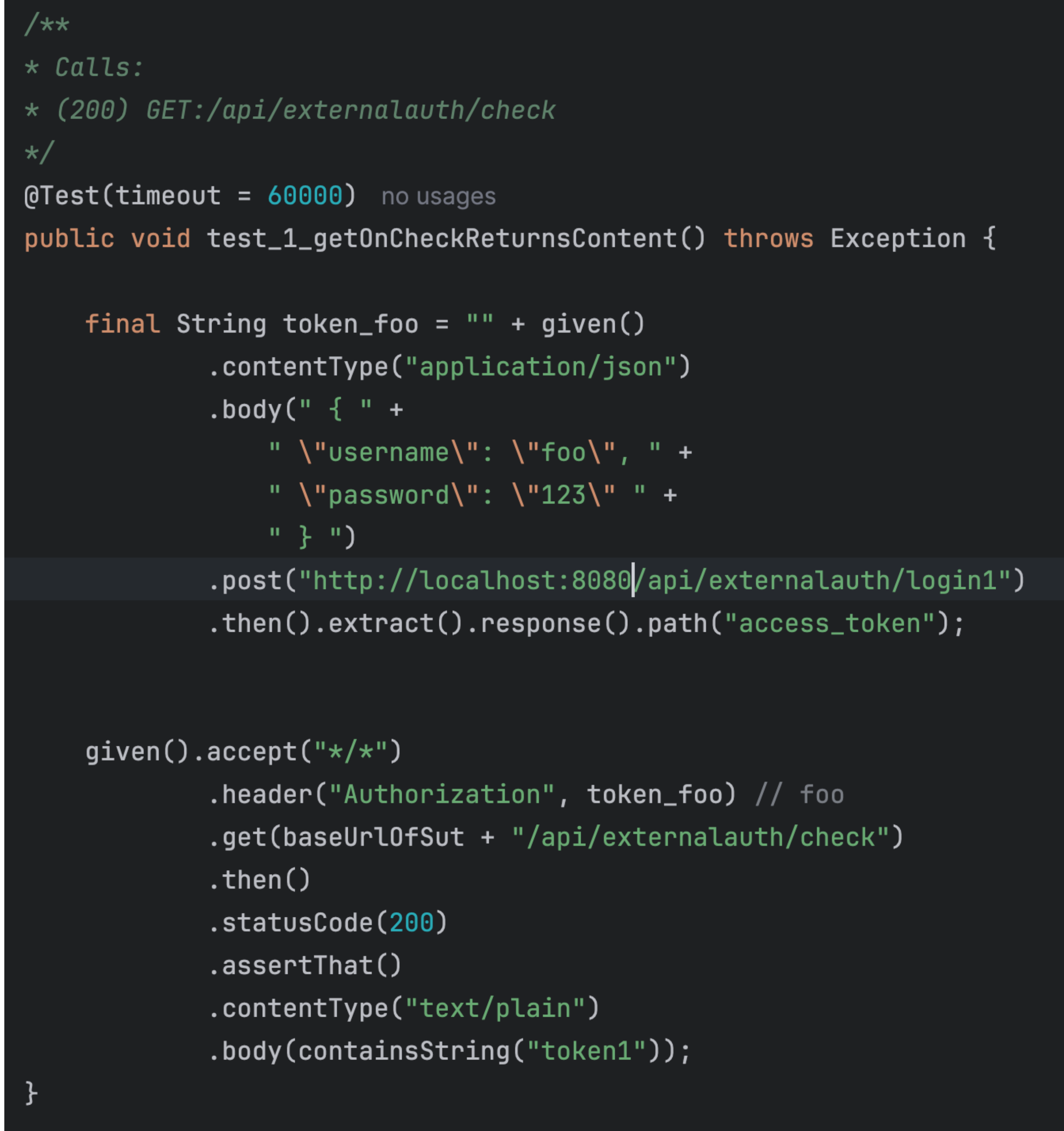
Configuration files in YAML, using WFC format



external_auth.yaml

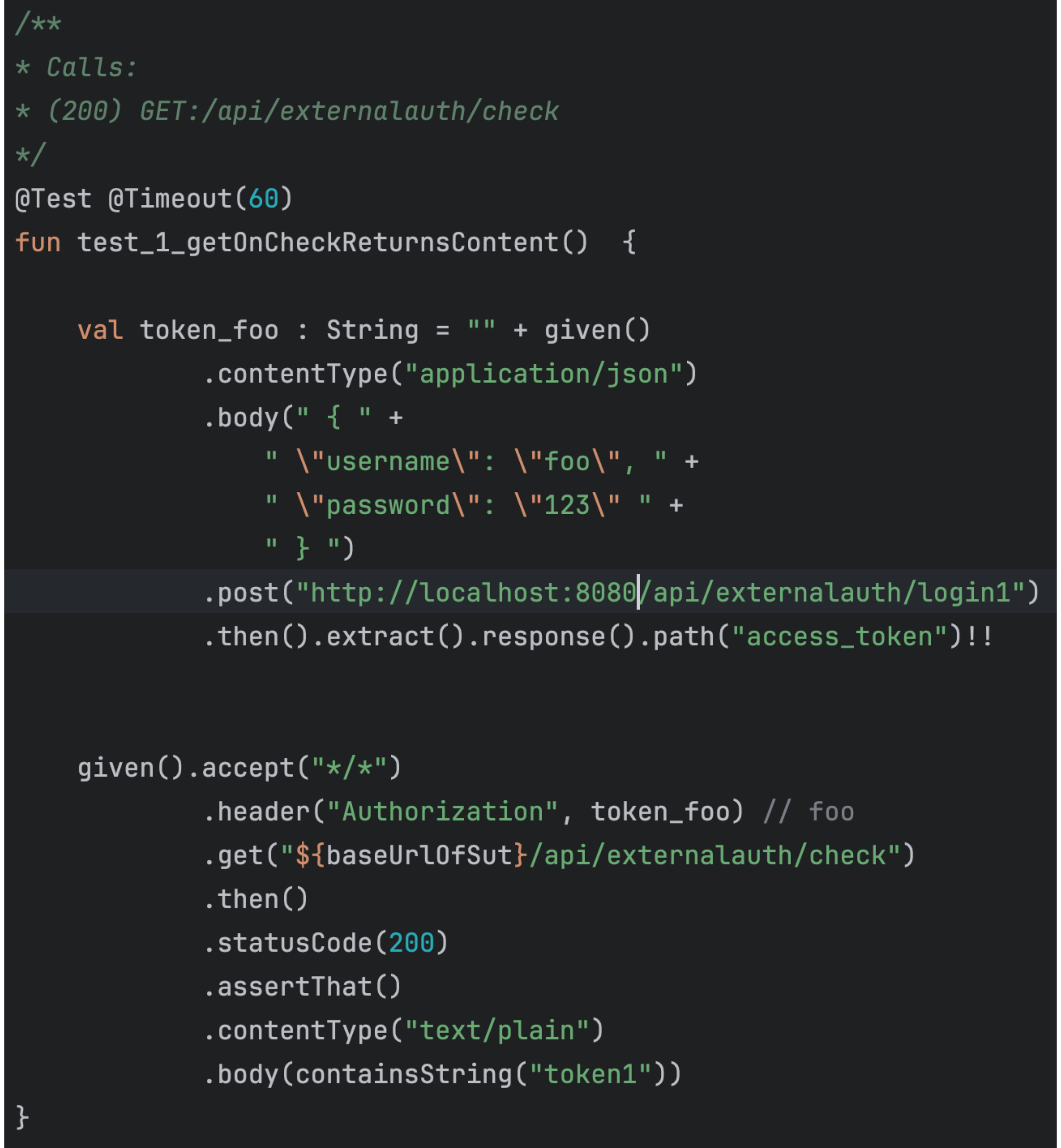
```yaml
auth:
  - name: "foo"
    loginEndpointAuth:
      payloadRaw: "{\"username\": \"foo\", \"password\": \"123\"}"

authTemplate:
  loginEndpointAuth:
    verb: POST
    externalEndpointURL: "http://localhost:8080/api/externalauth/login1"
    contentType: application/json
    token:
      extractFromField: /access_token
      httpHeaderName: Authorization
      headerPrefix: ""
```

Automatically fetch and extract tokens in generated tests (eg in Java)



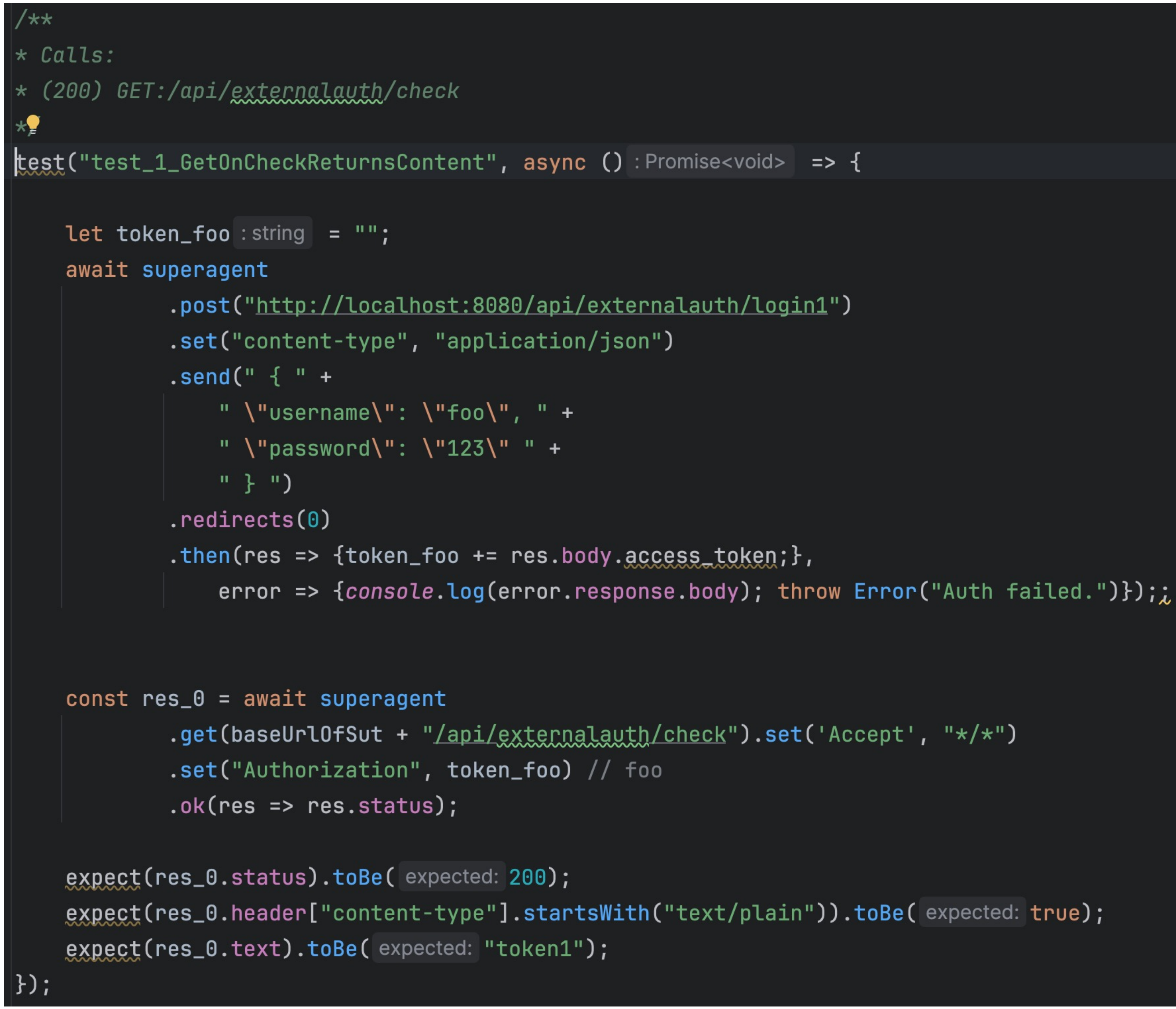
```
/**
* Calls:
* (200) GET:/api/externalauth/check
*/
@Test(timeout = 60000)  no usages
public void test_1_getOnCheckReturnsContent() throws Exception {

    final String token_foo = "" + given()
            .contentType("application/json")
            .body(" { " +
                " \"username\": \"foo\", " +
                " \"password\": \"123\" " +
                " } ")
            .post("http://localhost:8080/api/externalauth/login1")
            .then().extract().response().path("access_token");


    given().accept("*/*")
            .header("Authorization", token_foo) // foo
            .get(baseUrlOfSut + "/api/externalauth/check")
            .then()
            .statusCode(200)
            .assertThat()
            .contentType("text/plain")
            .body(containsString("token1"));
}
```

Or Kotlin…



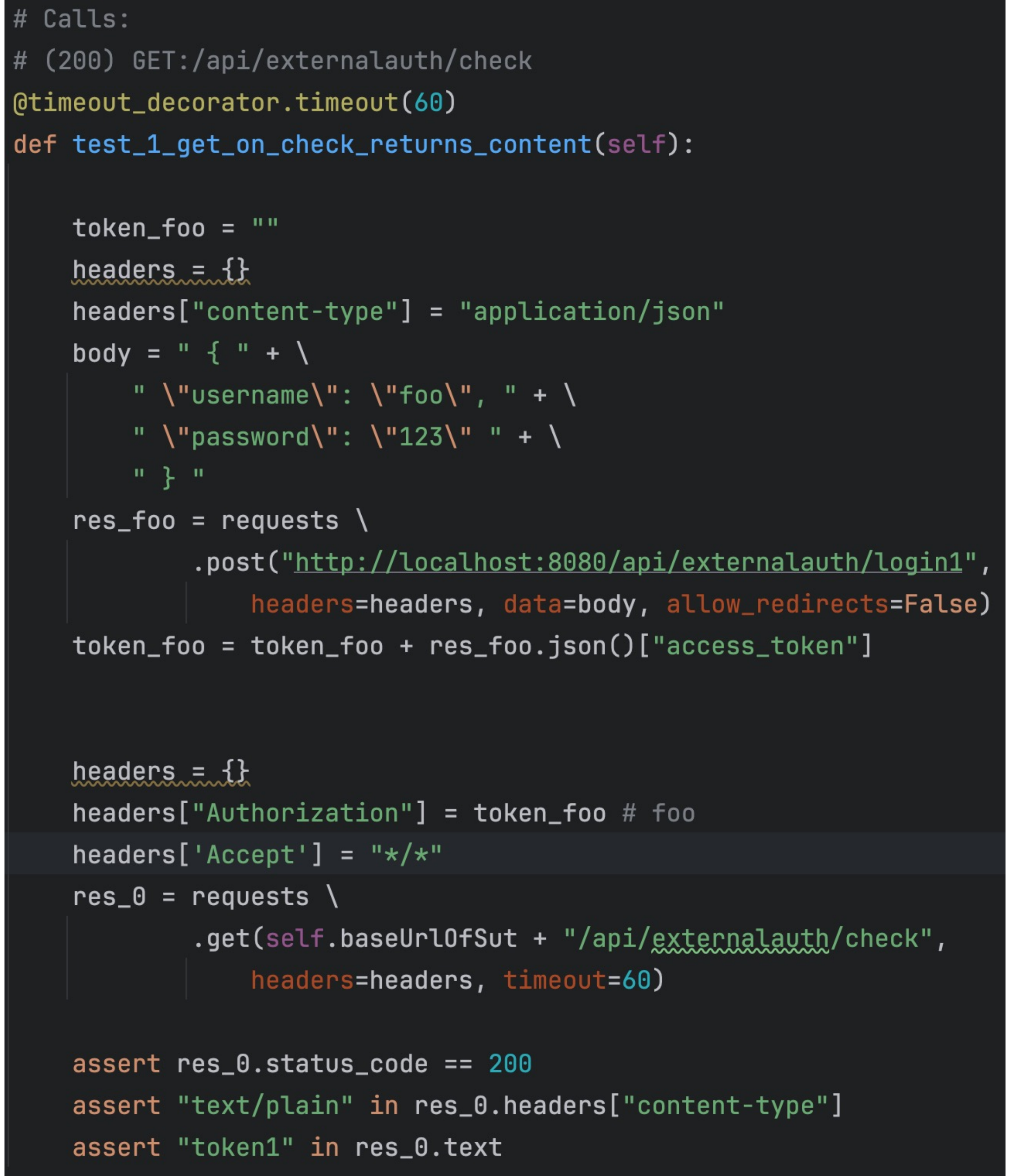
```
/**
* Calls:
* (200) GET:/api/externalauth/check
*/
@Test @Timeout(60)
fun test_1_getOnCheckReturnsContent()  {

    val token_foo : String = "" + given()
            .contentType("application/json")
            .body(" { " +
                " \"username\": \"foo\", " +
                " \"password\": \"123\" " +
                " } ")
            .post("http://localhost:8080/api/externalauth/login1")
            .then().extract().response().path("access_token")!!


    given().accept("*/*")
            .header("Authorization", token_foo) // foo
            .get("${baseUrlOfSut}/api/externalauth/check")
            .then()
            .statusCode(200)
            .assertThat()
            .contentType("text/plain")
            .body(containsString("token1"))
}
```

Or JavaScript...



```
/**
* Calls:
* (200) GET:/api/externalauth/check
*
test("test_1_GetOnCheckReturnsContent", async () : Promise<void>  => {

    let token_foo : string  = "";
    await superagent
            .post("http://localhost:8080/api/externalauth/login1")
            .set("content-type", "application/json")
            .send(" { " +
                " \"username\": \"foo\", " +
                " \"password\": \"123\" " +
                " } ")
            .redirects(0)
            .then(res => {token_foo += res.body.access_token;},
                error => {console.log(error.response.body); throw Error("Auth failed.")});;


    const res_0 = await superagent
            .get(baseUrlOfSut + "/api/externalauth/check").set('Accept', "*/*")
            .set("Authorization", token_foo) // foo
            .ok(res => res.status);

    expect(res_0.status).toBe( expected: 200);
    expect(res_0.header["content-type"].startsWith("text/plain")).toBe( expected: true);
    expect(res_0.text).toBe( expected: "token1");
});
```

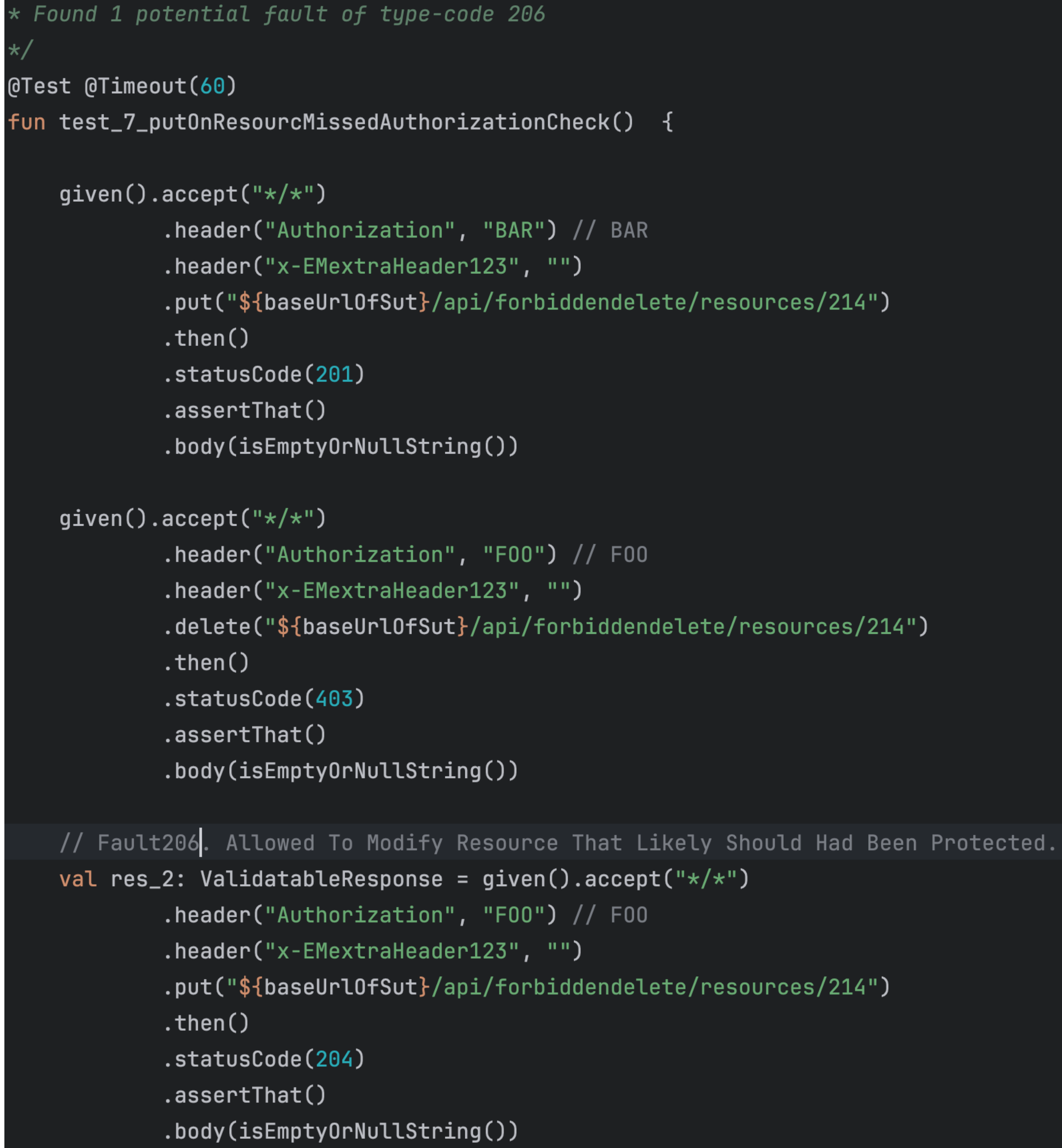
Or Python…



```python
# Calls:
# (200) GET:/api/externalauth/check
@timeout_decorator.timeout(60)
def test_1_get_on_check_returns_content(self):

    token_foo = ""
    headers = {}
    headers["content-type"] = "application/json"
    body = " { " + \
        " \"username\": \"foo\", " + \
        " \"password\": \"123\" " + \
        " } "
    res_foo = requests \
            .post("http://localhost:8080/api/externalauth/login1",
                headers=headers, data=body, allow_redirects=False)
    token_foo = token_foo + res_foo.json()["access_token"]


    headers = {}
    headers["Authorization"] = token_foo # foo
    headers['Accept'] = "*/*"
    res_0 = requests \
            .get(self.baseUrlOfSut + "/api/externalauth/check",
                headers=headers, timeout=60)

    assert res_0.status_code == 200
    assert "text/plain" in res_0.headers["content-type"]
    assert "token1" in res_0.text
```

Access Policies Validation Example

- Forbidden to delete a resouce of another user (403)…
- … but allowed (204) to modify it with a PUT???
- Excepted behavior?
- Or misconfigured authorization in the PUT???



```
* Found 1 potential fault of type-code 206
*/
@Test @Timeout(60)
fun test_7_putOnResourcMissedAuthorizationCheck()  {

    given().accept("*/*")
            .header("Authorization", "BAR") // BAR
            .header("x-EMextraHeader123", "")
            .put("${baseUrlOfSut}/api/forbiddendelete/resources/214")
            .then()
            .statusCode(201)
            .assertThat()
            .body(isEmptyOrNullString())

    given().accept("*/*")
            .header("Authorization", "FOO") // FOO
            .header("x-EMextraHeader123", "")
            .delete("${baseUrlOfSut}/api/forbiddendelete/resources/214")
            .then()
            .statusCode(403)
            .assertThat()
            .body(isEmptyOrNullString())

    // Fault206. Allowed To Modify Resource That Likely Should Had Been Protected.
    val res_2: ValidatableResponse = given().accept("*/*")
            .header("Authorization", "FOO") // FOO
            .header("x-EMextraHeader123", "")
            .put("${baseUrlOfSut}/api/forbiddendelete/resources/214")
            .then()
            .statusCode(204)
            .assertThat()
            .body(isEmptyOrNullString())
```

What about some more advanced cases?

White-box testing on JVM using **Evolutionary Computation** (eg Genetic Algorithms)

Dealing With SQL Databases

- Bytecode instrumentation to intercept all JDBC calls
- Find all SQL SELECT queries that return no data
  - eg due to WHERE clauses that are not satisfied
- Insert data directly into DB as part of the test case
  - Not always possible to create data with REST endpoints (eg POST/PUT)
  - using a JDBC connection
  - need to analyze DB’s schema
- *Goal*: insert data such that SELECT are not empty
- *Challenges*: WHERE clauses might have complex constraints. Need search
- *Why?* Can have impact on code execution flow

Java Example Using Spring

```java
@RequestMapping(
        path = "/{x}/{y}",
        method = RequestMethod.GET,
        produces = MediaType.APPLICATION_JSON
)
public ResponseEntity get(@PathVariable("x") int x, @PathVariable("y") int y) {

    List<DbDirectIntEntity> list = repository.findByXIsAndYIs(x, y);
    if (list.isEmpty()) {
        return ResponseEntity.status(400).build();
    } else {
        return ResponseEntity.status(200).build();
    }
}
```

Generated Test

```
@Test @Timeout(60)
fun test_1() {
   val insertions = sql().insertInto("DB_DIRECT_INT_ENTITY", 14L)
         .d("ID", "-65536")
         .d("X", "-67108182")
         .d("Y", "0")
      .dtos()
   val insertionsresult = controller.execInsertionsIntoDatabase(insertions)

   given().accept("*/*")
         .get("${baseUrlOfSut}/api/db/directint/-67108182/0")
         .then()
         .statusCode(200)
         .assertThat()
         .body(isEmptyOrNullString())

}
```

Taint Analysis

- Inputs can have constraint checks
  - eg, strings matching a regex, numbers in a certain range and strings representing dates
- Constraints might be in code and NOT in the OpenAPI schema
- Can evolve inputs till satisfy constraints… eg using SBST heuristics
- … but what if inputs are not modified and used as they are? Can we do better?

Java Example Using Spring

```java
@GetMapping(
        path = "/{date:\\d{4}-\\d{1,2}-\\d{1,2}}/{number}/{setting}",
        produces = MediaType.APPLICATION_JSON_VALUE)
public String getSeparated(
        @PathVariable("date") String date,
        @PathVariable("number") String number,
        @PathVariable("setting") String setting
){

    LocalDate d = LocalDate.parse(date);
    int n = Integer.parseInt(number);
    List<String> list = Arrays.asList("Foo", "Bar");

    if(d.getYear() == 2019 && n == 42 && list.contains(setting)){
        return "OK";
    }

    return "ERROR";
}
```

Solution

- Using bytecode instrumentation, check all JDK API usages
- Checking if input from HTTP is used without modification in a JDK call
- If yes, tell the search how input should be evolved
  - eg strings only representing valid dates, like for *LocalDate.parse(date)*
  - eg strings evolved always matching a particular regex
- Still need search to evolve the inputs
  - eg to handle constraints like *d.getYear() == 2019*
- Can dramatically boost the search efforts

Generated Test

```
@Test @Timeout(60)
fun test_4()  {

    given().accept("application/json")
            .get("${baseUrlOfSut}/api/testability/2019-12-10/42/Bar")
            .then()
            .statusCode(200)
            .assertThat()
            .contentType("application/json")
            .body(containsString("OK"))

}
```

Applications and Success Stories

Experience With EvoMaster

- Author’s of EvoMaster
- Academic tool, started in 2016
  - Around 3 millions Euro in funding from ERC and NFR
- Applied on many open-source APIs
  - found thousands of bugs
- Only tool supporting *white-box* testing for JVM
- Academic collaborations with industry

Open-Source Projects

- Found hundreds of faults in open-source projects
- Many APIs out there are not robust to receive invalid inputs, and so crashes
- Currently using 36 open-source APIs for experiments comparing fuzzers
- https://github.com/WebFuzzing/Dataset
- EvoMaster gives best results on those APIs

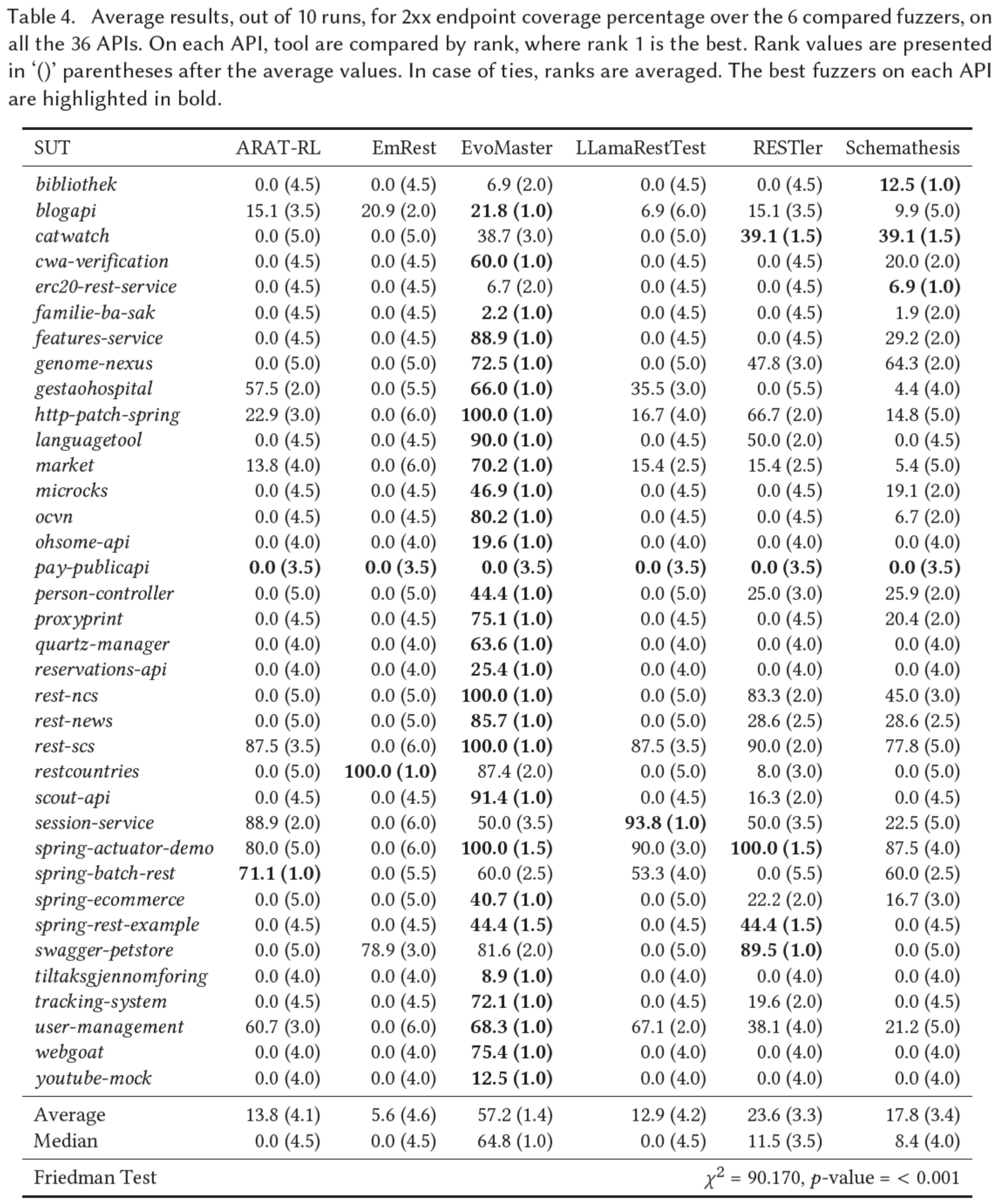


Table 4. Average results, out of 10 runs, for 2xx endpoint coverage percentage over the 6 compared fuzzers, on all the 36 APIs. On each API, tool are compared by rank, where rank 1 is the best. Rank values are presented in '()' parentheses after the average values. In case of ties, ranks are averaged. The best fuzzers on each API are highlighted in bold.

SUT	ARAT-RL	EmRest	EvoMaster	LLamaRestTest	RESTler	Schemathesis
bibliothek	0.0 (4.5)	0.0 (4.5)	6.9 (2.0)	0.0 (4.5)	0.0 (4.5)	**12.5 (1.0)**
blogapi	15.1 (3.5)	20.9 (2.0)	**21.8 (1.0)**	6.9 (6.0)	15.1 (3.5)	9.9 (5.0)
catwatch	0.0 (5.0)	0.0 (5.0)	38.7 (3.0)	0.0 (5.0)	**39.1 (1.5)**	**39.1 (1.5)**
cwa-verification	0.0 (4.5)	0.0 (4.5)	**60.0 (1.0)**	0.0 (4.5)	0.0 (4.5)	20.0 (2.0)
erc20-rest-service	0.0 (4.5)	0.0 (4.5)	6.7 (2.0)	0.0 (4.5)	0.0 (4.5)	**6.9 (1.0)**
familie-ba-sak	0.0 (4.5)	0.0 (4.5)	**2.2 (1.0)**	0.0 (4.5)	0.0 (4.5)	1.9 (2.0)
features-service	0.0 (4.5)	0.0 (4.5)	**88.9 (1.0)**	0.0 (4.5)	0.0 (4.5)	29.2 (2.0)
genome-nexus	0.0 (5.0)	0.0 (5.0)	**72.5 (1.0)**	0.0 (5.0)	47.8 (3.0)	64.3 (2.0)
gestaohospital	57.5 (2.0)	0.0 (5.5)	**66.0 (1.0)**	35.5 (3.0)	0.0 (5.5)	4.4 (4.0)
http-patch-spring	22.9 (3.0)	0.0 (6.0)	**100.0 (1.0)**	16.7 (4.0)	66.7 (2.0)	14.8 (5.0)
languagetool	0.0 (4.5)	0.0 (4.5)	**90.0 (1.0)**	0.0 (4.5)	50.0 (2.0)	0.0 (4.5)
market	13.8 (4.0)	0.0 (6.0)	**70.2 (1.0)**	15.4 (2.5)	15.4 (2.5)	5.4 (5.0)
microcks	0.0 (4.5)	0.0 (4.5)	**46.9 (1.0)**	0.0 (4.5)	0.0 (4.5)	19.1 (2.0)
ocvn	0.0 (4.5)	0.0 (4.5)	**80.2 (1.0)**	0.0 (4.5)	0.0 (4.5)	6.7 (2.0)
ohsome-api	0.0 (4.0)	0.0 (4.0)	**19.6 (1.0)**	0.0 (4.0)	0.0 (4.0)	0.0 (4.0)
pay-publicapi	**0.0 (3.5)**	**0.0 (3.5)**	**0.0 (3.5)**	**0.0 (3.5)**	**0.0 (3.5)**	**0.0 (3.5)**
person-controller	0.0 (5.0)	0.0 (5.0)	**44.4 (1.0)**	0.0 (5.0)	25.0 (3.0)	25.9 (2.0)
proxyprint	0.0 (4.5)	0.0 (4.5)	**75.1 (1.0)**	0.0 (4.5)	0.0 (4.5)	20.4 (2.0)
quartz-manager	0.0 (4.0)	0.0 (4.0)	**63.6 (1.0)**	0.0 (4.0)	0.0 (4.0)	0.0 (4.0)
reservations-api	0.0 (4.0)	0.0 (4.0)	**25.4 (1.0)**	0.0 (4.0)	0.0 (4.0)	0.0 (4.0)
rest-ncs	0.0 (5.0)	0.0 (5.0)	**100.0 (1.0)**	0.0 (5.0)	83.3 (2.0)	45.0 (3.0)
rest-news	0.0 (5.0)	0.0 (5.0)	**85.7 (1.0)**	0.0 (5.0)	28.6 (2.5)	28.6 (2.5)
rest-scs	87.5 (3.5)	0.0 (6.0)	**100.0 (1.0)**	87.5 (3.5)	90.0 (2.0)	77.8 (5.0)
restcountries	0.0 (5.0)	**100.0 (1.0)**	87.4 (2.0)	0.0 (5.0)	8.0 (3.0)	0.0 (5.0)
scout-api	0.0 (4.5)	0.0 (4.5)	**91.4 (1.0)**	0.0 (4.5)	16.3 (2.0)	0.0 (4.5)
session-service	88.9 (2.0)	0.0 (6.0)	50.0 (3.5)	**93.8 (1.0)**	50.0 (3.5)	22.5 (5.0)
spring-actuator-demo	80.0 (5.0)	0.0 (6.0)	**100.0 (1.5)**	90.0 (3.0)	**100.0 (1.5)**	87.5 (4.0)
spring-batch-rest	**71.1 (1.0)**	0.0 (5.5)	60.0 (2.5)	53.3 (4.0)	0.0 (5.5)	60.0 (2.5)
spring-ecommerce	0.0 (5.0)	0.0 (5.0)	**40.7 (1.0)**	0.0 (5.0)	22.2 (2.0)	16.7 (3.0)
spring-rest-example	0.0 (4.5)	0.0 (4.5)	**44.4 (1.5)**	0.0 (4.5)	**44.4 (1.5)**	0.0 (4.5)
swagger-petstore	0.0 (5.0)	78.9 (3.0)	81.6 (2.0)	0.0 (5.0)	**89.5 (1.0)**	0.0 (5.0)
tiltaksgjennomforing	0.0 (4.0)	0.0 (4.0)	**8.9 (1.0)**	0.0 (4.0)	0.0 (4.0)	0.0 (4.0)
tracking-system	0.0 (4.5)	0.0 (4.5)	**72.1 (1.0)**	0.0 (4.5)	19.6 (2.0)	0.0 (4.5)
user-management	60.7 (3.0)	0.0 (6.0)	**68.3 (1.0)**	67.1 (2.0)	38.1 (4.0)	21.2 (5.0)
webgoat	0.0 (4.0)	0.0 (4.0)	**75.4 (1.0)**	0.0 (4.0)	0.0 (4.0)	0.0 (4.0)
youtube-mock	0.0 (4.0)	0.0 (4.0)	**12.5 (1.0)**	0.0 (4.0)	0.0 (4.0)	0.0 (4.0)
Average	13.8 (4.1)	5.6 (4.6)	57.2 (1.4)	12.9 (4.2)	23.6 (3.3)	17.8 (3.4)
Median	0.0 (4.5)	0.0 (4.5)	64.8 (1.0)	0.0 (4.5)	11.5 (3.5)	8.4 (4.0)
Friedman Test						$\chi^2 = 90.170$, p-value = < 0.001

Experiments in the *lab* on open-source APIs gives you info on *coverage* and *fault* detection…

… but not on whether practitioners find it *useful*

Industry Collaborations

- We are university **scientists**, not tool vendors
- **Anyone** can use open-source tools
- Benefits **for industry**: priority on **feature requests** and bug fixing
- Benefits **for us**: access to industrial **case studies**

Industry Collaborations: EvoMaster at Meituan

- Large Chinese e-commerce enterprise
- EvoMaster used **daily** on **1700 microservices**, for millions of lines of code
- White-box testing Thrift RPC APIs

美团是一家科技零售公司

美团以“零售 + 科技”的战略践行“帮大家吃得更好，生活更好”的公司使命。

自2010年3月成立以来，美团持续推动服务零售和商品零售在需求侧和供给侧的数字化升级，和广大合作伙伴一起努力为消费者提供品质服务。2018年9月20日，美团在港交所挂牌上市。美团始终以客户为中心，不断加大在新技术上的研发投入。美团会和大家一起努力，更好承担社会责任，更多创造社会价值。

下载美团App

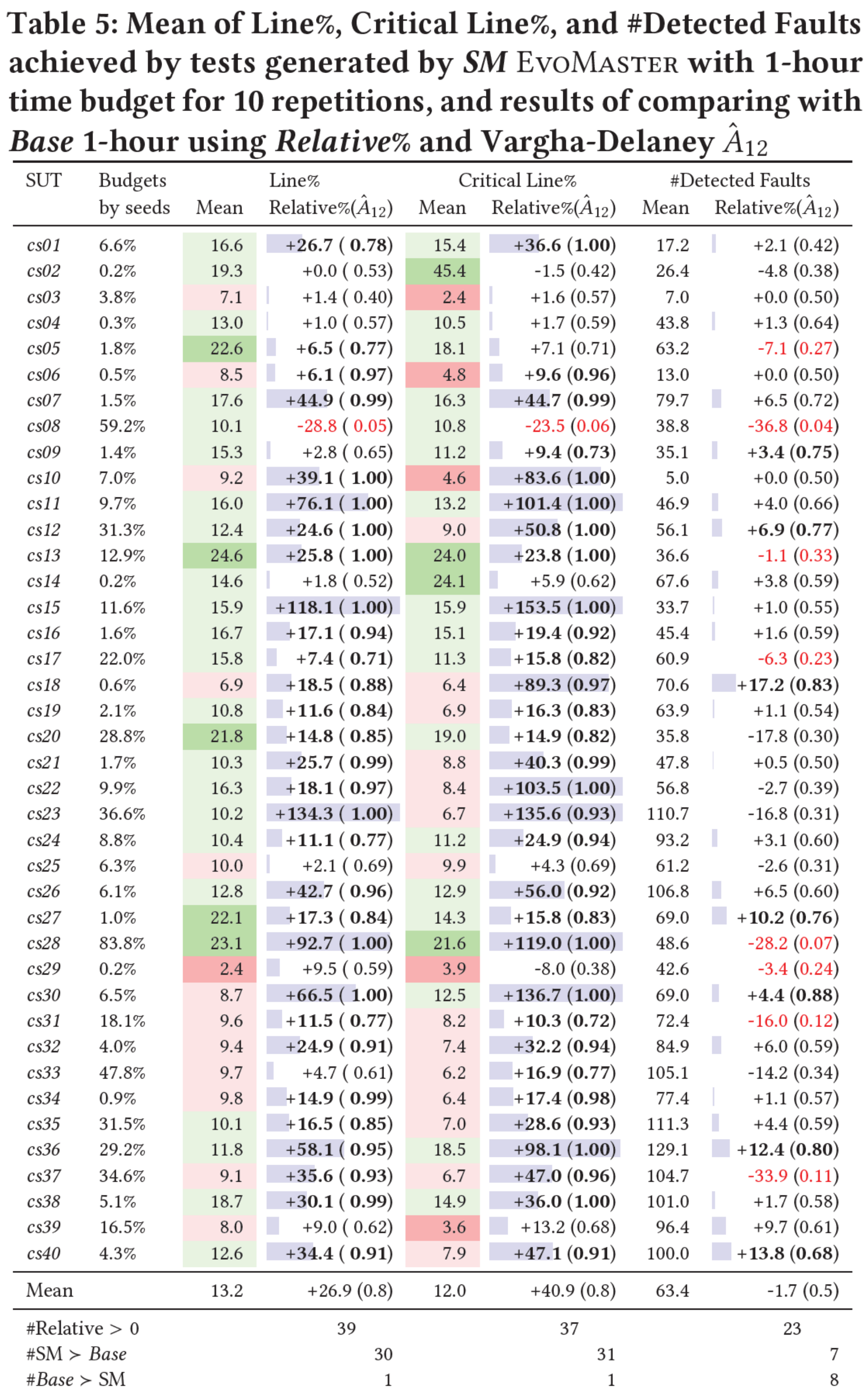


Table 5: Mean of Line%, Critical Line%, and #Detected Faults achieved by tests generated by *SM* EvoMaster with 1-hour time budget for 10 repetitions, and results of comparing with *Base* 1-hour using *Relative%* and Vargha-Delaney $\hat{A}_{12}$

SUT	Budgets by seeds	Line% Mean	Line% Relative%($\hat{A}_{12}$)	Critical Line% Mean	Critical Line% Relative%($\hat{A}_{12}$)	#Detected Faults Mean	#Detected Faults Relative%($\hat{A}_{12}$)
cs01	6.6%	16.6	**+26.7 (0.78)**	15.4	**+36.6 (1.00)**	17.2	+2.1 (0.42)
cs02	0.2%	19.3	+0.0 (0.53)	45.4	-1.5 (0.42)	26.4	-4.8 (0.38)
cs03	3.8%	7.1	+1.4 (0.40)	2.4	+1.6 (0.57)	7.0	+0.0 (0.50)
cs04	0.3%	13.0	+1.0 (0.57)	10.5	+1.7 (0.59)	43.8	+1.3 (0.64)
cs05	1.8%	22.6	**+6.5 (0.77)**	18.1	+7.1 (0.71)	63.2	-7.1 (0.27)
cs06	0.5%	8.5	**+6.1 (0.97)**	4.8	**+9.6 (0.96)**	13.0	+0.0 (0.50)
cs07	1.5%	17.6	**+44.9 (0.99)**	16.3	**+44.7 (0.99)**	79.7	+6.5 (0.72)
cs08	59.2%	10.1	-28.8 (0.05)	10.8	-23.5 (0.06)	38.8	-36.8 (0.04)
cs09	1.4%	15.3	+2.8 (0.65)	11.2	**+9.4 (0.73)**	35.1	**+3.4 (0.75)**
cs10	7.0%	9.2	**+39.1 (1.00)**	4.6	**+83.6 (1.00)**	5.0	+0.0 (0.50)
cs11	9.7%	16.0	**+76.1 (1.00)**	13.2	**+101.4 (1.00)**	46.9	+4.0 (0.66)
cs12	31.3%	12.4	**+24.6 (1.00)**	9.0	**+50.8 (1.00)**	56.1	**+6.9 (0.77)**
cs13	12.9%	24.6	**+25.8 (1.00)**	24.0	**+23.8 (1.00)**	36.6	-1.1 (0.33)
cs14	0.2%	14.6	+1.8 (0.52)	24.1	+5.9 (0.62)	67.6	+3.8 (0.59)
cs15	11.6%	15.9	**+118.1 (1.00)**	15.9	**+153.5 (1.00)**	33.7	+1.0 (0.55)
cs16	1.6%	16.7	**+17.1 (0.94)**	15.1	**+19.4 (0.92)**	45.4	+1.6 (0.59)
cs17	22.0%	15.8	**+7.4 (0.71)**	11.3	**+15.8 (0.82)**	60.9	-6.3 (0.23)
cs18	0.6%	6.9	**+18.5 (0.88)**	6.4	**+89.3 (0.97)**	70.6	**+17.2 (0.83)**
cs19	2.1%	10.8	**+11.6 (0.84)**	6.9	**+16.3 (0.83)**	63.9	+1.1 (0.54)
cs20	28.8%	21.8	**+14.8 (0.85)**	19.0	**+14.9 (0.82)**	35.8	-17.8 (0.30)
cs21	1.7%	10.3	**+25.7 (0.99)**	8.8	**+40.3 (0.99)**	47.8	+0.5 (0.50)
cs22	9.9%	16.3	**+18.1 (0.97)**	8.4	**+103.5 (1.00)**	56.8	-2.7 (0.39)
cs23	36.6%	10.2	**+134.3 (1.00)**	6.7	**+135.6 (0.93)**	110.7	-16.8 (0.31)
cs24	8.8%	10.4	**+11.1 (0.77)**	11.2	**+24.9 (0.94)**	93.2	+3.1 (0.60)
cs25	6.3%	10.0	+2.1 (0.69)	9.9	+4.3 (0.69)	61.2	-2.6 (0.31)
cs26	6.1%	12.8	**+42.7 (0.96)**	12.9	**+56.0 (0.92)**	106.8	+6.5 (0.60)
cs27	1.0%	22.1	**+17.3 (0.84)**	14.3	**+15.8 (0.83)**	69.0	**+10.2 (0.76)**
cs28	83.8%	23.1	**+92.7 (1.00)**	21.6	**+119.0 (1.00)**	48.6	-28.2 (0.07)
cs29	0.2%	2.4	+9.5 (0.59)	3.9	-8.0 (0.38)	42.6	-3.4 (0.24)
cs30	6.5%	8.7	**+66.5 (1.00)**	12.5	**+136.7 (1.00)**	69.0	**+4.4 (0.88)**
cs31	18.1%	9.6	**+11.5 (0.77)**	8.2	**+10.3 (0.72)**	72.4	-16.0 (0.12)
cs32	4.0%	9.4	**+24.9 (0.91)**	7.4	**+32.2 (0.94)**	84.9	+6.0 (0.59)
cs33	47.8%	9.7	+4.7 (0.61)	6.2	**+16.9 (0.77)**	105.1	-14.2 (0.34)
cs34	0.9%	9.8	**+14.9 (0.99)**	6.4	**+17.4 (0.98)**	77.4	+1.1 (0.57)
cs35	31.5%	10.1	**+16.5 (0.85)**	7.0	**+28.6 (0.93)**	111.3	+4.4 (0.59)
cs36	29.2%	11.8	**+58.1 (0.95)**	18.5	**+98.1 (1.00)**	129.1	**+12.4 (0.80)**
cs37	34.6%	9.1	**+35.6 (0.93)**	6.7	**+47.0 (0.96)**	104.7	-33.9 (0.11)
cs38	5.1%	18.7	**+30.1 (0.99)**	14.9	**+36.0 (1.00)**	101.0	+1.7 (0.58)
cs39	16.5%	8.0	+9.0 (0.62)	3.6	+13.2 (0.68)	96.4	+9.7 (0.61)
cs40	4.3%	12.6	**+34.4 (0.91)**	7.9	**+47.1 (0.91)**	100.0	**+13.8 (0.68)**
Mean		13.2	+26.9 (0.8)	12.0	+40.9 (0.8)	63.4	-1.7 (0.5)
#Relative > 0			39		37		23
#SM > *Base*			30		31		7
#*Base* > SM			1		1		8

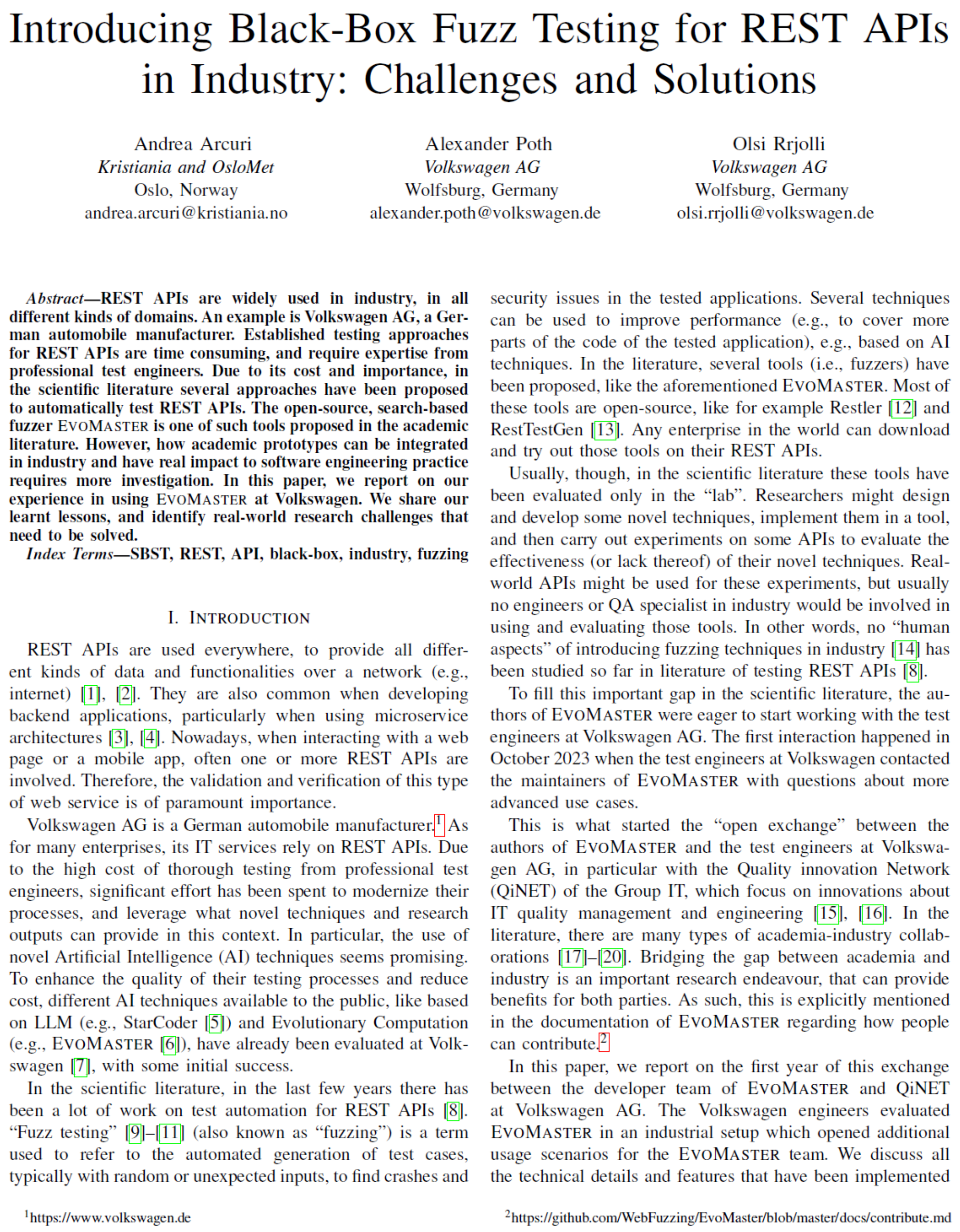
- ASE’24: “*Seeding and Mocking in White-Box Fuzzing Enterprise RPC APIs: An Industrial Case Study*”
- 40 APIs at Meituan
- More than 5M LOC
- Automatically found hundreds of faults

Industry Collaborations: EvoMaster at Volkswagen

- Large German car manufacturer
- EvoMaster used for black-box testing of REST APIs

https://www.volkswagen.no/no.html
Meny
Velg forhandler
Les mer
Bygg din ID.7 GTX stasjonsvogn

ID.7 GTX stasjonsvogn med fire-hjulstrekk: Fra kr 579 400



Introducing Black-Box Fuzz Testing for REST APIs in Industry: Challenges and Solutions

Andrea Arcuri
Kristiania and OsloMet
Oslo, Norway
andrea.arcuri@kristiania.no

Alexander Poth
Volkswagen AG
Wolfsburg, Germany
alexander.poth@volkswagen.de

Olsi Rrjolli
Volkswagen AG
Wolfsburg, Germany
olsi.rrjolli@volkswagen.de

***Abstract*—REST APIs are widely used in industry, in all different kinds of domains. An example is Volkswagen AG, a German automobile manufacturer. Established testing approaches for REST APIs are time consuming, and require expertise from professional test engineers. Due to its cost and importance, in the scientific literature several approaches have been proposed to automatically test REST APIs. The open-source, search-based fuzzer EVOMASTER is one of such tools proposed in the academic literature. However, how academic prototypes can be integrated in industry and have real impact to software engineering practice requires more investigation. In this paper, we report on our experience in using EVOMASTER at Volkswagen. We share our learnt lessons, and identify real-world research challenges that need to be solved.**

***Index Terms*—SBST, REST, API, black-box, industry, fuzzing**

I. Introduction

REST APIs are used everywhere, to provide all different kinds of data and functionalities over a network (e.g., internet) [1], [2]. They are also common when developing backend applications, particularly when using microservice architectures [3], [4]. Nowadays, when interacting with a web page or a mobile app, often one or more REST APIs are involved. Therefore, the validation and verification of this type of web service is of paramount importance.

Volkswagen AG is a German automobile manufacturer.[1] As for many enterprises, its IT services rely on REST APIs. Due to the high cost of thorough testing from professional test engineers, significant effort has been spent to modernize their processes, and leverage what novel techniques and research outputs can provide in this context. In particular, the use of novel Artificial Intelligence (AI) techniques seems promising. To enhance the quality of their testing processes and reduce cost, different AI techniques available to the public, like based on LLM (e.g., StarCoder [5]) and Evolutionary Computation (e.g., EVOMASTER [6]), have already been evaluated at Volkswagen [7], with some initial success.

In the scientific literature, in the last few years there has been a lot of work on test automation for REST APIs [8]. "Fuzz testing" [9]–[11] (also known as "fuzzing") is a term used to refer to the automated generation of test cases, typically with random or unexpected inputs, to find crashes and security issues in the tested applications. Several techniques can be used to improve performance (e.g., to cover more parts of the code of the tested application), e.g., based on AI techniques. In the literature, several tools (i.e., fuzzers) have been proposed, like the aforementioned EVOMASTER. Most of these tools are open-source, like for example Restler [12] and RestTestGen [13]. Any enterprise in the world can download and try out those tools on their REST APIs.

Usually, though, in the scientific literature these tools have been evaluated only in the "lab". Researchers might design and develop some novel techniques, implement them in a tool, and then carry out experiments on some APIs to evaluate the effectiveness (or lack thereof) of their novel techniques. Real-world APIs might be used for these experiments, but usually no engineers or QA specialist in industry would be involved in using and evaluating those tools. In other words, no "human aspects" of introducing fuzzing techniques in industry [14] has been studied so far in literature of testing REST APIs [8].

To fill this important gap in the scientific literature, the authors of EVOMASTER were eager to start working with the test engineers at Volkswagen AG. The first interaction happened in October 2023 when the test engineers at Volkswagen contacted the maintainers of EVOMASTER with questions about more advanced use cases.

This is what started the "open exchange" between the authors of EVOMASTER and the test engineers at Volkswagen AG, in particular with the Quality innovation Network (QiNET) of the Group IT, which focus on innovations about IT quality management and engineering [15], [16]. In the literature, there are many types of academia-industry collaborations [17]–[20]. Bridging the gap between academia and industry is an important research endeavour, that can provide benefits for both parties. As such, this is explicitly mentioned in the documentation of EVOMASTER regarding how people can contribute.[2]

In this paper, we report on the first year of this exchange between the developer team of EVOMASTER and QiNET at Volkswagen AG. The Volkswagen engineers evaluated EVOMASTER in an industrial setup which opened additional usage scenarios for the EVOMASTER team. We discuss all the technical details and features that have been implemented

[1] https://www.volkswagen.de

[2] https://github.com/WebFuzzing/EvoMaster/blob/master/docs/contribute.md

- Wrote on the experience of applying EM at VW
- Features needed
- Open challenges

Challenges

- Lot of research in academia for better test generation strategies
- Cover larger parts of API code
- Find more faults (and fault types)
    - not all faults have same severity
- Test readability
    - testers still need to look at generated tests

Hmmmm… why not just using a LLM?

- Input: OpenAPI schema
- Output: test cases

- Can work, but poor results
- You would miss all information from the responses of API
- No way to tell if a test case has found a fault
- You must interact with the API

Conclusion

- Many success stories about fuzzing
- REST fuzzing (and partially GraphQL and RPC) is getting momentum
- *Several open-source tools are available, to try out, today!*
    - we are biased about EvoMaster, but Schemathesis and CATS are good alternatives

linkedin.com/in/**arcuri82/**

Q/A

Thanks!

Tools: on GitHub:

- **WebFuzzing/EvoMaster**
- schemathesis/schemathesis
- Endava/cats
- microsoft/restler-fuzzer